DAFx’04 Conference Committees

Scientific Committee

Daniel Arfib CHRS-LMA, Marseille, France
Nicola Bernardini Media Innovation Unit - Firenze Tecnologia, Florence, Italy
Nishan Canaragajah Bristol University, UK
F. Javier Casajús ETSI Telecommunication-Universidad Politécnica de Madrid, Spain
Perry Cook Princeton University, USA
Laurent Daudet Laboratoire d’Acoustique Musicales, Université Pierre et Marie Curie (Paris VI), France
Philippe Depalle McGill University, Montréal, Canada
Giovanni De Poli University of Padova, Italy
Markus Erne Scopein Research, Aarau, Switzerland
Gianpaolo Evangelista Federico II University of Naples, Italy
Emmanuel Favreau Institut National de l’Audiovisuel, Paris, France
Simon Godsil Cambridge University, UK
Robert Hoeldrich Institute of Electronic Music and Acoustics, Graz, Austria
Jean-Marc Jot Creative Labs, USA
Matti Karjalainen Helsinki University of Technology, Finland
Søren Nielsen TC Electronic, Denmark
Rudolf Rabenstein Erlangen-Fraunhofer University, Germany
Davide Rocchesso University of Verona, Italy
Jøran Rudi NOTAM, Oslo, Norway
Francis Rumsey Surrey University, UK
Mark Sandler Queen Mary, University of London, UK
Xavier Serra Pompeu Fabra University, Barcelona, Spain
Julius Smith Stanford University, USA
Todor Todoroff ARTeM, Bruxelles, Belgium
Jan Tro Norwegian University of Science and Technology, Trondheim, Norway
Udo Zölzer Helmut-Schmidt University, Hamburg, Germany

Organizing Committee

Gianpaolo Evangelista Program Chair
Sergio Cavaliere Scientific Program
Giancarlo Sica Musical Events
Italo Testa Proceedings and Program
Welcome to DAFx’04, Naples

On behalf of the Organizing Committee, it is a great honor for me to welcome you all to DAFx’04. This 7th edition of the Digital Audio Effects Conference is hosted by one of the oldest Universities in the world. We are therefore bound to humbly ask the Magnificent Frederic II of Swabia, founder in 1224, to assist us throughout our journey...

Fortunately, our journey is far from being impervious. The previous DAFx conferences, held in Barcelona, Trondheim, Verona, Limerick, Hamburg and London, set the most important milestones, which highly simplified the organization of the present edition.

Yes, I can confirm that DAFx is a highly reputed event. Sprouted from the European COST G6 action, the organization of DAFx has continued well beyond the duration of the project and even in the absence of major sponsors. The number, quality level and geographical distribution of the scientific contributions received this year are by themselves sufficient to testify the world premiership of this event in digital audio signal processing.

I am grateful to all our sponsors who made the organization of this event possible. In particular, I wish to thank the Department of Physical Sciences and the University of Naples for financial and logistic support and the foundation Città della Scienza for hosting the concert-demonstration. I also wish to thank the non-financial sponsors, Associazione di Informatica Musicale Italiana (AIMI) and the European Association for Signal Image Processing (EURASIP).

I wish to thank the members of the DAFx’04 Organizing Committee, Italo Testa and Sergio Cavaliere, for their continuous and consistent help. I am particularly indebted to Maestro Giancarlo Sica for organizing the DAFx concert. Special thanks go to Christof Faller for organizing the special session on spatial audio processing, to the invited speakers Thomas Sporer, Juha Merimaa, Harald Viste, Jürgen Herre, and Heiko Purnhagen, and to the keynote speakers Antonio Camurri, Günther Theile and Curtis Roads.

I would like to dedicate this edition of DAFx to the memory of Prof. Aldo Piccialli who, with two workshops (Sorrento 1988 and Capri 1991) and two follow-up edited books, pioneered events that can be considered the precursors of the present DAFx meetings.

It comforts me to know that this 7th edition of DAFx is not going to be the last step of a major scale. Next year the conference will be hosted in Madrid, Spain and the following year in Montréal, Canada. At the moment, reservations to host DAFx conferences have been advanced up to the year 2009.

Therefore, “Long life to DAFx!” and my best wishes to you all to enjoy the event.

Gianpaolo Evangelista,
Chairman of DAFx’04 Organizing Committee
Contents

Keynote: Gesture Control
1 Multimodal Interfaces for Expressive Sound Control
   Antonio Camurri

Session: Gesture Control
5 The Sounding Gesture: An Overview
   Amalia de Götzen
11 Effect of Latency on Playing Accuracy of Two Gesture Controlled Continuous Sound Instruments Without Tactile Feedback
   Teemu Mäki-Patola and Perttu Hämäläinen
17 Audio-Based Gesture Extraction on the ESITAR Controller
   Ajay Kapur, George Tzanetakis, and Peter Driessen

Session: Time-Frequency and Spectral Processing
22 Sparse and Structured Decompositions of Audio Signals in Overcomplete Spaces
   Laurent Daudet
27 Detection of Clicks Using Sinusoidal Modeling for the Confirmation of the Clicks
   Esteban Alvarez, Rafael Mendez, and Guillermo Langwagen
33 Audio Analysis, Visualization, and Transformation with the Matching Pursuit Algorithm
   Garry Kling and Curtis Roads

Poster Session I
38 Removing Crackle from an LP Record via Wavelet Analysis
   Pavel Rajmic and Jaroslav Klimek
42 Spectral Delays with Frequency Domain Processing
   David Kim-Boyle
45 Exponential Weighting Method for Sample-by-Sample Update of Warped AR-Model
   Kari Roth and Ismo Nauppinen
49 Emulating Rough and Growl Voice in Spectral Domain
   Alex Loscos and Jordi Bonada
53 AIDE, A New Digital Audio Effects Development Environment
   Victor Lazzarini and Rory Walsh
57 ADAM - A 64 Channel General Purpose Real-Time Audio Signal Processor
   Robert Trausmuth and Michael Kollegger
61 Non-Linear Digital Implementation of the Moog Ladder Filter
   Antti Huovilainen

Session: Sound Transformations and Audio Effects I
65 A Fast Mellin Transform with Applications in DAFX
   Antonio De Sena and Davide Rocchesso
70 Transforming Singing Voice Expression - The Sweetness Effect
   Lars Fabig and Jordi Janer
76 Enhanced Time-Stretching Using Order-2 Sinusoidal Modeling
   Sylvain Marchand and Martin Raspaud
83 An Efficient Phasiness Reduction Technique for Moderate Audio Time-Scale Modification
   David Dorran, Robert Lawlor, and Eugene Coyle
**Session: Synthesis by Physical Models**

89  
A Piano Model Including Longitudinal String Vibrations  
Balázs Bank and László Sujbert

95  
The Feathered Clarinet Reed  
Tamara Smyth, Jonathan Abel, and Julius O. Smith

101  
Digital Sound Synthesis of Brass Instruments by Physical Modeling  
Bernhard Krach, Stefan Petrausch, and Rudolf Rabenstein

107  
Recent Advances in Physical Modeling with K- and W-Techniques  
Matti Karjalainen, Jyrki Pakarinen, Cumhur Erkut, Paulo A. A. Esquef, and Vesa Välimäki

113  
Computation of Nonlinear Filter Networks Containing Delay-Free Paths  
Federico Fontana, Federico Avanzini, and Davide Rocchesso

119  
Modal-Type Synthesis Techniques for Nonlinear Strings with an Energy Conservation Property  
Stefan Bilbao

**Keynote: Spatial Audio Processing**

125  
Wave Field Synthesis - A Promising Spatial Rendering Concept  
Günther Theile

**Session: Spatial Audio Processing (invited) I**

133  
Wave Field Synthesis - Generation and Reproduction of Natural Sound Environments  
Thomas Sporer

139  
Spatial Impulse Response Rendering  
Juha Merimaa and Ville Pulkki

145  
Binaural source localization  
Harald Viste and Gianpaolo Evangelista

**Session: Spatial Audio Processing (invited) II**

151  
Parametric Coding of Spatial Audio  
Christof Faller

157  
From Joint Stereo to Spatial Audio Coding - Recent Progress and Standardization  
Jürgen Herre

163  
Low Complexity Parametric Stereo Coding in MPEG-4  
Heiko Purnhagen

**Poster Session II**

169  
Computational Real-Time Sound Simulation of Rain  
Stanley J. Miklavcic, Andreas Zita, and Per Arvidsson

173  
A Strategy for the Modular Implementation of Physics-Based Models  
Matthias Rath

177  
Implementing Loudness Models in MATLAB  
Joseph Timoney, Thomas Lysaght, Marc Schoenwiesner, and Lorcán Mac Manus

181  
Timbral Attributes for Objective Quality Assessment of the Irish Tin Whistle  
Joseph Timoney, Thomas Lysaght, Lorcán Mac Manus, and Andreas Schwarzbacher

185  
Single-Note Ornamentation Transcription for the Irish Tin Whistle Based on Onset Detection  
Mikel Gainza, Eugene Coyle, and Robert Lawlor

189  
Efficiently Computable Similarity Measures for Query by Tapping Systems  
Gunnar Eisenberg, Jan-Mark Batke, and Thomas Sikora

193  
An Open Source Tool for Semi-Automatic Rhythmic Annotation  
Fabien Gouyon, Nicolas Wack, and Simon Dixon

197  
A Spectral-Filtering Approach to Music Signal Separation  
Mark Every and John Szymanski
Session: Sound Analysis and Information Retrieval

201  Audio Processing Using Haskell
    Henning Thielemann

207  A comparison Between Fixed and Multiresolution Analysis for Onset Detection in Musical Signals
    Chris Duxbury, Juan Pablo Bello, Mark Sandler, and Mike Davies

212  Piano Transcription Using Pattern Recognition: Aspects on Parameter Extraction
    Luis I. Ortiz-Berenguer, F. Javier Casajús-Quirós, Marisol Torres-Guijarro, and Jon Ander Beracoechea

217  On Finding Melodic Lines in Audio Recordings
    Matija Marolt

222  Musical Instrument Identification in Continuous Recordings
    Arie Livshin and Xavier Rodet

228  Bayesian Identification of Closely-Spaced Chords from Single-Frame STFT Peaks
    Randal Leistikow, Harvey Thomburg, Julius O. Smith, and Jonathan Berger

234  A New Score Function for Joint Evaluation of Multiple F0 Hypotheses
    Chunghsin Yeh and Axel Röbel

Session: Room Acoustics and Wave Field Synthesis I

240  Sound Source Separation: Azimuth Discrimination and Resynthesis
    Dan Barry, Robert Lawlor, and Eugene Coyle

245  Source Separation for WFS Acoustic Opening Applications
    Jon Ander Beracoechea, Marisol Torres-Guijarro, Luis I. Ortiz-Berenguer, and F. Javier Casajús-Quirós

250  Analysis of Certain Challenges for the Use of Wave Field Synthesis in Concert-Based Applications
    Terence Caulkins, Etienne Corteel, and Olivier Warusfel

256  A Maximum Likelihood Approach to Blind Audio De-Reverberation
    Massimiliano Tonelli, Nikolaos Mitianoudis, and Mike Davies

Session: Room Acoustics and Wave Field Synthesis II

262  Practical Implementation of the 3D Tetrahedral TLM Method and Visualization of Room Acoustics
    Stanley J. Miklavcic and Johan Ericsson

268  RoomWeaver: A Digital Waveguide Mesh Based Room Acoustics Research Tool
    Mark Beeson and Damian Murphy

274  Real Time Modeling of Acoustic Propagation in Complex Environments
    Fabio Antonacci, Marco Foco, Augusto Sarti, and Stefano Tubaro

280  Decorrelation Techniques for the Rendering of Apparent Sound Source Width in 3D Audio Displays
    Guillaume Potard and Ian Burnett

Session: Sound Transformations and Audio Effects II

285  Digital Emulation of Analog Compinging Algorithms for FM-Radio Transmission
    Jürgen Peissig, Jan Remmer ter Haseborg, Florian Keiler, and Udo Zölzer

291  High Quality Voice Transformations Based on Modeling Radiated Voice Pulses in Frequency Domain
    Jordi Bonada

296  Adaptive Effects Based on STFT, Using a Source-Filter Model
    Vincent Verfaille and Philippe Depalle

Keynote: Granular Synthesis

302  Gabor’s Legacy: New Developments in Granular Analysis and Synthesis
    Curtis Roads
Poster Session III

303 Software for Measuring and Improving Esophageal Voices
Begoña García and Javier Vicente

307 Improvement of Esophageal Voices' Pitch
Begoña García, Javier Vicente, Ibon Ruiz, Asier Alonso, and Estivaliz Loyo

311 Acoustical Simulations of the Human Vocal Tract Using the 1D and 2D Digital Waveguide Software Model
Jack Mullen, David Howard, and Damian Murphy

315 Room Impulse Response Shaping for Enhancement of Perceived Spaciousness and Auditory Distance
Dae-young Jang, Han-gil Moon, Kyeongok Kang, and Hoe-kyung Jung

319 Semi-automatic Ambience Generation
Pedro Cano, Lars Fabig, Fabien Gouyon, Markus Koppenberger, Alex Loscos, and Alvaro Barbosa

323 MATCONCAT: An Application for Exploring Concatenative Sound Synthesis Using MATLAB
Bob L. Sturm

327 Sonification of the Fission Model as an Event Generation System
Sinan Bokesoy

331 Memory Reduction Technique of Spreading Function in MPEG AAC Encoder
Shih-Way Huang, Liang-Gee Chen, and Tsung-Han Tsai

335 An MDCT-Based Psychoacoustic Model Co-Processor Design for MPEG-2/4 AAC Audio Encoder
Tsung-Han Tsai, Yi-Wen Wang, and Shih-Way Huang

Session: Sound Models

339 Synthesis by Mathematical Models
Domenico Vicinanza and Vittorio Cafagna

345 Sound Texture Modeling and Time-Frequency LPC
Xinglei Zhu and Lonce Wyse

Session: Compositional Issues

350 Making Sound with Numbers, Six Years Later
Nicola Bernardini, Damien Cirotteau, Free Ekanayaka, and Andrea Glorioso

356 Computer Instrument Development and the Composition Process
Victor Lazzarini

361 Event-Synchronous Music Synthesis
Tristan Jehan

Session: Audio for Multimedia

367 Musical Computer Games Played by Singing
Perttu Hämäläinen, Teemu Mäki-Patola, Ville Pulkki, and Matti Airas

372 Digital Audio Effects Applied Directly on a DSD Bitstream
Josh Reiss and Mark Sandler

378 Hierarchical Organization and Visualization of Drum Sample Libraries
Elias Pampalk, Peter Hlavac, and Perfecto Herrera

Session: Sonification and Auditory Displays

384 Spatial Auditory Displays - A study on the use of virtual audio environments as interfaces for users with visual disabilities
Christopher Frauenberger, Veronika Putz, and Robert Höldrich

390 Visualization of Sound as a Control Interface
Anne Sedes, Benoît Courribet, and Jean-Baptiste Thiébaut

Authors' Index (inside back cover)
MULTIMODAL INTERFACES FOR EXPRESSIVE SOUND CONTROL

Antonio Camurri

InfoMus Lab,
DIST - University of Genova, Genova, Italy
http://www.infomus.dist.unige.it antonio.camurri@unige.it

ABSTRACT

This paper introduces research issues on multimodal interaction and interfaces for expressive sound control. We introduce Multisensory Integrated Expressive Environments (MIEEs) as a framework for Mixed Reality applications in the performing arts. Paradigmatic contexts for applications of MIEEs are multimedia concerts, interactive dance/music/video installations, interactive museum exhibitions, distributed cooperative environments for theatre and artistic expression. MIEEs are user-centred systems able to interpret the high-level information conveyed by performers through their expressive gestures and to establish an effective multisensory experience taking into account expressive, emotional, affective content. The lecture discusses some main issues for MIEEs and presents the EyesWeb (www.eyesweb.org) open software platform which has been recently redesigned (version 4) in order to better address MIEE requirements. Short live demonstrations are also presented.

1. INTRODUCTION

A number of interactive systems are currently available to process audio and/or video streams: e.g., PureData, Max/MSP (www.cycling74.com), Isadora (www.troikatronix.com). These and other systems are particularly oriented toward a single modality of interaction, i.e., they might perform well when working with audio only (PureData, Max) or video only (Isadora). Support to sensory fusion and multimodality is very low if any. In the recent years several new requirements emerged for interactive systems (see e.g. [1]). Multimodal interfaces are not only a matter of working with streams of different types or with different sensors, but mainly concerns the ability to work at different abstraction levels and to support integrated processing of different channels [7]. The EU-IST project MEGA (Multisensory Expressive Gesture Applications, www.megaproject.org) defined a conceptual framework for multimodal expressive gesture processing, structured on four layers [17]. This paper gives a personal view on the perspective on next generations of interactive systems, by introducing Multisensory Integrated Expressive Environments (MIEEs), and introduces some latest developments around the EyesWeb research project, which aims at a partial implementation of MIEEs.

2. BEYOND INTERACTIVE SYSTEMS

The real-time multimodal processing and the modelling of expressive gesture in on-stage interactive performances is a challenge for both scientific and artistic research [1,9]. Multimodality and expressiveness [9,17] are intended to contribute to improve state of the art hyper- and virtual musical instruments [2,3,8], interactive dance, and interactive performances in general where technology is not only a tool, but rather it is integrated with art at the level of language and it becomes something intrinsic to the artwork. This leads to the concept of Multisensory Integrated Expressive Environments (MIEEs) [17]. MIEEs can be conceived of as a new generation of musical instruments based on real-time and intelligent human-machine interaction [18]: new musical instruments as a holistic human-machine concept based on an assembly of modular input/output devices and musical software components that are arranged according to essential human musical content processing capabilities. MIEEs aim at providing the extended digital platforms for the exchange of expressiveness through cross-modal interactions at levels that go beyond the classical multimedia approaches in art.

An example of a partial exploitation of the concept of MIEEs in large scale performances is in the work “Cronaca del Luogo” by Luciano Berto (opening of Salzburg Festival, 1999) in which the EyesWeb system (www.eyesweb.org) was used to control in real-time the processing of the voice of the main character depending on an analysis of the performer’s gestures. More recent examples include the public performances in the framework of the EU-IST Project MEGA (Multisensory Expressive Gesture Applications, www.megaproject.org), ranging from medium-scale events, e.g., the concert “Allegoria dell’opinione verbale” by Roberto Doati, to large-scale events, e.g., “Medea” by Adriano Guarnieri [5].

The design of MIEEs is challenging and many research issues have still to be faced. For example, systems must be endowed with the capability of interpreting performers’ gestures, and in particular expressiveness in the context where and when the gesture is performed. A MIEE should keep into account of spatial, temporal, and content memory. Information contained in performers’ gesture may be structured on several layers of complexity. A particular emphasis is on affective, expressive, emotional information. In fact, it is the capability of interpreting expressive information that allows interaction of technology and art at the level of the language art employs to convey content and to provide the audience with an aesthetical experience. In this framework, a central role is assumed by research on expressive gesture, i.e., on the high-level emotional, affective content gesture conveys, on how to analyse and process this content, on how to use it in the development of innovative multimodal interactive systems able to provide users with natural expressive interfaces [9]. Related previous research concerns Affective Computing [10], KANSEI Information Processing [11], recent studies on the communication of expressive content or “implicit messages” [12], work by psychologists (e.g.,[13,14,15]) and - from art and humanities - theories from...
choreography (e.g., Rudolf Laban’s Theory of Effort) and music composition (e.g., Schaeffer’s Morphology).

Another key issue is the development of strategies for controlling and/or generating audio in real-time. That is, even if algorithms are able to correctly and reliably interpret high-level expressive information from gesture were available, the problem of if and how to use such information in an artistic performance still remains open. In particular, a challenging direction is on the control of sound synthesis techniques using multimodal gesture analysis cues. A model for the mapping of cues (at different levels of detail) from multimodal expressive gesture to the parameters controlling sound synthesis is a very interesting direction in which some steps have already done (see for example [19]). For example, an interesting research direction is to explore models where the natural “physicality” and meaning of cues related to expressive gesture is mapped on parameters of synthesis techniques by physical models or similar techniques where parameters have “physical”, non abstract meanings.

A further difficulty is due to artistic choices of the designer of the performance, i.e., how much degrees of freedom the designer wishes to leave to the automatic systems in the control process, therefore abandoning the interaction metaphor of the “musical instrument” and going toward a dialog metaphor: in other words the role of technology in the artwork and, from a certain point of view, the concept of artwork. These aspects have been partially faced with the definition of the concept of expressive autonomy [16] and have been further investigated in the framework of the aforementioned EU IST MEGA project with particular reference to the definition of a conceptual architecture for modelling possible control (mapping) strategies.

3. MODELING EXPRESSIVE GESTURE

An important issue concerns the modeling and processing of expressive gesture. To this aim, a first problem concerns the identification of a suitable collection of descriptors (cues) that can be used for describing expressive gesture. Secondly, algorithms have to be defined and implemented to extract measures for such descriptors. Finally, data analysis has to be performed on these measures in order to obtain high-level information. This is only a rough sketch of the analysis process: a review of some of our recent research on algorithms for extraction of cues at several level of abstraction, from low-level signal-related cues to high-level analysis of expressive content is available in [9]. Such algorithms provide the input for the strategies for gestural control of multimedia output, and in particular of sound synthesis.

Expressive cues are likely to be structured on several layers of complexity. In analysis of dance fragments using video cameras for example, some cues can be directly measured on the video frames coming from a single video camera observing the dancer. Others may need more elaborate processing or 3D information. For example, it may be needed to identify and separate expressive gestures in a movement sequence in order to compute features that are strictly related to single gestures (e.g., duration, directness, fluency).

In the framework of the EU-IST project MEGA (Multisensory Expressive Gesture Applications, [www.megaproject.org](http://www.megaproject.org)) a conceptual framework for expressive gesture processing has been defined, structured on four layers [17]. Layer 1 (Physical Signals) includes algorithms for gathering data captured by sensors such as video cameras, microphones, on-body sensors (e.g., accelerome-
ters), sensors of a robotic system, environmental sensors. Layer 2 (Low-level features) extracts from the sensors data a collection of low-level cues describing the gesture being performed. In case of dance, for example, cues include kinematical measures (speed, acceleration of body parts), detected amount of motion, amount of body contraction/expansion. Layer 3 (Mid-level features and maps) deals with two main issues: segmentation of the input stream (movement, music) in its composing gestures, and representation of such gestures in suitable spaces. Thus, the first problem here is to identify relevant segments in the input stream and associate to them the cues deemed important for expressive communication. For example, in dance analysis a fragment of a performance might be segmented into a sequence of gestures where gesture’s boundaries are detected by studying velocity and direction variations. Measurements performed on a gesture are translated to a vector that identifies it in a semantic space representing categories of semantic features related to emotion and expression. Sequences of gestures in space and time are therefore transformed in trajectories in such a semantic space. Trajectories can then be analysed e.g., in order to find similarities among them and to group them in clusters. Layer 4 (Concepts and structures) is directly involved in data analysis and in extraction of high-level expressive information. In principle, it can be conceived as a conceptual network mapping the extracted features and gestures into (verbal) conceptual structures. For example, a dance performance can be analysed in term of the performer’s conveyed emotional intentions, e.g., the basic emotions anger, fear, grief, and joy. However, other outputs are also possible: for example, a structure can be envisaged describing the Laban’s conceptual framework of gesture Effort, i.e., Laban’s types of Effort such as “pushing”, “gliding”, etc. Experiments can also be carried out aiming at modelling spectators’ engagement. Machine learning techniques can be employed ranging from statistical techniques (e.g., multiple regression and generalized linear techniques), to fuzzy logics or probabilistic reasoning systems (e.g., Bayesian networks), to various kinds of neural networks (e.g., classical back-propagation networks, Kohonen networks), support vector machines, decision trees. In a recent experiment described in [4] we tried to classify expressive gesture in dance performance in term of the four basic emotions anger, fear, grief, and joy. Results showed a rate of correct classification for the automatic system (five decision tree models) in between chance level and spectators’ rate of correct classification. In another experiment, discussed in the same paper, we measured the engagement of listeners of a music performance (a Skrabin’s Etude) and analysed correlations with extracted audio cues and with cues obtained from the movement of the performer (a pianist).

4. THE EYESWEB OPEN PLATFORM

The EyesWeb open platform ([www.eyesweb.org](http://www.eyesweb.org)) has been designed with a special focus on the multimodal analysis and processing of non-verbal expressive gesture in human movement and music signals. It was developed at InfoMus Lab at DIST - University of Genova and recently it has been enhanced and re-engineered to support features of MIEEs (version 4). EyesWeb consists of a number of integrated hardware and software modules that can be easily interconnected and extended in a visual environment. The EyesWeb software includes a development environment and a set of libraries of reusable software components that can be assembled by the user in a visual language to build
patches as in common computer music languages inspired to analog synthesizers. EyesWeb is open so it is easily possible to extend it by third parties with libraries and plugins.

Besides its wide use in artistic projects, EyesWeb is used to support experiments on computational models of non-verbal expressive communication, on mapping, at different levels, gestures from different modalities (e.g., human full-body movement, music) onto real-time generation of multimedia output (e.g., sound, music, visual media, mobile scenery). It allows fast development and experiment cycles of interactive performance setups.

Recent improvements concern better support to cross-media and integrated real-time processing of different streams (e.g. audio and video), new libraries for real-time processing of expressive gesture, support to XML and many other features in the direction of support to MIEEs. Modules and patches collocated at different layers in the conceptual framework (see previous Section) are available: modules to extract low-level parameters from audio and motion data (e.g., the coordinates of the baricenter of a dancer’s silhouette and its bounding rectangle, the loudness and roughness of an audio excerpt), modules to extract mid-level features and expressive cues (e.g., body contraction/expansion, amount of detected motion, music tempo, articulation), high-level mappers (e.g., neural networks, Bayesian networks, Support Vector Machines). Many of such modules are included in the EyesWeb Expressive Gesture Processing Library [9], which now also includes 3D cues.

EyesWeb also supports distributed applications, e.g., patches running on several PCs, multi-user patches. In order to help programmers in developing blocks and extend the system, the EyesWeb Wizard software tool has been developed. Users can develop autonomously (i.e., possibly independently from EyesWeb) the algorithms and the basic software skeletons of their own modules. Then, the Wizard supports them in the process of transforming algorithms in integrated EyesWeb modules.

Multiple versions of modules (versioning mechanism) are supported by the system, e.g., allowing the use in patches of different versions of the same data-type or module. EyesWeb has been the basic platform of the MEGA EU IST project. In the EU V Framework Program it has also been adopted in the IST CARE HERE and IST MEDIATE projects on therapy and rehabilitation and by the MOSART network for training of young researchers. In the EU VI Framework Program it has been adopted by the TAI-CHI project (Tangible Acoustic Interfaces for Computer-Human Interaction) and by the Networks of Excellence ENACTIVE and HUMAINE. EyesWeb is fully available at its website [www.eyesweb.org](http://www.eyesweb.org). Public newsgroups also exist and are daily managed to support the growing EyesWeb community (several thousands of users), including individuals, universities, research institutes, and industries.

5. CONCLUSIONS

This paper introduces MIEEs and presents a partial implementation: EyesWeb 4 and related developments. This is only a first step in the directions sketched in the paper: much work remains to be done. We are in particular working at control strategies and on the so-called META-EyesWeb [9]. In particular, META-EyesWeb is a layer above EyesWeb able to supervise and to schedule execution of patches and subpatches according to adaptive interactive narrative structures (therefore beyond the metaphor of musical instrument). For example, the META-EyesWeb layer can support simple timelines of activation of patches in live electronics performances as, e.g., in Max. But, more interestingly, it supports a dynamic graph of execution (i.e., an interactive narrative structure) where each node is a (sub)patch and each link defines the semantics on how to pass from its input patch to its output patch. For example, it can be defined a “fading” behaviour between two patches, whose parameters can depend on previous history and concurrent active patches.

6. ACKNOWLEDGEMENTS

I am grateful to the colleagues of the Staff of the InfoMus Lab, with whom I had and have the pleasure to work in the research projects presented in this paper. Research is partially supported by the EU IST TAI-CHI project.

7. REFERENCES


Antonio Camurri (born in Genova, Italy, in 1959; ’84 Master Degree in Electrical Engineering in 1984; Ph.D. in Computer Engineering in 1991) is Associate Professor at DIST-University of Genova (Faculty of Engineering), where he teaches “Software engineering” and “Multimedia Systems”. He is founder and scientific director of the InfoMus Lab at DIST-University of Genova (www.infomus.dist.unige.it). InfoMus Lab participated to the exploitation of research results in several international artistic productions (e.g. Salzburg Festival 1999, Teatro La Scala, Milano, 1996), in museum and science centre interactive exhibits, in entertainment multimedia, in multimedia systems for therapy and rehabilitation. His research interests include sound and music computing, multimodal intelligent interfaces, computational models of non-verbal expressive gesture, interactive multimodal-multimedia systems for museum, theatre, music, entertainment, therapy and rehabilitation. Project Coordinator of the EU IST-E3 Project MEGA (Multisensory Expressive Gesture Applications, 2000-2005, www.megaproject.org), currently he is local project manager of several EU Projects in the VI Framework Programme (e.g. NoEs ENACTIVE, HUMAINE, IST TAI-CHI, CA 5252) and of industry contracts. He is member of the ExCom of the IEEE CS TC on Computer Generated Music and Associate Editor of the international “Journal of New Music Research”.
THE SOUNDING GESTURE: AN OVERVIEW

Amalia de Götzen

ABSTRACT

Sound control by gesture is a peculiar topic in Human-Computer Interaction: many different approaches to it are available, focusing each time on diversified perspectives. Our point of view is an interdisciplinary one: taking into account technical considerations about control theory and sound processing, we try to explore the expressiveness world which is closer to psychology theories. Starting from a state of the art which outlines two main approaches to the problem of “making sound with gestures”, we will delve into psychological theories about expressiveness, describing in particular possible applications dealing with intermodality and mixed reality environments related to the Gestalt Theory. HCI design can indeed benefit from this kind of approach because of the quantitative methods that can be applied to measure expressiveness. Interfaces can be used in order to convey expressiveness, which is a plus of information that can help interacting with the machine: this kind of information can be coded as spatio-temporal schemes, as it is stated in Gestalt theory.

1. INTRODUCTION

Sound control by gesture is a peculiar topic in Human-Computer Interaction: many different approaches to it are available, focusing each time on diversified perspectives. Our point of view is an interdisciplinary one: taking into account technical considerations about control theory and sound processing, we try to explore the expressiveness world which is closer to psychology theories. Starting from a state of the art which outlines two main approaches to the problem of “making sound with gestures”, we will delve into psychological theories about expressiveness, describing in particular possible applications dealing with intermodality and mixed reality environments related to the Gestalt Theory. HCI design can indeed benefit from this kind of approach because of the quantitative methods that can be applied to measure expressiveness. Interfaces can be used in order to convey expressiveness, which is a plus of information that can help interacting with the machine: this kind of information can be coded as spatio-temporal schemes, as it is stated in Gestalt theory.

2. GESTURES IN HUMAN-COMPUTER INTERACTION

The importance of gesture in the restricted community of Human-Computer Interaction which deals with sound is increasing. A significant sign of this increased attention can be found in carried out research in the field: in the last 5 years 2 research projects have been funded by the European Commission: MEGA (Multisensory Expressive Gesture Application) and the Cost287-ConGAS (Gesture CONtrolled Audio Systems) action. There are many reasons for that:

• this is an emerging subject which can benefit from advances in the other field (e.g. human performance, auditory perception, signal processing);
• it is a complex multi-disciplinary field which encompasses a strong scientific and technical side which does indeed include several kinds of communication, intelligent interaction, cognitive aspects, experimental psychology (related to gesture emotion, intention analysis and to aesthetic cognitive processes) augmented with humanistic research related to music. Communication of all these aspects happens through sound; however, the peculiarity of this communication is that messaging is esthetic rather than semantic (i.e. it cannot be coded in a unique and unambiguous way); this trans-disciplinary approach should be considered as a plus in a context like telecommunication where intelligent interaction and cognitive aspects are ever more relevant.

While both projects are dealing with gesture, they are quite different in scope, focusing in particular aspects of gesture interaction. The MEGA project (cf. http://www.megaproject.org), which ended in 2003, was centered on modelling and communication of expressive and emotional content in non-verbal interaction through the use of multi-sensory interfaces in shared interactive mixed reality environments: the attention was directed to the emotional content conveyed by gesture. In particular, the project focused on music performance and full-body movements as first class conveyors of expressive and emotional content. Main research issues were the analysis of expressive gestures (i.e. analysis of the expressive content conveyed through full body movement and musical gestures), the synthesis of expressive gestures (i.e. the communication of expressive content through computer generated expressive gestures, such as music performances, movement of virtual as well as real (robotic) characters, expressive use of visual media), the strategies for mapping the analysis data into multimodal output. MEGA showed a clear connection between gesture and expressive intention using approaches that integrate classical recognition techniques with novel analysis techniques by integrating consolidated scientific experiences along with theories from
music performance and from choreography, the Laban’s theory of effort. [3,4].

Laban and Lawrence use a 4D space to classify human movement based on effort in which the axes are exertion (light - strong), control (fluent - bound), effort (flexible - direct) and duration (sustained - quick); see Figure 1. Exertion is concerned with strength or weight (W); control with space (S), effort with flow (F) and duration with time (T). The eight basic (W,S,T) exertions are slashing, gliding, pressing, flicking, wringing, dabbing, punching and floating.

![Laban 4D space](image)

In this context, the meaning of gesture seems to be a very broad one: generally speaking, gesture is often referred to dancer movements and sometimes to specific body expressions, but gesture can be considered also a structure with definite semantics defined into an abstract space. Thus, a musical phrase can be considered a gesture which can express an emotion using only musical parameters, where the music is the abstract space. This connection between music and body movement is explicit in dance: a specific choreography can be used to better express a given musical work in a ballet and vice-versa; the emotional states carried by the music are the same ones expressed by body movement. Moreover, there are many psychological studies about music and emotion that maintain that music can represent the dynamic properties of emotions like speed, strength and intensity variations. In particular, Gurney asserts that music can express these emotions through the association with affinities between musical characters and body movements which can show them. [5]. Furthermore, Imberty [6] underlines that there are some kinetic tension and release schemes which are typical for both body and emotion; this leads to think that movements and emotional states are a coherent set and gesture is a communication channel.

Starting from the previous definition of gesture many works have been carried out in the following different (though connected together) areas:

- Analysis and synthesis of expressive content in human movement and gesture;
- Analysis and synthesis of expressive content in musical gesture performance.

In both cases the analysis process starts from gesture-derived information (physical movements or audio signals), captured by sensors into a computing system. These physical signals may be in different formats: they may consist of time variant signals such as sampled audio signals, sampled signals from tactile, infra-red sensors, signals from haptic devices, or events such as MIDI-messages or low-level data frames in video.

Several low-level features can be extracted and processed statistically in order to carry out subsequent expression-related analysis.

Considering the audio domain, these low-level features are related to tempo (=number of beats per minute), sound level, spectral shape (which is related to the timbre characteristics of the sound), articulation (eg. legato, staccato), attack velocity (which is related to the onset characteristics which can be fast or slow), pitch, pitch density, degree of accent on structurally important notes, periodicity (related to repetition in the energy of the signal), dynamics (intensity), roughness (or sensory dissonance), tonal tension, and so on.

Expressive cues can be located and extracted measuring these low-level physical parameters (e.g., kinematics data such as position of body joints or tracking of points on the body silhouette); they include global measures (i.e., cues depending on full body movement, such as the contraction index and the quantity of motion); measures depending on the current position of body joints such as the stability index, cues inspired by Rudolf Laban’s Effort Theory such as the directness index and the fluency index, cues inspired by psychological studies such as the durations of pause and motion phases.

The synthesis process begins from the human silhouette extracted from the videocamera or microdance recordings (a simple black and white bitmap). Some of the spatio-temporal cues extracted from movement in the analysis phase are mapped into visual changes/morphing of the silhouette: In other words, movement cues such as “quantity of movement (momentum)” or “direct/flexible” are mapped into visual cues synthesized in the silhouette.

Most of the research done has then been focused on so-called semantic spaces or maps. A semantic map represents categories of semantic features related to emotion and expression on a predefined grid. Typically, a gesture is then a trajectory in this space, and each trajectory can be seen as a point in a trajectory-related (super)space.

Energy-velocity spaces have been successfully used to synthesize musical performance. The space is derived from perceptual experiments [7] and it has been used in synthesis of different and varying expressive intentions in a musical performance thus far. The energy-velocity space is correlated with legato-staccato properties versus tempo. In this space, positions are used to define MIDI parameters as well as audio signal parameters which control the timing and the dynamics of the notes to be played during a performance. The MIDI parameters typically control tempo and key velocity. The audio-parameters control tempo, legato, loudness, brightness, attack time, vibrato and envelope shape.

The main characteristic of this kind of abstract spaces is that they are very related to the concept that interpret the physical space which can be of different nature: it is a sort of multimodal space. The interesting think is that, for instance, I can analyze an image extracting some expressive characteristics that I can use to synthesize sound. We can do a well known close loop where the input is the gesture and the feedback is the audio; see Figure 2.

All these elements show that gesture is a wide speculation on the topic in human-machine interaction research which can be studied in depth in systems that involve art and digital music [8] and, more generally, sound. Furthermore, audio systems (including Digital
sound synthesis and processing) are a wide field in which the control aspect (which is the focus of the Cost287-ConfGas action http://www.cost287-congas.org started in 2003) still needs to be studied and to be linked with scientific investigation upon gesture. In fact, since real-time digital signal processing has become a reality for many digital sound effects (including sophisticated ones), an increased knowledge of gestural devices and their interaction with digital sound effects is now necessary: the control of digital audio systems by gesture goes in that direction.

On the other hand, in traditional musical situations gesture usually produces sound. The relationship between gesture and sound is unique, it is a cause to effect link. In computer music, the possibility of uncoupling gesture from sound is due to the fact that computers can carry out all the aspects of sound production from composition up to interpretation and performance. Real time computing technology and development of human gesture tracking systems may enable gesture to be introduced again into the practice of computer music, but with a completely renewed approach. There is no longer a need to create direct cause-to-effect relationships for sound production, and gesture may be seen as another musical parameter to play with in the context of interactive musical performances.

From another point of view, with the emergence of new types of interfaces and technologies, including wireless devices, touch-weight-pressure sensitive devices, virtual reality interfaces, force-feedback devices, etc., new types of human computer interfaces are being provided to enhance and supplement the keyboard/slider/potentiometer controllers usually considered for digital sound production, getting closer to the wide variety of means and I/O modalities found in traditional music instruments.

The overall goal of such systems is to enhance the naturalness of human-computer interaction through more cognitive and intuitive interfaces in a field like digital sound production that requires an amount of communication precision and detail which is not usual in other Human-Computer Interaction (HCI) systems. This poses several challenges to the design and control of interaction between systems and humans where sound production is involved. Summing up, the control of sounds using gestural devices goes in two directions: on one hand it recovers a several centuries-long tradition tied to instrumental playing which proved to be extremely rich in nuance and detail while on the other it improves a new way to use machines to interact with sounds: it is more direct and natural, and it opens new and unexplored possibilities.

A pioneering work on gestural input devices was carried out during the analog electronic music period. The invention of the Theremin [9] anticipated the development of new expressive instruments. A noticeable aspect is that this first electronic musical instrument, a simple oscillator performed without physical contact, could produce very subtle and varied sounds because the generated sound reflected more the expressive quality of human movement than its own timbre quality.

Further recent works, using new technologies, have been done in this field [10,11,12,13,14] and a lot of new instruments with new control metaphors have been developed.

While the control aspect of the interaction is deemed interesting, the research interests are not limited to digital musical instrument control which can be considered just the first step of a deeper investigation. A few aspects have to be further investigated to find a way to use evaluation methods well known in HCI (e.g. Fitts’ law[15]) in a multimodal context, dealing with motion, sound and expressivity (e.g. the Gestalt approach).

3. EMOTION FROM MOTION: EXPRESSIVE GESTURES

The previous paragraph faced the problem of defining what gesture is and how it can convey expressiveness. The ‘MEGA approach’ is quite different from the ‘Gestalt one’ that we are about to consider now.

The gesture movement expressivity in fact has been deeply investigated by Gestalt theorists [16] and is concerned with the study of functional relations such as the perception of causality. Gestalt psychology stated that the phenomenical world is made by a lot of elements that interact with each other by means of a series of functional relationships. Many of these relationships form our common experience: for example, if we hit an object maybe something else will move, some shapes are difficult to grasp, pouring the wine into a glass, etc.. Situations like those mentioned above do not involve just spatial and kinetics aspects but also functional relationships: in the last example for instance we don’t see just the wine that changes position, but we actually see the wine coming out from the bottle neck and going down into the glass! Thus, functional relationships are the weaving of the phenomenical world, which is the world which can be described completely and naturally by means of direct experience. Through the Gestalt approach functional relationships give sense to everything which is around us: we know what things are by observing what they do. Psychology is then the science of actions and behavior and it has become essential to study perception like a phase of the action. In this context it may occur that an object influences a behavior just because of its meaning which is an element of perception itself. Such perception depends on the relationships which the object have with other objects: temporal and spatial relationships are experienced as functional relationships, be it causality or finalization.

3.1. Causality perception

Interesting studies have been carried out by Michotte [17] on the perception of causal relationships from which observations and researches about movement expressiveness are derived. These observations involves tertiary qualities of the objects which are often defined as physiognomic, that is “informing on the nature” of the object or event. We can in fact distinguish between:

- primary qualities: dimensions, shapes, weight etc.;
- secondary qualities: color, taste, affective valence;
Michotte found out that precise timing is needed to achieve perceived causality. The best-known experiment and result is about the launching effect: Michotte studied the perception of causal relationships between two light spots that move always along the same line with a variety of velocity patterns. See Figure 3.

![Figure 3: Michotte's launching effect (from Vicario 2004)](image)

Under particular experimental conditions Michotte found out that the A movement appear as active, while the B movement appear as passive. Both movements are simple, identical translations into space; however, under specific timing relationships between them they acquire an expressive quality which leads to perceived expressive individualities (or tertiary qualities) of both elements. This kind of characteristics are qualities of the perceptive structure and, because of that structure, objects and events are expressive. A very interesting propriety of that structures is their intermodality: a structure can maintain the same qualities in every sensorial modality: we perceive the same expressive quality with different senses (this is why, for example, we say that a voice (hearing) is sour (taste)). It is the principle for metaphor creation. An appropriate example of this property is shown in Figure 4.

![Figure 4: The intermodality of expressive qualities: Who is Takete and who is Maluma? (from Köhler 1984)](image)

In this example the absolute majority of people say that Maluma is the round figure while Takete is the other one. This is a fine example since both images and words make no sense to anyone, but because of their similar structure they obtain a coherent result. The angular figure has rapid changes like the sound of Takete, while the round figure has a continuous line without abrupt discontinuities, like the sound of Maluma. This kind of studies is very interesting for Human-Computer Interaction which deals with multimodality and, because of that structure, objects and events are expressive. A very interesting propriety of that structures is their intermodality: a structure can maintain the same qualities in every sensorial modality: we perceive the same expressive quality with different senses (this is why, for example, we say that a voice (hearing) is sour (taste)). It is the principle for metaphor creation. An appropriate example of this property is shown in Figure 4.

---

• tertiary qualities: good/bad, gloomy/happy, threatening/attractive.

---

3.2. Applications

Based on the Gestalt approach some application has been developed in information visualization and in animation [21, 22, 23, 24]. In fact motion has revealed itself as a perceptually rich and efficient display dimension: it may prove useful in visualizing complex information because of its preattentive and interpretative perceptual properties, reducing some of the over-use of the representation techniques. As Michotte discovered, the key factor to causal perception is movement amplification, where the movement of the motor object (A) extends into that of the projectile (B). To verify this Michotte considered the perceived effect of A’s movement on a qualitative change (appearance, disappearance, change in form or color) of object B. When there was no opportunity for movement amplification, no causality was remarked. Thus, merely causing objects to appear and disappear in temporal and spatial contiguity is insufficient for the impression of causality (as in flashing them on and off the screen in close coincidence); some form of kinematic integration must occur. It is quite natural to apply these results to new visual interface designs, which perhaps seem to be the counterpart of the audio-ecological approach. Using again Michotte results (such as the importance of movement speed in perception) Amaya et al. used signal processing techniques to analyze emotion in motion [25]. They captured movements from subjects performing two types of human activity, drinking from a cup and knocking on a door, in three emotional contexts (angry, sad and neutral). They identified two attributes which varied considerably over the different emotional movements: speed (frequency) and spatial amplitude (the range, or size of the motion). They divided the movements into basic periods (e.g. hand to cup, cup to mouth, cup down, hand back); determined the speed of the end effector along its trajectory in both the angry and neutral movements, and calculated speed transforms for both the neutral and angry movement by integrating the longest period along the trajectory and dividing it into frame templates. Such transforms can then be applied to a new, neutral movement by time-warping the frame distribution, interpolating between frames where needed. The spatial amplitude intensity for each joint was calculated for each movement period and nonlinear signal amplification is used to apply the amplitude transform to generate new joint positions from the new, neutral movement. They tested the approach by deriving...
angry and sad transforms from the cup-drinking data. Then they applied the transforms to the neutral knocking data, and found a close match between the generated and the real (motion-captured) angry knocking motion data.

Similar works have been done with gesture and sound [26] from analogies between music performance and body movement. The aim was to identify relevant gestures in sound control in order to develop sound models that respond to physical gestures: the authors found out that subjects correctly classified different types of motion produced by the model. Substituting then sampled sounds with physic-modeled sounds a model of expressive walking and running has been built [27]. This kind of study can be very useful, for instance, in VR applications and, in this specific case, in order to characterize footsteps of avatars controlled in accordance to gestures dynamics to produce natural and expressive sound variations and to investigate how sound feedback can affect vision.

The cross modal effect has been deeply investigated: auditory and visual spatial information, originating in the same event, usually results in one unified percept and the interaction represent a prerequisite for phenomenal causality [28]. Several cases of visual influence on auditory perception are reported, such as the McGurk effect [29], while fewer ones report the opposite influence [30, 31] (e.g. the disambiguation provided by audio stimuli when a visual ambiguity is introduced). Recent work has been carried out by Gusky and Troje on aud iovisual phenomenal causality [32]; following the Michotte method, the authors found out that the causality perception increases when additional auditory or visual information marks the onset of target motion. Similar studies have been carried out using physical modelling with continuous auditory feedback [33, 34]: continuous sound feedback can emphasize causality.

The main purpose of these works is to introduce interactive cartoon models of everyday sound scenarios with a new control approach through audio-visual-tangible systems. The idea is that human interaction in the world is essentially continuous, while the majority of sounds that right now are used in computer environments are totally unnatural (e.g. triggered samples). A few practical examples of that interaction have been elaborated by the SOb project (http://www.soundobject.org), developing rolling sound models based on physical models of impact [35].

- the invisiball: a thimble acts as the sender and the receiving antenna is placed under a 3D elastic surface. Finger position in the 3D space is detected in real-time and it is used by the algorithm controlling the rolling movement of a ball;
- the ballancer: the user has to balance a ball tilting a rectilinear track, and the modeled sound of the ball rolling over the surface of the track along with a tactile-visual response provides the feedback.

These examples of interaction have demonstrated that everyday sounds can be very useful because of their familiar control metaphor: no explanation nor learning are necessary [36]. Moreover, it is clear that continuous audio feedback affects the quality of the interaction and that the user makes continuous use of information driven by sounds to adopt a more precise behavior.

4. CONCLUSIONS

After novel expressivity paradigms and gesture analysis methods have been developed in an effort to map human gestures into quantitative scales (e.g. Fitts’ law, steering law, etc.), it has become crucial to deal with new, often specific sets of control parameters.

Furthermore, the role of multi-modality and multi-sensory communication will be central in the design of the next generation interfaces. As a consequence, non-speech communication will play an important role inside the information stream established between machines and users [37, 38].

Acoustic events play an important role in our general perception of the environment and can have a strong impact on our affects. This is especially true when acoustic cues are used to enhance or to complement the visual modality:

- audio feedback can be very informative when visual feedback is missing;
- sound conveys information about the environment (e.g. the material of a stroked surface, the brand of a car, the genre of a person walking or the voice of a person).

The study of the reaction of sounds to gestures will lead to more sophisticated ways to produce sounds. The typical structure of musical/instrumental gesture (division between control and sound production, that is between audio feedback, haptic feedback and visual cues, and sound production) has to be carefully studied and extended to lead to better (i.e. more natural) uses of sound effects.

Recent studies have shown that signal processing techniques can be fruitfully used to control the expressive content of a sound object through gesture [39]. A challenge task of the research effort can be the desire to place the emotion expression in the foreground, providing a model with a method enabling the interaction between users and machines through sound and musical cues.

5. REFERENCES


EFFECT OF LATENCY ON PLAYING ACCURACY OF TWO GESTURE CONTROLLED CONTINUOUS SOUND INSTRUMENTS WITHOUT TACTILE FEEDBACK

Teemu Mäki-Patola and Perttu Hämäläinen

Telecommunications Software and Multimedia Laboratory
Helsinki University of Technology, Finland
tmakipat@tml.hut.fi pjhamala@tml.hut.fi

ABSTRACT

The paper reports results from an experimental study quantifying how latency affects the playing accuracy of two continuous sound instruments. 11 subjects played a conventional Theremin and a virtual reality Theremin. Both instruments provided the user only audio feedback. The subjects performed two tasks under different instrument latencies. They attempted to match the pitch of the instrument to a sample pitch and they played along a short sample melody and a metronome. Both the sample sound and the instrument’s sound were recorded on different channels of a sound file. Later the pitch of the sounds was extracted and user performance analyzed. The results show that the time required to match a given pitch degrades about five times the introduced latency suggesting that the feedback latency cumulates over the whole task. Errors while playing along a sample melody increased 80% by average on the highest latency of 240ms. Latencies until 120ms increased the errors only slightly.

1. INTRODUCTION

Currently, physical sound modeling is an active research area. Real-time sound production makes it possible to alter any parameter of the sound model while playing. This creates a need for controllers whose input flexibility matches the complexity of the sound model. Virtual reality input technology, such as data gloves and location/orientation trackers with gesture analyses, is one option that offers several degrees of freedom. We are currently experimenting with virtual reality interfaces for the control of physical sound models in an EU funded project called ALMA [1].

Physical sound models are often computationally heavy, which introduces some latency. A virtual reality system also always introduces latency. Latency is a key issue also in networked co-operative playing. Thus, it is of importance to know how much latency can be allowed for different control paradigms.

An article by Paradiso [2] and a book edited by Wanderley and Battier [3] offer a good overview of existing electronic interfaces and controllers. Many have been created, especially during the last few decades. However, only few virtual reality interfaces for sound control exist [4, 5]. They have been interactive sound environments or interactive filters rather than standalone instruments. The interfaces and alternative controllers have been reported mostly as case studies.

There seems to be a lack of quantitative comparisons of the suitability of different interfaces for controlling sound. The importance of parameter mapping has only lately been considered [6, 7]. A comparison of three input devices for timbre space navigation exists [8]. The preliminary parameter mapping observations offer some suggestions to what direction to move on the issue. It would be beneficial to have similar guidelines based on properties of available input technology and its suitability for different kinds of sound control.

Earlier research suggests that tactile feedback improves playing accuracy of an instrument [9]. However, tactile feedback is currently difficult to elegantly integrate into virtual reality interfaces. Thus, if we want to use virtual reality for controlling sound it is of interest to know latency tolerance for cases where the subject does not obtain tactile feedback while playing an instrument.

Several studies have shown that latency degrades user performance in virtual reality [10, 11]. The degradation is gradual and depends of the task. The mentioned studies concentrated on reaching and target acquisition tasks. Feedback was visual and minimum latencies as high as the maximum latency in our test. Similar results by Watson et al. [12] show that latency slows down and reduces placement accuracy when the task requires feedback. They also studied the effect of variations in latency [13] concluding that only variations with standard deviation above 82ms affect performance in a grasping and placement task.

A classical experiment conducted by Michotte and reported by Card, Moran and Newell [14] shows that users perceive two events as connected by immediate causality if the delay between the events is less than 50ms. Dahl and Bresin [15] suggest that over 55ms of latency degrades use of a percussion instrument without tactile feedback while playing along with a metronome. Again the degradation was gradual. Only four professional musicians were tested with a baton instrument. The latency was increased in small steps while playing. Two subjects were tested also with tactile feedback (MIDI drum), concluding that the standard deviation of the flutter of consequent hits increased with increasing delay. However, again the change is slow. The standard deviation is no larger at 50ms than at zero. After 50ms it seems to rise gradually. The amount of subjects and samples in the test was too small for strong conclusions. The study verified also a hypothesis that the performer attempts to compensate the delay by matching sound to sound when he has to synchronize with other audio sources. Finney has shown that delay in auditory response caused large errors in performance of pianists [16]. There was no degradation if the performer did not receive auditory feedback. Discrepancy with sound and tactile feedback seems to be the main source of instrumental problems.

Less than 10ms latencies are often suggested for a music controller [17, 18] as professional piano players might already notice that much. However, tolerable latency is dependent on type of music, type of instrument sound [19] and presence or absence of
tactile feedback. We have earlier found that the just noticeable difference (JND) for latency for a continuous sound gesture controlled instrument without tactile feedback is about 30ms [20]. For an extreme perspective let us remind that latencies as high as several hundred milliseconds are not rare for church organs and yet they can be played when also practiced with the same latency.

2. USER TEST

The goal of our user test was to quantify how latency degrades the control accuracy of the tested gesture instruments. The test consisted of two parts on two similar instruments. In the first part the subjects heard a permutation of 16 consecutive sine wave notes on their left earphone. On their right earphone they heard the Theremin or virtual reality Theremin instrument they controlled. Each sample note played for five seconds during which the subject attempted to produce the same tone with the instrument. On the second part the subjects were played a short song (see Figure 1) seven times accompanied with a metronome. The subjects tried to play along the song as well as they could.

Each subject did both tests on five different latencies on the original Theremin and on three different latencies on the virtual reality Theremin. The latencies tested were 0ms (no latency), 30ms, 60ms, 120ms and 240ms on the original Theremin and 60ms, 120ms and 240ms on the virtual reality Theremin. The responsiveness of our virtual reality system was measured to be 60ms with a standard deviation of 8ms. Thus, only the last three latencies could be tested on this instrument.

Figure 1: Simple example melody used in the play-along test. The melody was played in a tempo of 120BPM. It is progressive, simple and uses all note lengths common for Theremin music.

Each individual pitch-matching test lasted for 80 seconds. The play-along tests were 56 seconds long each. Thus, the original Theremin part of the test took 11 minutes 20 seconds plus the practice and one minute breaks between the tests. The virtual reality Theremin part took 6 minutes and 48 seconds plus the practice and breaks. The whole test took about 40 minutes per subject.

The pitch matching permutation was randomised before the tests and was the same in each test for all subjects. The results were thus easily comparable to each other. None of subjects noticed that the notes came in the same order when asked after the test.

2.1. Subjects

The test subjects were 11 students and researchers from the Helsinki University of Technology. All of the subjects had at least six years of musical instrument practice; seven had more than 10 years of practice with several instruments. Four of the subjects currently practice more than five hours per week, five subjects for 1 to 3 hours and the remaining two less than an hour or not at all. Six of the subjects were 23 to 28 years of age, the rest were 30 to 50 years of age. Only one subject was female. One subject was left-handed. Nine of the subjects had previously participated in a test using the Theremin instrument.

2.2. Test equipment

The Theremin’s output was routed to a Boss GX-700 Guitar Effects processor. Using the effects processor the instrument’s sound could be delayed for a specified amount of milliseconds. The effects processor was preprogrammed with patches that had only a delay effect active and no direct sound. We tested the effects processor itself to produce less than 1ms of delay when the delay was deactivated. The output of the effects processor was routed to the right earphone of the test subject and to the laptop used for recording the session data (see Figure 2). Another computer was used for generating the sample pitches and melodies.

Figure 2: Test setting for Theremin. The subject (1) hears sample sounds from his left earphone and the Theremin instrument’s (2) sound from his right earphone. The instrument’s sound is delayed using a guitar effects processor (3). Both sounds are inputted also to a laptop computer (4) that records both sounds on different channels of a sound file for later pitch extraction. Another computer is used for producing the sample sounds (5) and also to produce a metronome sound on a speaker (not visible in figure).

The virtual reality version of the Theremin used our laboratory’s Cave-like virtual reality room, EVE. EVE uses an Ascension Technologies MotionStar magnetic tracker to track the location and orientation of the subject’s data gloves. The tracker reads the location and orientation of the sensors at a rate of 100Hz. We wanted the subject to receive only audio feedback. Thus, EVE’s stereo goggles were not used in the test and the visualization of the virtual reality Theremin was switched off. The EVE system uses SDT data gloves to measure the flexure of each finger.

The test setting for the virtual reality Theremin was similar to the test setting on the original Theremin. However, the delay was implemented on our audio program Mustajuuri [22] and not on the effects processor. Mustajuuri is a plugin based DSP program and serves as the sound control system of the EVE. It can be controlled over a local net from an SGI Onyx computer running the virtual reality application.

The sample sound and the instrument sound were outputted on different channels and routed similarly than in the test using the original Theremin. The Mustajuuri application was used in the...
other computer for procuing all sample sounds in both Theremin and virtual reality Theremin tests. The virtual reality Theremin’s sound was produced in the same instance of MaxJauiri.

In the virtual reality Theremin the height of the subject’s right hand controlled pitch. Closing the hand controlled volume. The pitch scale represented a vertical keyboard but was continuous.

The original Theremin’s sound is controlled through interaction with its two antennas. The distance from the player’s left hand from a horizontal loop shaped antenna controls the volume, the closer the hand to the volume antenna the smaller the amplitude of the sound. The distance of the user’s hand from a vertical antenna defines the pitch of the sound, the closer the hand the higher the frequency. The user is part of the instrument’s sensitive capacitance circuit.

2.3. Test procedure

The order of the test instruments was randomly chosen for each subject. On both instruments, the subject did first all pitch-matching tests and after that all play-along tests. The order of the latencies was randomized. Latencies were changed only between individual tests.

2.3.1. Original Theremin test

In the beginning of the half of the user test that used the original Theremin, the subject was first introduced to the instrument by letting him play it for couple minutes. Then the test procedure was explained and one 80 seconds long pitch matching was practiced without any latency in the instrument’s response. If the subject wanted to practice one more time it was allowed. Some subjects did. During the test the subject matched the 16 sound samples on all five latencies keeping a short break after each individual test.

After the pitch-matching test the sample melody was introduced to the subject and he was told to try to play along the melody as accurately as he could. Playing along the melody was practiced without latency for one or two test periods depending on when the subject felt ready for the test. After this the subject attempted to play along the melody on all five latencies. Each play-along test repeated the melody seven times on every latency setting. A one-minute break was kept after each individual test with different latency setting.

2.3.2. Virtual reality Theremin test

The virtual reality Theremin test was similar to the test using the original Theremin. Again the pitch-matching test was first. The user was allowed to play the instrument for few minutes and then practice one or two 80 second long pitch-matching tests with the minimum system latency of 60ms. After this the pitch-matching test began as with the original Theremin followed by the melody test. However, only the mentioned three latencies were tested.

3. RESULTS

The results indicate that the time taken to match a given pitch with the instrument lengthens about five times the introduced latency. When latency increases 60ms the matching time rises about 300ms. This suggests that the subject uses feedback several times/continuously during the task. First he moves quickly up or down depending on the relative location of the new target pitch. While he moves he uses the sound as feedback of making new estimates, just like an optimization algorithm or a mathematical control system. Latency cumulates over the whole task. The relative increase in matching times from 60ms to 240ms was 45%.

3.1. Valid subjects

Three of the eleven subjects were removed from the results in the final data analyses because they could not hear the differences in the pitches well. These subjects matched the pitches with an error of several halftones. The error was not a consistent amount of notes up or down but altered even during one test recording. The removed subjects were the ones with the least musical practice. Two of the three had not played any instrument for several years.

3.2. Extracting the pitch

We used Matlab for data analyses. The pitch was extracted from the recorded sound files by calculating piecewise Fourier transform for every consecutive 128 samples. The sample rate of the recording was 8000Hz on a 16bit resolution. Every piece was first zero padded to 256 samples resulting in double over sampling. Then the maximum spike (see Figure 3) was searched from each transformed slice with parabolic interpolation and the location of the maximum scaled to a MIDI key number scale.

The pitch was extracted from all of the recorded sound files. Figure 4 shows the extracted pitch of one such file. Each subject’s data produced eight similar pitch-matching curves and eight play-along melody curves.
3.3. Effect of latency in the pitch matching task

For the pitch-matching task we wanted to know how much longer it takes to reach the goal pitch when latency is introduced to the instrument’s response. We made a Matlab program that searched each sample pitch change from every pitch test file. It then isolated the next five seconds that the sample pitch remained unchanged while the subject was trying to match the instrument’s sound with it. We wanted to determine how long each matching took before the goal pitch was reached. Then the 14 matching times for each individual test were averaged to get the subject’s average pitch-matching times for each latency setting.

As there were 128 matching periods for each subject we needed some automatic measure for deciding when the subject had reached the goal pitch. We made an algorithm that evaluated each matching period to determine the time when the subject had reached the goal pitch. The algorithm moved a 500ms window forward in the five seconds long matching data. Two error measures were defined for the window (equations (1) and (2)).

\[
E_i(t) = \frac{1}{N} \sum_{n=1}^{N} \text{abs}(y(t-1+n)-x(t-1+n))
\]

(1)

In equation (1) the error E(t) is defined for a time t. y(t) is the sample pitch and x(t) is the instrument’s pitch at time t. The pitches have been scaled to a halftone MIDI key scale. The first error measure defines the average deviation from the target pitch inside the window of length N in semitones.

\[
E_s(t) = \text{Max}(\text{abs}(y(t+n)-x(t+n))), n \in [0, N-1]
\]

(2)

The second error measure defines the maximum deviation from the target pitch inside the window. We defined that the subject had reached the target pitch when the relationship of equation (3) was satisfied.

\[
E_s(t) < 0.5 \land E_i(t) < 0.5
\]

(3)

Thus, when the average error and the maximum error inside the window, forward from the point t, are less than a quartertone the pitch has been matched. The first t that satisfies equation (3) is marked as the time of reaching the target pitch. If no t satisfies the equation the matching time is marked as the full five seconds.

The average matching time was calculated for every latency setting for each individual. The individual results where then averaged over the test population. The matching times were calculated also with a 800ms window. The two windows gave quite similar results. The final results are the average of the results from analyses on both window lengths. They are presented in Table 1. Figure 5 shows an example of six five second long pitch-matching tasks extracted from the test data.

<table>
<thead>
<tr>
<th>Latency</th>
<th>0ms</th>
<th>30ms</th>
<th>60ms</th>
<th>120ms</th>
<th>240ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Theremin</td>
<td>2040</td>
<td>2440</td>
<td>2260</td>
<td>2610</td>
<td>3270</td>
</tr>
<tr>
<td>Normalized</td>
<td>0.90</td>
<td>1.08</td>
<td>1.00</td>
<td>1.16</td>
<td>1.45</td>
</tr>
<tr>
<td>Virtual Theremin</td>
<td>-</td>
<td>-</td>
<td>1930</td>
<td>2260</td>
<td>2780</td>
</tr>
<tr>
<td>Normalized</td>
<td>-</td>
<td>-</td>
<td>1.00</td>
<td>1.17</td>
<td>1.44</td>
</tr>
</tbody>
</table>

Table 1: Average pitch-matching times in milliseconds as a function of instrument latency. The normalized values are expressed as multiples of the matching time on the 60ms latency. The relative increase in matching times is very similar on both instruments.

As can be seen from Table 1 the results from the last three latencies from both instruments are highly consistent. The matching times grow 350ms and 330ms when the instrument latency rises from 60ms to 120ms. The matching times grow 660ms and 520ms when another 120ms of latency is introduced. Considering all changes after the 60ms latency the matching times grow 5.8, 5.5, 5.5, 4.3, 5.6 and 4.7 times the added latency. After 60ms, the matching time grows 5.2 times the amount of added latency by average. The relative change in the matching time is 45% from 60ms latency to 240ms latency.

The matching time with 30ms latency is larger than the matching time with 60ms of latency. We state the possible reasons for this in Section 4.

![Figure 5: Three individual pitch-matching curves of the same subject on the original Theremin. Upper row with no latency and the lower row with 240ms latency. It is clearly seen how larger latency causes more fluctuation around the target pitch (dashed line). The small ball on the curve marks the place where our algorithm decided that the target pitch had been reached.](image)

Table 2 presents an estimate of how much the matching curve fluctuates as a function of latency. The values were calculated by integrating the absolute difference of the sample pitch and the instrument’s pitch and dividing it with the length of the integrated area.

<table>
<thead>
<tr>
<th>Latency</th>
<th>0ms</th>
<th>30ms</th>
<th>60ms</th>
<th>120ms</th>
<th>240ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Theremin</td>
<td>1.26</td>
<td>1.31</td>
<td>1.31</td>
<td>1.52</td>
<td>1.93</td>
</tr>
<tr>
<td>Normalized</td>
<td>0.96</td>
<td>1.00</td>
<td>1.00</td>
<td>1.16</td>
<td>1.47</td>
</tr>
<tr>
<td>Virtual Theremin</td>
<td>-</td>
<td>-</td>
<td>1.01</td>
<td>1.17</td>
<td>1.29</td>
</tr>
<tr>
<td>Normalized</td>
<td>-</td>
<td>-</td>
<td>1.00</td>
<td>1.16</td>
<td>1.28</td>
</tr>
</tbody>
</table>

Table 2: The average difference in half tones of the subjects’ playing and the sample signal on the pitch-matching task. The normalized values are expressed as multiples of the average difference on 60ms latency.

The matching times of changing the pitch downwards were similar to the matching times of changing the pitch upwards. They differed less than 80ms by average. However, with the maximum latency of 240ms on the normal Theremin the matching times upwards were by average 700ms slower than downwards.

Differences of six of more half tones took 900ms more time to match by average than differences of less than 3 half tones. Interestingly this did not change as a function of latency but was nearly the same under all latencies. Matching times of all pitch differences shifted almost equally as a function of latency. Maximum differences in the data were 11 notes up and 15 notes down.
3.4. Effect of latency while playing along background music

Figure 6 shows an example performance of a subject playing along the sample melody on two different latencies. Table 3 shows the average pitch error during the playing tests in the whole population. As can be seen the differences between latencies of zero to 120ms are small. We assume that they fit inside the noise of the data. Only the 240ms latency brings forth clear performance degradation.

![Graph showing pitch error against latency]

Figure 6: Example patterns from one subject’s play-along tests. The solid line is the pitch of the Theremin instrument, the dashed red line is the pitch of the sample melody. The above graph is with zero latency and the lower graph is the same subject with 240ms latency. The first 16 seconds were cut away from all play-along data to give the subject time to start the test.

Interviewing the subjects it was found out that they relied mostly on kinaesthetic memory while playing along the music. They quickly learned the tune and the approximate hand locations for its notes and took little advantage of the audio feedback. They compensated the latency well on the range from zero to 120ms. The playing turned clearly more difficult only on the latency of 240ms. This largest latency seemed to be too much to compensate and the small refinements based on the audio feedback started to fluctuate.

Table 3: The average difference of the instrument and the sample signal in half tones on the play-along task. The normalized values are the errors divided by the error at the 60ms latency. The performance degrades 93% on the normal Theremin and 64% on the virtual reality Theremin as the latency rises to 240ms.

<table>
<thead>
<tr>
<th>Latency</th>
<th>0ms</th>
<th>30ms</th>
<th>60ms</th>
<th>120ms</th>
<th>240ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Theremin play-along</td>
<td>0.69</td>
<td>0.69</td>
<td>0.58</td>
<td>0.65</td>
<td>1.12</td>
</tr>
<tr>
<td>Normalized</td>
<td>1.19</td>
<td>1.19</td>
<td>1.00</td>
<td>1.13</td>
<td>1.93</td>
</tr>
<tr>
<td>Virtual Theremin play-along</td>
<td>-</td>
<td>-</td>
<td>0.80</td>
<td>0.92</td>
<td>1.32</td>
</tr>
<tr>
<td>Normalized</td>
<td>-</td>
<td>-</td>
<td>1.00</td>
<td>1.15</td>
<td>1.64</td>
</tr>
</tbody>
</table>

In the pitch-matching task the virtual reality Theremin seems to be more accurate than the original Theremin. However, the play-along test results are more erroneous on the virtual reality Theremin. This might be because the virtual reality theremin requires longer movements between the melody pitches, as the scale is less dense.

As mentioned, the matching times upwards were 700ms slower than downwards with 240ms latency on the original Theremin. This did not happen on the virtual reality Theremin. We suspect that the reason is the nonlinearity of the original Theremin’s scale in the high pitch end. A one-centimeter difference in the hand location becomes several tones when close to the pitch antenna. With such exact movements the iteration time is bound to rise when the feedback latency is high. The virtual reality Theremin had a tone-wise linear scale. The same amount of height difference always resulted in the same amount of change in half tones.

Table 1 and Table 2 show interestingly similar behavior for the increase in pitch matching times and average error. It should be noted that these are rather different things. Although that the matching time fluctuates under the 60ms latencies the average error is very similar on those latencies.

There is quite much noise in the data of individual subjects. Some of it is due to learning. Especially in the play-along test the subjects got better and better kinaesthetic memory of the hand poses for the melody. The order of the latencies was randomised but as we had only eight valid subjects it still leaves some noise to the data from learning and fluctuations in the subject’s concentration. The matching time with 30ms latency was larger than with 60ms latency. This is likely due to chance. Note that the error measure (Table 2) was the same for both cases.

The function for evaluating the matching times affects the answers a bit. With stricter error limits the lengthening of the matching times is a bit more severe. However, the relative differences of the answer distribution remain similar.

Dahl and Bresin’s study [15] with percussion instruments suggested that in the presence of background rhythm the subject attempts to match sound with sound. In the play-along part of our test the same thing was evident. The subjects compensated the latency quite well on all latencies except for the largest latency.

In the virtual reality Theremin the latency was not constant but had a standard deviation of 8ms. This variance was so small that its possible effects are shadowed by other sources of errors such as the noise in the data.

5. CONCLUSIONS AND FUTURE WORK

As can be seen from Table 1 the pitch matching times are smaller on the virtual reality Theremin compared to the original Theremin. Some of this is probably due to the slightly different user interface (height of the hand instead of the distance from a pole). Another reason is that the example sound is exactly similar to the instrument sound on the virtual reality Theremin. The original Theremin used in the test does not produce exact sine wave sound but rather heavily smoothed saw wave. The different color of the sound makes it a bit more difficult to match the pitches.

In the pitch-matching task the virtual reality Theremin seems to be more accurate than the original Theremin. However, the play-along test results are more erroneous on the virtual reality Theremin. This might be because the virtual reality theremin requires longer movements between the melody pitches, as the scale is less dense.
introduced latency. This suggests that the feedback latency cumulates over the whole task. The matching time differences on latencies below 60ms were within the noise threshold. Errors in following the sample pitch increased 40% on the maximum latency of 240ms as the subject’s playing started to fluctuate around the goal value. Starting the matching process over six half tone distances away from the goal pitch required 900ms (47%) more time compared to less than 3 half tone distances. Interestingly this time was not significantly increased by latency.

Errors while playing along a sample melody increased 80% by average on the highest latency of 240ms. Latencies until 120ms increased the errors by less than 20%. The subjects still managed to compensate the 120ms latency but not the maximum latency of 240ms. Interestingly the subjects produced least errors on 60ms latency while playing along a sample melody. In our preliminary studies there were also other anomalies around 60ms latency. It could be that this time constant has some special characteristic in human physiology, but from our part this is still a matter of further research.

As our test was to see the effect of latency on playing accuracy we did not examine our results from the perspective of Fitt’s law [23] or Meyer’s law [24]. However, it might be interesting to fit the pitch matching results to Meyer’s law, as it is a model for target reaching movement that consists of several sub-movements. We could then create a model for each latency and maybe come up with a latency dependent function for the model parameters. Our intention is to analyse the data further by trying to fit a second order control system model to the human pitch-matching behavior.

6. ACKNOWLEDGEMENTS

This research was supported by Pythagoras Graduate School funded by the Finnish Ministry of Education and the Academy of Finland and by an EU IST program (IST-2001-33059). The authors would also like to thank professor Tapio Takala for ideas, help and guidance.

7. REFERENCES

AUDIO-BASED GESTURE EXTRACTION ON THE ESITAR CONTROLLER

Ajay Kapur, George Tzanetakis, Peter F. Driessen

University of Victoria
Victoria, British Columbia, CANADA
ajay@ece.uvic.ca, gtzan@cs.uvic.ca, peter@ece.uvic.ca

ABSTRACT

Using sensors to extract gestural information for control parameters of digital audio effects is common practice. There has also been research using machine learning techniques to classify specific gestures based on audio feature analysis. In this paper, we will describe our experiments in training a computer to map the appropriate audio-based features to look like sensor data, in order to potentially eliminate the need for sensors. Specifically, we will show our experiments using the ESitar, a digitally enhanced sensor based controller modeled after the traditional North Indian sitar. We utilize multivariate linear regression to map continuous audio features to continuous gestural data.

1. INTRODUCTION

Using hyper instruments that utilize sensors to extract gestural information for control parameters of digital audio effects is common practice. However, there are many pitfalls in creating sensor-based controller systems. Purchasing microcontrollers and certain sensors can be expensive. The massive tangle of wires interconnecting one unit to the next can get failure-prone. Things that can go wrong include: simple analog circuitry break down, or sensors wearing out right before a performance forcing musicians to carry a soldering iron along with their tuning fork. The biggest problem with hyper-instruments, is that there usually is only one version, and the builder is the only one that can benefit from the data acquired. These problems have motivated our team to attempt to use sensor data so that sensors become obsolete. More specifically we use sensor data to train machine learning models, evaluate their performances and then use the trained acoustic-based models to replace the sensor.

The traditional use of machine learning in audio analysis has been in classification where the output of the system an ordinal value (for example the instrument name). In this work, we explore regression which refers to machine learning systems where the output is a continuous variable. One of the challenges in regression is obtaining data for training. To solve this problem, we use the sensor data to train the model that will replace the sensor. In our experiments, we use audio-based feature extraction with synchronized continuous sensor data to train a “pseudo” sensor using machine learning.

Specifically, we show our experiments using the Electronic Sitar (ESitar), a digitally enhanced sensor based controller modeled after the traditional North Indian sitar.

In this paper the following will be discussed:

• Background on related work using machine learning techniques for extraction of gesture information
• Description of sensor-based gesture extraction using the ESitar controller
• Audio-based feature extraction and multivariate feature extraction
• Experiments on audio-based gesture extraction on the ESitar controller

2. RELATED WORK

There has been a variety of research using machine learning techniques to classify specific gestures based on audio feature analysis. The extraction of control features from the timbre space of the clarinet is explored in [1]. Deriving gesture data from acoustic analysis of a guitar performance is explored in [2, 3, 4]. An important influence for our research is the concept of indirect acquisition of instrumental gesture described in [4]. Gesture extraction from drums is explored in [5, 6]. All the cited papers fall into two broad categories: (1) methods that rely on signal processing to directly map the sound to gesture parameter and typically work for continuous data, and (2) methods that use machine learning to extract categorical information. The former tend to be sensitive to noise and other uncertainties in the sensor data, and therefore not suitable for use in actual performance. The later although robust to noise don’t handle continuous gestural data. In addition, the mapping from sound to gesture in many cases is not straight-forward and machine learning is necessary to obtain a useful mapping. Using regression, our approach deals with continuous gestural data while retaining the robustness achieved by machine learning.

3. DIGITIZING SITAR GESTURES

In this Section we will briefly describe background of traditional sitar technique followed by a description of a sensor based controller which digitizes a sitar players gestures. This brief description is included to inform the presentation of the audio-based gestural analysis described later. More background and details about these topics can be found in [7].

3.1. Sitar and playing technique

The sitar is Saraswati’s (the Hindu Goddess of Music) 19-stringed, pumpkin shelled, traditional North Indian instrument. Its bulbous gourd (shown in Figure 1), cut flat on the top, is joined to a long necked hollowed concave stem that stretches three feet long and three inches wide. The sitar contains seven strings on the upper bridge, and twelve sympathetic strings below all tuned by tuning pegs. The upper strings include rhythm and drone strings, known as chikari. Melodies, which are primarily performed on the
upper-most string and occasionally the second string, induce sympathetic resonances in the twelve strings below. The sitar can have up to 22 moveable frets, tuned to the notes of a Raga (the melodic mode, scale, order, and rules of a particular piece of Indian classical music) [8,9,10,11].

Figure 1: The traditional North Indian sitar.

It is important to understand the traditional playing style of the sitar to comprehend how our controller captures its hand gestures. Our controller design has been informed by the needs and constraints of the long tradition and practice of sitar playing. It should be noted that there are two main styles of sitar technique: Ustad Vilayat Khan’s system and Pandit Ravi Shankar’s system.

Monophonic melodies are performed primarily on the outer main string, and occasionally on the copper string. The sitar player uses his left index finger and middle finger, as shown in Figure 2(a), to press the string to the fret to play the desired swara (note). The frets are elliptically curved so the string can be pulled downward, to bend to a higher note. This is how a performer incorporates the use of shruti (microtones) which is an essential characteristic of traditional classical Indian music.

On the right index finger, a sitar player wears a ring-like plectrum, known as a mizrab. The right hand thumb, remains securely on the edge of the dand (neck) as shown on Figure 3(a), as the entire right hand gets pulled up and down over the main seven strings, letting the mizrab strum the desired melody. An upward stroke is known as Dha and a downward stroke is known as Ra. The two main gestures we capture using sensors and subsequently try to model using audio-based analysis are: 1) the pitch/fret position and 2) the mizrab stroke direction. The corresponding sensors are described in the following Subsection.

3.2. The ESitar Controller

The ESitar was built with the goal of capturing a variety of gestural input data. For our experiments, we are interested in gestures that deduce monophonic pitch detection and mizrab pluck direction. A variety of different families of sensor technology and signal processing methods are used, combined with Atmel AVR ATMega16 microcontroller[12], which serves as the brain of the ESitar.

3.2.1. Fret Detection

The currently played fret is deduced using an exponentially distributed set of resistors which form a network interconnecting in series each of the frets on the ESitar (pictured in Figure 2(b)). When the left hand fingers depress the string to touch a fret (as shown in Figure 2(a)), current flows through the string and the segment of the resistor network between the bottom and the played fret. The voltage drop across the in-circuit segment of the resistor network is digitized by the microcontroller. Using a lookup table it maps each unique value to a corresponding fret number and sends it out as a MIDI message.

Figure 2: (a) Traditional fingers playing fret technique; (b) Network of resistors on the frets of the ESitar.

As mentioned before, the performer may pull the string downward, bending a pitch to a higher note (for example play a Cb from the A fret). To capture this additional information that is independent of the played fret, we fitted the instrument with a piezo pick-up to be fed into a pitch detector. We chose to implement the pitch detector as a pure data [13] external object using an auto-correlation based method [14]. The pitch detection is bounded below by the pitch of the currently played fret and allowed a range of eight semi-tones above to help eliminate octave errors.

3.2.2. Mizrab Pluck Direction

We are able to deduce the direction of a mizrab stroke using a force sensing resistor (FSR), which is placed directly under the right hand thumb, as shown in Figure 3. As mentioned before, the thumb never moves from this position while playing, however, the applied force varies based on mizrab stroke direction. A Dha stroke (upward stroke) produces more pressure on the thumb than a Ra stroke (downward stroke). We send a continuous stream of data from the FSR via MIDI, because this data is rhythmically in time and can be used compositionally for more then just deducing pluck direction.

Figure 3: (a) Traditional mizrab technique (notice thumb position); (b) FSR sensor used to measure thumb pressure.

4. AUDIO-BASED ANALYSIS

In this Section we describe how the audio signal is analyzed. For each short time segment of audio data numerical features are calculated. At the same time, sensor data is also captured. These two
steams of data potentially have different sampling rates. In addition, in some cases, the gestural data is not regularly sampled. We have developed tools to align the two streams of data for these cases. Once the features are aligned with the sensor data, we train a "pseudo" sensor using regression and explore its performance.

4.1. Audio-Based feature extraction

The feature set used for the experiments described in this paper is based on standard features used in isolated tone musical instrument classification, music and audio recognition. It consists of 4 features computed based on the Short Time Fourier Transform (STFT) magnitude of the incoming audio signal. It consists of the Spectral Centroid (defined as the first moment of the magnitude spectrum), Rolloff and Flux as well as RMS energy. More details about these features can be found in [15]. The features are calculated using a short time analysis window with duration 10-40 milliseconds. In addition, the means and variances of the features over a larger texture window (0.2-1.0 seconds) are computed resulting in a feature set with 8 dimensions. The large texture window captures the dynamic nature of spectral information over time and it was a necessary addition to achieve any results in mapping features to gestures. Ideally the size of the analysis and texture windows should correspond as closely as possible to the nature time resolution of the gesture we want to map. In our experiments we have looked at how these parameters affect the desired output. In addition, the range of values we explored was determined empirically by inspecting the data acquired by the sensors.

4.2. Audio-Based Pitch Extraction

The pitch of the melody string (without the presence of drones) is extracted directly from the audio signals using the method described in [16]. This method is an engineering simplification of a perceptually-based pitch detector and works by slitting the signal into two frequency bands (above and below 1000Hz), applying envelope extraction on the high-frequency band followed by enhanced autocorrelation (a method for reducing the effect of harmonic peaks in pitch estimation). Figure 4 shows a graph of a simple ascending diatonic scale calculated directly from audio analysis.

The audio-based pitch extraction is similar to many existing systems that do not utilize machine learning therefore it will not be further discussed. Currently the audio-based pitch extraction works only if the drone strings are not audible. We are planning to explore a machine learning approach to pitch extraction when the drone strings are sounding in the future.

The interaction between sensors and audio-based analysis can go both ways. For example we used the audio-based pitch extractor to debug and calibrate the fret-sensor. Then the fret sensor can be used as ground truth for machine learning of the pitch in the presence of drone strings. We believe that this bootstrapping process can be very handy in the design and development of gestural music interfaces in general.

4.3. Regression Analysis

Regression refers to the prediction of real-valued outputs from real-valued inputs. Multivariate regression refers to predicting a single real-valued output from multiple real-valued inputs. A classic example is predicting the height of a person using their measure weight and age. There are a variety of methods proposed in the machine learning [17] literature for regression. For the experiments described in this paper, we use linear regression where the output is formed as a linear combination of the inputs with an additional constant factor. Linear regression is fast to compute and therefore useful for doing repetitive experiments for exploring the parameter. We also employ a more powerful back propagation neural network [18] that can deal with non-linear combinations of the input data. The neural network is slower to train but provides better regression performance. Finally, the M5 prime decision tree based regression algorithm was also used [19]. The performance of regression is measured by a correlation coefficient which ranges from 0.0 to 1.0 where 1.0 indicates a perfect fit. In the case of gestural control, there is significant amount of noise and the sensor data doesn’t necessarily reflect directly the gesture to be captured. Therefore, the correlation coefficient can mainly be used as a relative performance measure between different algorithms rather than an absolute indication of audio-based gestural capturing.

5. AUDIO-BASED GESTURE EXTRACTION

5.1. Data Collection

In order to conduct the experiments the following tools were used to record audio and sensor data. Audio files were recorded with DigiDesign’s ProTools Digi 002 Console using a piezo pickup (shown in Figure 4) placed directly on the sitar’s tabli. Midi data was piped through pure data [13] (http://pure-data.sourceforge.net/) where it was filtered and sent to a custom built Midi Logger program which recorded time stamps and all midi signals. Feature extraction of the audio signals was performed using Marsyas [20] (http://marsyas.sourceforge.net). The sampling rate of the audio files and the sensor data were not the same. The audio data was sampled at 44100 Hz and then downsampled for processing to 22050 Hz. Also the sensor data was not regularly sampled. Tools were developed to align the data for use with Weka [21] (http://www.cs.waikato.ac.nz/ml/weka/), a tool for data mining with a collection of various machine learning algorithms.

![Image](image.png)

Figure 4: Graph of Audio-Based Pitch extraction on an ascending diatonic scale without drone strings being played.
For the experiments, excerpts of a jhala portion of a ragu were performed on the ESitar. Jhala is a portion of sitar performance characterized by the constant repetition of pitches, including the drone, creating a driving rhythm. [10] Because of the rhythmic nature of this type of playing we chose to explore the signals of the thumb sensor to get an inclination of mizrab pluck direction using audio-based feature analysis and regressive machine learning algorithms.

5.2. Experiments using Regression Analysis

Our first experiment was to analyze the effect of the analysis window size used for audio based feature extraction. Table 1 shows the results from this experiment. Take note that the texture size remained constant at 0.5 seconds and linear regression was used. The correlation coefficient for random inputs is 0.14. It is apparent based on the table that an analysis window of length 256 (which corresponds to 10 milliseconds) achieves the best results. It can also be seen that the results are significantly better than chance. We used this window size for the next experiment.

<table>
<thead>
<tr>
<th>Analysis Window Size (samples at 22.5 KHz)</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation Coefficient</td>
<td>0.2795</td>
<td>0.3226</td>
<td>0.2414</td>
</tr>
</tbody>
</table>

Table 1: Effect of analysis window size.

The next experiment explored the effect of texture window size and choice of regression method. Table 2 shows the results from this experiment. The rows correspond to regression methods and the columns correspond to texture window sizes expressed in number of analysis windows. For example, 40 corresponds to 40 windows of 256 samples at 22050 Hz sampling rate which is approximately 0.5 seconds. To avoid overfitting we use a percentage split where the first 50% of the audio and gesture data recording is used to train the regression algorithm which is subsequently used to predict the second half of recorded data.

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Input</td>
<td>0.14</td>
<td>0.14</td>
<td>0.14</td>
<td>0.14</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>0.28</td>
<td>0.33</td>
<td>0.28</td>
<td>0.27</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.27</td>
<td>0.45</td>
<td>0.37</td>
<td>0.43</td>
</tr>
<tr>
<td>M5' Regression Method</td>
<td>0.28</td>
<td>0.39</td>
<td>0.37</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Table 2: Effect of texture window size (columns) and regression method (rows).

It is evident from the table and Figure 5 that the best choice of texture window size is 20 which corresponds to 0.25 seconds. In addition, the best regression performance was obtained using the back propagation neural network. Another interesting observation is that the relation of inputs to outputs is non-linear as can be seen from the performance of the neural network and M5' regression algorithm compared to the linear regression.

Figure 5: Graph showing the effect of texture window size and regression method.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose the use of machine learning methods and more specifically regression for replacing sensors with analysis of the audio input data. A proof-of-concept experiment to verify this idea was conducted using the ESitar controller. Our preliminary results indicate that regression can be used to predict non-trivial continuous control data such as the thumb sensor of the ESitar without using sensors. Previous methods either relied on complex signal processing design and were not robust to noise or only dealt with classification into discrete labels. Our approach handles gracefully both problems with minimal user involvement as the training is performed using the sensor we are trying to replace without requiring any human annotation. We show how the choice of regression method and analysis parameters affects the results for a particular recording. There is a lot of work to be done in exploring how this approach can be used and this is only the beginning.

There are many directions for future work. We are exploring the use of additional audio-based features such as Mel-Frequency Cepstral Coefficients (MFCC) and Linear Prediction Coefficients (LPC). We are also gathering more recording to use for analysis. Creating tools for further processing the gesture data to reduce the noise and outliers is another direction for future research. We would also like to try and predict other types of gestural data such as fret position. Another eventual goal is to use this technique to transcribe the sitar and other Indian music. We are also interested in using this method for other instruments such as the tabla and snare drum.

7. ACKNOWLEDGEMENTS

Many thanks to Scott Wilson, Ari Lazier, Phil Davidson, Michael Gurevich, Bill Verplank, and Perry Cook for their contribution in building the ESitar. Another thanks to Phil Davidson for his Midi Logger program used to record sensor data and Tiffany Jenkins for her help with data acquisition.
8. REFERENCES


SPARSE AND STRUCTURED DECOMPOSITIONS OF AUDIO SIGNALS IN OVERCOMPLETE SPACES

Laurent Daudet

Laboratoire d’Acoustique Musicale
Université Pierre et Marie Curie (Paris 6)
11 rue de Lourmel 75015 Paris France
daudet@lam.jussieu.fr

ABSTRACT

We investigate the notion of “sparse decompositions” of audio signals in overcomplete spaces, i.e., when the number of basis functions is greater than the number of signal samples. We show that, with a low degree of overcompleteness (typically 2 or 3 times), it is possible to get good approximation of the signal that are sparse, provided that some “structural” information is taken into account, i.e., the localization of significant coefficients that appears to form clusters. This is illustrated with decompositions on a union of local cosines (MDCT) and discrete wavelets (DWT), that are shown to perform well on percussive signals, a class of signals that is difficult to sparsely represent on pure (local) Fourier bases. Finally, the obtained clusters of individuals atoms are shown to carry higher levels of information, such as a parametrization of partials or attacks, and this is potentially useful in an information retrieval context.

1. INTRODUCTION

Sparse decompositions of audio signals are extremely useful in many signal processing applications: compression, noise reduction, source separation, detection, etc. The goal is to decompose the signal onto a small number of basis functions, called “atoms” (typically time-frequency atoms, such as Gabor atoms or local cosines; or time-scale atoms, such as wavelets). The fundamental problem: the bigger your set of atoms (i.e., the more redundant), the more likely you will have a good match between your signal and the atoms, but the larger your set of possible solutions. The difficulty is to find the “optimal” (whatever that means) solution amongst them, usually a problem of high algorithmic complexity.

The work presented here tries to tackle this problem with practical, however suboptimal, methods. Optimality is here defined in terms of compression, i.e., bitrate vs. quality of approximation. Our space of representation is limited to the union of a basis of local cosines (MDCT) and a basis of dyadic discrete wavelets (DWT). This choice of basis is relevant for audio signals, especially for percussive sounds where the well-defined supports are difficult to capture with purely Fourier-based approaches; and where assumptions of harmonicity of the partials are not always verified. Based on the observation that significant coefficients are not randomly distributed across the time-frequency (scale) plane but rather tend to form clusters, we do not select individual large atoms as significant, but groups of neighboring large atoms, called “molecules” or sound “micro-objects”. In the MDCT domain, these form spectral lines; in the DWT domain, these form sub-trees located around transient parts of the signal. Besides a significant reduction in the algorithmic complexity, “micro-objects” are more meaningful from an analysis point of view than isolated coefficients, and are cheap to encode. In short, this work presents a tentative framework for the unification of two developments that have emerged independently in recent years: sparse representations in overcomplete spaces [1], and structured representations (such as SPIHT [2] in image coding).

This paper is organized as follows: after a short introduction to sparse representations (part 2), we will detail how this can be implemented within the framework of sparse representations (part 3). Finally, the conclusion (part 4) will discuss potential applications for information processing or musical purposes.

2. SPARSE OVERCOMPLETE DECOMPOSITIONS

2.1. What does “sparse” mean?

There are many definitions of sparsity for representations of signals. Here, we work in the context of representations (projections) of the signal on a set of pre-defined functions or “atoms” \( \{ b_n \} \), and the signal is simply “represented” by the set of mixing coefficients \( \{ \alpha_n \} \).

\[
x(t) = \sum_{n=0}^{N-1} \alpha_n b_n(t)
\]

A given representation is said to be sparse if the number of non-zero coefficients is small compared to the dimension \( N \) of the space (the total number of samples in the signal). Mathematically speaking, this corresponds to a so-called \( L_p \) measure of sparsity, that counts the amount of non-zero coefficients in a given set.

This definition can be generalized, and we can define a number of \( L_p \) sparsity measures, with \( p > 0 \), that represent how the “energy” is concentrated on a small number of coefficients:

\[
L_p(\{ \alpha_n \}) = \sum_{n=0}^{N-1} |\alpha_n|^p
\]

Amongst these, the \( L_1 \) measure is a popular choice since some algorithms can be implemented with linear programming techniques. Note that \( L_2 \) is just the standard measure of signal energy, and is invariant if the space of representation is an orthonormal basis, or a union thereof. Note that recent results [3] show that, if a given signal admits a “very sparse” representation under a given norm, then this representation is also the sparsest with respect to all \( L_p \) sparsity measures, \( 0 \leq p \leq 1 \).
Also commonly used are sparsity measures based in the normalised information measure:

\[ L_p(\{a_n\}) = \frac{1}{A} \sum_{n=0}^{N-1} \log(\frac{|a_n|}{A})^p \]  

(3)

where \( A \) is the \( L_2 \)-norm of the \( a \)'s.

2.2. Sparse approximations

In this study, we are going to relax the above constraints: we do not look for strictly sparse representations of signals, but for approximations of the signals that are sparse. More specifically, we assume that our signal is the sum of a signal that admits a sparse representation and a small noise component:

\[ x(t) = \sum_{n=0}^{N-1} a_n b_n(t) + \gamma(t) \]  

(4)

where \( \gamma \) is our approximation error that is assumed to be small. In the context of audio compression, the tradeoff between quality of approximation and sparsity can be formalized by way of rate-distortion curves.

2.3. Approximations in overcomplete spaces

In general, decompositions on orthonormal spaces do not provide sparse representations: it is indeed very unlikely that everywhere the signal locally resembles the basis functions. Actually, the basis functions have by themselves little to do with the signal - neither do they sound like the signal - , and it is the union of (usually a large number of) them that allow a good approximation. Figure 1 shows a few of this individual atoms, taken from two popular choices of orthonormal basis: the Modified Discrete Cosine Transform (MDCT), and the Dyadic Wavelet Transform (DWT).

![Figure 1: Individual atoms: top: three time-frequency (MDCT) atoms; bottom: four time-scale (DWT) atoms.](image)

Therefore, it is usually preferable to use overcomplete spaces of approximation, ie. spaces with a number of basis functions that is greater than the dimension of the space. One may for instance choose Gabor frames [4], which can be seen as a set of discretized windowed Fourier atoms.

The problem with overcompleteness is that one loses the nice orthogonality principle that grants us the uniqueness of the decomposition: indeed, for a given signal there is an infinity of possible decompositions. The problem here is to find, amongst these decompositions, the one that is the most sparse, or that admits a sparse approximation, according to one of the above definition of sparsity.

2.4. Previous approaches

The above problem is in general not tractable, since its algorithmic complexity is huge (they belong to the NP-complete class of problems). However, there are approaches that give practical solutions, at a cost of suboptimality. Amongst these, let us mention the approach of Matching Pursuits [5] and Basis Pursuits [6]. The main drawback of these methods that their algorithmic complexity is still too high to be used on high-dimension signals such as audio signals.

3. STRUCTURED DECOMPOSITIONS

The method proposed here relies on the observation that large (ie significant) coefficients are not randomly located, but form structures, or clusters, in the parameter plane. Here we restrict ourselves to spaces with a small degree of overcompleteness, typically 2 times or 3 times; and this allows for good visualization of the clusters. For the simplicity of the decompositions, we will choose the union of 2 (or 3) orthonormal bases. Preliminary results indicate that, if the bases are sufficiently different from each other, there is little to gain in choosing higher degrees of overcompleteness.

Our choice of basis will be the union of a basis of Modified Discrete Cosine Transform (MDCT) and a basis of Dyadic (discrete) Wavelet Transform (DWT). The MDCT is a popular choice in the audio coding community, since it is similar to a windowed Fourier transform while keeping the orthogonality for real signals. It is well adapted to the representation of locally tonal signals. The DWT, with short wavelets, is well known for its capacity to analyze transient portions of the signal, such as the attacks of percussive sounds.

With these notations our problem may be restated as follows. For a given signal \( x \) find the best overcomplete MDCT / DWT approximation of \( x \):

\[ x(t) = \sum_{n \in A} a_n a_n(t) + \sum_{m \in B} b_m b_m(t) + \gamma(t) \]  

(5)

where the \( a_n \) (resp. \( b_m \)) are the MDCT (resp. DWT) basis functions, and \( A \) and \( B \) the set of significant coefficients. Here, “best” means that the set of significant coefficients \( \{a_n\}_{n \in A} \cup \{b_m\}_{m \in B} \) is sparse according to our sparsity measure.

3.1. Tonal structures

In the MDCT time-frequency plane, large coefficients form tonal structures that appear along the spectral lines, as in Figure 2. On a practical point of view, tonal structures are detected as places where the MDCT spectrum pseudo-spectrum (a smoothed near-shift-invariant version of the MDCT spectrum \( |H_m| \), described in [7]) is strongly
correlated across time. A tonal structure is then described as a set of MDCT coefficients with a width of 3 frequency bins, extending over a number of adjacent windows (see Figure 2).

Figure 2: Top: MDCT spectrogram of a typical percussive signal. Bottom: detail of the structure of significant tonal coefficients for the partial framed on the top figure.

Figure 3: “Tonal molecule” corresponding to the coefficients in the bottom plot of Figure 2, centered around the frequency bin 98. Note that the selected molecule has connected separate groups of significant coefficients (the gaps in time frames 32 and 45 correspond to interference processes in the decomposition and do not bear a physical meaning).

Figure 4 represents the MDCT spectrogram of a glockenspiel recording, and Figure 5 shows the detected tonal structures on the same file. Although some partials are not detected, the obtained pattern can be seen as a signature of the original sound, and indeed it is very close to it from a perceptive point of view, except at the onset of the notes.

Figure 4: Time-frequency MDCT spectrogram of a glockenspiel.

Figure 5: Time-frequency MDCT spectrogram of the tonal molecules selected by the algorithm, same soundfile as Figure 4.

3.2. Transient structures

Here, we work on the residual of the previous tonal extraction, that appears to contain mainly the transient sequences at the onsets of the notes. These will be represented as structures in the Dyadic Wavelet Transform (DWT) domain, which is organized in a dyadic tree structure in the time-scale plane. We use compact wavelet with
short support, such as Haar or Daubechies-4, since they provide good time localization properties [8].

Here, transient structures will be defined as “trees” in the time-scale dyadic plane where large coefficients are correlated across scales [9], as shown in Figure 6. When a tree is selected, we prune out the small coefficients with a top-down search, that ensures that the remaining trees are connected and fully connected to the largest scale, i.e., the root of the dyadic tree. For very percussive signals, we observe that the selected structures correspond to the sharp attack transients of the sounds.

Figure 6: In the dyadic time-scale plane, large wavelet coefficients cluster around branches of sparse trees.

We have tested this algorithm on a number of signals. Preliminary results indicate that the best results are obtained on the sounds that are difficult to represent on classical Fourier-based spaces, namely the percussive signals. The main characteristics of these signals is that they are not harmonic (previous models extending Matching Pursuits have assumed harmonicity [10]), and they are strongly non-stationary at their attacks (hence requiring a large number of Fourier coefficients). Although this model is very general, the sparsity of the decompositions degrades quickly when the local frequencies of the tonal partials is not constant, e.g., in the case of a frequency chirp or vibrato. More complex tonal decompositions can be implemented to account for these classes of signals.

Furthermore, one of the main drawbacks of the algorithm at present is that one needs a priori estimate of the relative tonal / transient importance. Future improvements will offer a simultaneous rather than sequential estimation of these two components, through a modified version of the Matching Pursuit [11]. However, this is expected to induce a significant increase in the computational requirements.

It is also interesting to note that the reconstructed signals from the obtained structures are meaningful from an auditive point of view (whereas individual atoms were not). Tonal structures sound like individual partials, and transient structures sound like the attacks of each note. This may be seen as intermediate level of representations, between the low-level time-frequency / scale planes, and the (relatively) high-level MIDI-like representation as individual notes (see Figure 8). The difficulty of automatic transcription of audio files makes these intermediate levels (called “micro-objects”) an attractive option. This could also be useful for audio indexing purposes, used in information retrieval systems (for instance these objects bear information about the structure of the timbre, that is lost in the MIDI information).

Finally, one may wonder whether musical applications of this techniques are sensible. The author believes that this may be the case, since structures can be manipulated (sound transformations / effects) or completely created as reorganizations of molecules from different (possibly natural) sounds (this can be seen as an extension to the widely-used granular synthesis).

5. ACKNOWLEDGEMENTS

The author would like to thank Profs. B. Torrésani, Université de Provence, and M. Sandler, Queen Mary, University of London, for their contribution to this work.
6. REFERENCES


DETECTION OF CLICKS USING SINUSOIDAL MODELING FOR THE CONFIRMATION OF THE CLICKS

Esteban Alvarez, Rafael Mendez and Guillermo Langwagen
Faculty of Engineering,
University ORT Uruguay

ABSTRACT
This article presents methods for clicks detection in degraded audio recordings. It begins with a brief description of the method implemented in first instance for the detection of clicks in audio sources based on linear prediction. Looking for an improvement of the results obtained with this method, we propose a method based on sinusoidal modeling for the confirmation of the clicks. This method discards clicks that were wrongly detected. This allows the detection of clicks of small amplitude avoiding wrong detections. The results obtained by this method are shown, confirming the good operation. Finally, the method implemented for detection of clicks in naturally degraded audio sources is presented.

1. INTRODUCTION
There are several distinct types of degradation common in audio sources. These can be broadly classified into two groups: localized degradations and global degradations. Localized degradations are discontinuities in the waveform, which affect only certain samples.

The most common of this type of degradation are the clicks. Global degradations affect all samples of the waveform being hiss the most common.

The first step in the restoration of an audio source is the detection of the clicks. The algorithm to be implemented must fulfill the following requirements: detect most true clicks, detect properly the width of the clicks and avoid false detections that would degrade the signal even more. The detection methods are based on linear predictors.

2. DETECTION METHOD
In the following Sections we will explain some of the methods implemented to detect the clicks. These methods will be used before and in conjunction to sinusoidal modeling.

2.1. Linear Prediction
The audio signal will be modeled with a linear predictor that minimizes the mean squared error (see [1], [2] and [3]). With this method, the value of the predicted sample is based on a linear combination of the preceding samples:

$$\tilde{s}_n = \sum_{k=1}^{p} a_k s_{n-k},$$

where

$$\tilde{s}_n = \text{predicted sample}$$

$$s_n = \text{signal sample}$$

$$a_k = \text{coefficient of the AR model}$$

The prediction error (the difference between the value of the predicted sample and the original sample) is given by:

$$e_n = s_n - \tilde{s}_n = s_n + \sum_{k=1}^{p} a_k s_{n-k},$$

Considering \(\sigma_e\), as the standard deviation of the error signal, a sample belongs to a click if

$$|e_n| > K \cdot \sigma_e$$

The product \(K\cdot\sigma_e\) is called the detection threshold. The value of \(K\) (real number) must be sufficiently great to avoid false detections and small enough so that clicks are detected. The value that multiplies \(K\) may be other that the standard deviation of the error, since the module of the average or the module of the median can be used [1].

2.2. Determination of the width of the click
Because of the linear prediction, there is a problem with the detection inside the click. The detector works quite well in the borders of the click, however in the middle of it, the detector consider that some samples are not affected by the click. Figure 1a shows in thin line a click and in thick line the detection vector (clicks vector). This vector takes the value 1 when a sample is consider like click, otherwise takes the value 0.

2.2.1. Zeros consecutive parameter
In order to solve the problem previously mentioned, the parameter consecutive zeros was defined. This parameter counts the number of samples not being considered as clicks that are between two samples detected like clicks. If the value of the parameter consecutive zeros is less than 20, then the samples between clicks will be considered also as clicks. In fact, all these consecutive samples will be considered as pertaining to the same click (Figure 1b).

2.2.2. Detection in inverse sense of the music
The detection of the end of clicks is generally not correct: clicks finish before the detector indicates their end (Figure 1b). In order to solve this problem a detection of clicks is made in inverse sense of the music. This detection is made in the same way as previously explained, with the difference that the inverted musical signal vector is processed (Figure 1c).
2.2.3. Double-threshold method

From ideas obtained from [1] we implemented a system of double threshold. The first threshold (called detection threshold) will be used for the determination of the location of clicks, and a second threshold (called small threshold) lower than the first one will be used to determine the width of each click. The variant which we developed respect to [1] was to perform an additional detection in inverse sense of music with the small threshold to determine the correct width of the click.

In summary, to detect clicks the following steps are followed: first, detection of clicks with the detection threshold without taking in consideration the width of the click. After that, detection with the small threshold is made and next detection with this same threshold but in the inverse sense of music is made. The detections with the small threshold do not add new clicks, only widen or shorten them.

\[
\text{Amplitude} = \begin{cases} 
1 & \text{for clicks detected} \\
0 & \text{otherwise}
\end{cases}
\]

2.3. Iteration

In signals affected by clicks, there are clicks of high amplitude and others with low amplitude. The detector sometimes does not detect the clicks of low amplitude because the error is smaller than the threshold. The implemented procedure to overcome this disadvantage is explained next. After the first detection, the sections affected by clicks are recovered, using the interpolator algorithm LSAR (Least Squares Auto Regressive [1], [2], [3]). The new signal will have a smaller standard deviation error vector. Therefore, the detection threshold will be also smaller. In addition clicks that before were considered small will now be more representative. Then, the detection of clicks is again applied to the signal recovered in first instance, improving the detection of small clicks.

2.4. Use of windows to calculate the threshold

Because of the variable behavior of the musical signals, it becomes necessary to use a dynamic system of thresholds that changes its value throughout the recording. In order to implement the variation of the thresholds, the standard deviation of the error is calculated for each window (440 samples). A vector of the thresholds whose value changes from window to window is obtained. So the threshold adapts to the variations of the signal.

2.5. Detection of clicks in artificially degraded recordings

Having implemented a generic method for the detection of clicks, we used it to detect clicks of a recording and to interpolate these zones (for the interpolation LSAR was used).

The qualitative results of the recovered audio signals are in last instance a subjective question related to the perceptual results produced by the processing. In this part of the work we used non degraded signals to which artificial clicks were added to allow the use of quantitative measures that could validate the obtained results. For this purpose, the rate of wrongly detected clicks (clicks detected in samples not affected by click) and the rate of non detected clicks (clicks where the detector determined erroneously that the samples did not belong to click) were measured. In order to make these calculations, recordings with clicks artificially inserted, obtained from [4], were used, so that the location of clicks on the recordings were known.

The equations for obtaining the mentioned parameters are the following ones:

\[
\% \text{ wrongly detected} = \left(1 - \frac{\text{real clicks} - \text{wrongly detected clicks}}{\text{real clicks}}\right) \times 100
\]

(4)

\[
\% \text{ non detected} = \left(1 - \frac{\text{real clicks} - \text{non detected clicks}}{\text{real clicks}}\right) \times 100
\]

(5)

These rates were used to determine the values of the parameter used for the detection of clicks in artificially degraded recordings.

2.6. Detection of clicks in naturally degraded recordings

Once all the parameters for the detection of clicks were determined, we made tests in naturally degraded recordings. The results in first instance were not satisfactory since the detector fails in samples clearly affected by clicks. A possible cause of this behavior is that the waveform of clicks inserted artificially differs from clicks that appear in real recordings. Another cause may be that as the signal also is affected by hiss, the model of linear prediction is not as effective as in the signals that only have clicks. In order to solve this problem some parameters of the detector were modified and a method of confirmation of clicks was introduced. This will be explained next.

3. METHOD FOR CLICK CONFIRMATION USING SINUSOIDAL MODELING

The methods implemented for the detection of clicks are based on the comparison of the real signal with an estimated one. Thus we are looking for a threshold that let us detect all the clicks of the signal and avoid false clicks detection. Looking for a method to discard false click detection, we propose the method of click confirmation. We can use a smaller detection threshold than previously in order to detect all the clicks of the signal. Since the
threshold is small, there will be a lot of false detection clicks, but these will be discarded with the method presented in the following Sections.

3.1. Introduction

The objective of sinusoidal modeling (see [5], [6] and [7]) is to represent a signal based on variable sinusoids of frequency and amplitude in time. In order to do that, we perform the spectral analysis in windows of time. So it is possible to calculate the functions that describe the variations of frequency, amplitude and phase of each component of the signal.

These methods enable us to model a signal $x[n]$ as a sum of evolvemental sinusoids $[5]$

$$x[n] = \hat{x}[n] = \sum_{q=1}^{Q[n]} A_q[n] \cdot \cos \theta_q[n],$$

(6)

where $Q[n]$ is the component number in time $n$. The components will have variable amplitude $A_q[n]$ and phase $\theta_q[n]$ in time.

Sinusoidal modeling can be considered as a generalization of the Fourier series that allows us to describe the signal as the sum of sinusoids that changes in dependence with its behavior.

The sum of sinusoids that have slow variation in time is not effective for the modeling of impulsive events or noise. A term grouping these processes must be added to the model. That term is called residual, $r[n]$. So

$$x[n] = \hat{x}[n] + r[n]$$

(7)

Sinusoidal modeling can be interpreted as the evolution of the STFT (Shot Time Fourier Transform). In the following Sections we will present a brief description of the STFT and its relationship with sinusoidal modeling.

3.2. Short Time Fourier Transform

The goal of the STFT is to derive a time-localized representation of the frequency-domain behavior of a signal. The STFT is carried out by applying a sliding time window to the signal; this process isolates time-localized regions of the signal. Each of them is analyzed using a discrete Fourier transform (DFT).

Typically, the STFT is given by

$$\hat{X}[k,n] = \sum_{m=-N/2}^{N/2-1} w[n-m] \cdot x[m] \cdot e^{-j\omega_k m},$$

(8)

The analysis presented in this article will be based in the following definition of the STFT. The reference of time is changed to facilitate its interpretation as a filter bank and its relationship with sinusoidal modeling:

$$X[k,n] = \sum_{m=0}^{N-1} w[m] \cdot x[n + m] \cdot e^{-j\omega_k m},$$

(9)

where $\omega_k = 2\pi k/K$ and $w[m]$ is a window in the time domain with zero value outside of the interval $[0, N-1]$.

Equation (9) can be expressed as:

$$X[k,i] = \sum_{m=0}^{N-1} w[m] \cdot x[m + iL] \cdot e^{-j\omega_k m},$$

(10)

where $L$ is the time distance between successive applications of the window to the data.

In the first case, the reconstruction of the signal is obtained by calculating the inverse DFT for each window of the spectrum. From equation (8), we get:

$$\hat{x}[n] = \sum_{k=-N/2}^{N/2-1} \hat{X}[k,n] \cdot e^{j\omega_k n}.$$
3.3. Implemented method for click confirmation

Figure 3 shows samples of a song that includes a click. Also there is the reconstruction of the signal using five components with sinusoidal modeling.

It can be seen that the modeled signal does not follow the click closely because the reconstruction uses only few components. We propose to use this property to discard clicks that were detected incorrectly.

For each detected click we will reconstruct the signal using 50 samples previous to the start of the click and until 50 samples after its end. That signal will be compared with the real signal. If there is no sample that is bigger than the threshold (the definition will be explained later) we will consider that the samples involved are not a real click.

where \( k \) is 3 (determined experimentally) and \( \text{Var}(\text{difference}) \) is the variance of the difference of the mentioned signals.

For each click we calculate the threshold and then we compare the original signal with it. If any of the samples is bigger than the threshold, then we consider that the detected click is a real click. Otherwise the click is not confirmed and the vector that indicates the affected samples is modified (Clicks Vector).

![Figure 3: Signal modeled in the presence of a click.](image)

3.3.1. Algorithms

Figure 4 shows the flow chart of the algorithms that was implemented.

The program works as follows: first, it takes the vector that shows the samples that are affected with clicks. This is done using the detection algorithms previously explained with a small threshold. Because of that condition, we expect that some of the detected clicks are not real clicks.

Secondly, we define the threshold and model the signal. The idea is to include close to 100 samples without clicks so the modeled signal become similar to the music but not to the click (if it exists). For that purpose we model using 50 samples previous to the start of the click until 50 samples after the end of the click and the samples affected by clicks are substituted by a straight line. Then, the signal is reconstructed using five components with sinusoidal modeling.

To confirm the click we calculate the difference between the reconstructed signal and the original one in the samples that are not affected by clicks, that is to say in the 50 samples before the click and the 50 samples after the click. Then we calculate the comparison threshold using the following difference:

\[
\text{Threshold} = k \cdot \sqrt{\text{Var}(\text{difference})} \quad (14)
\]

3.3.2. Results

To evaluate the method we use the parameters previously explained in this article. We call “wrongly detected clicks,” to the detected clicks in samples that are not affected. With the term “non detected clicks” we mean the real clicks that are not detected.

The Table 1 shows the results of the click detection without using sinusoidal modeling (just the detection explained in the previous Sections) and using sinusoidal modeling.

The parameters used for the comparison are:

- Number of iterations: 6, the first 3 with a threshold of 8 and the other 3 with a threshold of 7.
- The value of \( k \) is 3.

<table>
<thead>
<tr>
<th>Signal</th>
<th>WITHOUT Sinusoidal Modeling</th>
<th>WITH Sinusoidal Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% non detected</td>
<td>% wrongly detected</td>
</tr>
<tr>
<td>Signal1</td>
<td>0</td>
<td>42.98</td>
</tr>
<tr>
<td>Signal2</td>
<td>1.65</td>
<td>9.92</td>
</tr>
<tr>
<td>Signal3</td>
<td>0</td>
<td>9.92</td>
</tr>
<tr>
<td>Signal4</td>
<td>1.65</td>
<td>8.26</td>
</tr>
<tr>
<td>Signal5</td>
<td>0.44</td>
<td>2.31</td>
</tr>
<tr>
<td>Signal6</td>
<td>0.88</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Table 1
Given that the algorithm of confirmation sometimes discard clicks that really are, it can be seen that there is a small increase in the value of “non detected clicks.” However, the percentage of “wrongly detected clicks” has been substantially decreased. The algorithm results efficient in the “no-confirmation” of detected clicks in samples that are not affected by clicks, so it achieves its primary objective.

4. ERROR VECTOR

To improve the detection in naturally degraded songs, we introduced an important modification: in the analysis shown in [1], there is a variant for the calculation of the threshold error. The objective is to improve the threshold calculation so that it becomes independent of the power of the signal and immune to the degradations of the song. In that way, we follow these steps to determine the prediction threshold error:

- Obtain the vector error of each window
- Square the elements of the vector
- Delete a determined percentage of the highest values of the vector
- Calculate the root of the elements of the vector
- The threshold error is obtained calculating the mean of the absolute value of the vector

The number of degraded samples can vary from a window to other. This can not be known beforehand, so we had to determine an arbitrary number of samples to be eliminated. In [1] the author proposes a value close to 10% of the highest samples of the error vector. In this article we propose the elimination of 5% of such samples. The statistic that we will use is the standard deviation because we got better results than with the mean of the error vector.

5. IMPLEMENTED CONFIGURATION

In songs with real clicks the behavior of the detection can only be determined by qualitative results. By qualitative results we mean comparing the signal with the detected click vector or listening to the restored song.

After several tests of the different configurations, we determined the following parameters for the detection of clicks:

- Number of coefficients of AR model: 20
- Length of the window: 440 samples
- Detection threshold: 7 · Standard deviation of the error vector without 5% of the highest samples.
- Small threshold: 5.5 · Standard deviation of the error vector without 5% of the highest samples.
- Number of iterations: 6

The detection threshold was not easy to determine. If the threshold is high (7), the clicks of high amplitude are detected but there are still many clicks not detected. If the threshold is small, there are a lot of wrong detections. Because of that we use sinusoidal modeling to discard clicks. As we explained previously, the algorithm that implements sinusoidal modeling takes the vector click with the detected clicks and discards the ones that are not confirmed.

The detection threshold will have two possible values, in accordance with the number of iterations that the process has. In the first four iterations the threshold will be 7, there the clicks of highest amplitude will be detected. In the following two iterations the threshold will be 6 and we use sinusoidal modeling to discard those clicks wrongly detected. The small threshold has a value of 5.5 for the first four iterations and 5 for the last two. In the following flow chart, the process of detection and interpolation of the clicks it can be seen.

![Flow chart of the detector’s process.](image)

Next, we can see a click detected with the mentioned method.

![Real click detected.](image)

To ensure the detection of the whole click, we decided to increase in two samples the start and the end of the click. Other option is to decrease the small threshold, but in this case there are risks to overestimate the width of the click. So with this option we get
better results in the detection in the edge of the clicks and do not add many samples to each click.

Using this configuration there is an average of 25.1 clicks detected per second in songs that do not have degradations. In spite of the numerical value, the important point is to determine if a song without clicks is degraded when it is submitted to the procedure of detection-interpolation. Thus, we applied the mentioned method to many songs without degradations. It can be seen that the difference between the original song and the restored song is practically null, so we confirm that using these parameters no degradations are introduced to the songs.

6. CONCLUSIONS

Respect to the determination of the width of the clicks, the implementation of the detection in inverse sense of the music allows us to adjust properly the end of the click. The double-threshold method allows us to reduce the number of samples to increase at the start and the end of the click. This improves noticeably the quality of the restored song.

The use of windows enables us to adapt the value of the detection threshold to the variations of the signal, following improvements in click detection. This is because the threshold adjusts to the statistic of the signal.

The click confirmation method based on sinusoidal modeling introduced in this article allows us to reduce the detection thresholds to detect the small amplitude clicks, while avoiding an increase in the wrongly detected click value. This method improves substantially the restoration of the songs that have clicks with small amplitude.

Finally, the complete methods implemented for the detection of clicks achieve excellent results in songs both natural and artificial degraded.

7. REFERENCES


http://www.lps.ufrj.br/audio/


http://www.cs.tut.fi/sgn/arg/music/tuomass/ssempdf
AUDI0 ANALYSIS, VISUALIZATION, AND TRANSFORMATION WITH THE 
MATCHING PURSUIT ALGORITHM

Garry Kling and Curtis Roads

Media Arts and Technology
3431 South Hall
University of California,
Santa Barbara, CA 93106-6065 USA
http://www.mat.ucsb.edu/
g.kling@mat.ucsb.edu clang@create.ucsb.edu

ABSTRACT
The matching pursuit (or MP) algorithm decomposes audio data 
into a collection of thousands of constituent sound particles or ga-
borets. These particles correspond to the “quantum” or granular 
model of sound posited by Dennis Gabor. This robust and high-
resolution analysis technique creates new possibilities for sound 
visualization and transformation. This paper presents an account 
of a first round of experiments with MP-based visualization and 
transformation techniques.

1. THE GRANULAR REPRESENTATION OF SOUND
In the 1940s, the Nobel Prize winning physicist Dennis Gabor pro-
posed that any sound could be decomposed into acoustical quanta 
bounded by discrete units of time and frequency [1,2,3]. This 
quantum representation formed the famous Gabor matrix. Like 
a sonogram, the vertical dimension of the Gabor matrix indicated 
the location of the frequency energy, while the horizontal dimen-
Sion indicated the time region in which this energy occurred. In 
a related project, Gabor built a machine to granulate sound into 
particles. This machine could alter the duration of a sound without 
shifting its pitch. In these two projects, the matrix and the granu-
lator, Gabor accounted for both important domains of sound rep-
resentation. The matrix was the original windowed frequency do-
main representation. The granulation machine, on the other hand, 
operated on a time domain representation.

Today such representations are labeled by a multiplicity of 
terms: “acoustic quantum,” “grain,” “gaboret,” “Gaussian elemen-
tary signal,” “short-time segment,” “Gabor atom,” “wavelet,” etc. 
Roads [4] cites 32 different names. In this paper, we refer to all 
such techniques as granular representations, echoing the term used 
by the composer-engineer Iannis Xenakis, who first proposed a 
granular representation of musical sound [5,6,7].

Techniques that exploit granular representations have emerged as 
highly useful tools for the synthesis and transformation of musi-
cal sound. Recent advances let us probe and explore the beauties of 
this formerly unseen world. Granular techniques dissolve the rigid 
bricks of music architecture—the notes—into a more fluid and sup-
ple medium. Sounds may coalesce, evaporate, or mutate into other 
sounds. The sensations of point, pulse (regular series of points), 
line (tone), and surface (texture) appear as the density of particles 
increases. Sparse emissions leave rhythmic traces. When the par-
ticles line up in rapid succession, they induce the illusion of tone 
continuity that we call pitch. As the particles meander, they flow 
into streams and rivulets. Dense agglomerations of particles form 
swirling sound clouds whose shapes evolve over time.

The potential of granular representations has yet to be fully ex-
plored [4]. New approaches to signal analysis have demonstrated a 
variety of techniques that serve as analytical correlates to granular 
synthesis, under the broad category of wavelet or atomic decompos-
tions [8]. For clarity and consistency, we refer to these analytical 
methods as “granular decompositions.”

1.1. The Matching Pursuit Algorithm
Amongst the garden of granular decompositions lies the matching 
pursuit (MP) algorithm, pioneered by Mallat and Zhang [9]. The 
MP is a generalized framework for computing adaptive, granular 
signal representations. Many different flavors of the MP have been 
proposed [10]. The concept of the algorithm derives from Gabor: 
given an input signal, elementary particles can be combined to re-
constitute that signal.

Figure 1 shows the operational flow of the MP algorithm. In 
step 1, the sound input, dictionary, residue, and output buffers are 
initialized. In step 2, the algorithm searches the particle dictionary 
to find the best match to the sound energy. The search procedure 
varies by implementation.

The core of this search algorithm is the inner product space 
used to calculate the correlation between the sound and the dictio-
nary particles. In the simplest case, this is the standard dot product. 
Some varieties of the search procedure also contain a step that re-
finements the current grain selection [13], or optimizes the search using 
previous results [10].

The dictionary construction varies according to the flavor of 
the MP being run. The structure and contents of the dictionary are 
flexible: the only requirement of the dictionary is that its vectors 
(grains) form a basis for the set of all input signals. In the common 
case, the dictionary is redundant, made up of pure sines, gaborets 
(a sine wave modulated by a gaussian envelope), and diracs (tran-
sient functions). This dictionary is useful for musical signals since 
it contains particles that can reconstitute the basic structures of a 
musical event: harmonic steady-state spectra and transient attack 
structures. In addition to this form, the MP algorithm has been 
adapted to use dictionaries of chirplets [11], and damped sinusoids 
[12].

Once a grain is chosen, it is subtracted from the signal in step 
3. The remainder is called the residue, and is used for subsequent
The Short-Time Fourier Transform (STFT) forms the core of most analysis and transformation methods employed by musicians. The toolbox created around the STFT is extensive, with new applications being added every year. However, certain properties of the Fourier transform make it less than ideal for certain applications. Initial experimentation with MP analysis has yielded provocative results that suggest that it may perform certain tasks at a higher level of quality than the STFT.

Perhaps the most notable defect of the STFT is poor localization of sound structures in both time and frequency. The STFT is not well suited to describe events that are smaller or larger than the hop size of the analysis [9]. The frequency resolution is directly tied to the window size, which further complicates event onset localization. The time-frequency tradeoff of the STFT distorts the reality of sound, fogging the acoustic lens of computer music with spectral clutter. Since MP analysis uses a multiscale dictionary of atoms with arbitrary frequency resolution, time-frequency resolution is not tied to the size of the analysis sample. Visual comparisons of this phenomenon are given below.

Another important difference between the STFT and MP is translation invariance. This property of the MP representation makes it useful for pattern search and recognition, since a feature's representation is not dependent on its time-frequency location. The STFT is translation invariant as well, but this property is destroyed by sampling the translation parameter uniformly [8]. The translation invariance of the MP representation lets us alter and extract information while conserving the energy content of the original signal. It is well known that changing the parameters of a single frequency bin in an STFT alters the entire spectrum, and thus character of a sound. Initial experiments have shown that deleting, transposing, or otherwise altering a selected group of grains does not destroy the timbral character of a sound.

2. SOUND VISUALIZATION WITH THE MATCHING PURSUIT ALGORITHM

The granular representation created by MP analysis affords a unique and powerful visualization technique. Utilizing the Wigner-Ville distribution in a simple but novel way, these visualizations, which we would like to term “microsonograms”, let one look inside the life of a sound. The following is a brief overview of the techniques involved with some pictorial examples. We will present video examples during our talk.

2.1. Spectral Visualizations

Visualizations of spectral analysis are essential tools for computer music. For a history of spectrum analysis and visualization see [14]. Spectral analysis entered the modern era in the 1960s when Cooley and Tukey pioneered the fast Fourier transform and digital computing made its calculation practical [14]. Researchers were finally able to capitalize on the local Fourier analysis proposed by Gabor [1].

The Fourier spectrogram depicts sound as continuous strata, zones of intensity blurred across time and frequency. Pointed sonic gestures become dull smears. The granular representation and its visualization, as presented by Mallat and Zhang [9], give us a more detailed visual account of sound’s inner structure.

2.2. The Wigner-Ville Distribution and the Microsonogram

Mallat and Zhang [9] introduced a visualization method for the MP analysis data that uses Wigner-Ville distribution to plot the time-frequency energy of an analyzed signal. The Wigner-Ville Distribution (WVD) is a development that parallels the theories of Gabor and representative of a second (and fundamentally different) direction in time-frequency analysis [15]. While brought into the signal processing world by Ville in 1948, Wigner originally developed it in 1932 in the context of quantum thermodynamics [8]. In contrast to the granular model of signal energy, the WVD is a quadratic time-frequency energy distribution that is computed by correlating a signal with a time and frequency translation of itself. The result is a time-frequency distribution with a continuous character, as opposed to the granular model above.

The utility of the standard WVD is limited due to interference terms that make interpretation problematic [15]. However, this property is overcome using the granular representation produced by the MP. The standard MP uses only static frequency gaborets.
Figure 2: A single cycle of a sine wave at 440 hertz. At top is the signal plot, in the middle the single particle found with MP analysis, and at the bottom is spectrogram of a 256 point STFT with a Hamming window.

The WVD of a single gaboret does not contain interference terms since it is well localized in time and frequency. The WVD of each grain found in the MP is summed [9], and the result is a representation that is distinct from the sonogram. It is a collection of grains whose parameters are independent, rather than a continuum of sine waves that are locked together in time.

It should be noted that although MP analysis is amenable to dictionaries composed of a wide variety of grain waveforms, certain waveforms cannot be visualized with the WVD without interference. Gaborets, chirplets, and other stationary or linearly modulated, single component particles can be visualized with good results. Grains with multiple frequency components and non-linear frequency modulations require more complicated procedures to visualize clearly, with a limited degree of success [15].

2.3. Visualization Examples

Figure 2 is an example using a single cycle of a sine wave at 440 cps. At top is a signal plot, in the middle is the WVD of the single grain found with an MP, and at bottom is a 256-point STFT with a Hamming window. It can easily be seen how the MP representation is better localized in time and frequency. Figure 3 is a brief excerpt from the composition Pictor alpha by Curtis Roads [18]. At top, the signal plot of the excerpt, composed of grains and a sharp transient, in the middle, a microsonogram of the first 100 particles found in an MP analysis, and at bottom, a 256 point STFT with a Hamming window. Note how the STFT (bottom) while recovering the main pitched component in rough form, dissolves the transient across the time-frequency plane. The microsonogram (middle) represents the transient well, as well as better showing the amplitude variation of the pitched component. All components of the sound are localized.

Figure 3: A 100ms excerpt from the composition Pictor alpha by Curtis Roads. At top is a plot of the excerpt. In the middle is a microsonogram of the first 100 particles of a MP analysis of the signal. At the bottom is a spectrogram of a 256-point STFT with a Hamming window.

3. SOUND TRANSFORMATIONS USING MATCHING PURSUIT ANALYSIS

Granular representations of sound present us with new possibilities for sound transformation that are impossible or impractical using STFT-based methods. Granular representations have advantages that free us of many of the conceptual and technical encumbrances of the Fourier representation. But in order to forge new sound effects with this data, we must, to some degree, leave our Fourier sensibilities behind us. This will free us to take full advantage of the granular nature of sound energy.

Additive synthesis techniques draw from STFT-based methods due to their direct conceptual relationship. Similarly, we can find a wealth of information to fertilize our experiments with granular representations by looking at granular synthesis and transformation techniques.

For our experiments, we wrote software that analyses sound data using the MP functions [19] in the LastWave software package [20], which contains the most complete implementation of MP analysis. The transformations are performed on the resultant data, which is resynthesized with the CSound score language. We will present an overview of the results of our experiments thus far. Further results will be presented during our demonstration.

3.1. Pitch-Time Effects

Experiments have shown the MP representation of sound to be well suited for creating transposition and time stretching effects. As with the standard time-domain granulation versions of these effects [21], there are myriad ways that we can use to achieve these results.
Transposition of the granular representation is particularly effective for percussive, noisy sounds, in contrast to phase vocoder techniques. Since transients are well localized, the attack portion is retained, which makes the effect convincing. Sounds that are harmonic are also transposed well, owing to better frequency localization than the STFT. Policies for transposing harmonics while keeping noise in place will be needed to improve this effect.

Time compression and expansion are effective even using a simple multiplicative method. Time compression is quite excellent using the granular representation, again owing to preservation of transient structures. An intelligent model, such as the deterministic plus stochastic model in SMS [23], will produce superior results. Grains cannot be lengthened or shortened limitlessly without losing the character of the original sound.

3.2. Thresholding

Amplitude thresholding is an effect that can sift noise or harmonic ity from a sound. This transformation is carried out by resynthesizing grains whose amplitude is lesser or greater than some coefficient. Resynthesis of grains that fall below some level tends to recover the noise components of a sound. Resynthesis of grains greater than some level extracts harmonic content and strong transients from a sound. The warbling effect created by an incomplete analysis or a threshold that recovers only a small portion of a sound is sure to become a cliché of MP based techniques.

3.3. Octave Splitting

In the parlance MP analysis, the octave of a grain denotes its duration in samples, expressed as a power of two. For efficiency reasons, the dictionary used in the LastWave implementation of the algorithm contains only grains of some power of two in duration.

Given that we have a representation that sorts grains by duration, an obvious experiment is to synthesize grains of each size separately. It was found that shorter octaves retain the prosody of the original signal while the longer octaves take on the harmonic characteristics of the input. Shorter octaves have a characteristic noisy and transient timbre, while the longer ones are “warbled” or “watery” and are more pitched. This effect was found to have a very consistent character no matter what sound sample was used.

3.4. Coalescence and Disintegration

Past implementations of these effects were carried out with the tracking phase vocoder (TPV) in the QuickMQ program [22]. In this program, the sinusoidal tracks can be altered with a variety of algorithms. The Granny algorithm realizes disintegration and coalescence [22] through a process that cuts holes in the time-frequency tracks of a sound. This transformation, however, is marred by transient artifacts that compromise the overall effect.

For the present realization of coalescence and disintegration, a time frequency grid is laid across the granular representation, as shown in Figure 4. The cells of this grid roughly correspond to the cells of the Gabor matrix, and grains whose centers fall within a given cell are said to occupy them. The duration in seconds and size in frequency of the grid are adjustable.

The amount of the effect is governed by an arbitrary function (Figure 4). When the function governing the effect deletes more grains as the sound progresses, disintegration results. Less and less grains are kept to produce coalescence. The function can, however, take any form. A random number is drawn for each cell in the matrix. If the value of this number is greater than the value of the function during that cell, the grains contained by that cell are deleted. The resulting grid is called a shattering gradient, which can be stored for later editing and reuse (Figure 5).

These effects are delicate, and require a great deal of attention for clarity. Successful coalescence results in a sound being reconstituted in all areas of the spectrum at once, but in a chaotic fashion. Well-constructed disintegrations leave a sound falling to pieces from within, as if by some internal force. In the future, the
simplest versions of these two cases will be automated for quick preview use, suitable for subsequent customization.

4. CONCLUSIONS AND FURTHER WORK

Granular representations of sound offer us powerful new ways to look at and interact with audio material. Early experiments have shown that these tools give us promising new directions in musical signal processing. The properties of the MP give us a robust, high resolution representation that affords new methods for understanding and transforming sound.

Extensive exploration is needed to realize the full potential of the MP and the resultant granular representation. Computation of the MP is very time-consuming, making the algorithm useful to only those with patience or powerful equipment. Optimizations, dictionary design, and search strategies are topics that are being actively researched.

The visualizations produced by MP analysis and the WVD can reveal structures that are blurred in the traditional spectrogram, particularly transients and finely spaced frequencies. This is something that shows promise for examining the acoustics of musical instruments and other detailed sonic phenomena.

Creation of granular synthesis-inspired transformations extends the artistic toolbox of composers and sound designers. The effects presented here are simple examples of the many possibilities that granular representations have to offer.

In the near future, we will be able to use the granular representation of sound to create a broad menu of audio analysis, visualization, and transformation techniques unheard of in the Fourier domain. The already versatile toolbox that the STFT affords is being augmented with ones using a particular theory of sound. This is a universe in which sound has a time pattern and a frequency pattern, an elementary principle [2] that was only partially acknowledged in Gabor’s time, but one that we fully appreciate in our own. It is only in the present time, with analytical tools such as MP analysis and the granular representation that we are beginning to exploit this property of sound directly.

5. REFERENCES

ABSTRACT

The familiar “crackling” is one of the undesirable phenomena which we deal with in an LP record. Wavelet analysis brings a new alternative approach to the removal of this feature in the restoration process of the recording. In the paper, the principle of this method is described. A theoretical discussion of how the selection of the wavelet basis affects the quality of the restoration is also included.

1. INTRODUCTION

The wavelet-type signal analysis has recently been a much used discipline, whose range of applications in one-dimensional and multi-dimensional signal processing spreads steadily. It is used for the time-frequency analysis, for reconstruction of non-complete or strongly disturbed signals and for data compression in many fields.

There were many algorithms developed how to restore the digitized audio signal from a vinyl record, for example SDROM (Signal Dependent Rank Order Mean) [3], methods based on linear prediction in AR and ARMA models [6] or the well-known median filtering. Another special class of restoration methods exploit the Bayesian statistics [5]. This paper introduces an alternative approach to the problem, using the wavelet signal processing.

In the paper, we first discuss the time and frequency characteristics of a “crackle”. After this an overview of wavelet transform and its properties necessary for the method’s derivation is presented. Then, the principle of the crackle removal is described. At the conclusion, the discussion of how the choice of the so-called mother wavelet affects the quality and effectiveness of the restoration is introduced.

1.1. Characterization of a “crackle"

The typical behavior of the crackle in the time domain usually corresponds to signal waveform “up and down”. We found that three quarters of these peaks stretch from 0.36 to 1.09 ms. The crackle has also bigger short-time energy than the rest of the signal. An arbitrary signal needs to decompose an arbitrary signal, even with discontinuities or sharp peaks. This type of decomposition is called multiresolution

2. WAVELET TRANSFORM

We will present some necessary basics of wavelet transform, first on signals with continuous time, after that we will switch to the discrete-time wavelet processing.

...
analysis (MRA). An example of such analysis can be found in Figure 1.

2.2. Compact-supported wavelets

There were even found orthonormal bases such that their elements have compact support, i.e. there can be found a closed interval such that outside it the basis function is zero. An example of such functions can be the Daubechies-type wavelets (1).

The compactness of the wavelet’s support plays an important role for our purposes of “decrackling”. The fact that a wavelet \( \psi \) vanishes outside a close interval means that every translation and dilation \( \psi_{j,k} \) contributes to the signal just locally.

2.3. Wavelets with vanishing moments

Vanishing moments is another important concept in wavelet signal processing. A wavelet is said to have \( k \) vanishing moments if

\[
\int_{-\infty}^{\infty} \psi(t) t^i \, dt = 0 \quad \text{for} \ i = 0, 1, \ldots, k - 1. \tag{2}
\]

Equation (2) can be interpreted as follows: if \( \psi \) has \( k \) vanishing moments, then every polynomial of order \( k - 1 \) or less can be represented as a linear combination of the scaling function. It means that in MRA of the polynomial-like signal all the coefficients representing the signal’s details will be zero. This is again an important feature for our purposes, because if there was a single, fast crackle in an audio signal, which we consider as locally polynomial, this would lead to non-zero detail coefficients right in place of the singularity.

2.4. Discrete-time wavelet analysis

In practical problems we most frequently work with discretized (sampled) signals of finite length. In this case we speak of the finite discrete wavelet transform (DTWT), which can be represented by an orthogonal matrix \( W \) of size \( n \times n \). Let \( x = [x_1, \ldots, x_n]^\top \) be a vector of length \( n \). Its wavelet transform is vector \( y = [y_1, \ldots, y_n]^\top \), obtained as \( y = Wx \). Due to the orthogonality of \( W \), the inverse wavelet transform is \( x = W^{-1}y = W^\top y \).

It is evident from the above text that the wavelet transform has an important property—linearity.

Instead of multiplying vectors \( x \) and \( y \) by orthogonal matrices \( W \) and \( W^\top \), respectively, more effective Mallat’s pyramid algorithm (4) is used for computing the transform. Each step of this algorithm corresponds to filtering a discrete series by specific low-pass and high-pass filters and then decimating the result. The coefficients from the low-pass branch are called “approximations” and those from the high-pass branch are called “details”. We can repeat this single transformation step with the approximations standing for the input signal. The number of repetitions is called transformation depth. Scheme of this algorithm is depicted in Figure 2.

This way the input is divided into a number of subbands. Figure 3 shows the idealized decomposition in frequency domain.

The algorithm of the inverse wavelet transform is similar: we pass through the decomposition “tree” in the opposite direction performing reverse operations.

Wavelets differ by the decomposing and reconstruction filters. The filters corresponding to the compact-supported wavelets are always FIR filters. The coefficients of the filters determine their frequency response and thus the quality of signal splitting into frequency subbands. Generally, the sharper slope is required, the more coefficient are needed.

3. PRINCIPLE OF THE WAVELET-TYPE PROCESSING OF CRACKLES

Our wavelet method starts from the assumption of additivity of the disturbing crackles. This means that the impulses are added to the signal which we desire to restore. Formally,

\[
y = x + p, \tag{3}\n\]

where \( x = [x_1, \ldots, x_n]^\top \) is the original music or speech signal without any undesirable artifacts, \( p = [p_1, \ldots, p_n]^\top \) is the random “crackling” signal, and \( y = [y_1, \ldots, y_n]^\top \) is the mixed signal, which we have observed.

Starting from this model, we can make following inference. As said above, the wavelet transform is linear, and thus it holds

\[
W y = Wx + Wp \tag{4}
\]

for the observed signal \( y \). Then for the original “clean” signal in the wavelet domain there must be \( Wx = Wy - Wp \). Applying the inverse transform via matrix \( W^{-1} \) we ideally obtain the desired signal without any corruption:

\[
x = W^{-1}(Wy - Wp). \tag{5}\n\]

Our method works on the principle formally stated in [5].
Figure 4: Multiresolution analysis of a signal with wavelet Daubechies of order 2. In the left column there are plots of contributions of single signal subspaces and in the right column there are their respective cumulative sums. It is clear that some subspaces contain more approximate view of the signal and others contain more details. The most up-right picture is the original input signal.

Figure 5: Restoring the recording via the wavelet algorithm. Daubechies of order 12 was used for the multiresolution analysis. The arrows indicate the detail wavelet coefficients that were set to zero. The output signal is the most upper one. The input signal was the same as the one in Figure 4.
We regard the detail wavelet coefficients in the close neighbourhood of a crackle as its transform, i.e. $W_p$. Setting them to zero (on principle we perform “anti-thresholding”) we locally suppress the contribution of the detail levels in the multiresolution analysis right in the place where the impulse was detected. Speaking of frequency domain, we locally and in gradual degree pass low frequencies in place of the impulse and suppress the high frequency components. The graduality is reached due to the fact that the support of the wavelet filters doubles in each multiresolution level. From the crackle position onwards, naturally, we again pass gradually more and more high frequencies. The described process lasts a bit longer than the crackle does, i.e. about 1 ms.

Figure 4 shows a successful application of the algorithm.

To be more precise, the algorithm consists of two parts, detection and elimination. Each part can use different wavelet. The main steps of the two parts are:

**detection**

1. signal transform with wavelet chosen for detection
2. passing through the detail levels and marking places suspected of impulse (based on an energetic criterion)
3. comparing detail coefficients belonging to these places with a properly set threshold
4. if the last step confirms there are detail coefficients above the threshold in the same place, the center of it becomes the center of the area to be modified. The width of this area depends on the amount of the coefficients above the threshold.

**elimination**

1. signal transform with wavelet chosen for elimination
2. detail coefficients belonging to the area specified above are set to zero, whereas in every subsequent decomposition level there are approximately half of the coefficients processed in the preceding level. This is due to the decimation step of the wavelet transform.
3. inverse wavelet transform

For better performance, the algorithm could be run recursively, i.e. multistage.

**4. FACTORS DETERMINING THE QUALITY OF RESTORATION**

In this Section, we discuss factors that affect the restoration quality.

The compactness of the wavelet support plays an important role. This is because the crackle is just a local artifact and can be expressed by only few wavelet coefficients which refer directly to the place where the impulse is situated. Thus, it is effective to use wavelets with compact support (Daubechies, Symlets etc.), which correspond to FIR filters.

Another important property of a wavelet is the above mentioned number of vanishing moments. This number is closely associated with the number of continuous derivatives (smoothness) and in the discrete-time processing, it is directly connected to the sharpness of the filters’ slopes. The more vanishing moments a wavelet has, the better is the signal frequency separation. Because for the purposes of de-crackling we don’t want the frequency bands to soak much into each other, it is better to use wavelets with more vanishing moments, i.e. wavelet filters of a bigger order. For example, see Figures 4 and 5—comparison of the two multiresolution analyses demonstrate the difference in decompositions’ smoothness. We have found that Daubechies wavelets of order about 10 give sufficient frequency separation, in proportion to the computational complexity.

The last factor affecting the restoration quality is the transform depth. By selecting improper transform depth, we could make an insufficient number of frequency subbands and thus not separate the crackle from the rest of the signal. For audio signals sampled at 44.1 kHz we found that choosing depth 5 or 6 is sufficient, i.e. it is adequate to process 5 or 6 sets of detail coefficients, which means we leave frequencies below 800 Hz intact.

**5. PERFORMANCE OF THE METHOD**

In the core of wavelet transform stands correlation of the signal with the dilated and translated wavelets. Due to the fact that the wavelets are of strongly oscillatory character, our method turned out to be successful for suppressing crackles of this character – attenuated oversights up & down. However, there also often occur crackles that do not observe this character, and for these, the algorithm’s performance is naturally worse.

**6. CONCLUSION**

In the paper, a new method of vinyl record restoration was introduced. The method is based on the wavelet-type signal analysis and forms an alternative to the commonly used methods. The crackling is suppressed via local processing of the so-called detail wavelet coefficients. There is also a discussion what factors determine the quality of the restoration process.

**7. ACKNOWLEDGEMENTS**

The paper was supported by the Grant Agency of Czech Republic – Czech Science Foundation, project No. 102/04/1097, and by the project COST No. OC277.

**8. REFERENCES**

SPECTRAL DELAYS WITH FREQUENCY DOMAIN PROCESSING

Dr. David Kim-Boyle

University of Maryland, Baltimore County, Dept. of Music, 1000 Hilltop Circle, Baltimore, Maryland, 21250 U.S.A.
kimboyle@umbc.edu

ABSTRACT

In this paper the author presents preliminary research undertaken on spectral delays using frequency domain processing. A Max/MSP patch is presented in which it is possible to delay individual bins of a Fourier transform and several musically interesting applications of the patch, including the ability to create distinct spatial images and spectral trajectories are outlined.

1. INTRODUCTION

Delaying individual FFT bins in a short-time Fourier transform, can create interesting musical effects that are unobtainable with more traditional types of delay techniques. By delaying select bins by a small time value on one channel of a stereo signal, for example, distinct spatial images for spectral bands can be realized which can take on even more musically interesting characteristics when the delay values for the bins are dynamically assigned or are determined through signal analysis.

While several commercially available software plug-ins allow one to realize spectral delays, the open-ended architecture of Cycling '74’s Max/MSP allows an implementation with greater levels of control and the ability to explore some unique musical applications, such as that briefly outlined above, which commercially available plug-ins do not facilitate [1].

2. FFT IMPLEMENTATION

The delay architecture employed in the Max/MSP patch is based on a model in which FFT frames are resynthesized from delayed FFT bins. The size of the delays, measured in integer multiples of the FFT length, are determined by indexing user-defined buffers which are updated at the signal level.

Figure 1 outlines the pfft~ subpatch which is used to transform the input signal. Windowing functions are automatically determined by the pfft~ object.

An FFT is performed on a signal with a Hanning window. Real and imaginary components are converted to magnitude and phase values before being written to two buffers via a pair of gate~ objects which open upon initiation of the delay process. A modulo on the index to these buffers and to the resynthesis abstractions is used to help conserve Max/MSP memory. An additional delay~ object is also used on the index to the resynthesis abstractions to provide delay headroom should the delay for a particular bin require indexing samples that have not yet been written.

The spectral delay process that constitutes the resynthesis part of the patch is contained within two abstractions, one for both the left and right channel. Due to limitations of the pfft~ object, each of these abstractions must be located at the same root level as the pfft~ sub-patch. Figure 2 shows the patch contained within each of these abstractions.

A signal is used to index a buffer which contains delays for each FFT bin. As mentioned, these delay values are integer multiples of the FFT length. The delay value is then subtracted from the current index to determine the sample number to read from the magnitude and phase buffers. A delay value of 3, for example, for bin #7 will mean that the magnitude and phase components of the resynthesized bin #7 will be read from bin #7 of the third previous FFT frame. While this is a crude way to realize these delays, it is computationally inexpensive and simple to implement.

As the delay values are integer multiples of the FFT length, the minimum delay time is defined by the FFT size. With a 2048-point FFT at a sampling rate of 44100Hz the minimum delay time is 46.44ms, a 1024-point FFT – 23.22ms, a 512-point FFT – 11.61 ms. While this makes it difficult to simulate interaural time differences between channels where the timing differences may be in

---

1 See Native Instruments’ “Spektral Delay” for example
the order of only a few milliseconds, it does nevertheless allow distinct spatial images to be realized through the precedence effect. This application will be expanded upon later.

Scaling functions, read from another user-defined buffer are also used to provide amplitude control over the frequency response of each abstraction. Like the delay buffers, these buffers can also be updated at the signal level. In addition, they can also be written to with values obtained from a separate FFT analysis of the input signal. This technique enables a degree of performance control over delay values and amplitude scaling which is particularly useful in interactive computer music applications.

Scaling functions, read from another user-defined buffer are also used to provide amplitude control over the frequency response of each abstraction. Like the delay buffers, these buffers can also be updated at the signal level. In addition, they can also be written to with values obtained from a separate FFT analysis of the input signal. This technique enables a degree of performance control over delay values and amplitude scaling which is particularly useful in interactive computer music applications.

3. CONTROL

Several waveform~ objects are used in the patch as basic controllers to determine FFT bin delays and values for amplitude scaling. While the waveform~ object is a somewhat unwieldy way to attribute the large amount of data required by the FFT, especially in the perceptually significant initial 25% or so of the bins, one of its more attractive features is that it provides an instantaneous method of writing to buffers, unlike other objects such as the multislider which requires additional levels of control.

In an attempt to facilitate greater control over the waveform~ object, several macro functions have been added. These include the ability to instantaneously copy data from one buffer to another, the ability to increment or decrement by a small amount the entire contents of the buffer and the ability to write a value to a specific range of bins.

The buffers indexed in the spectral delay abstractions can also be determined through signal analysis. The following Section of this paper will describe this application.

4. MUSICAL APPLICATIONS

The spectral delay patch allows several unique applications and musically interesting effects to be achieved. These include the following.

4.1. Signal analysis control

Through performing an FFT analysis of a control signal, which does not have to be the same signal as that processed, it is possible to establish correlations between the harmonic components of the signal and the corresponding delay times for the FFT bins. For example, strong harmonic components may produce long delay times for those corresponding bins while weak harmonic components may produce shorter delays. This implementation is presented in Figure 4.

Through a simple inverse function, it is also possible to map short delay times to weak harmonic components and longer delays to stronger components.

Other interesting results can also be obtained through gradually morphing from one set of delay values to another – for example from random, noise-like values to user-defined values.

4.2. Stereo spatial effects

By varying the delay times of one channel with respect to the other it is possible to create unusual spatial effects across certain spectral bands. For example, referring to Figure 5, if the delays for FFT bins 1-20 on the right channel are increased over time a gradual panning to the left for frequencies below around 860Hz, for a 1024-point FFT, will occur. Frequencies above 860Hz will remain spatially stable.
Figure 5: Spectral panning effects.

Unlike other types of spectral panning algorithms [2] that are based on the multiplication of a spectral band’s amplitude with a coefficient, the spatial images created through spectral delays are created by the precedence effect. As noted by Wallach, Newman and Rosenzweig in their seminal study of the effect [3] the ability to localize sound through the precedence effect is affected by the nature of the sound itself. Sharp, transient sounds cannot be spatialized with the spectral delay technique quite as successfully as sounds of a more continuous, complex nature.

4.3. Multichannel spatial effects

Working on the same principles as those involved in creating stereo effects, the addition of two or more spectral delay abstractions can allow spectral panning effects to take place in more than two channels.

By cascading delays between channels spectral bands can be made to move in circular motions around the listener, see Figure 6.

Figure 6: Circular spatial movement.

4.4. Spectral reverberation

By chaining delay abstractions together, a primitive type of spectral reverb can be created. With each abstraction simulating the effects of early reflections, it is possible to attribute different reverberation characteristics across the frequency spectrum. Striking effects can be created when these “reflections” are then sent to a series of all-pass filters which simulate a reverberant tail.

5. FUTURE WORK

The author is continuing to explore more refined methods of signal control and line message like control of delay times. Various methods of including spectral feedback within the patch are also being explored. Of particular interest as well is an exploration of whether it is possible to integrate head related transfer functions in order to simulate spectral movement that gives the spatial illusion of height.

6. ACKNOWLEDGEMENTS

The author would like to extend his thanks to Cort Lippe for his many helpful suggestions during research on this project.

7. REFERENCES


EXPERIMENTAL WEIGHTING METHOD FOR SAMPLE-BY-SAMPLE UPDATE OF WARPED AR-MODEL

Kari Roth
University of Turku / Department of Physics
Laboratory of Optics and Spectroscopy
FIN-20014 Turun Yliopisto, Finland
kari.roth@utu.fi

Ismo Kauppinen
Noveltech Solutions Ltd
Tykistökatu 4
FIN-20520 Turku, Finland
ismo@noveltechsolutions.com

ABSTRACT
Auto-regressive (AR) modeling is a powerful tool having many applications in audio signal processing. The modeling procedure can be focused to low or high frequency range using frequency warping. Conventionally the AR-modeling procedure is accomplished with frame-by-frame processing which introduces latency. As with any frame-by-frame algorithm full frame has to be available for the algorithm before any output can be produced. This latency makes AR-modeling more or less unusable in real-time sound effects especially when long frame lengths are required. In this paper we introduce an exponential weighting (EW) method for sample-by-sample update of the warped AR-model. This method reduces the latency down to the order of the AR-model.

1. INTRODUCTION
An auto-regressive model [1] assumes that a signal sample $y_n$ can be predicted as a linear combination of previous samples and is defined by

$$ y_n = - \sum_{m=1}^{p} a_m y_{n-m} + e_n, \quad (1) $$

where $p$ is the order of the model, $a_m$ are the model coefficients, and $e_n$ is the prediction error (i.e. residual). When signal samples are known the prediction error is given by

$$ e_n = y_n + \sum_{m=1}^{p} a_m y_{n-m}, \quad (2) $$

The model coefficients are chosen to minimize the prediction error usually in a least squares sense. There exist several algorithms for calculating the model coefficients (see e.g. [2]). In this paper Burg’s method [3] is used and a time-domain exponential weighting (EW) method for sample-by-sample update of the model coefficients is introduced.

Frequency warping [4] can be used to emphasize the modeling procedure to the low or high frequency range of the signal. This is highly useful in audio signal processing since audio signals usually have most of their energy concentrated to frequencies below 10 kHz. Frequency warping is achieved via bilinear mapping in the z-plane

$$ z \rightarrow D(z) = z^{-1} - \lambda \frac{1}{1 - \lambda z^{-1}}, \quad (3) $$

where $\lambda$ is the warping parameter. This implies that frequency warping can be incorporated into existing methods simply by replacing unit delays $z^{-1}$ by allpass filters $D(z)$.

2. WARPED BURG’S METHOD
Equation (2) represents a finite impulse response (FIR) filter and therefore can also be realized by a lattice structure depicted in Figure 1. The reflection coefficients are chosen to minimize the forward and backward prediction errors $f_n^{(m)}$ and $b_n^{(m)}$ at each stage independently. Solution to this problem is given as

$$ k_m = -\frac{2(\langle f_n^{(m-1)} \rangle \lambda b_n^{(m-1)})}{\langle f_n^{(m-1)} \rangle^{2} + \langle b_n^{(m-1)} \rangle^{2}}, \quad (4) $$

where $\langle \rangle$ denotes expectation value. This reflection coefficient is then used to obtain prediction errors of stage $m$

$$ f_n^{(m)} = f_n^{(m-1)} + k_m b_n^{(m-1)}, \quad b_n^{(m)} = b_n^{(m-1)} + k_m f_n^{(m-1)}. \quad (5) $$

Initially $f_n^{(0)} = b_n^{(0)} = y_n$. If the actual model coefficients are needed they can be obtained via Levinson-Durbin recursion [5, 6]. When frequency warping is used in Burg’s method the unit delays of the lattice filter are replaced by warped allpass filters $D(z)$. The resulting structure is shown in Figure 2. Frequency warping causes only minor modifications to the modeling equations. The warped reflection coefficient is given by

$$ \tilde{k}_m = -\frac{2(\langle f_n^{(m-1)} \rangle \lambda \tilde{b}_n^{(m-1)})}{\langle f_n^{(m-1)} \rangle^{2} + \langle \tilde{b}_n^{(m-1)} \rangle^{2}}, \quad (6) $$

where $m = 1, 2, \ldots, p$ and

$$ \tilde{b}_n^{(m)} = b_n^{(m-1)} - \lambda [\tilde{b}_n^{(m-1)} - \tilde{b}_n^{(m-1)}], \quad (7) $$

which is the warped backward prediction error. The reflection equations are respectively transformed to

$$ f_n^{(m)} = f_n^{(m-1)} + \tilde{k}_m \tilde{b}_n^{(m)}, \quad b_n^{(m)} = b_n^{(m-1)} + \tilde{k}_m f_n^{(m-1)}. \quad (8) $$

For a more detailed description see [7].

3. SAMPLE-BY-SAMPLE UPDATE
When using frame-by-frame AR-modeling the latency of the modeling phase is at least equal to the frame size used. This causes a disturbing delay if AR-modeling is used e.g. in a real-time sound
effect. By using sample-by-sample update this latency can be reduced to be equal to the order of the model.

Previously, a method has been developed for obtaining sample-by-sample update of the AR-model coefficients in a non-warped case, namely the gradient adaptive lattice method (GAL) [8]. This method can also be further developed to the frequency warped case. However, GAL does not ensure the stability of the model, i.e. \( k_m \) are not guaranteed to have absolute value less than unity although usually this is the case.

### 3.1. Gradient adaptive lattice method

GAL method uses the gradient of the prediction error energy

\[
\hat{J}_n^{(m)} = \left[ f_n^{(m)} \right]^2 + \left[ b_n^{(m)} \right]^2 .
\]  

(9)

The updated reflection coefficient is

\[
k_m(n) = k_m(n - 1) + \Delta k_m(n),
\]  

(10)

where \( \Delta k_m(n) \) can be estimated by using steepest descent approach which gives

\[
\Delta k_m(n) = -\mu_m \frac{\partial \hat{J}_n^{(m)}}{\partial k_m},
\]  

(11)

where \( \mu_m \) is a step size parameter and

\[
\frac{\partial \hat{J}_n^{(m)}}{\partial k_m} = 2 \left[ f_n^{(m)} b_{n-1}^{(m-1)} + b_n^{(m)} f_{n-1}^{(m-1)} \right].
\]  

(12)

The step size parameter \( \mu_m \) can be adapted to the prediction error energy estimate to have faster adaptation for fast changes in signal. This is done by choosing

\[
\mu_m = \frac{1 - \beta}{2 \hat{D}_n^{(m)}}
\]  

(13)

where \( \beta \) is a smoothing parameter and \( \hat{D}_n^{(m)} \) is the estimate of the prediction error energy after lattice stage \( m \) having value

\[
\hat{D}_n^{(m)} = \beta \hat{D}_{n-1}^{(m)} + (1 - \beta) \left\{ \left[ f_n^{(m-1)} \right]^2 + \left[ b_n^{(m-1)} \right]^2 \right\} .
\]  

(14)

### 3.2. Exponential weighting method

The proposed EW method for sample-by-sample update for the model parameters is to use time-domain exponential weighting to calculate the expectation values in equation (6). This can be achieved by

\[
(f_n^{(m)} f_{n}^{(m)}) \approx f_n^{(m)} f_{n}^{(m)} = \alpha f_{n-1}^{(m)} f_{n}^{(m)}
\]

\[
(b_n^{(m)} b_{n}^{(m)}) \approx b_n^{(m)} b_{n}^{(m)} = \alpha b_{n-1}^{(m)} b_{n}^{(m)}
\]

\[
(f_n^{(m)} b_{n}^{(m)}) \approx f_n^{(m)} b_{n}^{(m)} = \alpha f_{n-1}^{(m)} b_{n}^{(m)}
\]

\[
(b_n^{(m)} b_{n}^{(m)}) \approx b_n^{(m)} b_{n}^{(m)} = \alpha b_{n-1}^{(m)} b_{n}^{(m)}
\]

(15)

where \( \alpha \) is a smoothing parameter. The higher the value of \( \alpha \) is the more weight is given to the past values and the longer is the time required for the model to adapt to changes in the source. The time constant of the adaptation is

\[
\tau = \frac{1 - \alpha}{\alpha} \Delta t.
\]  

(16)
Table 1: Comparison of the operations needed per sample in each lattice stage for gradient adaptive lattice method, proposed EW method and EW with warping

<table>
<thead>
<tr>
<th></th>
<th>ADD</th>
<th>MULT</th>
<th>DIV</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAL</td>
<td>8</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>EW</td>
<td>7</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>EW + Warping</td>
<td>9</td>
<td>12</td>
<td>1</td>
</tr>
</tbody>
</table>

where $\Delta t$ is the sampling interval. Now the reflection coefficient $\tilde{k}_m$ can be calculated from

$$\tilde{k}_m(n) = -\frac{2X_{n}^{(m)}}{F_{n}^{(m)} + B_{n}^{(m)}}$$  \hspace{1cm} (17)

### 3.3. Computational costs

In Table 1 the computational costs of GAL method, the proposed EW method, and EW method with warping are presented. The computational costs of GAL and EW methods are nearly equal. The computational costs for GAL includes step size adaptation.

No quantitative comparison between frame-by-frame methods and sample-by-sample methods are shown here. Usually block processing methods have to use overlap-and-add which causes each sample to be processed at least twice making the computational costs greater than either of the sample-by-sample methods discussed in this paper.

### 3.4. Modeling performance

In Figures 3 and 4 the modeling performances of the EW method, GAL method, and the frame-by-frame method are compared. The modeled signal was a guitar tone sampled at 44.1 kHz. Model order $p$ was 6. The figures represent the tracks of reflection coefficients $k_2$, $k_3$, and $k_4$ as a function of time. The smoothing parameter $\alpha$ had value of 0.9999 in both experiments.

Figure 3 compares the EW method and the frame-by-frame method. Value $\lambda = 0.2$ was used for warping parameter. With frame-by-frame method frame size 1000 was used with 90% overlap. At each position the center of the frame used in frame-by-frame method is aligned with the sample that is fed into the EW algorithm. Figure 4 compares the EW method and the GAL method. In this case no warping was used. From these experiments it is easy to notice that the model obtained with the EW method is consistent with the model obtained using the other methods.

### 4. APPLICATIONS

AR-modeling has previously been successfully applied to e.g. partial tracking [9], transient sound analysis [10], click detection [11], and vocoding. With the proposed method frequency warping can be incorporated into these effects without losing real-time nature due to the long latency introduced by block processing.

In AR-model based click detection using frequency warping with negative warping factor enhances the performance [12]. In Figure 5 the proposed method is applied to click detection. The signal sample consist of 50000 samples of orchestral music sampled at 44.1 kHz and corrupted by crackle. The signal is fed through the algorithm and the resulting residual is shown in the lower figure. Model order $p = 6$, warping factor $\lambda = -0.5$, and smoothing parameter value $\alpha = 0.9999$ was used. In the original signal the degradations are well hidden. The modeling process enhances the disturbances and allows a restoration algorithm to focus only on the damaged sections that can be identified e.g. by simple thresholding.

### 5. CONCLUSIONS

In this paper we have adressed the problem of reducing latency in the AR-modeling procedure where frame-by-frame methods are conventionally used. We have introduced an exponential weighting (EW) method for sample-by-sample update of the AR-model including frequency warping. We have shown the calculation efficiency of the method and compared it to the existing gradient adaptive lattice (GAL) in the non-warped case. The latency is reduced to the order of the model in the proposed method.

### 6. REFERENCES

Figure 4: Tracks of the reflection coefficients using gradient adaptive lattice method (thick gray line) and the exponential weighting method (thin black line).


EMULATING ROUGH AND GROWL VOICE IN SPECTRAL DOMAIN

Alex Loscos, Jordi Bonada

Music Technology Group of the Institut Universitari Audiovisual Universitat Pompeu Fabra, Barcelona, Spain
{alex.loscos|jordi.bonada}@iua.upf.es

ABSTRACT
This paper presents a new approach on transforming a modal voice into a rough or growl voice. The goal of such transformations is to be able to enhance voice expressiveness in singing voice productions. Both techniques work with spectral models and are based on adding sub-harmonics in frequency domain to the original input voice spectrum.

1. INTRODUCTION

Vocal disorders have been largely studied as pathology in the field of phoniatry. However, in the context of popular singing, vocal disorders not always come from pathologies but sometimes healthy voices use it as an expressive recourse. The goal of the algorithms presented here is to achieve natural rough and growl effects in order to enhance singing voice in music productions. Decide where and how much effect to apply is a critic issue that will not be discussed here.

Unlike many of the studies concerning vocal disorders, the algorithms presented in this paper arise from spectral models and work with frequency domain techniques instead of working with physical models and use time domain techniques. More concretely, both rough and growl algorithms have been implemented on top of phase-locked vocoder techniques [1], [2].

1.1. Voice production mechanism

The first step in the voice production cycle takes place when air enters the lungs via the normal breathing mechanism. When this air in the lungs is pushed out by muscle force it excites the vocal mechanism through the bronchi and trachea. When the vocal folds are tensed, the airflow causes them to vibrate (Figure 1), producing voiced speech sound. In this case the airflow is chopped by the vocal folds into quasi-periodic pulses.

When the vocal folds are relaxed, in order to produce a noise, the airflow either must pass through a constriction in the vocal tract and thereby become turbulent, producing so-called unvoiced sound, or it can build up pressure behind a point of total closure (e.g. the lips), and when the closure is opened, the pressure is suddenly released, causing a brief transient sound.

The most common approach to model the voice production system is based on a source-filter decomposition assumption [4]. The speech processing discipline has been using widely this source-filter model in which two types of source signals (noise for unvoiced and a periodic pulse-train for voiced) are filtered by a dynamic filter that emulates the vocal tract (supra-laryngeal filter).

Figure 1: Video capture of a female larynx taken from [3] with permission of author. Vocal folds and aryepiglottic folds can be observed in the center and lower part of the laryngoscopic view respectively.

Note that from the voice-source model point of view, the vocal disorders that are being considered here come basically from the aperiodicities of the voiced excitation, that is, from the periodic train of pulse-like waveforms that corresponds to the voiced glottal excitation.

2. ROUGHNESS

Roughness in voice can come from different pathologies such as biphonia, or diplophaonia, and can combine with many other voice tags such as hoarse or creaky [5]. In this paper we will not stick to the rigorous rough voice definition but we will refer to rough voice as the one due to cycle to cycle variations of the fundamental frequency (jitter), and the period amplitude (shimmer).

Being aware of such nomenclature, we can say the most common techniques used to synthesize rough voices work with the source - filter model and reproduce the jitter and shimmer aperiodicities in time domain [6]. These aperiodicities can be applied to the voiced pulse-train excitation by taking real patterns that have been extracted from rough voices recordings or by using statistical models [7].

2.1. The roughness algorithm

The main idea underneath our algorithm for turning a normal phonation voice into a rough voice is to take the original input signal, transpose it down a certain number of octaves, take then the transposed signal and shift it and overlap it with a certain amount of randomness (Figure 2) to re-synthesize the original voice with its new rough character.
The shift applied to each of the $N$ shifted versions of the transposed signal is:

$$\text{Shift}_i = i \cdot T_0 + X_i \quad \text{for } i=0,1,\ldots,N-1$$  

where $T_0$ is the original period and $X_i$ is a zero mean random variable. These differently shifted $N$ versions of the transposed signal are then scaled by a unity mean random variable $Y_i$ and finally overlapped.

In order to take in what is the outcome of such system, Figure 3 shows a figurative time domain representation of all steps for $N=2$.

Figure 3 does not illustrate real results since our frame-based implementation does not take into account the relationship between the frame rate and the input period, nor the analyzed frame history. Also, the real implementation changes $X$ and $Y$ stochastic variables values at every frame time. This scenario does not allow generating patterns such as the ones represented in the figure. This is also the reason why even though theoretically, with such algorithm, the higher $N$ is the more control we may have over isolated periods of the signal, the implemented system does not fulfil this rule.

2.2. The simplified roughness implementation

The reason for implementing a very simplified version of the algorithm presented in previous Section is that the effect had to fit in a real time voice transformation environment. All performed simplifications are described next in this Section.

The one octave down transposition is accomplished by adding pure sinusoids to the spectrum in the sub-harmonic frequencies. More precisely, adding the main lobe bins of the analysis window and taking the phase from the closest harmonic peak and shifting it with the corresponding offset as in [8]:

$$\Delta \phi = 2\pi f_{sa} \left( \frac{f_{sa}}{f_h} - 1 \right) \Delta t$$  

where $f_{sa}$ is the sub-harmonic frequency to fill, $f_h$ is the frequency of the closest harmonic peak, and $\Delta t$ is the frame time.

Since the greater $N$ is, the more computationally expensive the effect is, we have taken the minimum $N$ value, $N=2$.

The jitter and shimmer stochastic variables of the first channel are set to its mean value $X_0=0$ and $Y_0=1$. Thus, the output of this first channel will be a one octave down transposition of the original input. This is a not very risky simplification for $N=2$ since it can be seen as moving $X_0$ and $Y_0$ randomness to $X_1$ and $Y_1$. The stochastic variables $X_1$ and $Y_1$ are defined to have a normal distribution with variances 3% of the input signal period, and 3 dBs respectively.
The random scaling due to $Y_1$, as well as the random time shift due to $T_0 + X_1$ are only applied to the sub-harmonics. The only reason for doing such oversimplification is to reduce the computational cost of the algorithm since with this only half of the peaks to which the random variables should be computed and applied are actually processed. The time shift is applied in frequency domain by adding the corresponding constant slope phase offset to the phase of the sub-harmonics spectrum as represented in the spectrum of Figure 4c.

Only sub-harmonics inside the $F_0$-8000 Hz band are added to the spectrum. Upper sub-harmonics are not significantly relevant in terms of acoustic perception to reproduce the rough effect, and the first sub-harmonic (placed at $0.5F_0$) is assumed to be, based on the observations, always masked by the amplitude of the fundamental peak.

3. GROWL

Singers in jazz, blues, pop and other music styles often use the growl phonation as an expressive accent. Perceptually, growl voices are close to other dysphonic voices such as hoarse or creaky, however, unlike these others, growl is always a vocal effect and not a permanent vocal disorder.

According to [9] growl comes from simultaneous vibrations of the vocal folds and supra glottal structures of the larynx. The vocals folds vibrate half periodically to the aryepiglottic fold vibration generating sub-harmonics.

The growl algorithm presented here adds these sub-harmonics in frequency domain to the original input voice spectrum to try to emulate the growl phonation. These sub-harmonics follow certain magnitude and phase patterns that have been extracted from the spectral analysis and observation of real growl voice recordings.

3.1. The growl observation

The behaviour of the growl sub-harmonics in terms of magnitude and phase vary quite a lot from one voice to another, from one pitch to another, from one phrase to another, etcetera. However certain patterns appear quite frequently. These patterns, which are explained next, are the ones that the growl effect applies.

If a growl utterance is observed in time domain, it is most of the times easy to recognize which is the real period of the signal and which is the macro period due to growling as it is in Figure 5. In the observations made growl phonation appeared to have from two to five sub-harmonics. In Figure 5 example, the spectrum presents three sub-harmonics placed at $F_r(m+k/4)$ (for $m=0$..number of harmonics, and $k=1..3$). Thus, three inner periods can be distinguished in between a growl macro period.

Regarding the magnitudes, in the band that goes from the fundamental frequency up to approximately 1500 Hz, the sub-harmonic peaks are commonly located below the spectral envelope (defined by the harmonic peaks). In this band, the closer the sub-harmonic is to the nearest harmonic, the higher its magnitude is. In the upper band, from approximately 1500 Hz to half the sampling rate, sub-harmonics go along with the harmonic spectral shape.

Regarding the phase, for a growl utterance with $N$ sub-harmonics, the typical behaviour of the phases of the sub-harmonics is to get approximately aligned with the phase of the left harmonic peak every $N+1$ periods as illustrated in Figure 6.

$$\varphi_{sh}^k = \varphi^i + \frac{2\pi}{N+1} (k+1)p \quad , \text{ for } k=0,1,2 \text{ and } p=0,1,2,3 \quad (3)$$

being $p$ the inner period index ($p=0$ for Figure 6a and $p=3$ for Figure 6d), $k$ the sub-harmonic peak index in between consecutive harmonic peaks, and $N$ the number of sub-harmonics.
3.2. The growl effect implementation

Based on the most frequently observed growl spectral symptoms, the implemented system fills the original spectrum with sub-harmonics. However, since growl is not a permanent disorder, the effect can not be applied all along the performance. For this reason the implementation includes an automatic growl control (as shown in Figure 7) by which we determine how much of the effect has to be applied at each time depending of the input singing voice. This control is mainly based on the first derivatives of the fundamental frequency and energy and has control on how many sub-harmonics have to be added, and their phase and magnitude patterns (including the gain of the sub-harmonics). With such implementation, the system is able to reproduce growl sub-period amplitude patterns as the one shown in Figure 8. In the waveform view of the transformed voice we can observe how each of the four periods of the growl macro period is set to have different amplitude. This amplitude modification is achieved by applying phase alignment patterns extracted from real growl analysis to the sub-harmonics.

4. CONCLUSIONS AND FURTHER WORK

The rough and growl algorithms presented in this paper have proven to be suitable in changing the voice character. However, the naturalness of the effect is highly dependent on input voice.

For different types of voice, different tessitura, different expressions, etc. different values of the transformation parameters are required. In that sense, a dynamic automatic control over these parameters has to be found. In the growl effect, this control would have to combine and work together with the current automatic growl control.

In the growl effect patterns extracted from real growl recordings are roughly reproduced in synthesis. This means the period to period amplitude envelope inside a growl macro-period is not only included in the phase alignment of the sub-harmonics but also in the sub-harmonics amplitudes. However, it is a tedious job to find the sub-harmonic amplitudes and phase alignment required for a certain made-up amplitude envelope. It is also remarkable no control over the jitter is available with the current growl implementation.

Concerning the rough effect, two interesting directions come up from the current implementation. First, perform a study of the system without any of the simplifications implemented. Second, take into account the frame rate / input period relationship and the analyzed frame history so that the system could follow the period. In that situation, increasing N would really improve the resolution of the algorithm. This could be considered as going towards a fusion of both techniques in which we would have control over the period to period jitter and the shimmer.

5. REFERENCES

AIDE, A NEW DIGITAL AUDIO EFFECTS DEVELOPMENT ENVIRONMENT

Victor Lazzarini  
Music Technology Laboratory  
NUI Maynooth, Ireland  
victor.lazzarini@may.ie

Rory Walsh  
13 Rue Louis de Cardonnel  
Grenoble, France  
rorywalsh@ear.ie

ABSTRACT

This paper describes a new rapid development environment for digital audio applications and computer instruments, AIDE (Audio Instrument Development Environment). The system is designed to help users build signal processing applications for use in music, multimedia and sound design. Based on a graphical patching principle, this system generates software using the V and Sound Object libraries. These provide the graphical interface/application framework and sound processing elements, respectively, for stand-alone programmes generated by AIDE. It is envisaged that the system will also generate application components in addition to stand-alone programs. The paper outlines in some detail the elements involved in the software. It discusses how the system is aimed at different types of users with different levels of interaction. The paper concludes with an overview of the typical application development cycle using the system.

1. INTRODUCTION

AIDE is a new software system for computer instrument and stand-alone audio software development in C++. It is intended for use by musicians, composers and programmers, as it provides different levels of user interaction. For those with no prior experience of programming, the software can be used as a pedagogical system, in which users can learn the intricacies of programming. Application development can be done through a user-friendly graphical approach. The system also provides lower-level interaction for more experienced programmers.

The first version of the system provides support for the development of stand-alone applications, but it is hoped that in the future users will also be able to develop software components. These would comprise Pure Data (PD) [1], MaxMSP [2] classes, as well as VST, LADSPA and DirectX plugins. AIDE is designed to assist all aspects of the development of practical tools for music signal processing.

2. SYSTEM FEATURES

The most important feature of AIDE is its versatility. As a basic tool it provides users with a way to create their own stand-alone audio applications. It is foreseen that users will employ the system to create task specific applications, such as computer instruments for works of electroacoustic music, research tools for audio processing, and educational tools for teaching computer music. The benefit of this is that the applications will be small in size and very easy to set up as the generated executable files do not need any third party libraries to run. The ‘ease of setup’ factor is very useful to musicians performing computer music. No more will they have to preinstall complete software to run their patches, the only software they will have to install is the executable which runs the process.

Work is now underway to allow for different types of target applications, not merely standalone executables. The generation of PD/MaxMSP externals is possible through the employment of Flext [3], a C++ layer for Max/MSP and PD externals. However, for now external objects are limited to a very basic interface. LADSPA, VST and DirectX plugins are other varieties of output that are planned for the future. This would provide users with a means to creating their own custom plugins, thus stretching the realms of user interaction found in many of today’s audio processing systems.

While the above examples illustrate how AIDE proves a very useful tool in the development of audio processing software, it must not be overlooked that it also provides users with access to low level signal processing operations. This type of access is often not available with other audio systems, in particular commercially available systems. While these systems may provide users with pre-built effects such as reverbs, delays, and filters, they do not allow the user to take these effects apart and rebuild them to their own specifications.

3. TECHNICAL ASPECTS

AIDE is being developed for Windows using the Borland© C++ Builder. It is hoped that, with the release of Borland’s Kylix© [4] a new Linux C++ development environment, a version for Linux will also be made available. The system works by creating the appropriate C++ code whenever the user places a new SndObj object or graphical component in the main patcher window. AIDE makes use of the freeware multi platform GNU gcc compiler to compile the stand-alone executables, therefore even though an application is created in Windows, the multi-platform makefile may be run on any Linux PC or Mac OSX providing that the correct libraries are installed.

3.1. The V library

The V GUI Library [5] is used to create the GUI interface for standalone applications. V is a C++ Graphical User Interface Framework designed to provide an easy to use system for building GUI applications. The framework is small, and provides all the tools one would need for building intuitive graphical user interfaces. V has also been designed to be portable and, currently, versions for Linux, Microsoft Windows, and OS/2 exist. The
V framework is freely available for use by anyone under the terms of the GNU Library General Public License.

3.2. The SndObj library

AIDE uses the Sound Object Library [6] to provide the audio processing operations for the generated software. The generated code is fully portable across Windows, Linux/Unix (with OSS), Irix and MacOS X. The SndObj library provides around 100 classes that can be used for time- and frequency-domain signal processing, as well as sound and MIDI input/output. With its SndThread class, it can also manage audio processing threads, something particularly important for this software.

The library is normally used by this software as a toolkit, but also serves as a framework for class development. The software, with its code generation capabilities, includes the possibility of user-defined custom SndObj classes. It is expected that AIDE will be used in the further development of the library.

4. LEVELS OF INTERACTION

The system was designed to be used at three different levels of interaction:

- **Introductory**: The first level is directed at the novice user, who has no prior knowledge of audio processing systems yet wants to learn about the many ways to manipulate and transform sounds digitally. By using AIDE they can start learning from the very beginning. By following interactive tutorials in which they follow flow charts of classic processing techniques they will begin to see the different ways in which the classic techniques can be realised using the system. They may then start to create their own applications by recreating these flow charts on screen in a modular based fashion to create new applications in which these techniques are embedded.

- **Advanced**: The second level is directed at users with prior knowledge of processing techniques. They may use AIDE to create and develop new techniques. As the system comes with a complete range of audio processing tools in the form of SndObj classes, it should make it possible to create or recreate a huge array of processing techniques. The user may then embed these new techniques into standalone software or plugins.

- **Developer**: The third level of interaction aims to support experienced users in processing techniques and in C++ programming. For them, the software will help them to realise more complex audio applications and enable fast development with code re-use. Skeleton applications which include just the GUI elements such as Menu items, buttons, scrollbars, etc, can be augmented by the user’s own C++ code. This can be edited within the application in a code text window. After the code is modified, it can be compiled as normal from within the Application Builder or from the command line. In this way users can benefit from code re-use, esp. when developing a graphical user interface for their application. This particular level of interaction provides experienced users with a great tool for researching new custom built audio processing techniques.

5. PROGRAM LAYOUT

AIDE incorporates a modular design ‘flow chart’ system, similar to many current audio processing systems, such as PD/MaxMSP, OSW [7] and CPS [8]. This system was chosen above others because of its clarity and simplicity of use. In addition, it is hoped that users of Pd/MaxMSP will not have many problems in adapting to AIDE as it incorporates the same ‘patcher’ paradigm. So as in many other audio processing systems the user simply drags and drops classes into the main patcher window to instantiate objects. These are then connected together in typical patch chord fashion. When the user starts a session with the AIDE they are presented with three main windows:

1) The Main Patcher Window: this contains the graphical representation of the audio processing flow.
2) The Source Code Editor: this contains the entire source for the project.
3) The GUI Layout, Data Structure and compiler output window: this contains the GUI designer where the user implements the graphical user interface for their application. It also contains the processing order of the classes on screen and also informs the users of which patch chord goes where. Finally it contains the compiler output to inform users of whether or not their projects compiled correctly.

6. IMPLEMENTING A SIMPLE APPLICATION

To illustrate a typical session it is best to implement a simple application, such as a Schroeder Reverb Unit. The user, upon opening the main program interface, begins by simply inserting the needed sound objects into the ‘patcher’ window. These objects are connected together with patch chords to determine the data flow of the application.

Figure 2 shows an implementation of our simple ‘Schroeder reverb’ unit. As you can see from the diagram the flow chart is quite simple to follow. There are four parallel comb filters, each with a constant for the gain and delay time parameters connected to two cascading allpass filters. The allpass filters also have two parameters, again for the gain control and the delay time. The SndRTIO objects handle real time audio IO and the SndlIn object
captures the audio stream and makes it accessible to other SndObjs. Each of the sound objects are named in the same way as they appear in the SndObj library so as not to confuse a user who wishes to make the move from using the AIDE to using the SndObj Library in another programming environment outside of AIDE.

After the signal processing patch is created, a Graphical User Interface for the application can be generated using the GUI designer. Again by a simple process of drag and drop a nice user friendly interface can be created for the application. In the above patch, constants have been used to provide parameters for the different class members. However if the 4 comb filter gains were made variable, they could be linked to GUI components, such as a scroll bar, as in Figure 2.

Each time a user places, deletes, or moves a graphical element, be it a sound object or a GUI object, C++ code is generated by AIDE to correspond with each new object. This C++ code is visible to the user through the source code editor giving them the opportunity to take a glimpse at how the code is structured and placed together. The automatically generated source code is put together in the most user friendly fashion possible, to give the user a clear idea of how the application works.

After completing the graphic interface, the user can proceed to the compilation of the new software. When the user compiles the project AIDE runs the project `makefile`. If there are errors in the compilation of the application, the user may view them through the compiler output window which is integrated into the software main interface. Providing that the source code has no user errors AIDE will compile everything into a standalone executable. As stated earlier the source code for each project and makefiles are cross-platform and can be compiled on many different platforms.

7. AIDE IN THE CONTEXT OF CURRENT AUDIO PROCESSING SYSTEMS

While AIDE obviously contains a certain likeness, in terms of user interaction, with other software systems such as PD, Max/MSP and OSW, the similarities stop there. There is no audio engine in AIDE, thus there is no run-time environment for the execution of the newly created programs. Instead, each patch must be carefully planned and implemented by the user before compilation in order for it to run and compile correctly. Through this approach users will find it easier to develop highly structured applications. None of the above systems are designed to allow users access to the actual processing code. In some ways this is where AIDE is most innovative. By offering access to the C++ code, users are being encouraged to look at the algorithms and classes for each of the SndObjs employed in their patch. With the freedom AIDE offers for exploration of processing classes, it is not difficult to see how the system can become a useful tool in the development and research of audio instrument design.

Another original aspect of this system is that it uses a platform-independent GUI C++ library for application development. While other software does offer users the options of creating GUI interfaces to control processing parameters, these are in general intricately connected to the main application, which sometimes is not desirable. Using a C++ application framework is possibly a more flexible and open way to provide GUI support for the generated software.

Among the comparable systems, CPS, a new realtime processing environment, appears to share some of the aims of AIDE. In this system, users can translate their graphical patches into C++ or Java source code. This source code can then be used together with a supplied SDK to create standalone applications. Unlike AIDE, however, CPS is not a development environment as such, lacking the compiler, text-editing and application-building support. CPS seems to be primarily a synthesis and processing system, with some secondary application development features. In addition, as opposed to AIDE, it is not free software.

Finally, it is also important to point out that, while all other systems discussed here provide their own basic signal processing and GUI functionality, AIDE depends on external libraries. This model allows for a more flexible evolution of the whole system. Separate upgrades of the libraries will add new functionality to AIDE, without the need for new versions of the program itself.
8. CONCLUSION

AIDE has been used very successfully by the authors and others in beta-testing. It has been employed in the development of computer instruments for live electroacoustic music, for sound processing tutorial materials and in general music applications. Further work will possibly involve also the addition of different types of output (plugins/components), as well as preview capabilities, whereby the sound processing operations can be tested prior to compilation. This would involve the design of a light-weight processing engine, based on the processing thread management services provided by the SndObj library. It is envisaged that, with the addition of these features, the system will become a comprehensive tool for audio application development. The beta-versions of the application, including source code and examples will soon be available on-line at the NUIM Music Technology Laboratory site: http://www.may.ie/academic/music/musictec.

9. ACKNOWLEDGEMENTS

The research project that led to the development of AIDE was carried out in the La Villa Media Institute, Grenoble, France, under the auspices of the French Ministry of Education. The authors would like to thank the Institute, its staff, and its Director, Bernard Cornu, for their support and encouragement.

10. REFERENCES

ADAM - A 64 CHANNEL GENERAL PURPOSE REALTIME AUDIO SIGNAL PROCESSOR

Dr. Robert Trausmuth, Michael Kollegger

IT Department
University of Applied Science Wiener Neustadt, Austria
geo@fhwn.ac.at

ABSTRACT

In this paper we introduce a 64 channel audio processing unit made in our department. The audio processor uses a 16 bit control unit (Infineon XC 167) with ethernet interface running a realtime operating system and two Analog Devices ADSP-TS101S high performance tigerSHARC DSPs for audio stream processing. The first project implemented on this equipment is a 64 channel in, 32 channel out audio mixer with a sampling frequency of 48 kHz (leaving another 32 channels for effect feedback loops) or 96 kHz, alternatively. The audio processor is fully remote controllable via TCP/IP.

1. INTRODUCTION

Teaching system programming is done best with a real world application. A sound system produces huge amounts of realtime data and the results can be heard immediately. So we started the 64 channel audio mixer 2 years ago from scratch. Every piece of hardware (except for the processor boards) was done in our labs. Protocols and system architectures had to be designed to meet the requirement of creating a fully remotely controllable audio mixer. The control is done via standard TCP/IP network, and the control console can either be a standard-mixer like console with faders and switches or a simple software application. Using a wireless LAN access point even gives the possibility of using handheld devices for stage personnel, allowing to adjust monitor channel settings directly at those devices with no need to contact the monitor sound engineer via walkie-talkie.

The mixer itself was built on a very general concept, leaving us the possibility of driving up to 256 output channels. For financial reasons we decided to implement only 32 output channels and use another 32 as feedback loops for effect calculations running on the second DSP.

Using ADAM means finally getting rid of multicore cables, since no audio data needs to leave the stage, although in the first implementation there is still the need of one audio line for the talkback monitor and one for the headset.

2. SYSTEM DESCRIPTION

The central player is a XILINX Spartan Ile 300 FPGA which handles all the data traffic and synchronizes the system components. It works with two different clock speeds to meet the needs of the CODECs and the DSP data ports. Communication with the XC 167 is done via dual port RAM to meet the different update speeds of slow control data and to avoid a third clock domain in the FPGA. Audio data is sent to and received from the CODECs in 32 bit words (24 bits significance) per channel at 48 kHz sampling rate.

The total amount of audio data handled by the FPGA is roughly 10 MB per second.

2.1. Data Flow Concepts

ADAM is contained in a 19 inch standard stage box. Communication with the outer world is done by Ethernet and TCP/IP protocols. Channel parameters like filter settings and fader controls are sent via TCP connection. VU meter data is provided by the stage-box via UDP broadcast telegrams. After implementing a realtime ethernet layer there will be 2 - 8 channels of realtime audio data for transmitting headset and talkback signals to the remote control. Another 8 channels will be available for transmitting output signals via standard WLAN equipment to distributed active speaker stations.

The first prototype also contains a touch panel directly on the stage box front side to check the function of the XC 167 module and to execute a variety of testing routines directly on the stage box.

2.2. Slow Control

The XC 167 handles all the slow control calculations. It is responsible for keeping up to 16 virtual audio mixer setups. Each setup can control 3 stereo main mix outputs with different delay times, 4 mono monitor channels and 8 stereo subgroups, which gives a total of 26 output channels. Of course not all of the channels have to be mapped to a real output channel.

Every time a parameter changes, the mixer matrix is recalculated and the new values are stored in the dual port RAM. The complete set of parameters is transferred to the DSP every 50 ms. At
the same time the DSP reports the last peak values to the slow control entity, which in turn broadcasts the signal values to all the remote controls. As shown in Fig. 2 audio data is first passed through the six band equalizer. The final mixing parameters are calculated depending on the selected signal routing. After the main channel fader the signal can be routed via one or more subgroups or sent directly to the master output. Each subgroup can be mapped to direct outputs (DACs) or to the master output. Four distinct AUX channels per setup can be routed pre or post the channel fader. The VU meter calculation is done directly after the ADC to check the input signal strength or directly after the mixer matrix, where the actual signal strength is available (per channel or for the subgroup/master sums).

Since only one DSP is working with the mixing parameters, the second DSP can be used for side-calculation like audio effects. A special parameter set gives the opportunity to set up 32 different audio effect chains which can be calculated in real time. This section of the audio mixer is currently under development.

The XILINX FPGA uses a 16 bit wide dual port RAM for communication with the XC 167. The DSP board is connected via one special synchronous 8 bit data port of the TS 101.

2.3. Audio Stream

Analog audio signals are sampled by high end 24 bit AD 1854 and AD 1871 CODECs. The regular sampling rate is 48 kHz, but this can be changed to 96 kHz if both DSPs are used for audio mixing calculation. The XILINX FPGA generates the sampling clock signal and synchronizes all other components. Data exchange with the DSP board is done via another of the 8 bit synchronous data ports of the TS 101 processor architecture, fully utilizing the internal 128 bit busses and the two independent computational units. Calculations are done in IEEE floating point format with single instruction multiple data operations. This special mode is used for calculating 4 channels at the same time. The optimization result is documented in [3]. Starting with a textbook application using roughly 62 µs, the optimization of the code leads to a final execution time of 3,22 µs for all 64 channels. Since the equalizer is realized as six stage second order IIR filter, careful simulations have been done on the parameter ranges for the filter coefficients. In his diploma thesis Franz Siegmeth has also proven the validity and possible implementation of these filter banks.

Two different memory segments are used to implement a working pipeline with a total of four steps. This gives a total of 83 µs response time per sample.

The audio mixer itself implements a 6 band fully parametrized equalizer for each input channel, followed by the 96 in / 64 out mixing matrix. Audio data is converted to floating point format prior to entering the input filter and reconverted to 24 bit integer after all calculations are finished. All calculations in the DSPs are done with floating point numbers.

Data for audio effect calculation is transferred via DMA to the second DSP. To grant more time for the effect calculations, the designated effect loop channels will be dealt with at the beginning of the calculation stage and transferred to the second DSP while the calculation on the “real” audio channels is done.

2.3.1. 6 Band Equalizer

Each input channel has its own 6 band equalizer with independent parameters (shelving and peak filters, $f$, gain $G$ and factor $Q = f_b/f_c$) are converted to corresponding parameters for calculation by slow control and sent to the DSP. The DSP code has been optimized for the TS 101 processor architecture, fully utilizing the internal 128 bit data busses and the two independent computational units. Calculations are done in IEEE floating point format with single instruction multiple data operations. This special mode is used for calculating 4 channels at the same time. The optimization result is documented in [3]. Starting with a textbook application using roughly 62 µs, the optimization of the code leads to a final execution time of 3,22 µs for all 64 channels. Since the equalizer is realized as six stage second order IIR filter, careful simulations have been done on the parameter ranges for the filter coefficients. In his diploma thesis Franz Siegmeth has also proven the validity and possible implementation of these filter banks.

2.3.2. Mixer Matrix

The mixer matrix itself is implemented using an adapted FIR filter algorithm. The coefficients are calculated by slow control and
actualized every 50 ms. The slow control microprocessor uses a three stage multiplication scheme to get the right values: first there are the per channel faders. Channels can be combined into subgroups (8 groups stereo). Subgroups can be configured as real output channels or again mixed to the main output. In parallel, every channel can be a part of the main output without belonging to a subgroup.

The matrix calculation starts with the 32 effect channels, which are subject to DMA transfer to the second DSP right after the calculation has finished. The calculation of the 32 output channels follows immediately.

For the main signal output channels (max. 3 pairs) there is a special feature in the output chain. Each main signal pair has its own volume control and delay line. The main signal can be delayed for up to 500 ms to compensate for sound wave propagation in a distributed speaker environment.

2.3.3. VU Meter

After the matrix calculation there is another series of calculation which gives the VU meter data sum for each channel. This sum is sent to all mixer terminals every 50 ms via UDP broadcast. The parameters for this calculation can be updated every 50 ms. This feature can be used to display the de facto contribution of each channel to the main signal, in which case the calculation parameters are the ones used in the matrix mixer section for the main output. When adjusting the input gain, the factors are changed to the channel fader coefficients. If the fader is set to 0 dB the input gain can be adjusted using the gain control.

For providing the standard VU meter display the 32 output channels are monitored, too, without any calculation parameter. Therefore the real signal strength can also be drawn from the VU meter broadcast package.

2.3.4. Audio Recording

Current development deals with the implementation of an IDE interface on the FPGA. Using ATA-2 (EIDE, 16 MBps) or even ATA-3 (Ultra ATA, 33 MBps) standard, the amount of realtime data should be no problem even if the recording is done uncompressed. Once this interface is included in the audio data stream, a control instance can use it to record the 64 channels of live data on hard disc. The FPGA can be used as source control and play back the recorded data to the DSP, making no difference to the live performance. Using this approach gives the possibility to record a live concert and use the raw sound data later in a studio the remix the recorded tracks for studio production.

2.4. Hardware Setup

The audio frontend is organized in 8 input / 4 output channels per unit. The audio signal enters a pre-ADC section where phantom power can be applied if necessary. Mic and Line signals can be sampled because prior to the ADC there is an analog gain controller (digitally controlled via a feedback DAC). The ADCs are chained together, so the ADC clock of 12,288 MHz gets 8 x 32 bits out of the chain per sampling interval. Data is left justified, so there is time enough after sampling the last channel to transfer the data to the DSP. Data transfer starts in the middle of the sampling clock cycle. This point is used for synchronization, since the DSP initializes its DMA channel after finishing the backtransfer of the output data and remains idle until new data is available. The possible data transfer rate to the DSP in this setup is 62 MBit x 8 per second, the ADC data rate is 12,288 MBit x 6 (one byte out of 4 is always 0), which gives enough safety margin for the DMA transfer.

DACs cannot be chained together, so each of them gets only two channel’s data. All DACs are set with the same clock signal, so each output channel has the new data available at the same time (which is trimmed to the end of the sampling interval). Following the DAC, there is a symmetric signal distributor, so the audio signal can be used either in a symmetric or asymmetric way, in which case the second line is shortcut to ground. The chip will rise the asymmetric signal level accordingly.

2.5. User Interface

We have two different approaches to the user interface. The first goal was to keep the traditional mixing console feeling for live performances. Thus we built a microcontroller based control system which looks familiar to the sound engineer [6]. On the other hand, we developed a computer application which can be used in an audio studio and provides full access to all the ADAM features. The application can be used during live performances to restore certain scene settings like fader setups or even filter and level setups. All those settings can be restored selectively so that the interference with live settings can be minimized. Since each fader module is an independent system controlled by the mixing console master (another XC 167), channels can be assigned to each fader as needed. The mixing console itself can store up to four different fader layouts. Using the software application, this number can be extended as needed.

Both application and hardware user interface communicate with the stage box over TCP/IP connections. A special protocol has been developed to meet the needs for exchanging setup and slow control data. The protocol has been prepared to include further extensions like transferring live audio data using a realtime ethernet protocol extension. The mixing console has a VU meter bridge to show all 64 input channels with the corresponding fader settings for the selected mixing console setup. The fader and control displays are updated by the stagebox so that more than one mixing console can be used for the same output setup. Faders and controls can be moved once their value does not longer correspond to the
actual stage box setting, and they have to “hook into” the actual value again before their movement will be sent to the stage box. This way we can avoid jumps in volume or filter settings even if more than one mixing console (or restored settings from the software application) have been applied to the stage box.

3. RESULTS

All parts of the system have been evaluated with realtime measurements. One sampling slot has about 21 $\mu$s. The FPGA uses the full slot time for sampling 8 x 8 channels of ADCs. During the last 650 ns of the time slot, there is time enough to transfer the last 8 channels to the DSP, so the calculation of the mixing stage starts nearly synchronously with the next sampling slot.

The mixing stage calculation uses 3,2 $\mu$s for the input amplifier (including the int to float conversion), leaving 16 $\mu$s for the matrix mixer. The calculation of the VU meter data uses another 1 $\mu$s, leaving nearly 1 $\mu$s as safety margin to cope with DMA transfer delays.

The second DSP is not used for the basic audio mixer setup and will be included in ongoing work to do some effect calculations on 32 extra channels.

4. CONCLUSIONS

ADAM is a powerful audio processing setup which can be adapted to other tasks as well. There are many ideas of further improvements, like 64 channel hard disc recording or realtime audio data transfer using the standard ethernet link to the remote control. The setup can also be altered to support less input channels but up to 256 output channels, which would be a great setup for wave front synthesis.

Ongoing developments are now dealing with realtime data compression which can be used for the hard disc recorder. Another project starting soon will be the development of an 8 channel audio WLAN box which can support active speaker setups with one of 8 channels coming from the ADAM mixer.

5. REFERENCES

NON-LINEAR DIGITAL IMPLEMENTATION OF THE MOOG LADDER FILTER

Antti Huovilainen

Laboratory of Acoustics and Audio Signal Processing
Helsinki University of Technology, P.O. Box 3000, FIN-02015 HUT, Espoo, Finland
ajhuovil@acoustics.hut.fi

ABSTRACT
This paper presents a non-linear digital implementation of the Moog ladder filter. The implementation is relatively efficient and suitable for inclusion into real-time systems, for example virtual analog synthesizers. The analog circuit is analyzed to produce a differential equation. This equation is solved using Euler’s method, and the result is shown to be equivalent to a cascade of first order IIR sections with embedded non-linearities. Finally, the filter structure is modified to improve tuning.

1. INTRODUCTION
Time varying filters are used in many musical applications - synthesizers, effects units and samplers. They are especially important nowadays in virtual analog synthesizers. The voltage-controlled filter published by Robert Moog in 1965 [1] is perhaps the most famous of them.

There exist several published digital filters suitable for musical applications, such as the State-Variable Filter [2]. The Moog filter itself has also been converted to digital form by Stilson and Smith [3]. These filters are linear and some people feel that they sound “digital” and lack the “warmth” that is characteristic of analog filters - especially the Moog filter. Rossum [4] has published a filter that claims to have the same “warm” sound by embedding a non-linearity within the filter.

In this paper the Moog filter circuit is analyzed and a digital implementation of it is presented. The implementation models the inherent non-linearities of the original circuit and should thus give more accurate results compared to linear digital filters. As the digital filter is directly based on the analog circuit, there are no extra coefficients that would need to be tuned by ear – the “warmth” is determined by the input amplitude. While the resulting filter requires more computation than traditional linear filters, modern advances in computing power allow it to be used in real-time systems. Since the large signal behavior of the circuit is analyzed, traditional pole-zero analysis is not directly usable. Not using pole-zero domain can also offer otherwise hidden insights.

Additional material and audio examples are available at http://www.acoustics.hut.fi/publications/papers/dafx2004-moog/

2. THE MOOG LADDER
Moog implemented his filter using an innovative transistor ladder circuit shown in Figure 1. It employs the base-to-emitter resistance of bipolar transistors to realize voltage-controlled RC-sections. The transistors also act as buffers between each stage.

The circuit has four stages, each of which consists of two transistors and a capacitor. Each stage is driven by the output current of the previous stage apart from the first stage, which is driven by a differential transistor pair. The bottom of the ladder is driven by control current \( I_c \). Differential output is taken from the emitters of the transistors in the last stage to a high impedance differential amplifier. To produce resonance, a portion of the output is routed to the other side of the input differential amplifier.

The current gain, or \( \beta \), of the transistors is high (values of > 200 are typical) and the resistors feeding the transistor bases are small valued, so transistor base voltages are effectively constant for all stages. It is a reasonable approximation to assume that the transistor \( \beta \) is infinite and therefore the base current is zero. Thus the stages are buffered from each other. Further, since the base-emitter voltage is logarithmically dependant on the collector current, the emitter voltage varies very little and the Early effect
[6] can be neglected. Therefore each stage depends only on the current state and the input current coming from the previous stage.

\[ I_{t1} - I_{t2} = (I_{t1} + I_{t2}) \tanh \left( \frac{V_{t1} - V_{t2}}{2V_t} \right), \]

where \( V_t \) is the so-called thermal voltage of a transistor [6]. Equation (1) holds when the transistors are assumed to be perfectly matched, beta is infinite and the Early effect is neglected.

3. DIFFERENTIAL EQUATION

We can now derive a differential equation for the filter. Since the stages are buffered from each other, it makes sense to first derive a differential equation for a single stage.

\[ \frac{dV_c}{dt} = \frac{I_{ctl}}{C} \left[ \tanh \left( \frac{V_{in}}{2V_t} \right) - \tanh \left( \frac{V_c}{2V_t} \right) \right]. \]

4. DISCRETIZING THE FILTER

4.1. Difference equation for a single stage

To be useful for a digital implementation, the differential equation must now be solved. The easiest way is to use Euler’s method. Although Euler’s method has some drawbacks, it is very useful for this particular case. As the differential equation is of first order, the solution is inherently stable. Since there is a non-linearity, oversampling must be used and this brings the Euler solution closer to the ideal solution. Runge-Kutta or some other higher order method could also be used, but it would require evaluation of the equation between samples. This is problematic as it is equivalent to having a higher sample rate for the input than the output. It also poses a problem for resonance, since that is achieved by feeding back some of the ladder output. Euler’s
method also has the advantage that the resulting difference equation is similar to a normal one-pole IIR lowpass filter as seen in Section 5.

Euler solution for equation (10) is
\[ V_c(n) = V_c(n-1) + T_s \left( \frac{V_c(n)}{2V_i} \right) \left( \frac{1}{2} \right) \]
where \( T_s \) is the time interval between samples. For sample rate \( F_s \),
\[ T_s = \frac{1}{F_s} \]
(12)

4.2. Difference equation for the complete filter
Difference equations can now be written for the full ladder filter.
\[ y_a(n) = y_a(n-1) + \frac{I_{ctl}}{C_{F_s}} \left( \frac{\tanh(x(n) - 2V_i)}{2V_i} - W(n-1) \right) \]
(13)
\[ y_b(n) = y_b(n-1) + \frac{I_{ctl}}{C_{F_s}} \left( W(n) - W(n-1) \right) \]
(14)
\[ y_c(n) = y_c(n-1) + \frac{I_{ctl}}{C_{F_s}} \left( W(n) - W(n-1) \right) \]
(15)
\[ y_d(n) = y_d(n-1) + \frac{I_{ctl}}{C_{F_s}} \left( W(n) - \tanh\left( \frac{y_d(n)}{2V_i} \right) \right) \]
(16)
where \( x(n) \) is the input, \( y_a(n), y_b(n), y_c(n) \) and \( y_d(n) \) are the outputs of individual filter stages, \( r \) is the resonance amount \((0 < r \leq 1)\) and
\[ W_{(a,b,c)}(n) = \tanh\left( \frac{y_{(a,b,c)}(n)}{2V_i} \right) \]
(17)

It can be seen that each stage uses as input the tanh of the output of the previous stage. This is also used by the previous stage during the next sample. The calculation result can be stored and thus only five tanh calculations per sample are required. These can be implemented efficiently with table lookups or polynomial approximations.

5. IMPROVING TUNING
5.1. Tuning of a single stage
While this paper is concerned with the large signal model of the Moog ladder filter, it is interesting to see what equation (11) is for low signal amplitudes. For small inputs \((-0.5 < x < 0.5)\), \( \tanh \) function is almost linear. Equation (11) then becomes
\[ V_c(n) = V_c(n-1) + \frac{I_{ctl}}{C_{F_s}} \left( \frac{V_c(n)}{2V_i} - \tanh\left( \frac{V_c(n-1)}{2V_i} \right) \right) \]
(18)
where \( F_s \) is the sample rate. Equation (18) is similar to that of a normal digital one-pole lowpass filter.

Scaled impulse invariant transform \([7]\) is impulse invariant transform scaled so that the dc gain is one. The difference equation for a one-pole lowpass filter transformed with scaled impulse invariant transform is
\[ y(n) = y(n-1) + g(x - y(n-1)) \]
(19)
which is the same as equation (18) with
\[ g = \frac{I_{ctl}}{2V_iC_{F_s}} \]
(20)
substitution. As can be seen, \( I_{ctl}, C \) and \( F_s \) determine the tuning. Since \( I_{ctl} \) and \( C \) do not affect anything else, their exact values are irrelevant. Coefficient \( g \) can therefore be computed the same way as with a normal scaled impulse invariant transformed one-pole filter
\[ g = e^{-2\pi r/F_s} \]
(21)
where \( F_s \) is the cutoff frequency.

Making this substitution to equation (13) and substituting \( x \) for input and \( y \) for output gives
\[ y(n) = y(n-1) + 2V_i \left( \tanh\left( \frac{x(n)}{2V_i} \right) - \tanh\left( \frac{y(n-1)}{2V_i} \right) \right) \]
(22)
Making the substitution to equations (14)–(17) gives the difference equations for the complete filter.

5.2. Resonance
To produce resonance, the filter output is fed back inverted. In the analog filter, each stage causes a 45 degree phase shift at the cutoff frequency, producing a combined phase shift of 180 degrees at the cutoff frequency. This phase shift, combined with inverting, causes the feedback to be positive at the cutoff frequency and thus it emphasizes frequencies around the cutoff. The attenuation for a single stage is 3 dB at cutoff, producing a total attenuation of 12 dB for the complete filter at the cutoff point.

In the digital implementation, the unit delay in the feedback path causes an additional phase shift, making the combined phase shift to be
\[ p = 4p_{stage}(f, F_s) + 180 \frac{f}{F_s} \]
(23)
where \( p \) is the total phase shift and \( p_{stage} \) is the phase shift of a single filter stage. This additional phase shift causes the resonance frequency to vary from the cutoff frequency. Another effect is that the attenuation at resonance frequency is no longer exactly 3 dB. This means that the feedback amount required to produce the desired resonance varies with frequency.

5.3. Compensation
Stilson and Smith show some methods to compensate this shifting of resonance frequency and amplitude \([3]\). However, these methods require the use of tuning and resonance amount compensation tables. This has an unfortunate side effect owing to the difference between tuning for zero resonance and tuning for self oscillation. Here, the tuning table must be two-dimensional, or some form of interpolation is needed between no tuning and full tuning (depending on the resonance amount). Further, methods where transfer function zeroes are introduced require two different coefficients to be used and interpolated in the filter loop.
Another approach is to stay with the scaled impulse invariant transformed filter stage and see how the filter structure might possibly be modified to compensate in order to get feedback phase as close to 180 degrees as possible. Figure 4 shows the phase shift for the filter both alone and with the feedback path unit delay using two times oversampling (88.2 kHz sample rate). It can be seen that the phase shift of the filter starts falling back to zero at higher frequencies and this somewhat compensates for the phase shift introduced by the unit delay.

The resulting phase shift is now slightly too small at the cutoff frequency. Addition of half-unit delay causes the phase shift to be almost exactly 180 degrees at cutoff up to about $F_s/4$. The half-unit delay can be realized by averaging two samples. At very high frequencies ($f > F_s/4$) the situation is now worse than without the extra delay, but as some oversampling is required because of the nonlinearities, this does not matter in practice. Figure 5 shows the tuning and amplitude error with and without the extra delay. With the extra delay added, the error in tuning is less than 10% for $f < F_s/4$ (or $f < 22$ kHz for $F_s$ of 88.2 kHz).

With two times oversampling, the remaining tuning error can be eliminated by using a tuning table. Since the error is so small, the resonance tuning compensation can be combined with the tuning for scaled impulse invariant transformed one-pole filter into a single table. The small error in the frequency response is unlikely to be audible.

6. CONCLUSIONS

A digital implementation of the Moog ladder filter has been presented with non-linearities of the circuit correctly modelled. The implementation is similar to a normal IIR filter made of cascaded first-order sections, but the first order sections have non-linearities embedded within them. The filters is directly based on the Moog transistor ladder circuit and thus requires no user tunable parameters other than the cutoff frequency and input amplitude.

While more computationally intensive than traditional IIR filters, the filter is still suitable for real-time and DSP implementations. As some of the calculations are shared between stages, the implementation requires only five tanh-function evaluations. These can be implemented efficiently with table lookups or polynomial approximations.

7. ACKNOWLEDGEMENTS

The author would like to thank Professor Vesa Välimäki for his valuable support and feedback, as well as Merlijn Blaauw, Martin Fay, John McMonagle, Andreas Tell and George Yohng for comments and proofreading.

This work has been supported by the Academy of Finland (project #104934).

8. REFERENCES

A FAST MELLIN TRANSFORM WITH APPLICATIONS IN DAFX

Antonio De Sena and Davide Rocchesso

Dipartimento di Informatica, Università di Verona

ABSTRACT

Many digital audio effects rely on transformations performed in the Fourier-transformed (frequency) domain. However, other transforms and domains exist and could be exploited. We propose to use the Mellin transform for a class of sound transformations. We present a fast implementation of the Mellin transform (more precisely a Fast Scale Transform), and we provide some examples on how it could be used in digital audio effects.

1. INTRODUCTION

Fast realizations of the Discrete Fourier Transform are widely used in order to produce audio signal transformations by operating in the transformed (frequency) domain [1]. These realizations are used as if they were approximations of an underlying continuous-time Fourier Transform, and the transformations rely on properties such as the magnitude invariance to time shifts, the relative auditory unimportance of phase for stationary signals, and the interpretation of spikes in the transformed domain as periodic components in the time domain [2,3].

Many other transforms with different properties have been devised in order to make certain operations easier or certain features more easily visible. Among these, the Mellin Transform, and its restricted version called the Scale Transform, can represent a signal in terms of scale. The scale can be interpreted, similarly to frequency, as a physical attribute of signals [4]. Thus, we can conceive digital audio effects that work by handling the signal in the scale domain, with transformation of the magnitude and/or phase of the Mellin image. This is technically feasible as long as fast and accurate realizations of these transforms are available.

Other useful applications can be done using the Mellin transform. For example Patterson and Irino [5] have proposed to use a particular bidimensional version of this transform for vowel normalization.

Digital audio effects, such as time/pitch scaling, using the Mellin transform and the phase vocoder were previously proposed [6], but the realization relied on non-uniform sampling or re-sampling, and no considerations on the speed, accuracy, and feasibility of these operations were given. In image processing, effects such as localized denoising have been proposed as based on the scale (Mellin) transform [7]. Since the Mellin transform can be interpreted as a Fourier transform working on logarithmic time, it relies on warping the time axis. Effects based on time and frequency warping, using the Fast Fourier Transform (FFT) or dispersive delay lines, were presented in [8].

In Section 2 we briefly introduce the Mellin and scale transforms, and we provide an interpretation of the transform and its relation with the Fourier transform. Section 3 shows how a fast discrete version of the scale transform is implemented, using exponential resampling and the FFT algorithm. Section 4 presents some digital audio effects obtained by transformations in the Mellin domain.

2. THE SCALE AND MELLIN TRANSFORMS

The Mellin transform of a function \( f(t) \) is defined as:

\[
M_f(p) = \int_{0}^{\infty} f(t) t^{p-1} \, dt,
\]

where \( p \in \mathbb{C} \) is the Mellin parameter. The scale transform is a particular restriction of the Mellin transform on the vertical line \( p = -jc + \frac{1}{2} \), with \( c \in \mathbb{R} \). Thus, the scale transform is defined as:

\[
D_f(c) = \frac{1}{\sqrt{2\pi}} \int_{0}^{\infty} f(t) e^{(-jc-\frac{1}{2})\ln t} \, dt.
\]

The scale inverse transform is given by

\[
f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} D_f(c) e^{(jc-\frac{1}{2})\ln t} \, dc.
\]

The key property of the scale transform is the scale invariance. This means that if \( f \) is a function and \( g \) is a scaled version of \( f \), the transform magnitude of both functions is the same. A scale modification is a compression or expansion of the time axis of the original function that preserves the signal energy. Thus, a function \( g(t) \) can be obtained with a scale modification from a function \( f(t) \), if \( g(t) = \sqrt{\alpha} f(\alpha t) \), with \( \alpha \in \mathbb{R}^+ \). When \( \alpha < 1 \) we get a scale expansion, when \( \alpha > 1 \) we get a scale compression. Given a scale modification with parameter \( \alpha \), the scale transforms of the original and scaled signals are related by

\[
D_g(c) = \alpha^{-c} D_f(c).
\]

This property derives from a similar property of the Mellin transform. In fact, if \( h(t) = f(\alpha t) \), then

\[
M_h(p) = \alpha^{-p} M_f(p).
\]

In both (4) and (5), scaling is reflected by a multiplicative factor for the transforms, and for (4) such factor reduces to a pure phase shift.

2.1. The scale transform interpretation

A parallel can be drawn between the properties of the Fourier and scale transforms. In particular, we can define a scale periodicity as follows: a function \( f(t) \) is said to be scale periodic with period \( T \) if it satisfies \( f(t) = \sqrt{T} f(t/T) \), where \( T = b/a \), with \( a \) and \( b \) period starting point and period ending point respectively (see Figure 1). \( C_0 = 2\pi / \ln T \) is the “fundamental scale” associated with the periodic function. By analogy with the Fourier theory, we can
define a “scale series” and Parseval theorem. Very important is the “exponential sampling theorem”\footnote{http://www.inrialpes.fr/is2/people/pgoncalv/} that, like the Shannon theorem, allows a perfect reconstruction of a scale-band limited signal from its samples. These samples must be distributed exponentially in time according to \( p_k = T_s \) with \( k \in \mathbb{Z} \), \( T_s = e^{p/\Sigma_0} \), and \( \Sigma_0 \) the signal maximum scale.

2.2. Relation with the Fourier transform

From its definition and interpretation, the Mellin transform provides a tight correspondence with the Fourier transform. More precisely, the Mellin transform with the parameter \( p = -jc \) can be interpreted as a logarithmic-time Fourier transform. Similarly, we can define the scale transform of a function \( f(t) \) using the Fourier transform of a function \( g(t) \), with \( g(t) \) obtained from \( f(t) \) by time-warping \( f \) and multiplying the result by an exponential function. This result can be generalized for any \( p \) defined as \( p = -jc + \beta \), with \( \beta \in \mathbb{R} \).

3. THE FAST MELLIN TRANSFORM

Practical modifications of signals in the Mellin domain can be achieved only if an accurate and fast discrete realization of the Mellin transform is available. In \cite{10}, an algorithm based on the extraction of the analytic signal was proposed, and it is now available for both \texttt{matlab} and \texttt{scilab}.\footnote{http://www.inrialpes.fr/is2/people/pgoncalv/} However, such realization requires the specification of a lower and upper frequency bounds, its complexity appears to be quadratic in the number of samples, and it displays strong side lobes in the scale domain (see Figure\ref{fig:2}).

We realized a Fast Mellin Transform (FMT) by exploiting the analogy between the Mellin and Fourier transforms, as a sequence of exponential time-warping, multiplication by an exponential, and Fast Fourier Transform, as represented in Figure\ref{fig:2}.

The Mellin transform with parameter \( p = -jc + \beta \) (with \( \beta \in \mathbb{R} \)) of \( f(t) \) is identical to the Fourier transform of \( e^{\beta}f(e^\tau) \):

\[
D[f(t)] = F[e^{\beta}f(e^\tau)],
\]  

where \( F[\cdot] \) and \( D[\cdot] \) refer to the Fourier transform and scale transform, respectively. The Fourier transform is commonly computed in time \( \mathcal{O}(N \log N) \) on \( N \) samples, by means of the FFT. While the multiplication by an exponential is trivially done in \( \mathcal{O}(N) \), we must find an algorithm for performing the exponential time-warping. This last problem can be seen as an exponential sampling of the continuous time signal. Generally we have only a uniformly (Shannon) sampled signal, thus the problem can be seen as resampling a discrete-time sequence, and this can be solved using interpolation. In theory, we should use a sinc interpolator (based on Shannon sampling theory), but the overall complexity turns out to be too high. However, we can approximate this interpolator by means of a natural cubic spline, and have the linear complexity associated with resolution of a tridiagonal matrix. Using this interpolator we can resample the original function obtaining an exponentially-sampled version.

The parameters needed by the resampling process are the exponential sampling step \( T_s \) and the number of exponential samples \( N_{\exp} \). If the original signal has been uniformly sampled by taking \( n \) samples with time step \( T_s \), starting at time \( T_s \), we can show that \( T_s = 1 + 1/n \) and \( N_{\exp} \approx n \ln n \). Figure\ref{fig:3} shows an example of distribution of exponential samples derived from a sequence of uniform samples.

The algorithm has an asymptotic complexity that depends only on the FFT, as this is the most (computationally) complex part of the entire process (the spline interpolation block is linear in \( N_{\exp} \) and the exponential multiplication block is also linear). The asymptotic complexity of the entire process is \( \mathcal{O}(N_{\exp} \ln N_{\exp}) \) or, in terms of the number \( n \) of uniform samples, \( \mathcal{O}(n \ln^2 n) \).

The accuracy of the Fast Mellin Transform in providing an approximation to the continuous-time Mellin transform is good.
Figure 3: Uniform sampling and (critical) exponential resampling

For example, Figure 4 provides a comparison of the magnitude of the FMT with the theoretical continuous-time Mellin transform and with the realization proposed in [10]. In this example we’ve worked with a step function signal created using 128 samples, a sampling frequency of 8000 Hz and setting the firsts 50 samples to 1 and the others to 0.

Figure 4: Scale transform (magnitude) of a step function: continuous-time transform (solid) and its approximations with the realizations by [10] (dashed) and by the authors (dashdotted)

4. DIGITAL AUDIO EFFECTS IN MELLIN (SCALE) DOMAIN

In this Section we show how to realize some digital audio effects using the scale domain.

4.1. Time scaling in Mellin domain

A straightforward yet useful effect is time compression or expansion with signal energy preservation. For two signals that are one the scaled copy (with factor $\alpha$) of the other, the scale transforms have the same magnitude, and a difference in the phase. Let $f(t)$ and $g(t)$ be those two signals, with $g(t) = \sqrt{\alpha} f(\alpha t)$ and $\alpha \in \mathbb{R}^+$. Using the fact that $D_x(c) = \alpha^{-c} D_f(c)$ we can obtain $g(t)$ from $f(t)$ by applying the scale transform to $f(t)$, adding the linear contribution $c \ln \alpha$ to the the phase, and anti-transforming the result. Some care has to be taken in the choice of $\alpha$: if it is too high the signal that we get from scale-compression will have frequency components that can cause aliasing. Conversely, if $\alpha$ is too low, we may end up cropping the signal in time.

Figure 5 shows a signal and its time-compressed version, obtained by adding a linear offset to the phase of the scale transform (6). What is depicted in Figure 5 is essentially a resampling.

Figure 5: Original audio signal and a scaled version (Scaled factor $\alpha = 2$) obtained using the FMT.

If the added phase contribution is not linear, then we can achieve simultaneous resampling and time warping. As compared to other resampling methods, such as the windowed-sinc interpolation [11] implemented in octave or matlab, using the scale

Figure 6: Signal phase and modified signal phase.
transform does not introduce any benefit neither in accuracy nor in efficiency. However, the possibility to work directly in the phase domain adding contributions to the nominal phase gives the possibility to “sculpt” the temporal behavior of the signal in just the same way as audio practitioners sculpt the frequency behavior with softwares such as audiosculpt[4].

4.2. Signal reconstruction using only magnitude or phase

In order to better understand the respective roles of magnitude and phase in the Mellin transform, we can transform a signal and reconstruct it using only the phase or only the magnitude. Figure[7] shows the sonogram of a test signal. Figure[8] reports the sonogram of the signal reconstructed by replacing the magnitude response with a constant (top) or by replacing the phase response. We notice that the phase-only reconstruction preserves the temporal location of the main events that remain well visible (and audible). This is similar to the highlighting of edges in images reconstructed from their (phase-only) Mellin transform [7].

4.3. Low-pass and high-pass filtering

In this Section we show what a low-pass filter or a high-pass filter do (in the Mellin domain). To low-pass filter, we simply set to zero all magnitude components that are found between a cutoff scale and the the signal maximum scale. Observing the results from a classical Fourier-based viewpoint (see Figure[9] top), we can interpret this filter like a time-varying low-pass filter. The filter cutoff frequency exponentially approaches zero in time. The speed of convergence depends on the cutoff scale. The high-pass filter behaves symmetrically, gradually moving the cutoff frequency toward zero (see Figure[9] bottom).

4.4. Phase with random deviations

In this experiment we introduce a random deviation to the phase. This deviation grows linear by the scale and adds up to the unwrapped phase. In this example the deviation doesn’t exceed the 0.04% of the phase, and this is enough to destroy the fine temporal structure without loosing the most important events (see Figure[10]).

5. CONCLUSION

We presented a new implementation and some audio applications of the discrete Mellin transform. The octave/matlab code will be made publicly available. In the future, more sophisticated interpolation schemes will be tried in order to improve the accuracy, and the extension to a sliced-time framework (a kind of Short-Time Mellin Transform) will be attempted.

6. REFERENCES

Figure 10: Phase with growing random deviation.


TRANSFORMING SINGING VOICE EXPRESSION – THE SWEETNESS EFFECT

Lars Fabig, Jordi Janer

Music Technology Group – IUA
Universitat Pompeu Fabra, Barcelona, Spain
{lfabig|jjaner}@iua.upf.es

ABSTRACT

We propose a real-time system which is targeted to music production in the context of vocal recordings. The aim is to transform the singer’s voice characteristics in order to achieve a sweet sounding voice. It combines three different transformations namely Sub-Harmonic Component Reduction (reduction of sub-harmonics, which are found in voices with vocal disorders), Vocal Tract Excitation Modification (to achieve a change in loudness) and the Intonation Modification (to achieve smoother transitions in pitch). The transformations are done in the frequency domain based on an enhanced phase-locked vocoder. The Expression Adaptive Control estimates the amount of present vocal disorder in the singer’s voice. This estimate automatically controls the amount of Sub-Harmonic Component reduction to assure a natural sounding transformation.

1. INTRODUCTION

1.1. Voice characters

A singer’s voice may have characteristics which on the one hand are described scientifically as vocal disorders, like a growl, creak, rough or hoarse voice. On the other hand, from the musical point of view, one can consider these characteristics as highly relevant expressiveness features, which creates its singer’s voice unique timbre. There are singers whose voices always have a timbre with vocal disorders, like Louis Armstrong, Janis Joplin and Tom Waits, but others control the amount of vocal disorders intentionally like Joe Cocker, Sting and Brian Adams. Obviously intentional vocal disorder is not the only possible expression in singing voice. Among many others there are vibrato, intonation, breathiness and loudness.

In this article we concentrate on Sweetness in the singing voice. The idea behind is to transform whatever type of singing voice into a sweet sounding voice, for example a sweet jazz voice, like the jazz singer Silje Nergaard or a smooth Bossa Nova voice like the singer Astrud Gilberto. Instead of adding expression we want to flatten it. This can be of interest in music production and post-production to be able to easily adjust the expressivity of a recorded singer’s voice posteriorly. A positive side effect is that we can avoid additional recording sessions, if we just want to retouch just a few phrases. Or a singer just wants to obtain a certain timbre that he is not able to sing or because of a steady hoarse voice.

To achieve the Sweetness effect we propose a combination of three different transformations:

- **Sub-Harmonic Reduction** intends to minimize the amount of vocal disorders.
- **Intonation Modification** smoothes pitch changes to achieve a sort of portamento effect.
- **Vocal Tract Excitation Modification** modifies the loudness of the voice.

1.2. Spectral View of Vocal Disorders

Voice disorders are caused by a variety of factors. Abuse or misuse of the voice with yelling, singing, or speaking is one common reason. Physical alterations in the vocal folds due to abusive lifestyle (smoking, alcohol) or aging are possible. Lack of movement and poor or improper function of the vocal folds are other causes. But also we are able to intentionally control our voice the way that vocal disorders become audible. Singers exploit it to enhance their timbre with personal expression. It may be used as an effect in a particular passage or phrase, e.g.: in a loud extended sung tone.

In this article we focus on the particular case of voice disorder that provokes “hoarseness”. This pathology is often referred in medical literature as Muscle Tension Dysphonia. This phenomenon occurs in the Larynx, the organ that includes the vocal folds (the source of voice production) due to an excessive muscular effort and usually due to pressed phonation (high sub-glottal pressure) [5].

![Figure 1: Spectra of singing voice without significant sub-harmonics (a) and with sub-harmonics (b). Sub-harmonics are indicated with square boxes.](Image)

A spectral analysis of a hoarse or growl voices reveals additional sub-harmonics which are superposed with the main harmonic partials. They appear due to modulations in amplitude and/or frequency of the glottal cycle periodicity in the voice source [1]. In one particular case depicted in Figure 1 there is not only one harmonic series present but several commensurable harmonic series, which are superposed. You see a harmonic series (a) which has been superposed with a sub-harmonic series in (b). The fundamental frequency is of the sub-harmonic series is approximately one third of $f_0$. In other cases when the irregularities.
of the vocal folds vibrations behave more stochastically, there may appear many noisy components in the spectrum, which can not be interpreted as a commensurable harmonic series anymore.

2. SWEETNESS ALGORITHM OVERVIEW

The proposed Sweetness Effect algorithm is represented by the block diagram seen in Figure 2. The applied processing technique is an enhanced version of the rigid phase-locked vocoder based on Short-Time-Fourier-Transformation (STFT) using an Analysis - Transformation - Synthesis scheme. The main difference is a Fundamental Frequency Estimator which allows us to classify spectral peaks as harmonic or spurious peaks (see Section 3.2 for details). First a frame-based-Spectral Analysis is done, which extracts the following spectral data:

- Spectral peaks (frequency, magnitude and phase)
- Fundamental frequency
- Harmonic peaks (frequency, magnitude and phase)
- The approximated Spectral Shape
- Vocal Tract Excitation Descriptors.

The estimated spectral data is used as control data and/or as data to be transformed in the Spectral Transformation block. Furthermore they are used as input data for the Expression Adaptive Control, which derives a control signal to regulate the necessary amount of Sub-harmonic Component Reduction depending on the amount of vocal disorder in the singer’s voice.

The Transformation block combines three types of transformation which are namely the Intonation Modification (IM) (stretches or smooths pitch changes), Vocal Tract Excitation Modification (VTEM) and the and Sub-Harmonic Reduction (SHR). While the amount of Intonation Modification and the amount of Vocal Tract Excitation Modification is controlled automatically by the Expression Adaptive Control (EAC). The whole transformation section is bypassed when the analysis spectral frame is considered to be unvoiced. The reason is that roughness and growl is only perceived in the stationary part of sung vowels. The spectral processing techniques used by the three transformations are transposition, equalization and spectral interpolation.

The transformed spectral data is then re-synthesized in the Spectral Synthesis block, which consist in an Inverse-Fast-Fourier-Transformation, inverse windowing and Overlap & Add procedure.

The algorithm has been implemented as a real-time application based on the Spectral Peak Processing technique [2], which was especially developed for transforming singing voice with high sound quality, using the C++ Library for Audio and Music (CLAM) [7].

In Section 3 we are going to give an overview about used techniques for spectral analysis and in Section 4 we explain the spectral transformations we apply to achieve the Sweetness effect. Section 5 deals with the Expression Adaptive Control and we conclude with results and improvements in Section 6.

3. SPECTRAL ANALYSIS

3.1. Peak and Pitch Detection

The Spectral Analysis is done using Short-Time-Fourier-Transformation. The time-signal is multiplied with the function of a Kaiser-Bessel window of 46 ms length and then a Fast-Fourier-Transformation is calculated. A simple peak detection algorithm is used to detect the local maxima in the magnitude spectrum (Spectral Peaks). The Fundamental Frequency Detector estimates the singer’s pitch from the detected Spectral Peaks. The Fundamental Frequency Estimator furthermore includes a decision algorithm to label a frame as voiced or unvoiced.

3.2. Harmonic Peak Selection

Considering the estimated fundamental frequency the spectrum is divided in regions of perfect harmonics according to multiples of the fundamental frequency $f_0$. For each of the regions the Spectral Peak of maximum magnitude is searched which is supposed to be the representative peak for this region, called Harmonic Peak [3]. It is represented as well as the Spectral Peaks as a data triplet of frequency, magnitude and phase.

![Figure 2: Sweetness Effect processing scheme.](image-url)
3.3. Harmonic Spectral Shape Approximation

We use a 3rd order spline-interpolation to approximate the Harmonic Spectral Shape, which interpolates between data pairs of logarithmic magnitude and frequency of the selected harmonic peaks.

3.4. Vocal Tract Excitation Estimation

The Vocal Tract Excitation Estimation takes advantage of the Excitation plus Resonance Model [2] (EpR), which is based on an extension of the well known source/filter approach [4]. The EpR filter can be decomposed into two cascade filters. The first of them models the differentiated glottal pulse frequency response and the second the vocal tract (resonance filter). The EpR source is modeled as a frequency domain curve and one source resonance, see Figure 4. The curve is defined by a gain and an exponential decay as follows:

\[
\text{Source}_{\text{dB}} = \text{Gain}_{\text{dB}} + \text{SlopeDepth}_{\text{dB}} \left( e^{\text{Slope} f} - 1 \right)
\]  

Figure 4: Approximation of the EpR source from [2].

It is obtained from an approximation to the harmonic spectral shape (HSS) determined by the harmonics identified in the Harmonic Peak Selection:

\[
\text{HSS}(f) = \text{envelope}_{i=0, n} \left[ f_i, 20 \log(a_i) \right]
\]  

where \( i \) is the index of the harmonic, \( n \) is the number of harmonics, \( f_i \) and \( a_i \) are the frequency and amplitude of the \( i \)th harmonic. On top of the curve, we add a second resonance in order to model the low frequency content of the spectrum below the first formant. The vocal tract resonance model has no impact on hoarseness or growl so that we do not consider it for our needs.

4. SPECTRAL TRANSFORMATIONS

4.1. Sub-Harmonic Reduction

The Sub-Harmonic Reduction reduces the amount of perceived hoarseness or growl minimizing the energy of sub-harmonics. Assuming that a pure harmonic series without sub-harmonics represents a clean sounding voice without vocal disorders, the approach is to synthesize the voice from the pure harmonic series, i.e. synthesizing just the harmonic peaks. To model the bandwidth of the harmonics, the Harmonic Peaks are convolved with the Fourier-transform of the analysis window function. This is done in the Sinusoidal Renderer which fills the surrounding areas of each harmonic.

In contrast, the Spectrum Renderer fills the spectrum with the data taken from the original spectrum (maintains the original shape of the magnitude spectrum). Afterwards both complex spectra are interpolated according to the interpolation factor \( k \) which is controlled by the user to adjust the amount sub-harmonic component reduction, see Figure 5.

In first experiments we found out that the pure harmonic series sounds unnatural, because of missing noisy components. We used two approaches to tackle this problem:

- The naturalness of the voice improves when the Sub-Harmonic Reduction is only applied in the frequency range from 0 Hz to 13 kHz, (with a given sample rate of 44.1 kHz). The frequency range from 13 kHz to 22.05 kHz remains untouched.

- Apply just the minimum amount of Sub-Harmonic Reduction so that perceptively we are not able listen the hoarseness or growl anymore.

Obviously in singer’s performance where roughness and growl are part of his expression the existing amount of sub-harmonics may be time-varying, so that we continuously had to adjust the interpolation factor \( k \) to achieve the best compromise between naturalness on the one hand and non-perceptive sub-harmonics on the other hand. To get rid of the continuous manual adjustments we propose the Expression Adaptive Control, which measures the amount of
sub-harmonic energy in the original spectrum and adaptively controls the amount of interpolation between original spectrum and the PHS (see Section 5).

Figure 5: Sub-Harmonic Component Reduction.

4.2. Vocal Tract Excitation Modification

The Vocal Tract Excitation Modification algorithm is based on the Excitation plus Resonance Model [1] (EpR). In singing voice the energy of higher harmonics depends on its singer’s vocal tract excitation. The EpR model describes this behaviour with a decaying exponential function.

The idea behind Vocal Tract Excitation Modification is to equalize the voice the way as it would have less excitation. It calculates the difference of the measured EpR curve and the desired one for all harmonics. The difference of both spectral envelopes is the resulting frequency domain filter curve to be applied to the signal. The user controls the excitation slope parameter of the desired EpR curve in a meaningful range that has been measured a-priori from a number of voice recordings.

Figure 6: Measured and desired vocal tract excitation and difference of both.

4.3. Intonation Modification

The Intonation Modification transformation is achieved by low-pass filtering the detected pitch, see Figure 8 (original and smoothed, i.e. low pass filtered). According to the amount of deviation from the original pitch the original spectrum is pitch transposed towards the smoothed pitch.

Figure 7: Vocal Tract Excitation Modification.

5. EXPRESSION ADAPTIVE CONTROL

As we have mentioned before, the vocal disorders that we intend to minimise are perceptually present mostly in steady vowels sounds. Therefore, it seems convenient to apply a control signal, so that the original signal is processed by the Sub-Harmonic Reduction algorithm only when the disorders are perceptually relevant. We call this signal Expression Adaptive Control (EAC), since it controls the Sub-Harmonic Reduction depending on the expressiveness of the singer’s voice. Basically, the EAC drives the Sub-Harmonic Reduction algorithm in such a way that the degree of interpolation between original spectrum and the pure sinusoidal spectrum, varies dynamically. The original voice is kept unaltered in case of transients and healthy phonation, which ensures a more natural sound.

Our first task is to define a method for identifying a voice with vocal disorders. As described in Section 1.2 in a growl or hoarse voice besides the harmonic partials additional sub-harmonics are found. In the field of Perceptual Audio Coding appears often the idea of identifying the noisiness of an audio signal. Common methods are the Spectral Flatness Measure (SFM) and Tonality [6]. Here we use a similar concept for identifying the sub-harmonic components.

As input data, the Expression Adaptive Control takes values from the Spectral Analysis block: Spectral Peaks, Harmonic Peaks, Pitch and Excitation Slope. The Sub-Harmonic Factor is computed using a formula that derives from Harmonic Peaks and Spectral Peaks values, which are stored in arrays. Both arrays contain peak information of magnitude, frequency and phase. The first step is to divide the spectrum in regions around each Harmonic Peak [2]. We assume that peaks of frequency above 3.5kHz...
are not relevant for our estimation, thus we consider only the lower frequency range. In the Figure 9, we observe the Harmonic Peaks, the Harmonic Region (centred on each harmonic partial), and all sub-harmonics with lower energy.

![Figure 9: Example of vocal disorder. The spectrum (0-3500 Hz) shows clearly the presence of sub-harmonics, in addition to the harmonic partials (marked with dark circles).](image)

5.1. Sub-Harmonic Factor

In a first study, we attempted to compute the Sub-Harmonic Factor (SHF) using the Spectral Flatness Measure of each Harmonic Region. It is defined by Johnston [6] as the ratio between arithmetic and geometric means of the spectral power density function, and computed directly from the FFT. Experimental results tended to be misleading, when the analysed frame contained harmonics of high bandwidth, for instance in case of vibrato. Therefore, we developed a new approach considering for each region only the magnitude of the spectral peaks as valid data. In the equation 3, the Spectral Peaks are represented by the vector $S_{Peak}[i]$, and the region’s Harmonic Peaks are represented by $H_{Peak}$. We call it Region Sub-Harmonicity ($RSH_i$):

$$RSH_i = \frac{\left[H_{Peak}\right]}{\sum_{i} S_{Peak}[i]}$$

Then we calculate the Sub-Harmonic Factor (SHF) as the average of all Region Sub-Harmonicity ($RSH_i$) values. We only consider up to a frequency of 3.5 kHz (Equation 4). Note that for frames with a high number of sub-harmonics, $RSH_i$ tends to 0. Thus, in the final formula (Equation 4), SHF tends to 1.0 for a high sub-harmonicity, and is 0 in case of a signal with solely pure harmonics:

$$SHF = \frac{1}{R} \sum_{i} (1 - RSH_i), \quad f_c \leq 3500Hz$$

5.2. Excitation Slope and Smoothing

Additionally to SHF, the EAC signal is controlled by the Excitation Slope. The Excitation Slope is an output parameter of the EpR model, and describes the exponential decay of the harmonic partials in the vocal excitation (see Section 3.4). After analysing a number of voice recording with vocal disorders, we observed that the “hoarseness” effect was perceptually more present in vowel sounds such as /a/ and /e/. We studied the results thoroughly using the EpR model [2] and observed that the Excitation Slope parameter was highly correlated with the mentioned vowel sounds.

The resulting EAC signal is the Sub-Harmonic Factor, weighted with the Excitation Slope and smoothed in time using a moving-average filter. Since our system works in frequency domain at a frame rate, the temporal smoothing here is also referred to a frame rate.

![Figure 10: The EAC signal consists of the Sub-Harmonic Factor, which is weighted by Excitation Slope and finally smoothed.](image)

Figure 11 shows a timeline with the evolution of different parameters: original audio, pitch, Sub-Harmonic Factor, Excitation Slope and EAC. In this example, the original audio only presents disorders in the first part. The EAC signal indicates the amount of sub-harmonic reduction that the Sub-Harmonic Reduction algorithm will apply. As we can see the sub-harmonic reduction will be applied mostly in the third note (/ie/). It is also noticeable that the first two notes (/uh/), also with a high SHF, are weighted with the Slope signal.

6. RESULTS AND CONCLUSIONS

The proposed system is able to transform a growl or hoarse a singing voice into a sweet sounding voice using Sub-Harmonic Component Reduction combined with Intonation Modification and Vocal Tract Excitation Modification. The system works offline or in real-time so that it can also be used for a live performance. The system provides professional sound quality at a sampling rate of 44.1 kHz.

The Sub-Harmonic Component Reduction works for most types of voices with vocal disorders. Singing voices performing at low pitch with a high amount of growl turned out to be still problematic. In this case harmonics and sub-harmonics are strongly frequency modulated so it becomes difficult to distinguish them in the spectral domain. If applied without Expression Adaptive Control with maximum reduction of sub-harmonics the transformed voice sounds less natural because of its pure sinusoidality. An approach to improve this behaviour is to preserve subtle noisy components of the original voice which do not belong to the category of sub-harmonics. Using the Expression Adaptive Control the unwanted effect of sinusoidality mostly disappears, except for voices with a strong growl expression.

Experimenting with a number of vocal recordings taken from several sampling libraries the Expression Adaptive Control performed well for most examples. Nevertheless it lacks robustness, because the control signal may deviate significantly depending on the number of spectral peaks detected for each harmonic region. Therefore unnecessarily Sub-Harmonic Component Reduction is applied although no sub-harmonics are present in the spectrum. This is perceived as a degradation of the voice’s naturalness. Although the system requires improvements regarding its robustness we consider it a very useful tool for any recording or post-production studio as well as for singer’s live performance. Different results can be auditioned at: [http://www.iua.upf.es/~lafabig/sweetnesseffect](http://www.iua.upf.es/~lafabig/sweetnesseffect)
Figure 11: The evolution of the different signals involved in the EA is shown in this example. The Pitch signal shows the different notes. The grey area marks the segment where hoarseness is present in the voice.

7. REFERENCES


http://www.iua.upf.es/mtg/clam
ENHANCED TIME-STRETCHING USING ORDER-2 SINUSOIDAL MODELING

Sylvain Marchand and Martin Raspaud
SCRIME – LaBRI, Université Bordeaux 1
351 cours de la Libération, F-33405 Talence cedex, France
{sm|mraspaud}@labri.fr

ABSTRACT
In this article, we introduce a 2-level sinusoidal model and demonstrate its aptitude for a challenging digital audio effect: time-stretching without audible artifacts. More precisely, sinusoidal modeling is used at the two levels of the new sound model. We consider the frequency and amplitude parameters of the partials of the classic sinusoidal model as (control) signals, that we propose to model again using a sinusoidal model. This way, higher-level musical structures such as the vibrato and tremolo in the original sound are captured in the “partials of partials” of this order-2 sinusoidal model. We propose then a new time-stretching method, based on this new hierarchical model, which preserves not only the pitch of the original sound, but also its natural vibrato and tremolo.

1. INTRODUCTION
Spectral models provide general representations of sound in which many audio effects can be performed in a very natural and musically expressive way. Based on additive synthesis, they contain a deterministic part consisting of a – often huge – number of partials, which are pseudo-sinusoidal tracks for which frequencies and amplitudes evolve slowly with time. The spectral modeling parameters of this deterministic part consist of the evolutions in time of the controls of the partials, thus leading to a large amount of data.

We have already shown in [1] that the redundancy in the evolutions of these parameters can be used to reduce these data [1] and that the re-analysis of spectral parameters can help us in extracting higher-level musical parameters such as the pitch [2].

In this article, we introduce a new 2-level sound model of great interest for digital audio effects. The first level of this new model is the well-known sinusoidal model, leading to partials whose parameters – frequencies and amplitudes – continuously evolve slowly with time. For the second level, we consider these parameters as (control) signals, that we propose to model again using a sinusoidal model. This way, higher-level musical structures such as the vibrato and tremolo in the original sound are captured in the “partials of partials” of this order-2 sinusoidal model.

We then demonstrate a straightforward application of this new model to digital audio effects: time-stretching. We chose to focus on this effect, although many others can be performed in this model. More precisely, we show how the new order-2 hierarchical model allows us to enhance the quality of this challenging audio effect, by preserving the natural vibrato and tremolo together with the pitch of the sounds while stretching them. We are specially interested in transforming the deterministic part – no noise or transients for now – of pseudo-harmonic instrumental sounds as well as the human voice.

After a brief introduction in Section 2 to the basic sinusoidal model and a survey of the existing time-stretching methods based on this model in Section 3, we introduce in Section 4 the new hierarchical model and we present in Section 5 a new method for time-stretching while preserving not only the pitch of the original sound, but also its natural microscopic variations such as its vibrato and tremolo.

2. SINUSOIDAL MODELING
2.1. Model and Parameters
Additive synthesis is the original spectrum modeling technique. It is rooted in Fourier’s theorem, which states that any periodic function can be modeled as a sum of sinusoids at various amplitudes and harmonic frequencies. For stationary pseudo-periodic sounds, these amplitudes and frequencies continuously evolve slowly with time, controlling a set of pseudo-sinusoidal oscillators commonly called partials. This is the well-known McAulay-Quatieri representation [3]. The audio signal \( a \) can be calculated from the additive parameters using Equations (1) and (2), where \( P \) is the number of partials and the functions \( f_p \), \( a_p \), and \( \phi_p \) are the instantaneous frequency, amplitude, and phase of the \( p \)-th partial, respectively. The \( P \) pairs \( (f_p, a_p) \) are the parameters of the additive model and represent points in the frequency-amplitude plane at time \( t \). This representation is used in many analysis / synthesis programs such as Lemur [4], SMS [5], or InSpect [6].

\[
a(t) = \sum_{p=1}^{P} a_p(t) \cos(\phi_p(t)) \quad (1)
\]

\[
\phi_p(t) = \phi_p(0) + 2\pi \int_0^t f_p(u) \, du \quad (2)
\]

2.2. Analysis Procedure
In order to faithfully imitate or transform existing sounds, this model requires an analysis method in order to extract the parameters of the partials from sounds which were usually recorded in the temporal model, that is audio signal amplitude as a function of time. The accuracy of the analysis method is extremely important since the perceived quality of the resulting spectral sounds depends mainly on it. Moreover, the main interest of an accurate analysis method, providing precise parameters for the model, is to allow ever deeper musical transformations on sound by minimizing deformations due to analysis artifacts.

The analysis method we use is made of two steps: spectral peaks are first extracted from the sound using a short-time spectral
2.2.2. Tracking of Partials

Since the Fourier analysis above delivers a short-time spectral representation of the analyzed sound, we consider local maxima in the magnitude spectrum (so-called peaks, see above) to be the instantaneous representation ofpartials. We have then to link peaks of successive frames to recover the continuous evolution of the partials. For this purpose, we use the enhanced partial-tracking algorithm we proposed in [8, 9]. This algorithm improves the classic McAulay-Quatieri algorithm [3] by using linear prediction in order to forecast, from their past, the future evolutions of the trajectories of the partials.

As for the practical side of this analysis, the maximal frequency difference between two successive frames for each partial was set to $\Delta = 10$ Hz. Partials whose amplitude was always below 0.001 or length was smaller than 0.1 s were considered as noise, since we are interested only in reliable – long and strong – partials.

2.3. Resampling the Parameters

In the remainder, we consider the frequency $f$, amplitude $a$, and phase $\phi$ parameters of the model as continuous signals. These parameters are measured at the center of each analysis frame. As a consequence, the corresponding signals get sampled at the analysis stage with a sampling period equals to the hop size of the analysis window (512 samples of the original sound – at 44100 Hz – here, see above). Since we need to know their values at each sound sample at the synthesis stage, we must be able to upsample these parameters (by a factor 512 in the example above).

More precisely, let us consider some signal $s$. We can reconstruct its continuous-time function $s(t)$ from its sampled (discrete-time) version $s[i]$, where $s[i] = s(i T_s)$, $T_s$ being the sampling period – that is the inverse of the sampling frequency $F_s$. For that purpose, we convolve the discrete signal by a reconstructor – a windowed sinc function – using an algorithm similar to the one proposed by Smith in [10, 11], except that we chose to use the Hann window instead of the family of Kaiser windows.

In theory, we consider the impulse train made of the samples of the discrete signal where they are known – at times multiple of the sampling period – and 0 (zero) elsewhere. The continuous version of the signal $s$ is reconstructed simply by convolving this impulse train by the ideal reconstructor, the sinc$(t F_s)$ function, using the sin cardinal function defined by:

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}, \text{ for } x \neq 0 \text{ and } \text{sinc}(0) = 1 \quad (3)$$

In practice, the ideal reconstructor cannot be used because of its infinite time support, and we need instead a reconstructor of finite support. In the remainder of this Section, let us denote by $N$ this finite size expressed in samples. This size allows us to tune the trade-off of reconstruction quality versus computation time in the resampling process. We obtain this practical reconstructor by multiplying the ideal reconstructor by some window of finite support. We chose a symmetric Hann window of odd size $N = 2k + 1$ ($k$ being some positive integer), defined by:

$$w_N(n) = \frac{1}{2} \left(1 - \cos \left(2\pi n/(N - 1) \right) \right) \quad (4)$$

for $n$ in the $[0; N - 1]$ range, and 0 (zero) elsewhere. The practical reconstructor is then given by:

$$r(t) = w_{2k+1}(k + t F_s) \cdot \text{sinc}(t F_s) \quad (5)$$
Another problem arises at the boundaries of the discrete signal $s$. Indeed, this signal is of finite support and, during the convolution, we need some of its values before its beginning and after its end. The signal has then to be extrapolated. One solution is to use the classic reflection method (i.e. for samples before the beginning, that is for any sample index $i$ which is a negative integer, we define $s[i] = 2s[0] - s[-i]$). This ensures the continuity of both the signal and its first derivative. However, this is not the best way for extrapolating the signal. For the amplitudes (of the audio signal $a$, of the partials $a_p$, etc.), extrapolating the signal with 0 (zero) values seems to be a natural choice, since those signals fade in and out from / to zero. However this will smooth the attack and this zero-padding technique cannot be used for other kinds of signals such as the frequencies $f_p$ or the phases $\phi_p$ of the partials. For this reason, we use the extrapolation by the Burg method as proposed in [12, 13] for sound signals and generalized in [8, 9] for partials.

Once we have the continuous $s(t)$ function, upsampling by a factor $u$ ($u \geq 1$) the signal $s$ is straightforward since we can compute this function at any time, all the more at multiples of the new sampling period $T_s/u$. Upsampling is like considering the $s(t/u)$ function. Downsampling $s$ by a factor $d$ ($d \geq 1$) is slightly more complicated, since high frequencies have to be filtered out in order to respect the Nyquist condition. This is done by replacing $F_s$ by $F_s/d$ in the sinc function of the reconstructor.

In the remaining, we will often upsample parameters by integral upsampling factors. More precisely, the analysis is done frame by frame, with more than one sample between to frames. In order to get the values of the parameters at each sound sample, the evolutions of these parameters have to be upsampled by a factor corresponding to the hop size $H$ used for the frames at the analysis stage.

### 3. SYNTHESIS AND TIME-STRETCHING

Once we have a sinusoidal model and an accurate analysis method for this model, we need a synthesis algorithm. Most synthesis methods can also be used to perform time-stretching. Let us now denote by $T_s$ the sampling period of the sound to be synthesized.

#### 3.1. Resampling the Frequency

The easiest way is to incrementally recompute the phase of each partial $p$ at each sound sample by a discrete approximation of Equation (2), for $k$ being any (positive) sample index:

$$\phi_p((k+1)T_s) = \phi_p(kT_s) + 2\pi f_p(kT_s)T_s$$

so that the relations between phases and frequencies are maintained, and then compute the complete sound by using Equation (1).

Since we need, for each partial $p$, the values of its frequency $f_p$ and amplitude $a_p$ at each sound sample, we have to upsample these parameters using the technique described in Section 2.

Then, in order to perform time-stretching by a factor $k$ ($k > 0$) during the synthesis, the frequency $f_p$ and amplitude $a_p$ of each partial can simply be resampled according to this $k$ factor prior to the synthesis algorithm itself, to match the targeted length.

Since this technique does not take the measured values of the phase into account, except $\phi_p(0)$ at the time origin, the resulting sound has not the same shape as the original sound (see Figure 2(b)), although this makes no audible difference for most sounds.

### 3.2. The McAulay-Quatieri Method

To be consistent with other articles on this topic (see for example [14]), let us introduce the following notations for the phase and frequency of partial $p$ measured at the center of frame number $k$:

$$\theta^k = \phi_p(kHT_s)$$
$$\omega^k = 2\pi f_p(kHT_s)$$

where $H$ is the number of samples between two consecutive frames. For simplicity sake, we will consider only one partial and omit the partial subscript.

The McAulay-Quatieri model [3] for phase reconstruction of each signal partial between the $k$-th and $(k+1)$-th synthesis frames consists of an order-3 polynomial, given by:

$$\theta(n) = \theta^k + \omega^k n + \alpha n^2 + \beta n^3$$

where $\theta^k$ and $\omega^k$ respectively denote the phase and frequency of the partial measured at the junction of synthesis frames $k$ and $k+1$ (which is chosen as the local origin $n = 0$). Assuming

1. continuity of the phases and frequencies – which are the derivatives of the phases – at frame junctions,
2. unwrapping of the phase with a “maximally smooth” constraint on the phase model

leads to the model parameters $\alpha$ and $\beta$, given by:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 3/N^2 \\ -2/N^3 \end{bmatrix} \begin{bmatrix} \theta^{k+1} - \theta^k - \omega^k N + 2\pi M \\ \omega^{k+1} - \omega^k \end{bmatrix}$$

where $N$ is the size of the synthesis frame (equals to the analysis hop size, so here $N = H$), and $M$ is the “phase unwrapping” integral factor given by:

$$M = e \left[ \frac{1}{2\pi} \left( \left( \theta^k - \theta^{k+1} \right) + \left( \omega^k + \omega^{k+1} \right) \right) \right]$$
where \(\lfloor x \rfloor\) denotes the nearest integer from \(x\). Since the phase, measured at the analysis stage, is known modulo \(2\pi\), the unwrapping factor \(M\) is used to find the real value of this phase, incrementally: if \(\theta^k\) has been unwrapped (at the previous frame), then the unwrapped version of \(\theta^{k+1}\) is \(\theta^{k+1} \mod 2\pi M\).

This phase model is a piecewise-polynomial model of order \(3\). Polynomial phase models of different orders have also been proposed (see [14]).

This model can also be used for time-stretching during the synthesis (see for example [15, 16]). For a stretching factor \(k (k > 0)\), the unwrapped phase must be multiplied by \(k\) (then re-wrapped), together with the size of the synthesis frame which becomes \(kN\).

3.3. Resampling the Unwrapped Phase

We show here that it is possible to consider the (unwrapped) phase of each partial as a continuous function. More precisely, we first unwrap the phase of each partial, from frame to frame, by considering the unwrapping factor \(M\) (see above). Then, we can upsample the unwrapped phase by a factor \(H\) in order to get the value of the phase at each sound sample, using the technique presented in Section 2. Then the amplitude of the partial is upsampled in the same way, and the complete sound can be computed using Equation (1), as it was for the technique presented in Section 3.1.

![Figure 3: Normal time-stretching (order 1) of the saxophone sound, by a factor 2. The frequencies (a) and amplitudes (b) of the partials are shown at the middle of the resulting sound. The rates of the vibrato and tremolo have changed (see Figure 1).](image)

The phase-resampling technique proposed here, and more precisely the underlying continuous phase model, has to be tested in the near future with both synthetic and natural sound examples. Phase modeling is a very interesting research topic. As with the McAulay-Quatieri method described above, phase models are often polynomial (see [14]). However, in this article we will model the amplitude and frequency parameters of the sinusoidal model using the same sinusoidal model (see Section 4 below). Thus, the amplitude and frequency parameters will be sums of sinusoids (more precisely cosine functions, see model Equation (1)), the first one being of frequency zero thus leading to a constant term. Since the frequency is the derivative of the phase (see model Equation (2)), the phase will be a linear term plus a sum of sinusoids. A similar model for the phase is presented in [17] for speech signals.

Series of tests for several polynomial phase models can be found in [14], the McAulay-Quatieri method described above being the order-3 polynomial of this study. Among these tests, the third synthetic example shows sinusoidal evolutions for the frequencies (vibrato). The vibrato (sinusoidal variation of the fundamental frequency) is tested with and without tremolo (sinusoidal variation of the amplitude). It is clear that these sinusoidal evolutions cannot be perfectly approximated by polynomials of finite degrees like the order-3 polynomial phase model of the McAulay-Quatieri method. But provided that the frequency of a sinusoidal evolution remains below the Nyquist frequency, this evolution can be – in theory – perfectly reconstructed using the sinus cardinal reconstructor (see Equation (3)). That is the reason why we could expect an improvement in quality when using the resampling technique.

In [14], the signal-to-noise ratio (SNR) measures the energy ratio between the original signal and the residual part (noise) obtained by subtracting the resynthesis from the original. The greater is the SNR, the more accurate is the phase model for the considered example. For the third – vibrato + tremolo – example, the SNR for the McAulay-Quatieri method is 76.21. For the phase-resampling method using Equation (5) with \(k = 32\), we compute a SNR of only 21.04, but with \(k = 256\) the SNR is then 76.61 and thus outperforms all the polynomial phase models listed in [14]. The SNR is a growing function of \(k\), and quality is at the expense of computation time. However, large values of \(k\) are needed only for the phase. Indeed, the resampling of the amplitude parameter is not problematic: by resampling only the amplitude with \(k = 32\) and using the ideal phase, the SNR would have been 110.32. The problem is with the imprecision on the reconstructed phase, because of its linear term. Although the error remains very small in percentage, the large values of the unwrapped phase at the end of the partials lead to error in the magnitude of \(\pi\). The synthesized signal is then from time to time put out of phase from the original one, thus the SNR is poor, but the perceived quality is still high. The SNR is indeed an imperfect perceptual metric.

3.4. Preserving the Vibrato

The previously presented time-stretching methods work well on really stationary sounds, but slight variations such as vibrato or tremolo are not properly conserved (see Figure 3). More precisely, their rate is changing with the stretching factor, leading to unnatural sound artifacts. To solve this problem, Artib and Delprat suggested in [18, 19] the use of a hybrid method. They start by doing an analysis of the sound in order to obtain the frequency of each partial and then find its mean frequency curve – the frequency envelope – by using a low-pass FIR filter. Subtracting this frequency
envelope from the frequency of the partial, they extract the vibrato (the modulation). Those two parts (envelope and modulation, illustrated for the amplitude instead of the frequency of the partial on Figures 4(a) and 4(b)) will then be stretched independently. From there, they find the mean frequency for the vibrato and then synthesize a new vibrato of the same frequency and of appropriate length. Adding this newly created vibrato to the resampled frequency envelope, they obtain the desired result: a stretching with vibrato conservation.

We wanted a more accurate vibrato, one that would be very close to the original, both in shape, rate, and depth, implying variations in frequency and amplitude, so that it would sound more natural.

## 4. Modeling the Parameters

On Figure 1, we see that partials issued from a sinusoidal analysis can contain sinusoidal-like components. In musical terms, these oscillations in both frequency and amplitude of a sound are respectively the vibrato and tremolo. Thus, our idea was to re-analyze those partials to extract the sinusoids from the sinusoidal parameters (this idea of re-analyzing the evolutions of the sinusoidal parameters appears for example in [1]). For that purpose, the same technique as presented in Section 2 was used to perform the analysis, simply considering the frequencies and amplitudes of the partials of a sound as regular signals.

In this article, we will denote by “order 0” the level corresponding to the sound signal in the temporal model, that is its amplitude as a function of time – \( a(t) \) in Equation (1). Then, the classical sinusoidal modeling is the order 1: we obtain partials from the order-0 signal. Re-analyzing those partials, and modeling them again with sinusoidal modeling, leads us to “partials of partials”, as illustrated in Figures 5, 6, and 7. This sinusoidal modeling of the parameters of the order-1 sinusoidal model constitutes another level in our hierarchical sinusoidal model: the order 2. In this article, we will stop at the order 2 since we are interested in simple microscopic sound structures such as the vibrato and tremolo, although further levels could help for macroscopic or higher-level musical sound structures.

On Figure 5 we can see the frequency of the second partial of the saxophone sound (plain) as well as the frequencies of the associated order-2 partials (dashed). We can clearly see the vibrato around 5 Hz and the corresponding order-2 partial. Since this vibrato is not perfectly sinusoidal, other order-2 partials (harmonics) are also present. Figure 6 shows the frequency of the order-2 partials of the amplitude of the second partial of the saxophone. This time, we can clearly see the tremolo, with the same frequency as the one of the vibrato. Figure 7 shows the amplitude of the order-2 partials of the amplitude of the second partial of the saxophone. The order-2 partial with the greatest amplitude is the DC component (corresponding to a frequency of 0 Hz, see Section 2), and is very close to the envelope as defined by Arfib and Delprat (see Section 3.4 and Figure 4(a)). However, our version of this envelope is not as good as it should be, because of the main drawback of sinusoidal modeling: the fast variations, such as the attack and decay, are smoothed. The depth of the tremolo can be read on the amplitude of the next order-2 partial. The partial-tracking algorithm is not perfect, and failed around the frame 25 (the order-2 partial was wrongly split into two parts). The amplitudes of the other order-2 partials are very low, because the shape of the tremolo is nearly a perfect sinusoid.

However the settings for the order-2 sinusoidal analysis had to be changed. Indeed, for the order-1 analysis, we considered the “interesting” frequencies to be audible, between 20 and 20000 Hz. Here, the frequencies we are looking for are around 5 or 10 Hz only, and not audible. Hence, the analysis window having to be large enough to contain at least 2 periods of the sinusoids we are looking for, we chose a window size of 64 samples at the sampling frequency of the (order-1) partials, that is 44100/512 \( \approx 86.13 \) Hz. This allows us to take up sinusoids down to a frequency of 2.69 Hz, thus sufficient to take a regular vibrato or tremolo into account. The analysis window was moved by steps of \( H' = 1 \) sample. The maximal frequency difference between two successive frames for each order-2 partial in the partial-tracking algorithm was set to \( \Delta' = 0.2 \) Hz. Partials whose amplitude was always below 0.0001 or length was smaller than 0.2 were considered as noise.

Moreover, to obtain a correct analysis, we usually need extra samples before and after the signal itself. For example, if we center our first analysis window on the first sample, then the first half of this window should be filled with some extrapolated samples. The same problem as the one we faced at the end of Section 2 occurs, hence we used the same solution: extrapolation using the Burg method.

Figure 4: Envelope (a) plus modulation (b) decomposition of the amplitude of the second partial of the saxophone. The envelope is the low-pass filtered version of the amplitude of the partial. The modulation is obtained by subtracting this envelope from the original amplitude (plain curve of the Figures 6 and 7).
The synthesis within the order-2 model consists of two levels.

First, the frequency and amplitude parameters of the (order-1) partials are reconstructed – resynthesized – from the order-2 parameters (phase and amplitude only) using the synthesis technique presented in Section 3.3. This method is shape-invariant, and thus the vibrato and tremolo are kept as close to the original as possible. We obtain partial trajectories almost identical to the original, except for the transients though.

Second, the (order-0) audio signal is synthesized from these synthetic order-1 partials (using their frequency and amplitude) by the technique described in Section 3.1. Note that this technique is not shape-invariant, but for now we can only use the frequencies and amplitudes, not the phases, of the synthetic partials to generate the result. However, as mentioned before, this makes almost no audible difference for most sounds.

Using the classic (order-1) sinusoidal parameters, the time-stretching operation would consist in resampling these parameters to the desired length. This was explained in Section 3, and we concluded that vibrato and tremolo were not conserved. We use the same technique for our enhanced time-stretching, however applied to order-2 partials (i.e. partials of partials) instead of order-1 partials. By resampling order-2 partials to the desired length, we obtain after the synthesis a stretched sound with the original vibrato and tremolo rate. Moreover, since we use a shape-invariant method to synthesize the extended vibrato and tremolo, they are very similar to the original in shape too.

Figure 8 is an example of this enhanced time-stretching on the saxophone sound shown on Figure 1(b). We can see that the tremolo shape and rate on the first figure are very close to the ones of the second, but that the evolution of this tremolo was slowed down, as the global envelope was stretched.

However, the drawback of this method is the drawback of sinusoidal modeling in general, that is the smoothing of sudden changes such as the attack.

6. CONCLUSIONS AND FUTURE WORK

In this article, in the context of sinusoidal modeling, we have considered the frequency, amplitude, and phase parameters of the partials as continuous signals. By considering these signals and not their piecewise-polynomial approximation as it is generally done,
we were able for example to resample them and re-analyze them with the same kind of methods and models that those used for audio signals.

This unified approach lead us to a 2-level sinusoidal model of great interest for digital audio effects, such as time-stretching while preserving not only the pitch but also higher-order sound structures such as the vibrato and tremolo of musical sounds.

Of course, the results presented here are still preliminary. They were computed using a software we developed in Common Lisp. This is a high-level language allowing a level of abstraction necessary to handle easily complex data structures. This software is still in early stages of development, thus no public release is available yet. However, some sound examples – including the saxophone used for the figures of this article – are available online.

In the near future, we will study the possibility of preserving the shape of the signal (order 0) together with the shape of the vibrato and tremolo (order 1). We will investigate hierarchical models of orders greater than 2, allowing us to deal with higher-level musical sound structures. We also intend to take into account transients and noise in the basic sinusoidal model.

7. ACKNOWLEDGMENTS

This research was carried out in the context of the SCRIME project which is funded by the DDMTS of the French Culture Ministry, the Aquitaine Regional Council, the General Council of the Gironde Department and IDDAC of the Gironde Department. The SCRIME project is the result of a cooperation convention between the Conservatoire National de Région of Bordeaux, ENSEIRB (school of research in computer science of Bordeaux). It is made of composers of electro-acoustic music and scientific researchers. It is managed by the LaBRI (laboratory of research in computer science of Bordeaux). Its main goals are research and creation, diffusion and pedagogy.

8. REFERENCES


AN EFFICIENT PHASINESS REDUCTION TECHNIQUE FOR MODERATE AUDIO TIME-SCALE MODIFICATION

David Dorran, Eugene Coyle
Dublin Institute of Technology
Dublin, Ireland.
david.dorran@dit.ie

Robert Lawlor
National University of Ireland
Maynooth, Ireland.
rlawlor@eeng.may.ie

ABSTRACT
Phase vocoder approaches to time-scale modification of audio introduce a reverberant/phasy artifact into the time-scaled output due to a loss in phase coherence between short-time Fourier transform (STFT) bins. Recent improvements to the phase vocoder have reduced the presence of this artifact, however, it remains a problem. A method of time-scaling is presented that results in a further reduction in phasiness, for moderate time-scale factors, by taking advantage of some flexibility that exists in the choice of phase required so as to maintain horizontal phase coherence between related STFT bins. Furthermore, the approach leads to a reduction in computational load within the range of time-scaling factors for which phasiness is reduced.

1. INTRODUCTION

Time-scale modification of audio alters the duration of an audio signal while retaining the signals local frequency content, resulting in the overall effect of speeding up or slowing down the perceived playback rate of a recorded audio signal without affecting the quality, pitch, timbre or naturalness of the original signal. This facility is useful for such applications as enhancement of degraded speech, language and music learning, fast playback for telephone answering machines and audio-video synchronization in broadcasting applications.

The phase vocoder is a popular method for time-scaling audio due to its ability to achieve high quality modifications on a variety of signals within a wide range of time-scaling factors. However, the phase vocoder suffers from an artifact known as phasiness that exists predominantly due to a loss in vertical phase coherence between modified short-time Fourier transform (STFT) bins, as explained in [1]. In [1] an improvement to the phase vocoder is presented that reduces the presence of the phasiness artifact by providing a more accurate estimate of the phase of STFT components in the neighborhood of STFT peaks. However, the artifact remains audible and is particularly objectionable in speech.

This paper presents a technique that offers a further reduction in the phasiness artifact for moderate time-scaling, in the range of ± 10%. The approach takes advantage of a certain amount of flexibility that exists in the choice of phase for modified, time-scaled, STFT bins to achieve horizontal phase coherence, and uses this flexibility to improve upon vertical phase coherence, thus reducing the phasiness effect. Section 2 outlines the operation of a phase vocoder implementation that has the same analysis and synthesis STFT hop size, as used in [2]. Section 3 presents an analysis of horizontal phase coherence under ‘ideal’ conditions, which is then used to determine the amount of flexibility in the phase used so as to maintain horizontal phase coherence. Section 4 demonstrates how the flexibility in the choice of phase can be used to improve vertical phase coherence and outlines the computational benefits associated with the technique. Section 5 discusses the limitations of the approach and the results of informal listening tests. Section 6 concludes this paper.

2. THE PHASE VOCODER

The phase vocoder was first described in [3], with an efficient STFT implementation given in [4]. A tutorial article in [5] provides an excellent insight into the fundamental operation of the phase vocoder and [6] presents some detail of a MATLAB based implementation. The concept and problems of vertical phase coherence are described in detail in [1] and a mathematical description is also provided. In the rest of this Section we briefly outline the phase vocoder and how it can achieve time-scale modification, using the same analysis and synthesis STFT hop size, as used in [2].

The first step is to obtain an STFT representation, \( X(t_m, \Omega_k) \), of the input, as given in [1]

\[
X(t_m, \Omega_k) = \sum_{n=-\infty}^{\infty} h(n) x(n) e^{-j2\pi n t_m \Omega_k}
\]

(1)

where \( x(n) \) is the input signal, \( h(n) \) is the analysis window, \( \Omega_k \) is the center frequency of the \( k^{th} \) vocoder channel and \( n=0 \) is the \( u^{th} \) analysis time instant and \( t_m = u R \), where \( R \) is the rate (and synthesis) hop size and \( u \) is a set of successive integer values, starting at 0.

In [2] time-scale expansion is achieved by appropriately repeating STFT frames e.g. to time-scale by a factor of 1.5 every second frame is repeated, as illustrated in Figure 1; similarly time-scale compression is achieved by omitting frames e.g. to time scale by a factor of 0.9 every tenth analysis frame is omitted. Like traditional implementations of the phase vocoder, the magnitudes of the modified, time-scaled, STFT remains unaltered i.e.

\[
|Y(t_m, \Omega_k)| = |X(t_m, \Omega_k)| \quad \text{for all } k
\]

(2)

where \( n = \text{round}(m/a) \), \( m \) is a set of successive integer values starting at 0, \( t_m \) and \( t_n \) are a set of analysis and synthesis time instants, respectively.

The phases of the modified STFT, \( \angle Y(t_m, \Omega_k) \), are determined so as to maintain both horizontal and vertical phase coherence. To achieve phase coherence, first the peaks representing the dominant components of each frame are detected. In [1] a peak is defined as...
any bin whose magnitude is greater than its four nearest neighbours. In the simplest, most efficient, implementation phases of peaks are updated by maintaining the same phase difference between consecutive synthesis frames that exists between corresponding analysis frames i.e.
\[
∠Y(t_k,Ω) = ∠Ω + ∠Ω − ∠Ω
\]
which becomes
\[
∠Y(t_k,Ω) = ∠Y(t_{k+1},Ω) + ∠Ω − ∠Ω
\]
for all \( k \).

Having determined the phases of the synthesis peaks, the phases of bins in each peak’s region of influence are updated by maintaining the same phase difference between peaks and the bins in their region of influence that exists in the mapped analysis frame. In [1] the upper limit of the region of influence of a peak is set to the middle frequency between that peak and the next one. Then
\[
∠Y(t_k,Ω) = ∠Y(t_{k+1},Ω) + ∠Ω − ∠Ω
\]
for all \( k \) in each peak’s region of influence.

A better method for updating phases requires sinusoidal modeling based peak tracking, as explained in [1], however, no advantage was found in using a peak tracking approach when employing the phasiness reduction technique, described later in Section 4, in the range of time-scale factors for which the techniques offer a significant improvement i.e. 0.9-1.1.

A time-scaled version of the original signal is obtained by calculating the inverse STFT of \( Y(t_m,Ω) \).

The inverse STFT of a given STFT is found by calculating the inverse discrete Fourier transform (IDFT) of each STFT frame. Successive inverse STFT frames are then overlapped and added together to produce the time-domain signal. A single iteration of the overlap and add process is illustrated in the upper three waveforms of Figure 2, where two frames of a sinusoidal signal are overlapped and summed together to reproduce a perfect sinusoid. Now consider the case where the overlapping frames are no longer overlapped and summed together to produce the time-domain signal. A single iteration of the overlap and add process is illustrated in the upper three waveforms of Figure 2, where two frames of a sinusoidal signal are overlapped and summed together to reproduce a perfect sinusoid.

The first step in achieving this aim is to describe the above situation through the use of a vector representation. From Figure 3, the ramped sinusoidal components are represented by the vectors \( a(t) \) and \( b(t) \), which vary with time, according to the ramping function, but are constantly separated in phase by \( θ \), and which sum to produce vector \( c(t) \).

\[
|c(t)| = |a(t)| + |b(t)|
\]

where \( C = π - θ \) radians.

Typically, a hanning window is used within a phase vocoder implementation, therefore, if the magnitude of the original sinusoid is normalized to one, \( |a(t)| \) is given by
\[
|a(t)| = 0.5(\cos(πt / L) + 1)
\]

where \( L \) is the duration of the overlap and \( 0 ≤ t ≤ L \). The sum of \( |b(t)| \) and \( |a(t)| \) must be one for perfect reconstruction, therefore
\[
|b(t)| = 1 - |a(t)|
\]

To determine the maximum phase difference that can be introduced without introducing audible distortion a set of equations representing the situation described above is derived.

The first step in achieving this aim is to describe the above situation through the use of a vector representation. From Figure 3, the ramped sinusoidal components are represented by the vectors \( a(t) \) and \( b(t) \), which vary with time, according to the ramping function, but are constantly separated in phase by \( θ \), and which sum to produce vector \( c(t) \).
Since \( t = L/2 \) provides the only non trivial solution. Therefore, the maximum amplitude variation is given by

\[
1 - |c(L/2)| = 1 - \sqrt{0.5^2 + 0.5^2 - 2(0.5)(0.5)\cos \theta}
\]

\[
= 1 - \sqrt{0.5 + 0.5 \cos \theta}
\]

(10a)

since the magnitude of the original sinusoid has been normalized to one, \( C = \pi - \theta \) radians and \( |c(L/2)| = 0.5 \).

From [7], the human ear is insensitive to amplitude variations of tones, introduced by sinusoidal amplitude modulation, for degrees of modulation that are less than 2% for tones that are less than 80dB. It is important to note that the total variation in amplitude from a maximum to a minimum is twice the degree of modulation. This value varies significantly with pressure levels, for example for a pure tone of pressure level 40dB the degree of modulation increases to 4% while at 100dB it decreases to 1%. These values are independent of the frequency of the tone. It should also be noted that, from [7], these values are dependent on the frequency of modulation, but the values given above are based on the modulating frequency at which human hearing is most sensitive. Also, for white noise the degree of modulation tolerated is 4% for pressure levels greater than 30dB. It can be shown that the amplitude modulation of \( c(t) \) is quasi-sinusoidal in nature, with the degree of modulation, \( D_m \), given by, from equation (10b)

\[
D_m = \left| 1 - \sqrt{0.5 + 0.5 \cos \theta} \right| / 2
\]

(11)

where the divisor of 2 is required since the degree of modulation is half the total variation in amplitude.

By making the assumption that maximum pressure levels of total components of the signals being analysed are below 80dB, the degree of modulation of \( c(t) \) must then be kept below 2%. So, from equation (11)

\[
\left| 1 - \sqrt{0.5 + 0.5 \cos \theta} \right| / 2 \leq 0.02 \text{ radians}
\]

(12)

Therefore

\[
\theta \leq 0.5676 \text{ radians}
\]

(13)

to ensure no perceivable amplitude modulations are introduced.

It should be noted that the amplitude modulation introduced results in an average decrease in signal amplitude level, however, the decrease is within the just noticeable amplitude level difference, as given in [7], if equation (13) is satisfied.

\( B(t) \) represents the time-varying phase variation between \( a(t) \) and \( c(t) \) and, from the well known sine-rule, is given by

\[
B(t) = \sin^{-1}\left( \frac{B(t) \sin C}{|c(t)|} \right)
\]

(14)

then

\[
\frac{dB(t)}{dt} = C \left( \sqrt{1 - \frac{(a(t))^2}{|c(t)|^2}} \right) \frac{\frac{\partial (|c(t)|)}{dt}}{\frac{\partial |c(t)|}{dt}}
\]

(15)

The frequency \( f_c \) of the quasi-sinusoidal component \( c(t) \) is given by

\[
f_c = f_x + \frac{df(t)}{dt} \text{ rads/second}
\]

(16)

where \( f_x \) is the frequency of the sinusoidal component \( a(t) \).

Since \( f_x \) is constant, the derivative of the \( B(t) \) with respect to \( t \) represents the frequency modulating component of \( f_c \). The maximum frequency modulation is determined by first finding the derivative of \( f_c \) with respect to \( t \), setting it to zero and solving for \( t \). Then

\[
\frac{d^2 B(t)}{dt^2} = \frac{\pi}{L} \tan \left( \frac{\theta}{2} \right)
\]

(17)

and when (17) is set to zero it can, once again, be shown that \( t = L/2 \) provides the only non trivial solution. Therefore, it can be shown that the maximum frequency deviation is given by

\[
\frac{dB(L/2)}{dt} = \frac{\pi}{L} \tan \left( \frac{\theta}{2} \right)
\]

(18)

Also from [7], the human ear is insensitive to frequency variations introduced by frequency modulation; for tones greater than 500Hz, modulations less than 0.7% are not perceived and for tones less than 500Hz, a fixed modulation of 3.6Hz is tolerated. Once again, these values are dependent on the frequency of modulation, however the values given above are based on the modulating frequency at which the human ear is most sensitive. Therefore, in order to ensure the ear does not perceive distortion for any frequency, the variation of \( f_c \) must be kept below 3.6Hz or 22.62 radians/second. So, from equation (18) and setting \( L = 23.22m \), which corresponds to half the length of a 2048 point window at a sampling frequency of 44.1kHz.

\[
\frac{\pi}{0.0232} \tan \left( \frac{\theta}{2} \right) \leq 22.62 \text{ radians}
\]

(19)

Then

\[
\theta \leq 0.3313 \text{ radians}
\]

(20)

From (13) and (20) the maximum phase deviation, \( \Psi_{\max} \), that can be introduced without introducing audible modulations is

\[
\Psi_{\max} = 0.3313 \text{ radians}
\]

(21)

This value only strictly applies to frequencies less than 500Hz, if the dependence of modulations on frequency is considered then \( \Psi_{\max} \) could be increased to 0.5676 radians for frequencies greater than

\[
\frac{\pi}{0.0232} \tan \left( \frac{0.5676}{2} \right) 2\pi = 897.23Hz
\]

(22)

and varied accordingly between 0.3313 and 0.5767 radians for all other frequencies.

The above analysis is carried out based on a single pure sinusoidal tone, however, most audio signals of interest are, for the most part, a sum of quasi-sinusoidal components, a feature exploited by sinusoidal modeling techniques [8] and is the underlying assumption of the phase vocoder. It is assumed that the sum of sinusoids that have been amplitude and frequency modulated to the maximum limit, such that they are perceptually equivalent to the original individual sinusoids, results in a signal that is perceptually equivalent to the sum of the non-modulated sinusoids. Informal listening tests in a quiet office environment support this assumption.

The above analysis is also based on an ‘ideal’ horizontal phase shift i.e. vertical phase coherence is maintained. Such a phase shift is easy to achieve with synthesized pure sinusoids but is difficult with real audio signals; this difficulty is, of course, the reason for the existence of the phaselessness artifact in the first place. However, the above analysis does suggest that a certain amount of flexibility exists in the choice of phase in order to maintain horizontal phase coherence of dominant sinusoidal components. This is further supported by the fact that phase vocoder implementations are capable of producing high quality time-scale modifications even though frequency estimates, used in [1] to determine synthesis phases, are prone to inaccuracies [9], [10].
The derivation of amplitude and frequency modulations introduced due to phase deviation was based on a hop size of half the analysis window length. A similar, albeit more tedious, approach can be used to determine modulations introduced for the case of different hop sizes; a hop size of half the analysis window length is used in this Section for its intuitive appeal and mathematical simplicity. Another commonly used hop size is one quarter of the analysis frame length, for which it can be shown that $\Psi_{\text{max}} \approx 0.24$ radians for analysis window lengths of 46.44 ms.

4. REDUCTION IN PHASINESS AND COMPUTATIONS

In the previous Section it was shown that a certain amount of flexibility exists in the choice of phase required to achieve horizontal phase coherence within a phase vocoder implementation. This flexibility can be used to ‘push’ or ‘pull’ modified STFT frames into a phase coherent state; however a set of coherent target phases for each frame are first required. One set of target phases that would guarantee vertical phase coherence are the phases of the original frames that are mapped to each synthesis frame. So, having determined an estimate of the synthesis phases using the procedure described in Section 2, the synthesis phases are updated further using the following rules:

If

$$\text{princ}_\text{-} \arg \left( \angle Y(t_n, \Omega_k) - \angle X(t_n, \Omega_k) \right) \leq \Psi_{\text{max}}$$

(23a)

then

$$\angle Y(t_n, \Omega_k) = \angle X(t_n, \Omega_k)$$

(23b)

else

$$\angle Y(t_n, \Omega_k) = \angle Y(t_n, \Omega_k) + \text{sign} \left( \text{princ}_\text{-} \arg \left( \angle Y(t_n, \Omega_k) - \angle X(t_n, \Omega_k) \right) \right)$$

(23c)

where $\Psi_{\text{max}}$ is the maximum deviation in phase, as determined in Section 3, $\text{sign}$ is a function that returns the sign of the submitted value i.e. 1 or $-1$ and $\text{princ}_\text{-} \arg$ returns the principle argument of the submitted value between $\pm \pi$.

For the following paragraphs it is important to be aware of two situations; the first situation is where consecutive analysis frames are mapped to consecutive synthesis frames e.g. in Figure 1 the consecutive analysis frames 2, 3 and 4 are mapped to three consecutive synthesis frames 3', 4' and 5', this case can be described more generally as the situation when $t_n \rightarrow t_n$ and $t_{n+1} \rightarrow t_{n+1}$; the second situation covers all other cases.

It should be noted that for the case where consecutive analysis frames are not mapped to consecutive synthesis frames, $\Psi_{\text{max}}$ should be reduced to take the likelihood of increased inaccuracies of phase estimates into consideration when using equation (4). Phase estimates of consecutive analysis frames that are mapped to consecutive synthesis frames are likely to be accurate, at least for peaks, since the same phase differences are kept between consecutive analysis frames as consecutive synthesis frames; the same cannot be said for the case where consecutive analysis frames are not mapped to consecutive synthesis frames. It is difficult to determine a precise figure for the inaccuracy of the phase estimate; consequently it is difficult to determine a value for the maximum phase deviation that can be introduced. From experimentation it was found that reducing $\Psi_{\text{max}}$ to $\Psi_{\text{max}}/2$ is an adequate choice.

It should also be noted that, for the case where multiple consecutive analysis frames are mapped to multiple consecutive synthesis frames, a reduction in phase differences between one synthesis frame and its corresponding, mapped, analysis frame results in the same phase reduction for all consecutive synthesis frames that follow; since from equation (4) the phase modifications are propagated through the remaining synthesis frames. Following from this observation, it can be noted that if $(\pi - \Psi_{\text{max}}/2)/\Psi_{\text{max}}$ consecutive analysis frames are mapped to $(\pi - \Psi_{\text{max}}/2)/\Psi_{\text{max}}$ consecutive synthesis frames the phase coherence is guaranteed to be recovered for at least one of the consecutive synthesis frames (the $\Psi_{\text{max}}/2$ value represents the phase deviation introduced for non-consecutive synthesis frames). Therefore, the closer the time-scale factor is to one the greater the opportunity to recover phase coherence, since the number of consecutive analysis frames mapped to consecutive synthesis frames, $k$, is given by

$$k = 1/|1-\alpha|$$

(24)

It then follows that phase coherence is guaranteed to be recovered at least once every $k$ frames if

$$\alpha > (\pi - 3 \Psi_{\text{max}}/2)/ (\Psi_{\text{max}}/2 - \pi)$$

for $\alpha < 1$ (25a)

or

$$\alpha < (\pi + 3 \Psi_{\text{max}}/2)/ (\pi - \Psi_{\text{max}}/2)$$

for $\alpha > 1$ (25b)

Since phase coherence is ensured for some sections of the time-scaled output if equation (25a) or (25b) is satisfied, it follows that these sections are copies of the sections of the input. Therefore, these ‘copied’ sections do not have to be processed in the frequency domain and can be simply overlapped and added to the time-scaled output; resulting in a reduction in the computational requirements of the approach. This process is illustrated in Figure 4, where the analysis frame marked B would achieve phase coherence and the synthesis frame marked A' is almost phase coherent i.e. all STFT bins of frame A' are within $\Psi_{\text{max}}$ radians of the phase of the mapped analysis frame marked A.

![Figure 4: Copying a time-domain segment to the output](image)

The phases of the analysis frame marked C are required to calculate equation (4), therefore, given a set of analysis time instants $t_n = uR$, where $u$ is a set of consecutive integer values starting at 0, the STFT needs only be calculated, at most, for the cases when $floor(u|1-\alpha|)|1-\alpha| - 1 \leq u \leq floor(u|1-\alpha|)|1-\alpha| + ceil((\pi - \Psi_{\text{max}}/2)/\Psi_{\text{max}})$

(26)

where $\text{ceil}$ and $\text{floor}$ are functions that return the nearest integer greater than and less than the value submitted, respectively.

Equation (26) provides the maximum number of analysis time instants at which the STFT must be calculated to ensure phase coherence. Further computational savings can be achieved by recognizing that phase coherence can be achieved at any frame...
within a set of \((\pi - \Psi_{\text{max}}/2)\Psi_{\text{max}}\), consecutive synthesis frames. So, given that the synthesis frame mapped to the analysis frame at the analysis time instant \(R/(\lfloor u(1-\alpha)\rfloor - 1) + h\) is almost phase coherent i.e. all bins are within \(\Psi_{\text{max}}\) radians of the phase of the mapped analysis frame, then no frequency domain processing is required at the analysis time instants, \(uR\), for \(u\) in the range

\[
\lfloor u(1-\alpha)\rfloor - 1 + h < u < \lfloor (u+1)(1-\alpha)\rfloor / (1-\alpha)
\]

where \(h\) is an integer less than \(1/|1-\alpha|\).

By making the assumption that all computations other than calculating the STFT and Inverse STFT are negligible, Figure 5 illustrates the computational advantage of the phasiness reduction technique; the vertical axis shows the ratio of computations of the standard phase vocoder to the computations of the phase vocoder that utilizes the phasiness reduction technique described in this paper. The solid line is plotted for \(\Psi_{\text{max}} = 0.3313\) radians and the dashed line is plotted for \(\Psi_{\text{max}} = 0.24\) radians.

Figure 5: Computational advantage of the technique

5. SUBJECTIVE TESTING AND DISCUSSION

Eight test subjects undertook a number of subjective listening tests. The results indicate that the improvement in the quality of time-scaled output achieved by using this approach is most effective for time-scale factors close to one with a significant improvement noticed for moderate time-scale factors in the range 0.9-1.1. Beyond this limit, the reduction in phasiness is less significant and no improvement in quality was perceived for time-scale factors outside the range 0.85-1.15. The results also indicate a greater improvement for speech signals, due to the fact that the phasiness artifact is more objectionable in speech to begin with. Phasiness appears to be more objectionable in speech because reverberation, which is similar to phasiness, is not often noticeably present in a speech signal, so when it is inadvertently introduced it tends to be obvious; whereas in music reverberation is often noticeably present, and is even synthetically added to music recordings, consequently, when additional reverberation, or phasiness, is introduced into a music signal it is less obvious and therefore less objectionable. The reduction in phasiness is also particularly noticeable in gravelly type speech. This was attributed to the fact that the phase update procedure proposed in [1] is most applicable to signals composed of strong sinusoidal components and gravelly speech seems to violate this model to a greater degree than other types of speech.

Figure 6 illustrates the effects of the phasiness reduction technique on a speech signal. It should be noted that while the preservation of the waveform shape, i.e. shape invariance, does not ensure phase coherence, the loss of shape invariance can be attributed to a loss of phase coherence.

The range of time-scale factors over which the technique has a significant reduction in phasiness is quite restrictive for many applications, however, it is ideally suited to such applications as audio-video synchronization in broadcasting application, which require time-scale modifications in the range 24/25-25/24 [11].

The phasiness reduction technique described in this paper has similarities with time-domain approaches [12], in that, for moderate time-scaling, certain segments of the time-scaled signal are a copy of the original, as is the case in time-domain approaches, the phase vocoder, however, has the advantage of producing better results for complex polyphonic audio. The technique also has similarities to the synchronised time-domain/subband approach described in [13], where individual subbands are ‘pulled’ or ‘pushed’ into a synchronised state by taking advantage of some psychoacoustic properties.

Figure 6: The effects of the reduction of phasiness

6. CONCLUSION

Time-scale modification of audio using phase vocoder based approaches require both horizontal and vertical phase coherence between modified STFT bins to produce a high quality output. In this paper it is shown that some flexibility exists in the choice of phase required to ensure horizontal phase coherence, when psychoacoustic properties are considered. This flexibility in horizontal phase is then used to ‘push’ or ‘pull’ the modified STFT into a phase coherent state, resulting in a reduction in the phasiness artifact associated with phase vocoder time-scaling implementations, for moderate time-scale factors in the range 0.9-1.1. It is also shown that the phasiness reduction technique results in a significant reduction in computational overhead for moderate time-scaling.

7. REFERENCES


A PIANO MODEL INCLUDING LONGITUDINAL STRING VIBRATIONS

Balázs Bank and László Sujbert

Department of Measurement and Information Systems
Budapest University of Technology and Economics
{bank|sujbert}@mit.bme.hu

ABSTRACT

In this paper a mixed-paradigm piano model is presented. The major development is the ability of modeling longitudinal string vibrations. Longitudinal string motion is the reason for the metallic sound of low piano notes, therefore its modeling greatly improves the perceptual quality of synthesized piano sound. In this novel approach the transversal displacement of the string is computed by a finite-difference string model and the longitudinal motion is calculated by a set of second-order resonators, which are nonlinearly excited by the transversal vibration. The soundboard is modeled by a multi-rate filter based on measurements of real pianos. The piano model is able to produce high-quality piano sounds in real-time with about 5–10 note polyphony on an average personal computer.

1. INTRODUCTION

As the piano has hundreds of strings, the bottleneck of piano modeling lies in the strings. For string modeling the digital waveguide [1] is by far the most efficient approach. Instead of discretizing the wave equation, it discretizes its traveling-wave solution. Since in most of the cases the string motion has to be computed correctly at the excitation and observation points only, the model is reduced to a delay line and a low-order ($N = 10..20$) filter in a feedback loop.

Already high-quality real-time physical models of the piano have been presented based on digital waveguide modeling [2, 3, 4, 5]. However, in these models the effect of longitudinal vibrations was neglected. The quality of these models can be greatly improved by including the longitudinal motion of piano strings. In the low range of real pianos the pitch of the longitudinal components can be perceived by the listener, and the subjective quality of the instrument is highly dependent on the frequency of these modes [6], pointing out that the longitudinal string motion has an important perceptual effect.

The longitudinal vibration of piano strings is made up of the free vibration of longitudinal modes and the forced motion excited by the transversal displacement. The spectral peaks corresponding to the forced motion are called “phantom partials” [6]. We have presented a detailed analysis on how these partials are generated in [7]. Fig. 1 shows an extract of the spectrum of a $G_1$ piano note, recorded at 2 m distance from the piano. The phantom partials are clearly visible between the transversal partial series, so is one longitudinal mode (marked by circle), which has even larger amplitude than the neighboring transversal ones.

Accordingly, it would be highly beneficial to incorporate the longitudinal modes in the efficient digital-waveguide based piano models. Borin [8] have amended his real-time piano model with independent digital waveguides for the longitudinal polarization. We have also made experiments with similar solutions [7]. In these models the longitudinal modes are excited during the hammer-string contact only, therefore the forced longitudinal motion is not simulated. These simple models capture some aspects of low piano tones, but sound unnatural. The longitudinal modes sound separated from the transversal ones, unlike in real piano sounds and in finite-difference simulations. We suppose that the key to having coherence between transversal and longitudinal components is that the longitudinal vibration is continuously excited by the transversal one.

In the next Sections, we will present a new approach to string modeling, which is able to model the continuous interaction between the transversal and longitudinal polarizations efficiently. After the derivation of the basic equations a finite-difference model is described. This is followed by the new approach, which is basically the simplification of the finite-difference model, having the same perceptual quality at around 10% computational cost. Then, parameter estimation techniques are described, and simulations presented. Possible directions of future research conclude the paper.

Figure 1: Spectrum of the first second of a $G_1$ piano note (forte playing) between 1 and 2 kHz. A prominent longitudinal mode is marked by a circle.
2. THE BASIC EQUATIONS

A real piano string is vibrating in two transversal planes, and in the longitudinal direction as well. Principally, piano hammers excite one transversal polarization of the string, the other two are gaining energy through coupling. These polarizations interact with each other as a result of nonlinear behavior of the string.

For simplicity, let us assume that the string is vibrating in one plane, thus, one transversal and one longitudinal polarization is present. We will see later that modeling these two polarizations can only excite the longitudinal vibration if the square of the string slope is significant, i.e., the transversal displacement is relatively large.

Accordingly, the longitudinal vibration is approximately described by the following equation:

$$\mu \frac{\partial^2 \xi}{\partial t^2} = ES \frac{\partial^2 \xi}{\partial x^2} + \frac{1}{2} ES \frac{\partial \left( \frac{\partial \xi}{\partial x} \right)^2}{\partial x}$$

which is again a standard wave equation with an additional force term depending on the transversal vibration of the string according to a second-order nonlinearity. Note that the transversal string motion can only excite the longitudinal vibration if the square of the string slope is significant, i.e., the transversal displacement is relatively large.

After similar derivations, the wave equation for the transversal motion can be written as follows:

$$\mu \frac{\partial^2 y}{\partial t^2} = T_0 \frac{\partial^2 y}{\partial x^2} + ES \frac{\partial \left( \frac{\partial y}{\partial x} \right)^2}{\partial x}$$

which is again a standard wave equation with an additional force term depending on the product of transversal and longitudinal string slope. Consequently, the longitudinal vibration influences the transversal one at large displacements only.

By looking at Eqs. (6) and (7) we can conclude that the coupling of transversal and longitudinal string motion depends on the magnitude of vibration according to a square law and that the coupling is bi-directional.

3. FINITE-DIFFERENCE MODELING

A straightforward choice for computing the solution of the above equations is finite-difference modeling, but first the partial differential equations Eqs. (6) and (7) have to be extended by additional terms. The equation for transversal vibration including dispersion and frequency-dependent losses takes the form:

$$\frac{\partial^2 y}{\partial t^2} = T_0 \frac{\partial^2 y}{\partial x^2} + ES \kappa \frac{\partial^2 y}{\partial x^2} - 2b_1 \frac{\partial y}{\partial t} + 2b_2 \mu \frac{\partial^2 y}{\partial^2 x \partial t} + ES \frac{\partial \left( \frac{\partial y}{\partial x} \right)^2}{\partial x}$$

where $y = y(x,t)$ and $\xi = \xi(x,t)$ are the transversal and longitudinal displacements of the string with respect to time $t$ and space $x$. As the length of the element changes varies the tension $T = T(x,t)$ which equals to $T_0$ at rest) of the string according to the Hooke’s law:

$$T = T_0 + ES \left( \frac{ds}{dx} - 1 \right)$$

where $E$ is the Young’s modulus and $S$ is the cross-section area of the string. By substituting Eq. (2) into Eq. (3) the string tension can be approximated as:

$$T \approx T_0 + ES \left( \frac{\partial \xi}{\partial x} + \frac{1}{2} \left( \frac{\partial y}{\partial x} \right)^2 \right)$$

As the segment $ds$ is nearly parallel to the x axis, the longitudinal force on the segment $ds$ is the difference of the tension at the sides of the segment:

$$F_l \approx \frac{\partial T}{\partial x} dx \approx ES \left( \frac{\partial^2 \xi}{\partial x^2} + \frac{1}{2} \left( \frac{\partial y}{\partial x} \right)^2 \right) dx$$

This force acts on a mass $\mu dx$, where $\mu$ is the mass per unit length. Accordingly, the longitudinal vibration is approximately described by the following equation:

$$\mu \frac{\partial^2 \xi}{\partial t^2} = ES \frac{\partial^2 \xi}{\partial x^2} + \frac{1}{2} ES \frac{\partial \left( \frac{\partial \xi}{\partial x} \right)^2}{\partial x}$$

which is the standard wave equation with an additional force term depending on the transversal vibration of the string according to a second-order nonlinearity. Note that the transversal string motion can only excite the longitudinal vibration if the square of the string slope is significant, i.e., the transversal displacement is relatively large.

Likewise, the equation for longitudinal vibration has to be completed by frequency-dependent loss terms similar to what was used for transversal vibration in [5, 10]:

$$\frac{\partial^2 \xi}{\partial t^2} = ES \frac{\partial^2 \xi}{\partial x^2} + 2b_1 \mu \frac{\partial \xi}{\partial t} + 2b_2 \mu \frac{\partial^2 \xi}{\partial x \partial t} + \frac{1}{2} ES \frac{\partial \left( \frac{\partial \xi}{\partial x} \right)^2}{\partial x}$$

which is again a standard wave equation with an additional force term depending on the product of transversal and longitudinal string slope. Consequently, the longitudinal vibration influences the transversal one at large displacements only.

By looking at Eqs. (6) and (7) we can conclude that the coupling of transversal and longitudinal string motion depends on the magnitude of vibration according to a square law and that the coupling is bi-directional.
where $b_{11}$ and $b_{22}$ set the decay times of the longitudinal modes in the same way as $b_1$ and $b_2$ for the transversal vibration (see Eq. (9)). The longitudinal modal frequencies are not in a perfect harmonic series in real pianos [7]. The simplest (although not physically meaningful) way of achieving this effect in the model is having a not uniform mass density $\mu(x)$ in Eq. (10) along the dimension $x$.

As the string is assumed to be hinged at both ends, the corresponding boundary conditions become [11]:

$$y(0, t) = y(L, t) = \xi(0, t) = \xi(L, t) = 0$$

$$\left. \frac{\partial^2 y(x, t)}{\partial x^2} \right|_{x=0} = \left. \frac{\partial^2 y(x, t)}{\partial x^2} \right|_{x=L} = 0$$

(11)

The solution of the partial differential equations is computed on a grid $x_m = m\Delta x, t_n = n\Delta t$ by substituting the differentials by finite differences:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x_{m-1}, t_n) - 2u(x_m, t_n) + u(x_{m+1}, t_n)}{\Delta x^2}$$

$$\frac{\partial^2 u}{\partial t^2} \approx \frac{u(x_m, t_{n-1}) - 2u(x_m, t_n) + u(x_m, t_{n+1})}{\Delta t^2}$$

$$\left. \frac{\partial u}{\partial t} \right|_{x_m, t_n} \approx \frac{u(x_m, t_n) - u(x_m, t_{n-1})}{\Delta t}$$

(12)

where $u$ is used as a general notation for either $\xi$ or $y$. The fourth-order term in Eq. (8) is computed by using the first line of Eq. (12) twice. The string model is then connected to a simple finite-difference hammer model simulating the hammer-string interaction [11]. The resulting finite-difference equations are explicit, meaning that the next values of $y(x_m, t_n)$ and $\xi(x_m, t_n)$ can be easily computed from the previous values without the need of iterations.

We have developed such a model consisting of 100 string elements [7]. The finite-difference string model produces high sound quality, but requires a large amount of computation. It is mainly because of the high traveling speed $c_1$ in the longitudinal direction, which makes large sampling rates (e.g., $f_s = 500$ kHz) necessary in order to avoid numerical instability. With today’s personal computers this would mean one note polyphony (i.e., monophony). However, for experimental purposes, this kind of approach is still beneficial. For example, a commercial computer program based on similar principles was written by Berghard [12], to help piano builders in scale design.

A complete finite-difference string model made it possible to experiment whether it is reasonable to neglect the coupling from longitudinal to transversal motion or not. We have found that although the produced waveforms are slightly different, the perceptual difference is insignificant. In general, this means that it is enough to model the coupling from transversal to longitudinal vibrations, allowing large simplifications, which will be presented in Sec. 4.

However, if the transversal-to-longitudinal excitation force (the rightmost term of Eq. (6)) has strong peaks at the longitudinal modal frequencies, the longitudinal modes may reach extreme amplitudes if the longitudinal-to-transversal coupling is neglected. This would not happen if the coupling from longitudinal motion to the transversal one was also realized, since in that case the longitudinal mode would diminish the amplitude of those transversal partials from where it originates [7]. On the other hand, piano builders try to avoid these constellations anyway, therefore we can do the same by setting the longitudinal modal frequencies different from the peaks of the transversal-to-longitudinal excitation force (see also Sec. 5).

4. THE NEW APPROACH

The starting point of our composite model is the finite-difference approach described in Sec. 3, as the transversal displacement needs to be known precisely for each point along the string for computing the transversal-to-longitudinal coupling precisely.

The basic idea allowing the simplification of the original finite-difference string model described in Sec. 3 is that the longitudinal displacement should be known at the termination only, since the feedback from the longitudinal motion to the transversal one is neglected. Therefore, there is no need for a finite-difference model for computing longitudinal vibrations, which eliminates the problem of high (e.g., 500 kHz) sampling rates.

Accordingly, the transversal string displacement $y(x, t)$ is computed by a finite-difference model similar to the Eq. (8) excluding the coupling from longitudinal vibrations at audio sampling rate ($f_s = 44.1$ kHz). On the contrary, the longitudinal motion is described by its modal form [13]:

$$\xi(x, t) = \sum_{k=1}^{N} a_k \sin\left(\frac{kx}{L}\right)\cos(2\pi f_k t)e^{-\frac{\pi k}{192}}$$

(13)

where $k$ is the mode number, $N$ is the total number of modes to be computed, $L$ is the length of the string, $a_k$, $f_k$, and $\tau_k$ are the amplitude, frequency, and decay time of mode $k$, respectively. The force at the bridge $F_{\text{br}, k}(t)$ can be approximated by:

$$F_{\text{br}, k}(t) \approx ES \left. \frac{\partial \xi(x, t)}{\partial x} \right|_{x=L} = \frac{\pi}{L} \sum_{k=1}^{N} a_k (-1)^{k-1} \cos(2\pi f_k t)e^{-\frac{\pi k}{192}}$$

(14)

which can be implemented as second-order resonators in parallel.

Eq. (14) describes only the excitation-free motion of the longitudinal modes. Regarding the forced motion, first the excitation force distribution $F_{\text{i}, \text{exc}, k}(x, t)$ is computed from the transversal displacement according to Eq. (6):

$$F_{\text{i}, \text{exc}, k}(x, t) = \frac{1}{2} ES \left. \frac{\partial \left( \frac{\partial u(x, t)}{\partial x} \right)^2}{\partial x} \right|_{x=L}$$

(15)

Then, the force input of mode $k$ is calculated by the following way:

$$F_{\text{i}, k}(t) = \int_{x=0}^{x=L} \sin\left(\frac{kx}{L}\right)F_{\text{i}, \text{exc}, k}(x, t)dx$$

(16)

which is the scalar product of the force distribution $F_{\text{i}, \text{exc}, k}(x, t)$ and the modal shape of mode $k$ [13].

The equations were presented in continuous time for clarity. In the synthesis model the differential and integral operations are substituted by finite difference and summation. The computationally heavy part of longitudinal-vibration simulation lies in Eqs. (15) and (16). Especially the load of Eq. (16) is heavy, since it means that the force input $F_{\text{i}, k}(t)$ is computed for all the modes ($N \approx 10$ in practice) separately. Therefore, further simplifications are necessary.
The excitation spectrum (the Fourier transform of $F_{l,k}(t)$) of all the odd and all the even longitudinal modes are very similar, respectively. It can be seen in Figs. 3 and 4, that the only difference is that the frequency peaks are slightly shifted as a function of mode number $k$ because of the inharmonicity of the string [7]. The amplitudes are also somewhat different, but the general envelopes are of quite similar structure. Therefore, it is a logical choice to substitute the excitation force $F_{l,k}(t)$ of all the odd longitudinal modes by the excitation force of one odd longitudinal mode (e.g., $F_{l,k}(t) = F_{l,6}(t)$ for odd $k$). The same can be done for the even longitudinal modes. However, it is important to incorporate at least one odd and one even modal shape, since odd longitudinal modes give a rise to odd phantom partials, and even modes to even phantoms. Having only one modal shape in the model would lead to an excitation spectrum with odd or even harmonics only. Accordingly, the model can be simplified by computing the force input for two modes (e.g., $F_{res} = F_{l,5} + F_{l,6}$, but any other odd and even mode would do) and using this as a common excitation for all the resonators. This leads to almost identical perceptual results compared to the full model of Eq. (16) for $k = 1...N$.

The string model is depicted in Fig. 5. It can be seen that the transversal string displacement is computed by a finite-difference string model, which is excited by a finite-difference hammer model [14]. The transversal force $F_l$ at the bridge is the force at the right-hand side termination.

Then the excitation force of the resonators $F_{res}$ is computed by squaring the string slope at each point, differentiating along the dimension $x$ (i.e., approximating Eq. (15)) and computing a scalar product with the modal shape of two consecutive longitudinal modes (similarly to Eq. (16)). The longitudinal force at the bridge is then computed by feeding the excitation signal to a resonator bank $R_l(z)...R_N(z)$. This signal is filtered by $H_l(z)$ to take into account that the soundboard has a different response to longitudinal bridge deflection compared to the transversal one. We have found that already a simple differentiation $H_l(z) = 1 - z^{-1}$ produces good results.

The force signals of the two polarization $F_l$ and $F_t$ are added and sent to a soundboard model based on multi-rate filtering [4]. The soundboard model is depicted in Fig. 6. The string signal is split into two parts: the signal below 2.2 kHz is downsampled by a factor of 8 and filtered by a 4000 tap length at $f_s/8$ FIR filter $H_{low}(z)$ precisely synthesizing the amplitude and phase response of the soundboard for the low frequencies. The signal above 2.2 kHz is filtered by a 1000 tap (ca. 20 ms at $f_s = 44.1$ kHz) FIR filter. This simplification in the high-frequency chain can be done because the higher modes of the soundboard decay faster than the lower ones, while the ear is also less sensitive in this region. The signal of the high frequency chain is delayed by $N$ samples to compensate for the latency of decimation and interpolation filters of the low frequency chain. The sound produced by this model is indistinguishable from that calculated by a 16000 tap FIR filter directly implementing the soundboard impulse response.
5. PARAMETER ESTIMATION

The parameters of the hammer model are taken from [14]. The lengths of the strings are measured on a real piano. Instead of measuring the mass of the string, the mass density $\mu$ is computed from the tension $T_0$ (set to 700 N), the length, and the fundamental frequency. String inharmonicity and the loss parameters of the transversal vibration are estimated from recorded piano tones by polynomial regression. As for the losses, the method is basically the first step of the one-pole filter design method presented in [15].

The frequencies of the longitudinal modes (i.e., the frequencies of $R_1, \ldots, R_2$) can be set according to the spectra of real piano tones. However, the peaks of longitudinal modes cannot be easily found between the transversal ones automatically, which results in a huge amount of work. Alternatively, the longitudinal modal frequencies can be set in a way that they should correspond to a note which is in harmonic relationship to the transversal vibration (e.g., the longitudinal component sounds four octaves higher than the transversal one). This is what piano builders wish to achieve in real pianos as well [6]. However, these frequencies should not lie on the peaks of the excitation force $F_{res}$ (i.e., the solid lines in Figs. 3 and 3), since that would lead to undesirable ringing. This can be done automatically by computing the spectrum of the excitation signal $F_{res}$ and shifting those longitudinal mode frequencies which are too close to some of the peaks. The decay time of the resonators were set to around 0.1 sec in most of the cases. The ratio between the transversal and longitudinal vibration is controlled by the amplitudes of the resonators and was set manually.

The parameters of the soundboard model were taken from force-hammer measurements of a real piano soundboard [4]. The filters $H_{low}(z)$ and $H_{high}(z)$ are computed as follows: first a 16000 tap target impulse response $H_I(z)$ is calculated by measuring the force–pressure transfer function of the soundboard. This is low-pass filtered and downsampled by a factor of 8 to produce an FIR filter $H_{low}(z)$. The impulse response of the low frequency chain is now subtracted from the target response $H_I(z)$ providing a residual response containing energy above 2.2 kHz. This residual response windowed to a shorter length (1000 tap).

6. RESULTS

In Fig. 7 the spectrum of a synthesized $G_1$ piano tone is depicted with 5 m/s hammer impact speed (forte playing). The phantom partials and the free response of the second longitudinal mode (marked by a circle) are clearly visible between the transversal partials and sometimes reach higher amplitude levels than the transversal partials (e.g., around 1.5 kHz). It is important to note that the most important part of the longitudinal vibration is the forced motion (phanton partials) and not the free mode marked by a circle. For comparison, the spectrum coming from the transversal string vibration is displayed in Fig. 8. Note that without longitudinal modeling the spectrum is clean and contains a quasi-harmonic series only. Both signals were generated including the multi-rate soundboard in the model.

Comparing Fig. 7 to Fig. 1 shows that the spectrum of the synthesized piano tone is much closer to the original if longitudinal string motion is also considered in the modeling. There are still some differences between the synthesized and original. However, this is not considered as a drawback, since in physics-based sound synthesis the goal is rather to develop a model which has a realistic piano sound than to imitate a particular type of piano.

Although it is not depicted, the model responds to the variations of playing dynamics (piano to forte) realistically. The longitudinal components become significant at high dynamic levels only similarly to real pianos.

The model is capable of producing similar sound quality compared to the finite-difference model of Sec. 3 at around 10% of computational cost. Computational savings are achieved by eliminating the need of huge sampling rates, which were necessary to assure numerical stability. This is the reason why the perceptual quality of the finite-difference model is still preserved: the major difference is that the new model does not compute anything above 20 kHz, which we would not hear anyway. The new approach has an other advantage over the finite-difference method: the flexibility of setting the longitudinal modal frequencies.

Compared to earlier digital waveguide or finite-difference based piano models the sound quality improved significantly for low pi-
ano tones. The computational requirements are 10–20 times higher than that of a digital-waveguide based string model, still allowing 5–10 note polyphony in real-time on an average PC, in C++ implementation. Practically, this means that those piano pieces can be played which do not require sustain pedal. Sound examples can be listened at: http://www.mit.bme.hu/~bank/publist/dafx04.

7. CONCLUSIONS AND FUTURE WORK

In this paper a novel approach was presented for modeling the longitudinal vibration of piano strings. The method is based on a finite-difference string model for transversal vibrations, driving second-order resonators for longitudinal-vibration simulation. Large computational savings have been achieved compared to the complete finite-difference string model with no significant effect on the produced sound quality, allowing the use of the method in real-time. Simplifications were done along perceptual lines, i.e., those factors were neglected, which have no significant effect on the produced sound.

As for further improvements in sound quality, the force-pressure transfer function of the soundboard for the longitudinal polarization could be measured on real pianos. This would allow the use of a separate soundboard model for the longitudinal polarization. Alternatively, a precise shaping filter $H(z)$ in Fig. 5 could be designed. Another natural choice can be incorporating the other transversal polarization in the model.

The main area of future investigations should be reducing the computational complexity. One attempt to this can be substituting the finite-difference model in Fig. 5 by a digital waveguide for computing the transversal vibration. Now the difficulty is that the digital waveguide in its efficient form is intended to compute the string motion at the observation point (the bridge in this case) only. Therefore the modal shapes will be different from that of a real string, since the wavetrains will close through the dispersion filter at the termination. A solution can be having multiple (50..100) observation points by distributing the all-pass filters between the delay elements, similarly to what was done in the case of the kantele [16] for a different reason, namely, for tension-modulation modeling. Unfortunately, this leads to computational requirements almost as heavy as of the finite-difference string model.

Larger computational savings could be achieved by concentrating more on the perceptual aspects of longitudinal vibration. As little is known about how these components are perceived, this calls for psychoacoustic studies.

8. ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers and other researchers for their helpful comments.

9. REFERENCES


THE FEATHERED CLARINET REED

Tamara Smyth, Jonathan Abel, Julius O. Smith

CCRMA, Stanford University
Stanford, CA 94305, USA
{tamara|jonathan|jos}@ccrma.stanford.edu

ABSTRACT

In this research, a method previously applied to improve a digital simulation of the avian syrinx is adapted to the geometry of the clarinet reed. The clarinet model is studied with particular attention to the case when the reed beats against the lay of the mouthpiece, closing off airflow to the bore once each period. In place of the standard reed table which gives steady-state volume flow as a function of constant pressure difference across the reed, a more realistic dynamic volume flow model is proposed. The differential equation governing volume flow dynamics is seen to have a singularity at the point of reed closure, where both the volume flow and reed channel area become zero. The feathered clarinet reed refers to the method, first used in the syrinx, to smooth or feather the volume flow cutoff in a closing valve. The feathered valve eliminates the singularity and reduces artifacts in the simulated clarinet output.

1. INTRODUCTION

Many sounds are produced by coupling the mechanical vibrations of a source to the resonance of an acoustic tube. In the bird’s vocal organ, the syrinx, air pressure from the lungs controls the oscillation of a membrane (by changing the pressure across the membrane, which creates a variable constriction through which air flows before reaching the upper bronchus and trachea [1, 2, 3]). Similarly, blowing into the mouthpiece of a clarinet will cause the reed to vibrate, narrowing and widening the airflow aperture to the bore. Sound sources of this kind are referred to as pressure-controlled valves and they have been simulated in various ways to create musical synthesis models of woodwind and brass instruments as well as animal vocal systems.

When airflow sets a valve into motion, it causes a change in the height of the valve channel and, if the motion is extreme, it can potentially close off the channel completely, creating a sudden termination in airflow. Examples of this occur when a clarinet reed beats against the lay of the mouthpiece or when the syrinx membrane touches the opposite wall of the upper bronchus. Such abrupt changes are difficult to synthesize because of undesirable time quantization effects that result when using relatively low audio sampling rates.

As described in [3] and [4], if the vibrating membrane in the syrinx were to reach to opposite wall of the bronchus, the membrane’s flexible biological material would likely cause it to touch gradually, starting with the center bulge and the remainder settling gently on either side before finally closing off the channel. That is, as illustrated in Figure 1, instead of the channel being sealed the moment the membrane touches the opposite wall, it is more likely that flow will be able to seep through side corners and any other potential openings before the channel is closed completely.

In previous research [3], the behaviour of the differential equation governing volume flow through a syrinx valve was re-examined, paying particular attention to this troublesome transition between an open and closed valve. A closed-form solution for the time evolution of volume flow was given and used to derive an update for the volume flow which, in effect, sampled the continuous output of the differential equation governing volume flow. The result was a sort of leaky valve, with the leakage decreasing as the volume flow decreases, which smoothed or feathered the transition between the two states, significantly reducing the aliasing associated with the closing of the valve.

In this paper, also described in [4], improvements made to the syrinx valve [3] are applied to the clarinet reed, effectively feathering the beating clarinet reed. Though the cane reed is much more rigid than the bird’s syrinx membrane, its simulation can benefit from the same principles of feathering—particularly when meeting the demands of relatively low audio sampling rates.

We begin by describing the current status of the clarinet simulation which employs a so-called reed table to relate the pressure difference across the valve to the volume flow through the valve channel. This quasi-static model ignores the dynamic relationship between the pressure difference and volume flow, yielding a set volume flow for each value of input pressure difference. We address this shortcoming by then introducing a dynamic model for airflow through the reed channel, and replace the reed table with differential equations governing the reed’s displacement of the reed and the resulting volume flow through the valve channel. With the dynamic model in place, the feathered valve can then be incorporated, taking into account the geometry of the clarinet valve.

2. THE QUASI-STATIC CLARINET MODEL

Most current models of the clarinet reed are implemented using a lookup table which matches values for flow with the pressure...
drop across the reed valve. This is known as a quasi-static model since the value of flow, $U$, is established by using a lookup table (see Figure 3) relating pressure difference and volume flow under constant-flow conditions [5]. One of the benefits of the reed table approach is that it produces very satisfying results with low computational cost. One difficulty however, occurs when the reed beats against the lay, terminating air flow every cycle, and the reed and the lay is often too abrupt, the sound produced can be metallic and artificial. Furthermore, in the case that the reed beats against the lay, terminating air flow every cycle, it is clear that a static model is not entirely accurate [6].

An excellent description of the quasi-static clarinet reed model was published by Dalmont, Gilbert and Ollivier in [7], by Fletcher and Rossing in [8], and by Hirschberg et al. in [9], and is summarized here to give context to the discussion that follows.

The steady flow through a valve is determined based on an input (or blowing) pressure $p_m$, and a resulting output (or mouthpiece) pressure $p_b$ (see Figure 2). The difference between these two pressures is denoted $\Delta p$, and is related to volume flow via the stationary Bernoulli equation [8],

$$U = A \sqrt{\frac{2 \Delta p}{\rho}},$$

where $A$ is the cross section area of the air column (and the jet) and $\rho$ is the density of air. The steady state reed position, and therefore the jet cross-sectional area, is a function of the pressure difference alone, and (1) can be used to generate the reed table shown in Figure 3.

The geometry of the clarinet valve is given by the width of the reed channel $w$ and the height of the opening $H$ (or alternatively, the distance between the reed and the lay). The area of the valve opening $A$ is therefore given by

$$A = wH.$$

The motion of the reed follows the familiar equation

$$\frac{d^2 x}{dt^2} + 2\gamma \frac{dx}{dt} + \omega_r^2(x - x_0) = \frac{A}{m} \Delta p,$$

where $\gamma$ is a damping coefficient, $m$ is the effective mass of the reed and $\omega_r$ is the reed’s resonant frequency. Since $\omega_r$ is related to the stiffness and the mass by

$$\omega_r = \sqrt{\frac{k}{m}},$$

where $k$ is a constant describing the reed stiffness (in $\text{Pa/m}$), equation (3) can be rewritten for convenience as

$$\mu \frac{d^2 x}{dt^2} + \rho g \frac{dx}{dt} + \kappa x = \Delta p,$$

where $\mu$ is mass per meter square and $g$ is a viscous-damping coefficient (in $\text{s}^{-1}$) [7]. In the quasi-static model, the time derivatives in (5) are set to zero, rendering the mechanical reed effectively massless, with the stiffness being the only reactive element. The equation for the quasi-static reed therefore becomes

$$x = \frac{\Delta p}{\kappa}.$$

If $H_0$ is the equilibrium opening, that is, the opening of the valve in the absence of flow, the displacement of the reed determines the valve opening, $H$, by

$$H = H_0 - x.$$

From (6) and (7), the steady-state pressure difference corresponding to a just closed reed is determined by setting the opening to zero, that is setting the displacement to its maximum, $x = H_0$. Stated mathematically,

$$H_0 - H = \Delta p,$$

$$\Delta p_{max} = \kappa H_0,$$

where $\Delta p_{max}$ represents the maximum pressure difference, below which the valve is open. By applying (7) and (8), the area of the reed opening becomes

$$A = w \left( H_0 - \frac{\Delta p}{k} \right).$$

Figure 2: A simplified diagram of a clarinet reed. The variable $p_m$ represents the mouth pressure, $p_b$ is the pressure in the bore, $U$ is the volume flow, $x$ is the displacement of the reed, $y$ indicates the position along the reed and $\lambda$ is the length of the unclamped end of the reed.

Figure 3: The reed table provides a value for volume flow $U$ corresponding to a change in pressure across the reed. The region of oscillation is between the two dotted lines.
which can be further reduced to

\[ A = w H_0 \left( 1 - \frac{\Delta p}{\Delta p_{\text{max}}} \right) \]  

by applying (9). The stationary volume flow from (1) therefore becomes

\[ U = w_0 \left( 1 - \frac{\Delta p}{\Delta p_{\text{max}}} \right) \sqrt{\frac{2 \Delta p}{\rho}} \]  

Note that a pressure difference of \( \Delta p_{\text{max}} / 3 \) gives the maximum value for steady state flow \( U_{\text{max}} \).

\[ U_{\text{max}} = \frac{2}{3} w H_0 \sqrt{\frac{2 \Delta p_{\text{max}}}{3 \rho}}. \]  

It is just above this value of differential pressure that the reed can oscillate in response to an applied pressure [7].

If the pressure difference is greater than \( P_{\text{max}} \), it is assumed that the reed is closed, and there is no flow through the valve channel (see Figure 3) and \( U \) is set to zero. This handling of the flow between open and closed valves can be improved by feathering the collision between the open and closed states.

### Table 1: Example value ranges for variables of the quasi-static clarinet model (some values are taken from [7]).

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Symbol</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equilibrium reed opening</td>
<td>( H_0 )</td>
<td>0.04 – 0.1 cm</td>
</tr>
<tr>
<td>Reed stiffness</td>
<td>( \kappa )</td>
<td>800 – 1300 hPa/cm</td>
</tr>
<tr>
<td>Effective width of jet</td>
<td>( \omega )</td>
<td>1.2 – 1.8 cm</td>
</tr>
<tr>
<td>Maximum volume flow</td>
<td>( U_{\text{max}} )</td>
<td>200 – 600 cm³/s</td>
</tr>
<tr>
<td>Flow on the reed</td>
<td>( \nu )</td>
<td>0.9 cm</td>
</tr>
<tr>
<td>Damping coefficients</td>
<td>( \gamma )</td>
<td>1000/s</td>
</tr>
<tr>
<td>Density of air</td>
<td>( \rho )</td>
<td>0.0000012 kg/cm³</td>
</tr>
<tr>
<td>Frequency of reed</td>
<td>( f_1 )</td>
<td>1045 Hz</td>
</tr>
<tr>
<td>Reed length</td>
<td>( \lambda )</td>
<td>3.4 cm</td>
</tr>
</tbody>
</table>

3. REPLACING THE REED TABLE WITH THE DYNAMIC MODEL

Since volume flow doesn’t instantaneously respond to changes in differential pressure, a dynamic model is needed. Before applying the feathered beating reed, the equations for flow and displacement in the quasi-static model must first be replaced with their corresponding differential equations, incorporating the appropriate valve geometry for the clarinet.

#### 3.1. Volume Flow

The strategy for determining the volume flow derivative in the clarinet reed model is similar to that for the syrinx discussed in [3]. In the clarinet reed, only a short section of length \( \nu \) along the \( y \) axis (see Figure 2) is in contact with the flow before the flow separates from the surface of the reed and forms a jet. The force on a thin slice \( dy \) along this part of the reed is given by

\[ F = A(y; \nu) \Delta p(y). \]  

where \( A(y; \nu) \) is the area of the valve channel at this position and \( \Delta p(y) \) is the pressure drop across this section of the reed.

This force is applied to a mass of

\[ m = \rho A(y; \nu) dy, \]  

where \( \rho \) is the air density and \( A(y; \nu) dy \) is the volume to which the force is applied. Newton’s second law, \( F = ma \), can then be applied to (14) and (15) to obtain

\[ \Delta p(y) = \rho A(y; \nu) \frac{dy}{dt}. \]  

where acceleration is given by the time derivative of the particle velocity, \( dy/dt \), assumed constant over this section of the reed.

Since volume flow is equal to particle velocity scaled by area, the expression for differential pressure as a function of position \( y \) along the reed channel is given by

\[ \Delta p(y) = \rho \frac{dU}{dy} / A(y; \nu). \]  

Equation (17) is then integrated over the length of the channel to obtain

\[ p(0) \rightleftharpoons p(\nu) = \rho \frac{dU}{dy} \int_{y=0}^{y=\nu} dy / A(y; \nu), \]  

where \( y = 0 \) is the channel entrance and \( y = \nu \) is the point at which the flow separates from the surface of the reed and forms a jet.

The pressure at the channel entrance is obtained using Bernoulli’s equation given by

\[ p(y) = p_0 + \frac{\nu^2}{2} - v(y)^2 \]  

where \( v \) is the particle velocity. Again substituting volume flow divided by area for particle velocity, the pressure at the channel entrance \( p(0) \) is given by

\[ p(0) = p_m - \frac{\rho}{2} \left( \frac{U}{A(0; x)} \right)^2, \]  

where \( p_m \) is the mouth pressure. Since the pressure at the point of flow separation \( p(\nu) \) is equal to the bore pressure \( p_b \), equation (18) becomes

\[ p_m - p_b = \frac{\rho}{2} \left( \frac{U}{A(0; x)} \right)^2 = \frac{\rho}{2} \frac{dU}{dy} \int_{y=0}^{y=\nu} dy / A(y; \nu). \]  

The differential equation governing volume flow is then given by

\[ \frac{dU}{dt} = (p_m - p_b) \frac{A(x)}{\nu \rho} - \frac{U^2}{2 \nu A(x)} \]  

where the flow is assumed to be in contact with the reed for a distance of \( \nu \), at an assumed constant area equal to

\[ A(x) = A(0; x) = w (H_0 - x), \]  

where \( w \) is the width of the reed (or jet) and \( H_0 \) is the opening of the valve channel in the absence of flow.

Note that in steady state, the dynamic equation for air flow (22) reduces to (1). Setting the time derivative \( dU/dt \) to zero, implies

\[ (p_m - p_b) \frac{A(x)}{\nu \rho} = \frac{U^2}{2 \nu A(x)}. \]  

Solving for the steady state flow, we get

\[ U = A(x) \left( \frac{2}{\rho} (p_m - p_b) \right). \]
3.2. Reed Displacement

The equation for displacement is determined by considering the force acting on the clarinet reed. If the reed is rigid and hinged with spring constant \( k \) at a point far from the mouth side of the lip, let \( \lambda \) be the length of the reed, roughly along the \( y \) axis, which sees the mouth pressure \( p_m \).

There is a force closing the reed which is given by

\[
F_m = w \lambda p_m, \tag{26}
\]

and in contrast, the force on the bore side of the reed away from the jet, given by

\[
F_b = -w(\lambda - \nu)p_b, \tag{27}
\]

where \( \nu \) is defined as before, forces the reed open. The force applied by the flow (which also forces the reed open) is found by integrating the pressure along the flow and is given by

\[
F_v = -w \int_{y=0}^{y_{max}} \left( p_m - \frac{\rho}{2} \left( \frac{U}{A(x)} \right)^2 \right) dy. \tag{28}
\]

The overall force acting of the reed is obtained by summing (26), (27) and (28) and is given by

\[
F = w(\lambda - \nu)(p_m - p_b) - w\rho \frac{U}{A(x)}^2. \tag{29}
\]

Once the force is known, the displacement of the reed is obtained using the familiar differential equation (comparable to 3)

\[
\frac{d^2x}{dt^2} + 2\nu \frac{dx}{dt} + \frac{k}{m} (x - x_0) = \frac{F}{m}, \tag{30}
\]

where \( F \) is defined by (29).

3.3. Model Aliasing

In simulating the valve, care must be taken in computing the volume flow between open and closed states. This is made more difficult by the singularity in the equation for the volume flow derivative (22) as the valve opening approaches zero.

Frequently this type of situation is handled by solving the equation only when it is stable, and substituting a fixed value at the point of singularity. It is well known, however, that switching based on a level threshold causes aliasing in discrete time signals. In this case, aliasing is caused by setting the volume flow \( U(t) \) to zero when the valve closes.

As illustrated by the magnified plot of volume flow in Figure 4, \( U(t) \) (the dashed line) is forced to zero on a sample boundary. Regardless of the value of \( U(t) \) predicted by updating (22), air volume flow is still being set to zero when the valve is closed.

Theoretically, this approach seems correct, since no air should flow through a closed valve. Truncating the volume flow on sample boundaries, however, is problematic (both in the static and dynamic model). Depending on the period of the signal, the clipping may not happen at the correct phase and aliased components will be generated. This is illustrated in Figure 5 which shows a sinusoid and its truncated version along with their respective power spectra. Aliased components appear as peaks at nonharmonic frequencies.

4. IMPROVING THE DYNAMIC MODEL BY FEATHERING THE BEATING REED

The difficulty with discretizing (22) in the presence of small valve areas is illustrated in Figure 6. Since the slope of \( U(t) \) is decreasing with decreasing volume flow (this can be seen in Figure 4), predictions of the slope based on (22) tend to overshoot zero volume flow [3]. It would therefore be preferable to use a small area solution of \( dU(t)/dt \) to update the volume flow when the valve is closing. In [3] the small area solution to the volume flow update is solved for the syrinx geometry; here it is solved for the clarinet reed.

When the valve channel area \( A(t) \) is sufficiently small, the first term of (22) can be ignored and the differential equation for \( U(t) \) is approximated by

\[
\frac{dU}{dt} \approx -\frac{U^2}{2\nu A(t)}, \quad A(t) \ll 1, \tag{31}
\]

which is in the form of a so-called Bernoulli differential equation [10]. Though this differential equation is nonlinear in \( U(t) \), it may be converted to a linear form by the substitution

\[
W(t) = \frac{1}{U(t)}. \tag{32}
\]

Writing (31) in terms of \( W(t) \) gives the following new differential equation for \( U(t) \)

\[
\frac{dU}{dt} = -\frac{1}{W(t)^2} \frac{dW}{dt} \tag{33}
\]

where

\[
\frac{dW}{dt} = \frac{1}{2\nu A(t)} \tag{34}
\]
Figure 5: Figure (a) shows a full and truncated version of a sine wave. Figure (b) shows the desired power spectrum of the truncated waveform, and Figure (c) shows the artifacts in the spectrum if the truncation is done too abruptly for the sampling rate used.

Figure 6: In the case of a large sampling period $T$, updating the volume flow using (22) can cause $U$ to overshoot. The dotted line represents the actual value of $U$.

This equation is easily integrated to solve for volume flow:

$$ W(t) = \int_{t_0}^{t} \frac{d\tau}{2\nu A(\tau)} + C, $$

$$ U(t) = \int_{t_0}^{t} \frac{d\tau}{2\nu A(\tau)} + C, $$

where the constant of integration $C$ may be set given knowledge of $U(t)$ at a particular time $t_0$ and

$$ U(t) = \frac{U(t_0)}{1 + U(t_0) \int_{t_0}^{t} \frac{d\tau}{2\nu A(\tau)}.} \tag{37} $$

Note that when the area $A(t)$ is small, the integral in the denominator of (37) is large, and any initial positive value of volume flow is quickly reduced to zero without crossing zero, as would be expected for a closing valve. This observation provides justification for having zero volume flow when the valve area is zero. The small valve area solution to (37) suggests a possible alternative to truncating $U$ when the valve is closed (which would be otherwise necessary given the singularity in (22)). If the valve were slightly leaky, e.g.,

$$ \tilde{A}(t) = A(t) + \lambda \tag{38} $$

for a small leakage area $\lambda$, the singularity at zero area would be avoided, and the volume flow behaviour would be relatively unchanged. However, it is not sufficient to use a leaky valve in place of one that is truncated because though this may reduce the slope of $U(t)$ it also introduces the undesirable behaviour of volume flow oscillating about zero.

In order to see how this solution should be incorporated into the volume flow update, consider the value of $U(t)$ at time $t_0 + T$, where $T$ is the sampling period. Given the small area solution for volume flow (37), but in a more convenient form

$$ U(t) = \left[ \frac{1}{U(t_0)} + \frac{1}{2\nu \tilde{A}(t_0)}(t - t_0) \right]^{-1}, \tag{39} $$

the valve channel area can be substituted by $A(t_0)$, since it is assumed to be constant during the time interval $[t_0, t_0 + T]$. Substituting into (39) we obtain

$$ U(t) = \left[ \frac{1}{U(t_0)} + \frac{1}{2\nu A(t_0)}(t - t_0) \right]^{-1}, \tag{40} $$

and the volume flow at $t_0 + T$ is

$$ U(t_0 + T) = U(t_0) \left[ 1 + \frac{U(t_0)}{2\nu A(t_0)T} \right]^{-1}. \tag{41} $$

Using the first order backwards difference approximation, the new differential equation for $U(t)$ becomes

$$ \frac{dU}{dt} = \frac{U(t_0 + T) - U(t_0)}{T} \tag{42} $$

$$ \frac{dU}{dt} = -\frac{U(t_0)^2}{2\nu A(t_0)} \left[ 1 + \frac{U(t_0)}{2\nu A(t_0)T} \right]^{-1}. \tag{43} $$

Comparing the form of (43) to (31) note that the Bernoulli terms are identical, save a factor of $[1 + U(t_0)/2\nu A(t_0)T]^{-1}$. This factor has the effect of reducing the derivative in the presence of small channel areas or large sample periods. Rewriting (43) gives

$$ \frac{dU}{dt} = -\frac{U(t_0)^2}{2\nu A(t_0)} \left[ 1 + \frac{U(t_0)}{2\nu A(t_0)T} \right]. \tag{44} $$

Note that in this form the Bernoulli term is similar to that of (31), with a valve having increased area. In other words, it has become a leaky valve whose leakage increases with increasing volume flow. The final step is to replace the second term in (22) with this new Bernoulli term (44) to obtain

$$ \frac{dU}{dt} = (p_a - p_b) \frac{A(t_0)}{\rho p} - \frac{U(t_0)^2}{2\nu A(t_0) + U(t_0)T}. \tag{45} $$

which is the final feathered differential equation for volume flow.
5. CONCLUSIONS

The differential equation (22) describing the behaviour of volume flow (22) can be numerically unstable because of the singularity in the Bernoulli term when the valve closes. Since abruptly setting the flow to zero causes aliasing, the problem is addressed by incorporating the new small-area solution for $U(t)$. The volume flow is now updated in a way which produces smoother transitions between open and close valves.

By feathering the collisions of the beating reed in the clarinet simulation, the sound is greatly improved. This is illustrated in the output spectrum of the static model (Figure 7) and that of the feathered dynamic model (Figure 8) where the aliasing is significantly reduced.

Figure 7: Output of the quasi-static clarinet model using a sampling rate of 44.1 kHz. Control parameter values were mouth pressure, 70 - 10 kPa, and frequency, 300-600 Hz. Lines through the spectrum illustrate undesirable artifacts.

Figure 8: Output of the “feathered” dynamic clarinet model with the same control parameters and the same sampling rate as Figure 7. The output is improved overall and almost free of artifacts.

6. REFERENCES


ABSTRACT

The Functional Transformation Method (FTM) is an established method for sound synthesis by physical modeling, which has proven its feasibility so far by the application to strings and membranes. Based on integral transformations, it provides a discrete solution for continuous physical problems given in form of initial-boundary-value problems. This paper extends the range of applications of the FTM to brass instruments. A full continuous physical model of the instrument, consisting of an air column, a mouthpiece and the player’s lips is introduced and solved in the discrete domain. It is shown, that the FTM is a suitable method also for sound synthesis of brass instruments.

1. INTRODUCTION

In the field of sound synthesis physical modeling has gained much interest in the last decade. The FTM steps in, where other physical based sound synthesis techniques loose direct physical relations or require high computational cost. The FTM starts with a description of a musical instrument in form of a partial differential equation (PDE) with initial conditions (IC) and boundary conditions (BC). By performing integral transformations on both time and space variable, a multi-dimensional transfer-function-model (MD TFM) is achieved, that allows the real time solution of the entire system. The FTM provides full access to all physical parameters of the underlying model. For that reason interaction of control parameters with the model is close to the control mechanisms of real musical instruments, which allows an intuitive way of interaction with the model. Furthermore fully accessible physical models can help to understand the mechanism of the sound production and therefore the influence and optimization of physical parameters. The general procedures of the FTM are depicted in Figure 1. Detailed information is available in [1].

\[ L\{\cdot\} \rightarrow \text{ODE BC} \rightarrow T\{\cdot\} \rightarrow \text{MD TFM} \]

Figure 1: General procedures of the FTM.

In this paper a physical model of a brass instrument is built of several separate models; the air column, the mouthpiece and the player’s lips. All models are chosen to be as complex as necessary and as simple as possible in order to keep the basic physical properties and to be able to show the feasibility of the FTM synthesis for brass instruments in general.

For the air column an extended version of Webster’s horn equation is the underlying PDE of the model. The mouthpiece model is assumed to be a lumped model, where advantage is taken of analogous acoustic network circuits for the description of acoustical problems. This allows a description of the mouthpiece with a set of coupled ordinary differential equations (ODEs). Furthermore a lip model in form of a pressure controlled valve is used, that is coupled to the mouthpiece ODEs in a nonlinear way.

The paper is organized as follows. All parts of the entire model are treated in separate chapters. First they are introduced in the continuous form and then the way for obtaining a discrete model is shown respectively. Section 2 treats the air column and the used FTM procedures, Section 3 the mouthpiece model and Section 4 the lip model. Some details on the connection of the discrete models are given in Section 5. Section 6 provides results of the MATLAB implementation. Section 7 concludes this paper and shows possibilities of improvement of the physical model.

2. AIR COLUMN

Brass instruments consist roughly of long cylindrical pipes, mostly curved, that flare to a horn at one end. The sound waves inside propagate lossy through the instrument and are both, radiated and reflected at the horn. Due to viscous and thermal effects at the walls of the instrument and high pressure amplitudes the significant loss mechanisms are quite complex. Details are available in [2]. However in that first approach towards brass instruments the model is kept quite simple.

2.1. Continuous Model

Starting point of the air column model is a mathematical description of the brass instrument as an initial-boundary-value problem. Therefore, corresponding to [1], the following description with a PDE in vector notation with initial values \( y_i \) and boundary values \( y_b \) is used.

\[
\begin{align*}
[L + CD] y(x, t) &= f_e(x, t), \quad x \in V \\
f_{b1}^H y(x, t) &= y_1(x, t), \quad x \in \partial V \\
_f y(x, t)|_{t=0} &= y_0(x), \quad x \in V
\end{align*}
\]

The vector of unknown quantities is \( y(x, t) \). The vector \( f_e(x, t) \) indicates the excitation functions. The variable \( t \) is the time. \( x \) denotes the vector of spatial coordinates defined in the spatial volume \( V \), which is bounded by \( \partial V \). The operator \( D_t \) denotes the 1st order temporal derivative. \( f_e \) and \( f_b \) are vector operators specifying...
initial and boundary values. $(\cdot)^H$ is the hermitian operation and $L$ is a matrix operator of the form

$$L = A + BV \cdot. \quad (2)$$

At first the model is simplified to one spatial dimension. In consequence the nabla operator $V$ simplifies to the 1st order derivative with respect to the spatial coordinate $x$. The model divides the spatial volume $V$ of the brass instrument model into a cylindrical part $V_1$ and an horn shaped part $V_2$, which are both assumed to be round in diameter.

![Diagram of a brass instrument with labeled parts](image)

**Figure 2: Profile of a brass instrument. Details of the shape vary with the instrument. At the coordinate $x_h$ the air column is connected to the mouthpiece. $x_s$ is the coordinate of the bell’s end. The cylindrical and the horn shaped part are connected at $x_s$.**

For a concise notation the velocity potential $\Phi$ is introduced, that is related to the pressure $p$ and the sound particle velocity $v$ by

$$\frac{\partial \Phi(x,t)}{\partial x} = \Phi'(x,t) = v(x,t), \quad (3)$$

$$\frac{\partial \Phi(x,t)}{\partial t} = \Phi(x,t) = -\frac{1}{\varrho_0} p(x,t). \quad (4)$$

The constant $\varrho_0$ is the static density of air. The vector $y$ of unknown quantities for the initial-boundary-value problem is

$$y(x,t) = \left(\begin{array}{c} \Phi'(x,t) \\ \Phi(x,t) \end{array}\right). \quad (5)$$

It contains the flux quantity in form of $\Phi'$ and the potential quantity in form of $\Phi$. The standard PDE used for the air column model is the well known horn equation of Webster. Based on known eigenfunctions for hyperbolic horns, it has been already investigated in [3]. Here, we consider an extended version of Webster’s horn equation with an additional term $d_1 \Phi$, that causes a frequency-independent damping effect in the air column model. This is a severe simplification, as the damping in real brass instruments is frequency-dependent. Details are available in [2]. The resulting equation is valid for the sound particle velocity potential $\Phi$ as well as for the pressure $p$

$$\Phi'' + A'(x) \Phi' = \frac{1}{c^2} \Phi + d_1 \Phi. \quad (6)$$

The function $A(x)$ is the diameter of the wavefront within the instrument dependent on the coordinate $x$. $A'(x)$ denotes the 1st order derivative of $A(x)$ with respect to the variable $x$. The constant $c$ is the speed of sound in air.

The scalar PDE of equation (6) is turned into a vector PDE, corresponding to the notation in equation (1), with the following matrices $A$ and $C$:

$$A = \begin{pmatrix} \frac{A'(x)}{A(x)} & -d_1 \\ 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & -\frac{1}{c^2} \\ -1 & 0 \end{pmatrix} \quad (7)$$

The matrix $B$ equals the identity matrix $I_0$. In the cylindrical part of the instrument the radius of the instrument’s pipe is constant. Thus we have a constant radius of $r_c$ with

$$r(x) = r_c, \quad A'(x) = 0, \quad x \in V_1. \quad (8)$$

For the horn shaped part we assume that the shape of a brass instrument’s horn is close to a Bessel horn. The radius function $r$ of such a horn is given with

$$r(x) = r_h x^{-\varepsilon}, \quad A'(x) = -\frac{2\varepsilon}{x}, \quad x \in V_2, \quad (9)$$

where $\varepsilon$ is the flare parameter of the horn and $r_h$ is a scaling factor. A brass instrument’s shape is achieved by adjusting the parameters $x_h, x_s, r_c, r_h$ and $\varepsilon$ properly. The excitation forces in $V$ are set to zero and the initial conditions are assumed to be homogeneous.

$$f_i(x,t) = 0, \quad (10)$$

$$y_i(x) = 0. \quad (11)$$

The boundary conditions have to be adapted to our problem. At the point $x_{b1}$, where the mouthpiece is connected to the air column, we assume the sound particle velocity being equal to a given boundary excitation function $\psi(t)$. At the boundary point $x_{b2}$ a radiation load can model sound radiation from the instrument. To keep the model simple, the radiation load is set to zero, which means the pressure is zero at $x_{b2}$. Thus the boundary conditions are determined as follows:

$$f_{b1}^H y(x_{b1}, t) = \psi(t), \quad (12)$$

$$f_{b2}^H y(x_{b2}, t) = 0. \quad (13)$$

$$f_{b1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad f_{b2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (14)$$

### 2.2. Discrete Model

A discrete solution of the continuous problem can be obtained by performing the steps of FTM procedure as shown in Figure 1. Performing the Laplace-transformation on the problem of equation (1) leads to the following ODE with boundary conditions.

$$[L + sC] Y(x,s) = 0, \quad x \in V \quad (15)$$

$$f_{b1}^H Y(x_{b1}, s) = \psi(s), \quad (16)$$

$$f_{b2}^H Y(x_{b2}, s) = 0. \quad (17)$$

Next step of the FTM is the Sturm-Liouville-Transformation (SLT) that is used for the transformation of the space variable. The SLT is an integral transformation with a kernel function $K(x, \beta)$ that has to be designed carefully in order to obtain a transfer function model from the problem (see [1]). The transformation is defined by

$$T \{ Y(x) \} = \tilde{Y}(\tilde{\beta}) = \int V \tilde{K}^H(x, \tilde{\beta}) C \tilde{Y}(x) dx. \quad (18)$$

Using a well designed kernel function transformation of equation (15) yields an algebraic equation in the transformed domain that includes the transformed boundary values $\tilde{Y}_b$ as follows:

$$\beta \tilde{Y}(\tilde{\beta}, s) + s \tilde{Y}(\tilde{\beta}, s) - \tilde{Y}_b(\tilde{\beta}, s) = 0. \quad (19)$$
In order to obtain an algebraic equation corresponding to (19) from equation (15) a set of Sturm-Liouville eigenvalue problems has to be solved. These problems are called the eigenvalue problem and the adjoint eigenvalue problem. Details on this step are left out here and can be found in [1]. Firstly the adjoint operator $\tilde{L}$ is introduced.

$$\tilde{L} = A^H - B^H \nabla \label{eq20}$$

Because the model divides the instrument in two parts described with different PDEs due to the different matrix $A$ in $V_1$ and $V_2$, the eigenvalue problem and the adjoint eigenvalue problem have to be solved in sections for the definition range $V_1$ and $V_2$ respectively. The kernel functions $K$ and $\tilde{K}$ are then defined section-wise (see [4] for details). The problems are indexed with $n = 1, 2$.

$$LK_n(x, \beta) = \beta CK_n(x, \beta), \quad x \in V_n \label{eq21}$$
$$\tilde{K}_{1n}(x_0, \beta) = 0 \label{eq22}$$
$$\tilde{K}_1(x_0, \beta) = K_2(x_0, \beta) \label{eq23}$$

$$\tilde{L}K_{2n}(x, \tilde{\beta}) = \tilde{\beta}C^H K_n(x, \tilde{\beta}), \quad x \in V_n \label{eq24}$$
$$\tilde{\tilde{K}}_{1n}(x_0, \tilde{\beta}) = 0 \label{eq25}$$
$$\tilde{\tilde{K}}_1(x_0, \tilde{\beta}) = \tilde{K}_2^H(x_0, \tilde{\beta}) \label{eq26}$$

The adjoint boundary operators $\tilde{f}_1$ and $\tilde{f}_2$ are estimated

$$\tilde{f}_1 = [0 \ 1], \quad \tilde{f}_2 = [0] \label{eq27}$$

For concise notation of the solution of (21) and (24) introduce $\nu = z + \frac{1}{4}, \quad M = \frac{\sqrt{\nu^2 - \beta^2}}{\nu} \quad \text{and} \quad \tilde{M} = \frac{\sqrt{\nu^2 - \tilde{\beta}^2}}{\nu}. \quad J_0$ and $Y_0$ denote the Bessel-Functions of order $n$ of 1st and 2nd kind. The following solutions can be obtained by any symbolic mathematical program as MAPLE, for instance. The solution of equation (21) is

$$K_{1}(x, \beta) = \left( \begin{array}{c} -\frac{\nu}{\beta} e^{-Mx} \\ \frac{\nu}{\beta} e^{Mx} \end{array} \right) C^{(1)}_K \label{eq28}$$

$$K_{2}(x, \beta) = x e^{M(\nu - Mx)} \left( \begin{array}{c} \nu \frac{\nu}{\beta} e^{-Mx} J_{\nu - \beta}(jMx) \\ \nu \frac{\nu}{\beta} e^{-Mx} J_{\nu + \beta}(jMx) \end{array} \right) C^{(2)}_K \label{eq29}$$

with $a^{(1)}_K$ and $a^{(2)}_K$ being constants. The solution of equation (24) is

$$\tilde{K}_{1}(x, \tilde{\beta}) = \left( \begin{array}{c} \frac{\nu}{\beta} e^{-Mx} \\ -\frac{\nu}{\beta} e^{Mx} \end{array} \right) \tilde{C}^{(1)}_K \label{eq31}$$

$$\tilde{K}_{2}(x, \tilde{\beta}) = x^{-1} e^{M(\nu - Mx)} \left( \begin{array}{c} \nu \frac{\nu}{\beta} e^{-Mx} J_{\nu - \beta}(j\tilde{M}x) \\ \nu \frac{\nu}{\beta} e^{-Mx} J_{\nu + \beta}(j\tilde{M}x) \end{array} \right) \tilde{C}^{(2)}_K \label{eq32}$$

with $\tilde{a}^{(1)}_K$ and $\tilde{a}^{(2)}_K$ being constants. Then the eigenvalues $\beta_n$ and the adjoint eigenvalues $\tilde{\beta}_n$ can be obtained from equation (23) and

\begin{equation}
\frac{\nu}{\beta} e^{-Mx} J_{\nu - \beta}(jMx) - \frac{\nu}{\beta} e^{Mx} J_{\nu + \beta}(jMx) = 0 \tag{33}
\end{equation}

respectively. Inserting the kernels (28) and (29) in equation (23) leads to the following complex equation:

\begin{equation}
\frac{j\tanh(M(x_n - x_{n+1}))}{Y_{\nu}(jMx_{n+1})} = \frac{\nu}{\beta} \left[ \frac{J_{\nu - \beta}(jMx_{n+1}) - J_{\nu + \beta}(jMx_{n+1})}{Y_{\nu}(jMx_{n+1})} \right] \tag{34}
\end{equation}

In order to avoid a numerical search for the eigenvalues in the entire complex plane, we can take advantage of the frequency-independent damping effects. It is known a priori that this kind of damping gives the same real part for all eigenvalues $\beta_n$. Under the assumption $\beta_n = \sigma + j\omega_n = \frac{\nu}{\beta} \frac{\nu}{\beta} + j\omega_n$ the term $M$ simplifies to $M = j\sqrt{\frac{\nu^2 - \beta^2}{\nu^2}}$. With this assumption equation (34) simplifies to a real equation depending only on the variable $\omega_n$.

Solutions of this simplified equation can be found numerically in a much easier way. The eigenvalues are related to the adjoint eigenvalues with $\tilde{\beta}_n = \beta_n$. The operation $(\cdot)^*$ denotes the conjugate complex of $(\cdot)$.

Application of the SLT provides the transformed boundary conditions $\tilde{Y}_n$. Details on this step are available in [1].

$$\tilde{Y}_n(\tilde{\beta}_n, s) = \Psi(s) \tag{35}$$

The operator $\tilde{g}_n^H$ is a vector operator of the form $\begin{bmatrix} -1 & 0 \end{bmatrix}$. Reordering equation (19) and discretization by using the impulse-invariant-transformation yields a discrete-time transfer function model ($T$ is the sampling interval).

$$\tilde{Y}(\tilde{\beta}_n, z) = T \frac{\tilde{g}_n^H \tilde{K}(x_{n+1}, \tilde{\beta}_n) \Psi(s)}{z - e^{-\beta_n T}} \tag{36}$$

The boundary excitation function $\psi$ has the quantity of a sound particle velocity and is connected to the excitation volume flow $u_n$ at $x = x_{n+1}$ by $\psi(t) = \frac{u_n(t)}{A(x_{n+1})}$. Thus we can rewrite equation (36) using equation (35) to

$$\tilde{Y}(\tilde{\beta}_n, z) = \tilde{Y}(\tilde{\beta}_n, z) e^{-\beta_n T} z^{-1} \tag{37}$$

The vector of unknown quantities $y$ can be obtained by performing the inverse SLT. Now it is possible to compute the pressure and the sound particle velocity of the air column for all values of $x$.

$$Y(x, z) = T^{-1} \{ \tilde{Y}(\tilde{\beta}_n, z) \} = \sum_{\beta_n} \frac{\tilde{K}(x, \tilde{\beta}_n)}{N_{\beta_n}} \tilde{Y}(\tilde{\beta}_n, z) \tag{38}$$

In order to avoid aliasing the summation is limited up to the eigenvalues $\beta_n$ that fulfill the condition $\Im \{ \tilde{\beta}_n \} < \frac{\nu}{\beta}$. The norm factor $N_{\beta_n}$ computes with

$$N_{\beta_n} = \int_{x_{n+1}}^{x_{n+2}} \tilde{K}^H(x, \tilde{\beta}_n) \tilde{K}(x, \tilde{\beta}_n) dx \tag{39}$$
3. MOUTHPIECE

The mouthpiece is the connection between the lips of the player and the air column. There are brass mouthpieces in various shapes and sizes. The details vary with the style of the instrument and the player’s preferences, but all mouthpieces have the common general design shown in Figure 3. The player presses the lips against the surface of the mouthpiece cup, which has a characteristic volume $V_b$. A narrower passage of the diameter $S_c$ and a length of $l_c$ connects the cup to the main bore of the instrument.

![Figure 3: Profile of a typical brass instrument mouthpiece](image)

3.1. Continuous Model

A simple physical model of a mouthpiece can be found in [2]. The analogous acoustic compliance as an acoustic network depicted in Figure 4 is used to describe the basic physical behavior of a brass mouthpiece. The cup is modeled as an acoustic compliance $C$ and the passage of constriction as an acoustic inertance $L$. Lossy effects in the mouthpiece are included in the model with a dissipative element $R$. The impedance quantities denote the ratio of the pressure $p$ to the volume flow $u$.

For a common linear network like the mouthpiece model the underlying ODEs are simple. Therefore a description of the mouthpiece model is given directly in the frequency domain with the following $2 \times 2$ input-output-system matrix

$$
\begin{pmatrix}
U_1(s) \\
U_2(s)
\end{pmatrix} = \begin{pmatrix}
sL + sC + 1 & sC \\
sl + R & 1
\end{pmatrix} \begin{pmatrix}
U_1(s) \\
U_2(s)
\end{pmatrix}.
$$

The coefficients $a_1, ..., a_8$ can be obtained from the discretization procedure.

3.2. Discrete Model

A discrete mouthpiece model can be obtained by performing the impulse-invariant-transformation on the matrix description in equation (40). In the $z$-domain we get for $U_2$ and $P_1$

$$
U_2(z) = \left[a_1z^{-1} + a_2z^{-2}\right]U_2(z) + a_3z^{-1}U_1(z) + [a_4 + a_5z^{-1}]P_2(z),
$$

$$
P_1(z) = [a_6 + a_7z^{-1}]U_2(z) + a_8P_2(z).
$$

The coefficients $a_1, ..., a_8$ can be obtained from the discretization procedure.

4. LIPS

So far a model for a brass instrument has been introduced. Beside the properties of the instrument itself the interaction mechanism of the player with the instrument has a strong influence on the produced sound and is fundamental for the characteristic sound of every instrument. When playing a brass instrument the player interacts with the instrument by using his lips, that are pressed against the mouthpiece. Thereby the lips behave as an oscillator, that excites the air column inside the instrument. The lip oscillation is supported with energy provided by the player himself and reflections coming back from the instrument. The lip model is chosen to provide the behavior of lips towards a brass instrument in general and neglects details.

4.1. Continuous Model

Physical lip models of different kinds are available from the publications of N.H. Fletcher, e.g. [5], and from various publications of the group of X.Rodet at IRCAM, Paris, e.g. [6]. All these lip models are pressure controlled valve models. Here a basic upward striking model is used.

![Figure 5: Schematic of a physical model for the lips of a brass player according to [6].](image)
related to the geometric details of the lip model, \( P_2 \) indicates the blowing pressure inside the mouth and \( p \) is the pressure inside the mouthpiece. The volume flow \( u \) entering the mouthpiece is set to zero except when the condition \( x_3(t) > 0 \land P_2 - p(t) > 0 \) is fulfilled. Then it is computed with

\[
u(t) = l \sqrt{-\frac{\partial}{\partial \varphi} x_1(t) \sqrt{P_2 - p(t)}}.
\]

The parameter \( l \) is a constant describing the width of the lip. The initial conditions are denoted with \( x_i = [x_1(0), \dot{x}_1(0)] \).

### 4.2. Discrete Model

A discrete lip model can be easily obtained by performing the Laplace-transformation and the impulse-invariant-transformation on equation (45)

\[
X_l(z) = \left[w_1 z^{-1} + w_2 z^{-2}\right] X_1(z) + w_3 z^{-1} (P_2 - P(z))
\]

(47)

+ \( x_i w_0(z) \).

The coefficients \( w_1, w_2, \) and \( w_3, \) and the vector \( w_0(z) \) are available from the discretization procedure.

### 5. CONNECTING THE MODELS

When the mouthpiece model is connected to the air column, the volume flow \( u_2 \) equals the excitation flow \( u_\text{e} \) of the air column model. Furthermore the pressure \( p_2 \) equals the pressure at the coordinate \( x_{3,1} \) of the air column. The lip model can be connected to the mouthpiece by setting the lip flow \( u \) equal \( u_1 \) and the mouthpiece pressure \( p \) equal \( p_1 \). But this straightforward approach of joining the models yields an algorithm containing a delay free loop. In order to get an implementable synthesis algorithm all delay free loops have to be eliminated. Computing \( u_2 \) with equation (43) does require \( p_2 \) to be known at the recent time step. But with equation (38) the state vector of the FTM, and consequently \( p_2 \), can only be computed when all FTM computations (equation (37)) of the recent time step are already performed. This would require knowledge about the still unknown flow \( u_2 \). Therefore this straightforward way is not possible. An successful way of joining the models is shown now. The output pressure \( p_2 \) can be obtained from the state vector of the FTM.

\[
p_2(t) = -\varrho_0 \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \gamma(x_{3,1}, t)
\]

Using equation (37), (38) and (48) yields the following expression for \( p_2 \):

\[
P_2(z) = -\varrho_0 h_0 \sum_{\mu} \zeta_{\mu, x_{3,1}} \xi_{\mu}(z) z^{-1}
\]

\[
-\varrho_0 h_p \sum_{\mu} \zeta_{\mu, x_{3,1}} \xi_{\mu} U_2(z).
\]

Equation (49) can now be inserted into equation (43) and (44).

Reordering yields the following equations with the coefficients \( e_1, e_8, \) that are derived from the reordering procedure

\[
U_2(z) = \left[e_1 z^{-1} + e_2 z^{-2}\right] U_1(z) + e_3 z^{-1} (P_1 - P_z(z))
\]

(50)

\[
P_2(z) = \left[e_6 + e_7 z^{-1}\right] U_2(z) + e_8 z^{-1} (P_1 - P_z(z)).
\]

(51)

With equation (50) \( u_2 \) can be computed without the knowledge of \( p_2 \), which means the delay free loop has been eliminated. So the

---

**Figure 6: Synthesis algorithm for brass instruments. Simulation of the air column is performed with several complex 1st-order resonators in parallel. The excitation is computed with a nonlinear loop back containing mouthpiece and lip model computations.**

The following algorithm provides a discrete solution of the previously introduced continuous model. All equations given in the \( z \)-domain are transformed to the discrete time domain by performing the inverse \( x \)-transformation:

for \( k \) in simulation time

\[
u_2[k] = e_1 u_2[k - 1] + e_2 u_2[k - 2] + e_3 u_1[k - 1]
\]

\[
+ e_4 \eta[k - 1] + e_5 \eta[k - 2]
\]

for \( \mu = 1 : N \)

\[
\tilde{y}[\mu, k] = \xi[\mu] \bar{y}[\mu, k - 1] + \zeta[\mu] u_2[k]
\]

\[
\eta[k] = \sum_{\mu} \varrho_0 h_0 \zeta_{\mu, x_{3,1}} \xi_{\mu} \tilde{y}[\mu, k]
\]

\[
x_1[k] = w_1 x_1[k - 1] + w_2 x_1[k - 2]
\]

\[
+ w_3 (P_2[k - 1] - p_1[k - 1])
\]

\[
p_1[k] = e_6 u_2[k] + e_7 u_2[k - 1] + e_8 \eta[k - 1]
\]

if \( x_1[k] > 0 \land P_2[k] - p_1[k] > 0 \)

\[
u_1[k] = l \sqrt{-\frac{\partial}{\partial \varphi} x_1[k] \sqrt{P_2[k] - p_1[k]}}
\]

else

\[
u_1[k] = 0
\]
6. RESULTS

The synthesis algorithm was implemented in MATLAB. The model parameters were adjusted to fit the profile of a trumpet and a trumpet player. At first the instrument model without joined lips was investigated by computing the impulse response of the system. This allowed comparisons of the physical modeled computer instrument to measurements performed on real instruments. Comparable data is available in [6]. The results showed conformity at least in the basic matters, which is reasonable when looking at the rough simplifications of the underlying physical model. In the next step the lip model was connected to the instrument. Simulations yield results similar to those published in [7], where a similar lip model and a measured impedance function of a real instrument was used.

7. CONCLUSIONS

In this paper the way from a continuous physical model to a sound synthesis algorithm is shown for a brass instrument. Basic physical models for air column, mouthpiece and lips are introduced. Then the continuous models are turned into the discrete domain. The air column model is solved by performing the FTM procedures, the mouthpiece model and the lip model are solved by performing Laplace-transformation, impulse-invariant-transformation and inverse z-transformation. The discrete models are connected successfully by eliminating a delay-free loop. Feasibility of FTM based sound synthesis for brass instruments is demonstrated by implementing the algorithm in MATLAB. The simulation results show conformity to the basic properties of real brass instruments but also the need for improvement in order to achieve a more realistic sound. On one hand it is possible to use a more sophisticated lip model, on the other hand the instrument model itself can be modified. Two suggestions seem to be useful to achieve a more exact instrument model. Modeling frequency-dependent damping close to the characteristic of real brass instruments may be possible when extending equation (6) with an additional damping term $d_2 \Phi^\alpha$. Using a high-pass impedance as radiation load at the point $x_{12}$ may also yield improved results.

8. REFERENCES

RECENT ADVANCES IN PHYSICAL MODELING WITH K- AND W-TECHNIQUES

Matti Karjalainen, Jyri Pakarinen, Cumhur Erkut, Paulo A. A. Esquef, and Vesa Välimäki

Helsinki University of Technology
Laboratory of Acoustics and Audio Signal Processing
{firstname.lastname}@hut.fi

ABSTRACT
Physical (or physics-based) modeling of musical instruments is one of the main research fields in computer music. A basic question, with increasing research interest recently, is to understand how different discrete-time modeling paradigms are interrelated and can be combined, whereby wave modeling with wave quantities (W-methods) and Kirchhoff quantities (K-methods) can be understood in the same theoretical framework. This paper presents recent results from the HUT Sound Source Modeling group, both in the form of theoretical discussions and by examples of K-vs. W-modeling in sound synthesis of musical instruments.

1. INTRODUCTION
Real-world systems of interest in acoustics and computer music are typically continuous in time and space, and therefore their dynamic behavior is inherently described by partial differential equations [1]. Computer-based modeling and simulation of them requires, however, discretization of the underlying PDEs, which in a general case corresponds to the continuous analog system only when sample rate approaches infinity, i.e., temporal and spatial sample intervals are made infinitesimally small.

In this paper we discuss several discrete-time modeling paradigms, particularly the digital waveguides (DWGs), wave digital filters (WDFs), and finite difference time domain schemes (FDTDs). Their properties are compared and their mixing to hybrid modeling techniques are probed further from previous studies. Two cases (wave digital bell and distributed nonlinearity by FDTDs) are used to characterize realization principles.

1.1. General viewpoints
Discretization in time and space leads to interesting and difficult problems that are not found in the ordinary continuous case. Particularly when systems are simulated or synthesized efficiently in the time domain by discrete techniques, in contrast to being solved from equations, the question of localized discretization (blockwise construction of models through interconnection of elements) and consistent scheduling (ordering of operations) are of major importance.

In the analog world we may in the limit assume arbitrarily short (infinitesimal) delays between spatial points of interest, and the order of events follows the causality principle of physics. In the discrete-time world, however, a single sample period is the shortest possible non-zero time interval, in which explicit two-way physical interaction can happen. This leads easily to the problem of delay-free loops, i.e., implicit equations where the output of an operation needs an input value that may be dependent on the output value not yet known. Particularly with nonlinear elements this is a fundamental problem [2], in addition to aliasing.

It is now well-known that the delay-free loop problem is easier to overcome if computations are formulated by wave components instead of ordinary physical quantities [3]. Referring to electric circuits, the latter ones are often called the Kirchhoff quantities, in contrast to wave quantities. Using dual K-variables (K for Kirchhoff), such as voltage and current or force and velocity, is very intuitive for circuits and their mechanical equivalents, but in the discrete methodology they are not as easy to use. K-elements formulated as transfer functions between dual K-variables cannot form circuits and networks directly, but they in general must be converted into wave-based formulation, in order to compute them by explicit relations as local interactions. This is different from solving system equations’ which permits global interactions.

This paper is organized as follows. Section 2 presents a condensed overview of the physical modeling paradigms of interest in this study. In Section 3 we investigate the interrelations of these paradigms, followed by a case study of a wave digital bell model in Section 4. In Section 5, distributed nonlinear W-modeling is investigated, followed by a summary of the paper.

2. PARADIGMS OF DISCRETE-TIME MODELING
A short characterization of different physical modeling paradigms is presented in this Section, including digital waveguides, wave digital filters, finite difference time domain schemes, and modal decomposition.

2.1. Digital waveguides
Digital waveguide modeling is based on the fact that wave propagation in a medium can be simply and efficiently simulated by two delay lines [4], one for each directional wave component. This is closely related to the d’Alembert solution of the wave equation. By proper discretization in time and space the delay lines can be updated simply by

\[ y_{k,n+1} = y_{k-1,n} \quad \text{and} \quad y_{k,n+1} = y_{k+1,n} \]  

(1)

where arrows denote the right- and left-traveling components of the total waveform, and indices \( k \) and \( n \) refer to discrete position and time, respectively. Lowpass and allpass filters can be cascaded with delay elements to simulate damping and dispersion. Delay sequences between points of signal observation (output) and feed-in (input) can be consolidated into subsystems that are computationally highly efficient [4].

In addition to delays we need connecting junctions that fulfill physical continuity constraints, i.e., the Kirchhoff rules. For a parallel junction of acoustic components we may write:

---

1Time-domain simulation can also be based on solving global equations for each time step, but that tends to be highly inefficient compared to block-based simulation by localized interactions.
where $P_1$, $P_2$, ..., $P_N$ are pressure, and $U_1$, $U_2$, ..., $U_N$ are volume velocities at the ports of the junction, $P_3$ is the common pressure of coupled branches and $U_{ext}$ is an external volume velocity to the junction. When port pressures are represented by incoming wave components $P_i^+$ and outgoing wave components $P_i^-$, admittances attached to each port by $Y_i$, and

$$P_i = P_i^+ + P_i^- \quad \text{and} \quad U_i = U_i^+ Y_i^+$$

the junction pressure $P_3$ can be obtained as:

$$P_3 = \frac{1}{\sum_{i=0}^{N-1} Y_i} (U_{ext} + 2 \sum_{i=0}^{N-1} Y_i^+ P_i^+)$$

where $Y_{ext} = \sum_{i=0}^{N-1} Y_i$ is the sum of all admittances to the junction. Outgoing (scattered) pressure waves, obtained from Eq. (4), are then $P_i^- = P_3 - P_i^+$. Figure 1 depicts this as a signal flow diagram for the computation of such a scattering junction.

The same diagram can be applied to a series connection and volume velocity waves so that pressures and volume velocities are interchanged and admittances are replaced by impedances.

**2.2. Wave digital filters**

Wave digital filters (WDFs) are models that were originally developed for discrete-time simulation of lumped element circuits and systems as they were known from the analog electric domain [3]. The close relationship between them and digital waveguides is well known [4, 5]. While DWGs emphasize delays and wave propagation, WDFs have emphasis on lumped element modeling. However, both are capable to both types, and actually they are compatible and complementary approaches to wave-based modeling.

The WDF formalism is based on a notation of (‘voltage’) waves $a$ and $b$ as

$$a = V + R I \quad \leftrightarrow \quad V = (a + b)/2 \quad \text{and} \quad I = (a - b)/2R$$

where $a$ is in-coming and $b$ is out-going wave in a port, $V$ is voltage and $I$ is current as Kirchhoff variables, and $R$ is port resistance (reference resistance). In Fig. 2 a model for series connection of inductor L, capacitor C, and resistor R is depicted. It is constructed by two three-port series adaptors (SA1 and SA2) that implement wave scattering according to Kirchhoff laws. The implementation of the R, C, and L components is shown. Delay-free loops are avoided by the structure of the components and by impedance-matched reflection free ports denoted by $-$ in the adaptors [3].

In a closer comparison of DWGs (Eq. (3-5) and Fig. 1) and WDFs (Eq. (6) and Fig. 2) we can see that through acoustical-to-electrical analogies $P_i^+ = a_i$, $P_i^- = b_i$, $P_i = V_i$, $U_i = I_i$, and $1/Y_i = R_i$ for the DWG we get

$$V = \frac{(a + b)}{R}$$

which shows a difference in scaling compared to the WDF convention in (6). In fact we may select the scaling quite freely if we are interested just to get physically proper values of the Kirchhoff quantities $V$ and $I$. One useful convention is to apply power-normalized waves [3, 4] by

$$V = \frac{(a + b)}{\sqrt{R}} \quad \text{and} \quad I = \frac{(a - b)}{\sqrt{R}}$$

which have the favorable property that port power $P = V \cdot I = a^2 - b^2$ is independent of changes in $R$.

If the port admittances/impedances in DWG junctions are real-valued, the DWG junctions and WDF adaptors implement the same computation of K-variables, thus they are just two slightly different formulations of the same W-modeling principle. The WDF theory supports combinations of parallel and series connections (typically by 2- and 3-port adaptors) using reflection-free ports as in Fig. 2. The WDF theory includes a set of lumped one- and two-ports, and it is generalized also to multidimensional modeling [3].

**2.3. Finite difference models**

Finite difference approximation is a popular method of numerical integration of partial differential equations [6]. In physical modeling it is used particularly for multidimensional mesh structures [5] but also for example in string modeling [7].

Second order differences applied to the wave equation with a proper space and time discretization yield a simple recursion formula

$$y_{k,n+1} = y_{k-1,n} + y_{k+1,n} - y_{k,n-1}$$

for the computation of an FDTD node variable for a 1-D FDTD structure, whereby indices $k$ and $n$ refer to spatial and temporal indices, respectively [4]. In [8] we have derived an extension of the scheme that allows FDTD structures with arbitrary connection admittance to be formed, see Fig. 3, in a way very similar to the

![Figure 1: A 3-port parallel scattering junction for acoustic pressure waves. Port 1 (left) is terminated by admittance $Y_1$, port 2 (right) is connected to a delay-line, and port 3 (top) is not connected.](image1)

![Figure 2: (Left) A WDF series connection of resistor (R), capacitor (C), and inductor (L) constructed by two three-port series adaptors (SA1 and SA2). (Right) Equivalent analog circuit.](image2)
3. K- VS. W-MODELING

As mentioned above, it is well known that block-based modeling with K-methods is difficult or impossible due to the delay-free loop problem. There are special techniques, however, to use K-modeling, and the combination of K- and W-blocks is of particular interest. In this Section we discuss some related questions.

3.1. Comparison of DWG vs. FDTD models

In [8] we have presented a careful analysis of DWGs and FDTDs as shown in Figs. 1 and 3, by proving their functional equivalence in processing related K-variables. There are, however, a couple of special questions related to the equivalence need to be addressed.

The first one deals with the "sporadic" or "spurious" oscillations that easily appear in the FDTD structure but not in DWGs. The potential instabilities in FDTD are due to its inherent dual-feedback, see the bottom part of Fig. 3. This creates poles at DC and Nyquist frequency that need to be counteracted by the term \( H(z) = 1 - z^{-2} \) when feeding external excitation \( U_{\text{ext}} \). This pole cancellation means lack of numerical robustness close to these frequencies. Without this term an impulse fed to the junction results in continuous Nyquist frequency oscillation plus step function propagating from the junction in an FDTD array, which means unbound generation of energy, thus being nonphysical in the sense of passive systems.

One may ask if the instability is possible in a DWG. It is only possible if the junction is fed through the inverse of \( H(z) = 1 - z^{-2} \). However, the equivalence of the DWG in Fig. 1 and the FDTD structure in Fig. 3 has been proven in [8] and details of their relations will be discussed in Section 3.1.

2.4. Modal decomposition techniques

While the DWG, WDF, and FDTD methods above are explicit time-domain simulation techniques, the modal decomposition methods rely on frequency-domain formulations of systems under study. They decompose the behavior of a system into decaying exponentials, whereby oscillatory components represent eigenmodes of the system. Modal decomposition methods include the traditional modal synthesis [9] and a newer approach called functional transformation method (FTM) [10]. In Section 4 we will study semiphenomenal modeling of bells based on a modal decomposition paradigm.

Figure 3: A three-port parallel connection of FDTD type, corresponding to the DWG in Fig. 1.

Figure 4: KW-converter for mixed modeling with FDTDs (left-hand side) and DWGs (right-hand side).
cases of impedance \( H(z) \) specification: Polynomial (FIR), rational (IIR), and a modal filterbank of second-order resonators.

**FIR type impedance:** When \( Z(z) \) is given as a Z-domain polynomial expression, i.e.,

\[
Z_p(z) = \sum_{i=0}^{N-1} q_i z^{-i}
\]

(13)

then the \( a \) to \( b \) scattering function \( H(z) \) will in a delay-containing case \( R = q_0 \) become

\[
H_p(z) = \frac{\sum_{i=0}^{N-1} q_i z^{-i}}{\sum_{i=0}^{N-1} q_i z^{-i} + R} = \frac{\sum_{i=0}^{N-1} q_i z^{-i}}{1 + \sum_{i=1}^{N-1} \frac{q_i}{2q_0} z^{-i}}
\]

(14)

which is an IIR filter cascaded with a unit delay.

**IIR type impedance:** When \( Z(z) \) is given as a Z-domain rational expression (for simplicity numerator and denominator orders are the same), i.e.,

\[
Z_c(z) = \sum_{i=0}^{N-1} q_i z^{-i}
\]

(15)

then the delay-containing requirement for port resistance becomes

\[
R = q_0 / p_0 \quad \text{and} \quad H(z) \quad \text{according to Eq. (12) will be}
\]

\[
H_c(z) = \frac{\sum_{i=0}^{N-2} q_i z^{-i}}{1 + \sum_{i=1}^{N-2} \frac{q_i}{2p_0} z^{-i}}
\]

(16)

which is again an IIR filter with a cascaded unit delay.

**Second order resonators:** Second order resonators are of special interest because they are useful in building modal decomposition models. For example a series RLc-resonator having poles at DC and Nyquist frequency can be simulated by

\[
Z_a = K \frac{1 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-2}}
\]

(17)

that with \( R = K \) leads to realization

\[
H_a(z) = 0.5 \frac{q_1 + (q_2 + 1) z^{-1}}{1 + 0.5 q_1 z^{-1} + 0.5 (q_2 - 1) z^{-2}}
\]

(18)

### 4. WAVE DIGITAL BELL

As a W-modeling case using modal decomposition, a ‘semiphysical’ driving point impedance of a bell is investigated next. In earlier papers we have studied sound synthesis of bells using inharmonic digital waveguides and modal filterbanks [12, 13]. High-resolution analysis of modal data was achieved using the frequency-zooming ARMA analysis (FZ-ARMA) technique as developed in [14]. This is able to resolve for each partial a set of modes, very close in frequency, which produce the beating inherent in typical bell sounds.

Real bells are physical objects that, as 3-D structures, have an inharmonic set of partials [1]. By proper tuning several of the lower partials can be made approximately harmonic, but there always remains at least one perceptually important inharmonic component. In untuned small handbells there may not be any harmonic structure. Slight asymmetries are the reason to partials as mode groups and the perceivable beating (warble).

In a detailed model of a real bell the modal decomposition should describe the spatial distribution of modal shapes so that a force excitation in any point could be solved for the spatial distribution of the exponentially decaying modes as well as for the sound radiated from bell surfaces. Instead we in this paper develop a wave digital model based on modal decomposition that considers the bell only as a driving point impedance with related force and velocity. We may calibrate such a semiphysical model simply according to a sound recording of an existing bell and realize a model that sounds realistic, being efficient for real-time synthesis.

We have taken a particular bell and analyze its prominent modal components using the FZ-ARMA analysis. Table 1 lists the modal data from a bell recording. Two modes are fitted to each partial below 10 kHz to realize proper beating for the partials.

For a wave-based modeling we may use the analyzed modal data by constructing a wave port with a corresponding driving-point impedance. The task is now to construct a port compatible with W-modeling that implements a driving point impedance so that when the bell is hit by a proper hammer-like object (force excitation), it makes a desired sound (port velocity as output).

There are several ways to construct the port impedance of interest, for example:

1. Make the modes by basic WDF components as series resonators, as shown in Fig. 2. Pairs of modes are then combined in parallel into partials, which are further connected in parallel to make a full port impedance.

2. Implement each mode or a mode group for a partial as a filter structure according to the rules described in Section 3.3, and connect the partials in parallel.

3. Realize the whole impedance, composed of all modes, as a single filter, according to Eq. (16).

It is obvious that progressing from case 1 to 3 the computational efficiency of implementation can be improved, since consolidating functionality can utilize the advantages of DSP more than composing a model from lumped WDF elements. Case 2 benefits from the possibility of controlling the properties of each mode separately, while in case 3 only the composite filter coefficients are directly accessible. In case 1 the equivalent lumped components, corresponding in some way to masses, spring constants and damping factors, are directly controllable.

One particular design issue with case 1 is the bilinear mapping when creating reactances with WDFs. The formulas of reference resistance for capacitance \( C \) and inductance \( L \) in electrical circuits are:

\[
R_C = 1 / 2 C f_s \quad \text{and} \quad R_L = 2 L f_s
\]

(19)

where \( R_C \) and \( R_L \) are the related WDF port resistances and \( f_s \) is the sample rate. However, due to the bilinear mapping between

---

Table 1: Modal data for the bell of the case study including two modes per partial: modal frequencies (\( f_1 \) and \( f_2 \)), initial amplitudes (\( A_1 \) and \( A_2 \)), and decay time constants (\( \tau_1 \) and \( \tau_2 \)) are given.

<table>
<thead>
<tr>
<th>n</th>
<th>( f_1/\text{Hz} )</th>
<th>( f_2/\text{Hz} )</th>
<th>( \tau_1/\text{s} )</th>
<th>( \tau_2/\text{s} )</th>
<th>( A_1 )</th>
<th>( A_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>850.8</td>
<td>0.165</td>
<td>0.0723</td>
<td>851.3</td>
<td>0.749</td>
<td>0.0965</td>
</tr>
<tr>
<td>2</td>
<td>1702.3</td>
<td>0.464</td>
<td>0.1497</td>
<td>1703.1</td>
<td>0.421</td>
<td>0.0514</td>
</tr>
<tr>
<td>3</td>
<td>2026.7</td>
<td>0.355</td>
<td>0.1258</td>
<td>2032.8</td>
<td>0.048</td>
<td>0.0734</td>
</tr>
<tr>
<td>4</td>
<td>2787.2</td>
<td>0.131</td>
<td>0.0763</td>
<td>2792.5</td>
<td>0.079</td>
<td>0.0364</td>
</tr>
<tr>
<td>5</td>
<td>3404.7</td>
<td>0.251</td>
<td>0.0610</td>
<td>3407.0</td>
<td>0.098</td>
<td>0.0716</td>
</tr>
<tr>
<td>6</td>
<td>4552.1</td>
<td>0.028</td>
<td>0.0290</td>
<td>4559.6</td>
<td>0.110</td>
<td>0.0278</td>
</tr>
<tr>
<td>7</td>
<td>4889.6</td>
<td>0.149</td>
<td>0.0554</td>
<td>5050.5</td>
<td>0.259</td>
<td>0.0511</td>
</tr>
<tr>
<td>8</td>
<td>6881.5</td>
<td>0.149</td>
<td>0.1261</td>
<td>6889.2</td>
<td>0.051</td>
<td>0.0088</td>
</tr>
<tr>
<td>9</td>
<td>8549.8</td>
<td>0.153</td>
<td>0.0029</td>
<td>8631.9</td>
<td>0.023</td>
<td>0.0047</td>
</tr>
<tr>
<td>10</td>
<td>8695.0</td>
<td>0.109</td>
<td>0.0313</td>
<td>8842.0</td>
<td>0.153</td>
<td>0.0191</td>
</tr>
</tbody>
</table>

---

*This is a relatively high-pitch bell from the Belfort bell recordings, provided by Mark Leman.
realized is functionally equivalent. In each case 1–3 the wave digital port
gets correct modal frequencies and decay times. In cases 2 and 3
secondly, reverberation and re
needs a more complex bell model for more realistic sound, and
piano hammer [15].

To make a full bell model with a hammer striking the driving
point, the hammer has also to be modeled. The contact is a nonlin-
er (time-varying) one, thus special techniques are needed to make
it energetically correctly to guarantee stability. There are two prin-
ciples available to this: using power-normalized waves discussed
where
the frequency scale is warped [3] so that while \( \Omega \rightarrow f_0/2 \), the
responding analog angle frequency \( \omega \rightarrow \infty \). Thus the inductances
and capacitances in a WDF realization have to be prewarped to
get correct modal frequencies and decay times. In cases 2 and 3
above the filter design process takes automatically care of proper
frequencies and decay rates. In each case 1–3 the wave digital port
realized is functionally equivalent.

Figure 5 shows the admittance function (dB scaled) of the port
implemented from data in Table 1, and Fig. 6 plots the temporal
envelope of partial number 3 showing beating in decay.

To make a full bell model with a hammer striking the driving
point, the hammer also has to be modeled. The contact is a nonlin-
er (time-varying) one, thus special techniques are needed to make
it energetically correctly to guarantee stability. There are two prin-
ciples available to this: using power-normalized waves discussed
above (see also [15]) or by nonlinear reactance through mutator
type of adaptor [2]. These techniques allow for complex nonlinear-
contact of the hammer and the bell as it happens also with the
 piano hammer [15].

The WDF bell with impulsive hit sounds different from the
recording used for calibration for two reasons. First, the attack
needs a more complex bell model for more realistic sound, and
secondly, reverberation and reflections of the space surrounding
the bell. In other aspects the timbre is very realistic, including
correct type of beating and decay envelope.

5. DISTRIBUTED NONLINEARITIES BY DWG’S

Distributed nonlinearities are among the most challenging tasks
in physics-based modeling. They are quite common in musical
instruments, although in many cases a linearized model is a use-
ful approximation. In this Section we investigate how the tension
modulation nonlinearity in strings can be realized through digital
waveguide principles.

When a real string is displaced, it is elongated causing an in-
crease in its tension and thus also in its fundamental frequency.
The elongation of the string can be expressed as [16]

\[
l_{\text{el}}(t) = \int_{0}^{l_{\text{nom}}} \sqrt{1 + \left(y(t, x)\right)^2} \, dx - l_{\text{nom}}
\]

(21)

where \( l_{\text{nom}} \) is the nominal string length, \( x \) is the spatial coordinate
along the string, and \( y \) is the displacement of the string. Note, that
since the elongation calculation is done globally, i.e. for the whole
string in one piece, we lose the information concerning local elon-
gations and effectively consider the tension as being uniform for
the whole string. In real strings this is not the case, although longi-
tudinal waves do propagate considerably faster than the transversal
ones. The elongation can be approximated for the digital waveg-
uide as [17]

\[
L_{\text{el}}(n) \approx \frac{1}{N} \sum_{m=0}^{l_{\text{nom}}} \left[s_n(n, m) + s_1(n, m)\right]^2
\]

(22)

where \( s_n(n, m) \) and \( s_1(n, m) \) are the slope waves at time instant \( n \)
and position \( m \), propagating right and left, respectively, and \( l_{\text{nom}} \)
is the rounded nominal string length.

Since the transversal wave propagation velocity is proportional
to the string tension, the change in tension corresponds to a change
in the wave velocities. In DWGs this can be implemented by mod-
ulating the delay line lengths. This is done by inserting first-order
allpass filters between each unit delay in the delay line and then
varying the allpass-filter coefficients. This is illustrated in Fig. 7.
The excitation to the string can be inserted during run-time as a
force signal using an interaction block, denoted by \( I \). This block is
essentially a 3-port parallel junction of Fig. 1, with the third port
omitted and the excitation signal used as an external force.

The first-order allpass filter is of the form

\[
A(z) = \frac{-a + z^{-1}}{1 - az^{-1}}
\]

(23)

where \( a \) is the filter coefficient which defines the length of the
delay caused by the filter. The approximated delay for each allpass
filter can be expressed as [17], [18]

\[
d_{\text{ap}}(n) \approx -\frac{1}{2N} \sum_{m=-1}^{l_{\text{nom}}} \left(1 + \frac{EA}{K_0} \right) l_{\text{el}}(n, m) / l_{\text{nom}}
\]

(24)

where \( E \) is Young’s modulus, \( A \) is the cross-sectional area of
the string, and \( K_0 \) is the nominal tension corresponding to the string at
rest. \( N \) denotes the total number of allpass filters in the structure.
The allpass-filter coefficient \( a \) can now be written as

\[
a = (1 - d_{\text{ap}})/(1 + d_{\text{ap}})
\]

(25)

It is important to note that the string model presented in Fig. 7
simulates a string vibrating in one polarization only. For a more
realistic model, another such structure should be used for modeling
the second polarization. This is especially true in the case of the
kantele, a traditional Finnish plucked-string instrument, where
the two vibration polarizations have different effective lengths due to
a knotted termination of the string [19]. Using two nonlinear DWG
string models with slightly different lengths, a synthesis model of
the kantele can be generated. For a more thorough discussion of
the kantele model and the nonlinear DWG string, see [18].

The synthesis results reveal that the initial pitch glide phe
omenon can be realistically modeled using the nonlinear DWG
string. The fundamental frequencies of a real kantele recording
and the synthesized tone are illustrated in Figure 8.
Figure 7: A nonlinear DWG string. The string consists of allpass lters, denoted A ( ), connected via unit delays for avoiding delay-free loops. $H_L$ and $H_R$ denote loop lters simulating the frequency-dependent losses and $g$ is a scaling coefficient for modeling frequency-independent losses. $F(n)$ represents the excitation force signal applied on the string.

The generation of missing harmonics can also be simulated, if the boxcar integration of Eq. (24) is replaced with a leaky integrator, as suggested in [17]. Then, the leaky integrator parameters can be used to control the amplitudes of the missing harmonics. Note that this solution is not physics-based, since the error in the integration creates the mode-coupling, but it still is an efficient way of emulating the phenomenon.

6. SUMMARY

In this paper we have presented recent results from the HUT Sound Source Modeling group. A theoretical discussion covered different discrete-time modeling paradigms, divided in W- and K-modeling approaches, their interrelationships, and how they can be combined. Two particular cases have been described: a wave digital bell based on modal decomposition of a port impedance and distributed nonlinearity modeling by digital waveguides.

7. ACKNOWLEDGEMENTS

This work is supported by the IST/ALMA project (ALgorithms for the Modelling of Acoustic interactions, IST-2001-33059) and the Academy of Finland project SA 53537. Paulo Esquef has been supported by the Brazilian National Council for Scientific and Technological Development (CNPq).

8. REFERENCES

COMPUTATION OF NONLINEAR FILTER NETWORKS CONTAINING DELAY-FREE PATHS

Federico Fontana, Davide Rocchesso
Dept. of Computer Science
University of Verona
Strada le Grazie 15, 37134 Verona, Italy
fontana@sci.univr.it
davide.rocchesso@univr.it

Federico Avanzini
Dept. of Information Engineering
University of Padova
Via Gradenigo 6/A, 35131 Padova, Italy
avanzini@dei.unipd.it

ABSTRACT
A method for solving filter networks made of linear and nonlinear filters is presented. The method is valid independently of the presence of delay-free paths in the network, provided that the nonlinearities in the system respect certain (weak) hypotheses verified by a wide class of real components: in particular, that the contribution to the output due to the memory of the nonlinear blocks can be extracted from each nonlinearity separately. The method translates into a general procedure for computing the filter network, hence it can serve as a testbed for offline testing of complex audio systems and as a starting point toward further code optimizations aimed at achieving real time.

1. INTRODUCTION

The history of audio effects design traces back to the world of analog circuits, perhaps even before [1, 2]. It was not long after the digital architectures had appeared, that scientists considered the possibility to reproduce in the digital domain the analog and electro-acoustic mechanisms the early audio effects were based upon: at first with the aim of improving their quality by modeling them through digital filters; later on, with the goal of simulating these effects on digital equipment ranging from specialized hardware available in the studio to more common PC’s, audio cards and other consumer electronic devices.

The conversion into a sequence of discrete computations of a continuous-time process, such as the one realized by an analog electronic, electro-acoustic, fluid-dynamic or mechanical system, must move through a number of steps each one potentially introducing approximations that progressively shift the model away from the original system. Sometimes such approximations introduce negligible effects, if any at all. Conversely, there are cases in which they generate intolerable problems, often resulting in heavy artifacts affecting the system response or, even, preventing the model to work correctly.

The theory of digital signal processing suggests how a discrete-time system can be realized in the form of a network of filters, suitable for implementation on a real-time processor. In the most simple case these filters are linear, furthermore they all run at one single sampling frequency over networks that do not contain delay-free paths [3]. Various methods taken from the numerical analysis allow to represent continuous-time systems in terms of networks made of individual interconnected processing blocks [4, 5], that have a direct counterpart in a corresponding linear digital filter network. Alternatively, analog transfer characteristics of linear elasto-mechanic and fluid-dynamic systems can be computed in the Laplace domain [6]. From there, several techniques can be invoked to transform the analog characteristics into corresponding discrete-time transfer functions and, finally, into a digital filter network [7, 8].

In principle the discretization of an analog system does not prevent the final digital filter network to contain delay-free paths. In the linear case such a network can be always rearranged into a new filter network in which the delay-free paths have been formally solved by composing the filters belonging to them into bigger linear structures that ”embed” the loop [3]. Nevertheless there are cases where this rearrangement is deprecated. These include situations in which the access to the filter parameters becomes too complicated after the rearrangement. Furthermore, the elimination of a delay-free path implies that all the branches belonging to it cannot be used any longer as input/output points where to inject/extract the signal to/from the system: this problem is particularly relevant in the design of virtual musical instruments by physical modeling [9, 10].

The analog-to-digital conversion becomes even more complicated when one or more nonlinearities exist in the continuous-time system. Linearizing the system before the conversion is often forbidden since the nonlinearity adds unique features, impossible to reproduce with a linear behavior. As long as nonlinearities are involved in the conversion new problems come into play, the most important of which is certainly the preservation of the system stability along with a precise simulation of the nonlinear characteristic [11, 12, 13, 14]. One further complication arises if a nonlinearity is part of a delay-free path: in this case there is no general procedure to rearrange the transfer functions of the loop to come up with a new linear structure in which to embed the delay-free path.

Moved by the idea of defining a general method for the representation of nonlinear filter networks, we have pointed out a strategy that in principle enables to model a network of (one-dimensional, although the method can be extended to the multi-dimensional case) nonlinear and linear filter realizations, even in presence of delay-free paths. Here we will assume that each filter block—both linear and nonlinear—has been already precisely modeled, and it is ready to be included in the network.

We also implicitly assume the stability of the resulting system. This assumption, because the method does not deal with the analog-to-digital conversion problem and the related stability issues [19, 4, 14]. Instead, it focuses on the way linear and nonlinear
filters can be connected each other from a purely structural viewpoint, preserving their original position in the network in terms of input/output mutual relations. In particular we will deal with the case when nonlinear blocks are part of a delay-free loop, and provide a method to compute those loops without rearranging them into a different topology.

2. FORMAL SOLUTION OF NONLINEAR DELAY-FREE PATHS

A technique to compute simple linear delay-free paths without rearranging the loopback topology has been first proposed by Härmä [15]. This technique has been successfully employed for the computation of warped IIR filters [16] and magnitude-complementary parametric equalizers [17], and it has been generalized to linear filter networks containing an arbitrary delay-free path configuration [18]. Here we will extend the same technique to networks containing nonlinear blocks.

We need to:
1. rearrange every linear and nonlinear transfer function, in order to separate the contribution of the instantaneous from the historical component in the function;
2. define additional equations accounting for the connections between blocks;
3. evaluate the historical component in every (both linear and nonlinear) block;
4. substitute the linear constraints into the nonlinear equations.
5. compute, if possible, the output from every linear block;
6. correctly update the historical components in each block.

In the following we address each point in detail.

2.1. Rearrangement of the transfer characteristics

The network is made of \( m_L \) linear filters, here expressed using their transfer function

\[
H_i(z) = \sum_{k=0}^{Z_i} b_{i,k} z^{-k}, \quad i = 1, \ldots, m_L, \tag{1}
\]

and \( m_N \) nonlinear blocks, specified by writing their transfer characteristic in the discrete time from the input \( x_i \) to the output \( y_i \):

\[
y_i[n] = f_i(x_i[n], p_i[n]) \quad i = 1, \ldots, m_N. \tag{2}
\]

In writing (2), it is implicitly assumed that the nonlinear blocks respect two hypotheses:

- the output \( y_i \) can be made explicit. More weakly, that every nonlinearity admits the existence of a transfer function \( f_i \) in the form expressed by (2) [19];
- \( p_i \) contains only the contribution of the historical components in the function in a way that we can evaluate the nonlinearity for past input and output values, thus obtaining a new function in the single variable \( x_i \):

\[
y_i[n] = f_i(x_i[n], p_i[n]) \quad i = 1, \ldots, m_N
\]

\[
p_i[n] = p_i(x_i[n-1], y_i[n-1], p_i[n-1], \ldots) \tag{3}
\]

In practice this class of nonlinear functions is sufficiently expressive for a wide range of audio applications [12, 14, 20].

On the other hand, in the linear case the historical component \( q_i \) can be immediately found out in every transfer function by gathering all past components together in the time-domain version of (1):

\[
y_i[n] = b_i x_i[n] + q_i[n] \quad i = 1, \ldots, m_L \tag{4}
\]

\[
q_i[n] = \sum_{k=1}^{Z_i} b_{i,k} x_i[n-k] + \sum_{k=1}^{P_i} a_{i,k} y_i[n-k] \tag{5}
\]

in which the coefficient \( b_{i,k} \) has been simply denoted as \( b_i \).

2.2. Connections between blocks

Without loss of generality we can assume that the whole network forms one single delay-free loop. Alternatively we can extract from the filter network a number of subgraphs forming individual delay-free subnetworks, then treat each of them separately using the method presented below, and finally feed the remaining blocks in the network with the output from such subnetworks\(^1\) [21].

For each of the \( m_L + m_N \) block inputs we consider the \( R_i \) outputs from other blocks that feed that input, possibly with the addition of an external signal \( u_i \) directed to the same input:

\[
x_i[n] = \sum_{k=1}^{R_i} y_{ik}[n] + u_i[n] \quad i = 1, \ldots, m_L + m_N. \tag{6}
\]

Note that the paths that connect \( y_i \) and \( x_i \) directly must be treated as separate network branches. In other words, the condition

\[
i_k \neq i \quad k = 1, \ldots, R_i \tag{7}
\]

must be satisfied for any \( i \). For this purpose, such branches must result in corresponding equations that are a particular case of (4):

\[
y_i = x_i.
\]

Equations (6) establish the correction between all the blocks. Figure 1 depicts the situation in the case of a branch formed by a linear block: in this case the historical component is computed simply by feeding the filter with a null value [15].

\[
\begin{align*}
\text{Figure 1: Structure of a linear filter branch. The output } y_i \text{ is obtained as a superposition of the instantaneous component } x_i, \text{ plus the historical component } q_i. \text{ The input is the result of summing the outputs } y_{i1}, \ldots, y_{iR_i} \text{ from other branches plus one (possibly null) external input } u_i. \\
\end{align*}
\]

\(^1\) Although in the pure linear case the proposed method works for any network topology [18].
2.3. Solution of the system

Once the previous algebra has been carried out we come up with the following equations:

\[
\begin{align*}
&y_i[n] = f_i(x_i[n], p_i[n]), &i = 1, \ldots, m_N \\
&y_i[n] = b_i x_i[n] + q_i[n], &i = m_N + 1, \ldots, m_N + m_L \\
&x_i[n] = \sum_{k=1}^{N} y_{ka}[n] + u_i[n], &i = 1, \ldots, m_N + m_L
\end{align*}
\]

Such equations can be rewritten in matrix form as

\[
\begin{align*}
y_N[n] &= f(x_N[n], p[n]), \\
y_L[n] &= B x_L[n] + q[n], \\
x_R[n] &= C y[n] + u[n],
\end{align*}
\]

in which the column vectors \(x_N, y_N, x_L, y_L, p, q, u\) collect the respective signal components and \(y, x, f\) are defined as

\[
y = \begin{bmatrix} y_N \\ y_L \end{bmatrix}, \quad x = \begin{bmatrix} x_N \\ x_L \end{bmatrix}, \quad f(x_N, p) = \begin{bmatrix} f_1(x_1, p_1) \\ \vdots \\ f_{Nm}(x_{Nm}, p_{Nm}) \end{bmatrix}.
\]

Furthermore \(B\) is a diagonal matrix containing the linear coefficients \(b_1, \ldots, b_{m_L}\) and \(C\) accounts for the connections between blocks: if the element \(c_{ij}\) is equal to one then the \(j\)th input block is connected to the \(i\)th input block, otherwise it is equal to zero. Note that the property (7) translates into the fact that every diagonal element in \(C\) is equal to zero.

In particular, \(C\) can be seen as composed of four sub-matrices respectively accounting for the connections from \((C_{NN})\) nonlinear to linear, \((C_{NL})\) linear to nonlinear, \((C_{LN})\) nonlinear to linear, and finally \((C_{LL})\) linear to linear blocks. For this reason in (8c) we also split the vector \(u\) into \(u_N\) and \(u_L\):

\[
x_N \quad x_L = \begin{bmatrix} C_{NN} & C_{NL} \\ C_{LN} & C_{LL} \end{bmatrix} \begin{bmatrix} y_N \\ y_L \end{bmatrix} + \begin{bmatrix} u_N \\ u_L \end{bmatrix}.
\]

Substitution of equations (8a) and (8b) in (9) gives

\[
\begin{align*}
x_N &= C_{NN} f(x_N, p) + C_{NL} (B x_L + q) + u_N, \\
x_L &= C_{LN} f(x_N, p) + C_{LL} (B x_L + q) + u_L.
\end{align*}
\]

Let \(I\) denote the identity matrix, then in (10a) and (10b) we can move all terms to the left-hand side except for the linear historical components and the external inputs, respectively:

\[
\begin{align*}
x_N - C_{NN} f(x_N, p) - C_{NL} B x_L &= C_{NL} q + u_N, \quad (11a) \\
- C_{LN} f(x_N, p) + (I - C_{LL} B) x_L &= C_{LL} q + u_L. \quad (11b)
\end{align*}
\]

If the matrix \(I - C_{LL} B \nRightarrow F_{LL}\) is invertible\(^2\) then we can isolate \(x_L\) in (11b):

\[
x_L = F_{LL}^{-1} C_{LN} f(x_N, p) + F_{LL}^{-1} (C_{LL} q + u_L),
\]

and substitute this formula inside (11a) in a way that \(x_L\) is eliminated and \(x_N\) remains the only unknown vector in that equation. After some algebraic manipulation (11a) is in fact carried out as:

\[
x_N = W_{1} f(x_N, p) + W_{2} q + W_{3} u_L + u_N
\]

with

\[
\begin{align*}
W_3 &= C_{NL} B F_{LL}^{-1} \\
W_1 &= W_{3} C_{LN} + C_{NN} \\
W_2 &= W_{3} C_{LL} + C_{NL}
\end{align*}
\]

2.4. Solving nonlinearities

Equation (13) defines the inputs \(x_N[n]\) to the nonlinear blocks in terms of known quantities: the historical components \(q[n]\) and the external inputs \(u_L[n], u_N[n]\). In addition the matrix \(W_1\) isolates the instantaneous dependence of \(x_N[n]\) on \(y_N[n]\). From (8a) and (13), the nonlinear equations can be written as

\[
y_N[n] = f(W_1 y_N[n] + W_2 q[n] + W_3 u_L[n] + u_N[n], p[n]). \quad (15)
\]

Recall that \(p\) contains only the contributions of historical components (see equation (3)). Therefore at each time step \(y_N\) is the only unknown in (15) and \(p\) parametrizes the function \(f\):

\[
y_N[n] = \tilde{f}_p(W_1 y_N[n] + \tilde{x}_N[n]), \quad (16)
\]

where \(\tilde{f}_p(\cdot) = f(\cdot, p[n])\) and \(\tilde{x}_N[n] = W_{2} q[n] + W_{3} u_L[n] + u_N[n]\) collects the contribution of known quantities to the input \(x_N\).

Equation (16) provides a formulation which has strong similarities with that developed in [19]. According to it, the nonlinear function \(\tilde{f}_p\) defines implicitly the dependence of \(y_N[n]\) on \(\tilde{x}_N[n]\).

In addition to the formulation given in [19], the nonlinearity is here parametrized by the historical components \(p\).

It was shown in [19] that, under appropriate conditions for the matrix \(W_1\), the implicit dependence \(y_N(\tilde{x}_N)\) admits a global representation, and can therefore be precomputed and stored in a look-up table for efficient implementation. However, the efficiency of a table look-up drops dramatically with increasing dimensionality of the input. Furthermore, in our case the input to the table comprises not only \(\tilde{x}_N\), but also the historical components \(p\).

An alternative choice to table look-up is iterative search: at each time step the value \(y_N[n]\) is found by searching a local zero of the function

\[
g[y_N] = \tilde{f}_p(W_1 y_N + \tilde{x}_N) - y_N \quad (17)
\]

The Newton-Raphson algorithm operates the search in this way:

\[
y_N[k] = y_N[n] \quad k = 1, \quad \text{while } \text{err} > \text{Errmax} \\
\text{Compute } g(y_N[k]) \text{ from Eq. (17)} \\
\text{Compute } y_N[k+1] = y_N[k] - J_k^{-1} g(y_N[k]) \\
\text{Compute } \text{err} = \text{abs}(y_N[k+1] - y_N[k]) \\
\text{end}
\]

where \(J_k = \left[ \frac{\partial g}{\partial y_N[k]} \right] \) is the Jacobian of \(g\) evaluated in \(y_N[k]\).
2.5. Filter update

Once the values of $x_N[n]$ and $y_N[n]$ have been found, we can compute $x_L[n]$ from (12), and $y_L[n]$ from (8b).

At this point of the procedure the system not only produces the output $y[n]$ but it is also ready to update all the nonlinear and linear blocks correctly, that is, to inject the components of $x[n]$ to the respective blocks. As a consequence of the latter operation new values of the historical components, $p[n+1]$ and $q[n+1]$, become available for the next step of the procedure.

2.6. Summary of the procedure

It is convenient to summarize the overall computations that take place in the system during each step (refer also to Figure 2):

1. $y_N[n]$ is computed by means of (16) using the external inputs $u_N[n]$ and $u_L[n]$, and the historical components $p[n]$ and $q[n]$;
2. $x_N[n]$ is computed from (13);
3. $x_L[n]$ is computed from (12);
4. $y_L[n]$ is computed from (8b);
5. $p[n+1]$ is found out by collecting the historical components from every nonlinear block, according to (3)—as already mentioned, we do not investigate particular forms that this equation takes;
6. $q[n+1]$ is produced by collecting the historical components from every linear filter, for instance using (5) or, alternatively, by feeding each filter with a null signal [15].

Though, no computations are necessary if the filters are realized in transposed direct form [3, 18].

In the time-invariant case the method requires $O((m_N + m_L)^2)$ computations to compute the output at each temporal step, plus those needed to solve (17). Although the quadratic complexity is competitive especially when a network contains many filter connections [18], nevertheless the effort taken to solve the nonlinearity can play a major role in the overall computational complexity if the Newton-Raphson algorithm converges slowly. For this reason, a real time implementation should rather consider to use pre-computed solutions of (17), hence turning time into space (i.e., memory) consumption, except for specific cases in which Newton-Raphson is guaranteed to converge in a given number of steps.

3. AN EXAMPLE: NONLINEAR CONTACTING RESONATORS

In order to provide an example application, the method presented above has been tested on a physical model of two colliding resonators.

3.1. Modal description

The contacting objects are modeled as modal resonators, following the approach described in [22]. However, here the modal description is extended in order to account for non-constant center frequency of the resonator modes. Each mode of the resonator is...
therefore described in the continuous-time domain as:
\[ y_j^{(r)} + g_j y_j^{(r)} + \left[ \omega_{0j}^2 + \omega_{ij}^2 \cdot y_j^{(r)} \right] y_j^{(r)} = \frac{1}{m_j} f_{tot}^{(r)}, \quad (18) \]
where \( y_j^{(r)} \) is the displacement of the \( j \)-th mode of the resonator and \( f_{tot}^{(r)} \) represents the net sum of forces that drive the resonator. The parameter \( g_j \) is the modal damping coefficient, and \( 1/m_j \) controls the “inertial” properties of the mode (\( m_j \) has the dimension of a mass). Note that a nonlinear correction \( \omega_{ij}^2 \cdot y_j^{(r)} \) has been added to the center frequency \( \omega_{0j} \).

The effect of this correction is to increase the instantaneous center frequency with increasing displacement\(^3\).

Equation (18) can be rewritten as
\[ y_j^{(r)} + g_j y_j^{(r)} + \omega_{0j}^2 y_j^{(r)} = \frac{1}{m_j} \left[ f_{tot}^{(r)} - m_j \omega_{0j}^2 \right] y_j^{(r)} + \mathrm{sgn}(y_j^{(r)}) \left| y_j^{(r)} \right|^\alpha, \quad (19) \]
where a second order linear oscillator is recognized on the left-hand side, and the nonlinear correction is provided as an input to the oscillator on the right-hand side.

### 3.2. Contact

Given two nonlinear modal resonators \((r = 1, 2)\) as described above, an impulsive contact (collision) between them is modeled by assuming that the force acting on each mode is given by
\[ f_{tot}^{(r)} = f_{ext}^{(r)} + (-1)^r f_c, \quad (20) \]
where \( f_c \) represents the contact force that occurs during collision. This is generated using the nonlinear model discussed in [22]:
\[ f_c(y_c, \dot{y}_c) = \left\{ \begin{array}{ll}
ky_c^2 + \lambda y_c^3 \cdot \dot{y}_c & y_c > 0, \\
0 & y_c \leq 0,
\end{array} \right. \quad (21) \]
where \( y_c = y^{(1)}_c - y^{(2)}_c \) is the inter-penetration of the two objects and \( y^{(1)}_c, y^{(2)}_c \) are the object displacements, obtained as linear combinations of the respective modal displacements (see [22]).

### 3.3. Discrete-time system

The simple mechanical system described above can be transposed into the discrete-time domain using the formulation outlined in Section 2. The number \( m_L \) of linear blocks equals the total number of linear oscillators (19). The number \( m_N \) of nonlinear blocks equals the number of non-null correction terms \( \omega_{ij} \) plus one, i.e. the contact force (21).

### Nonlinear blocks

The inputs to the nonlinear blocks are defined as follows:
\[ x_i := y_i^{(r)}, \quad (i = 1, \ldots m_N - 1, \ j : \omega_{ij} \neq 0), \quad (22) \]
\[ x_{m_N} := y_c, \]

The external inputs \( u_L \) are zero. The historical components \( p \) are all zero except for the last one:
\[ p_i = 0 \quad (i = 1, \ldots m_N - 1). \quad (23) \]

In order to compute the contact force (21), the inter-penetration velocity \( \dot{y}_c \) is also needed. This is estimated in the discrete-time domain using the bilinear transformation:
\[ \dot{y}_c[n] = 2F_c (y_c[n] - y_c[n-1]) - y_c[n-1] \]
\[ = 2F_c x_{m_N}[n] + p_{m_N}[n]. \quad (24) \]

where the historical component \( p_{m_N}[n] \) is defined as
\[ p_{m_N}[n] := 2F_c x_{m_N}[n-1] + y_c[n-1] \]
\[ = 4F_c x_{m_N}[n-1] - p_{m_N}[n-1]. \quad (25) \]

The nonlinear characteristics are then given as
\[ f_c(x_i, p_i) := -m_j \omega_{ij}^2 \cdot (x_i)^{y_j} + 1, \quad (26a) \]
\[ (i = 1, \ldots m_N - 1, \ j : \omega_{ij} \neq 0), \]
\[ f_{m_N}(x_{m_N}, p_{m_N}) := f_c(x_{m_N}, 2F_c x_{m_N} + p_{m_N}). \quad (26b) \]

### Linear blocks

The inputs to the linear blocks are defined as follows:
\[ x_i := y_i - m_N + (-1)^r y_{m_N}, \quad (i = 1, \ldots m_N + 1, \ldots m_N + m_L - 1), \quad (27) \]
\[ x_i := (-1)^r y_{m_N}, \quad (i = 2m_N, \ldots m_N + m_L), \]

with \( r(= 1, 2) \) such that the \( i \)-th input belongs to resonator \( r \). The external inputs \( u_L \) are simply
\[ u_i := f_{ext}^{(r)}, \quad (i = m_N + 1, \ldots m_N + m_L) \quad (28) \]

with \( r(= 1, 2) \) such that the \( i \)-th input belongs to resonator \( r \).

Correspondingly, the outputs from the linear blocks are the modal displacements:
\[ y_1 := y_j^{(r)} \quad (i = m_N + 1, \ldots m_N + m_L, \ j = 1, \ldots m_L). \quad (29) \]

In order to compute the transfer functions \( H_i(z) \) between the forces \( x_i \) and the modal displacements \( y_i \), equation (19) is discretized with the bilinear transformation. It can be seen that the filter coefficients \( a_{1,i}, a_{2,i} \) defined in (1) are given by
\[ a_{1,i} = \omega_{ij}^2 + 4F_c^2 \]
\[ a_{2,i} = \frac{\omega_{ij}^2 + 4F_c^2}{2\Delta_j}, \quad (i = m_N + 1, \ldots m_N + m_L). \quad (30) \]

where the quantity \( \Delta_j \) is given by
\[ \Delta_j = F_c^2 + g_j F_c/2[\omega_{0j}]^2 / 4. \]

### 3.4. Summary of the example

The above discussion has shown that the mechanical system under consideration can be assembled using three kind of computational blocks:

- The second-order filter whose coefficients are given in (30) is instantiated as many times as the number of modeled resonating modes.
- The nonlinear memoryless block (26a) is instantiated as many times as the number of non-null correction terms \( \omega_{ij} \).
4. CONCLUSION

We have presented a general procedure that enables to model a generic network of nonlinear and linear computational blocks that satisfy weak hypotheses.

It has been shown that the network can be solved even in the presence of delay-free paths that involve nonlinear blocks. Moreover, the proposed solution does not require any rearrangement of the network and preserves the original topology of the network, in terms of mutual input/output relations between blocks.

The example application provided in Section 3 has shown that the proposed procedure allows for a highly modular formulation of the system. Each computational block has a clear physical meaning: in the example the linear blocks are second order oscillators that represent modes of resonating objects, while nonlinear blocks are forces.

Each computational block can therefore be modeled independently. Then, provided that a connection topology is specified, the global computational structure for the complete system is constructed automatically using the procedure given in Section 2.

In the context of this research, the most desirable feature of this methodology would be to establish general criteria to guarantee the preservation of the structural properties of the system while moving from the analog to the digital domain, in a way that at the end of such a translation each filter block has a clear physical meaning. We are currently working on extending the scope of the method in this direction.

5. REFERENCES

MODAL-TYPE SYNTHESIS TECHNIQUES FOR NONLINEAR STRINGS WITH AN ENERGY CONSERVATION PROPERTY

Stefan Bilbao
Sonic Arts Research Centre
Queen's University Belfast
s.bilbao@qub.ac.uk

ABSTRACT
There has recently been increased interest in the modelling of string vibration under large amplitude conditions, for sound synthesis purposes. A simple nonlinear model is given by the Kirchhoff-Carrier equation, which can be thought of as a generalization of the wave equation to the case for which the string tension is "modulated" by variations in the length of the string under deformation. Finite difference schemes are one means of approach for the simulation of nonlinear PDE systems: in this case, however, as the nonlinearity is spatially invariant, the solution may be broken down into sinusoidal components, much as in the linear case. More importantly, if time discretization is carried out in a particular way, it is possible to obtain a conserved energy in the numerical scheme, leading to a useful numerical stability guarantee, which can be difficult to obtain for strongly nonlinear systems. Numerical results are presented.

1. INTRODUCTION
When it is desired to simulate the transverse motion of a vibrating string, in a single polarization, a simple starting point is the 1D wave equation [1]. There are many approaches to the simulation of such an equation: the most straightforward makes use of finite difference approximations [2, 3, 4]. In this case, the defining PDE is discretized, leading to a solution approximated at various grid points, and at a finite set of time instants. Digital waveguides [5] represent a particularly elegant special case of a finite difference method, for which the numerical solution may be calculated as a sum of discrete travelling waves, at a greatly reduced computational cost. When the wave equation is complemented by additional terms which model such perceptually crucial effects as loss and dispersion [6, 7], finite difference schemes may still be used; for certain schemes, an equivalent digital waveguide formulation also follows [8], and it is also possible to construct quasi-physical waveguide structures which remain relatively cheap, computationally speaking [9].

Because the wave equation (and its extensions as mentioned above) is linear and time-invariant, a complete description is available in the frequency domain; the behavior of the string may be broken down into independent contributions from various modes, each of a particular shape and frequency. The shapes and frequencies are strongly dependent on the particular type of boundary conditions applied. Synthesis based on such a decomposition is often referred to as modal synthesis [10]; it may be extended to deal with general linear and time-invariant systems. If the system is, in addition, spatially-invariant (i.e., if there is no material parameter variation), all solutions may be expressed in terms of modal contributions, the shapes of which are complex exponentials; the wave equation for a string of constant density is of this form, but the vocal tract, for instance, is not (it still possesses modes whose time-dependence is complex exponential, but these are not spatially complex exponentials).

Under high-amplitude conditions, the 1D wave equation is no longer a good model of transverse wave propagation. The most general models of string vibration are nonlinear, and involve pointwise coupling among the two transverse polarizations and the longitudinal displacement [1, 11]. Under certain assumptions [12], generally valid for many types of strings which appear in a musical setting, transverse motion in a single polarization may be decoupled, and a simplified nonlinear equation, sometimes called the Kirchhoff-Carrier equation [13, 14], results. In this model, the effect of the nonlinearity is global—in other words, though nonlinear, the equation remains spatially shift-invariant. Such a model has served as the starting point for extensions of digital waveguides [15, 16, 17], as well as finite difference schemes [18].

Due to the spatial invariance of the nonlinearity, one might expect that a modal-type description will be available for this nonlinear model; although the string does not possess modes as such, a breakdown of the string into simple sinusoidal shapes suggests a numerical simulation approach which is similar to modal synthesis. One of the great benefits of such an approach is that it is possible to arrive at a useful stability condition on the numerical scheme; stability-checking machinery such as von Neumann type analysis [19, 20, 21] is generally not valid in the nonlinear case. Such analysis, as in the case of finite difference schemes, relies on strict energy conservation properties [22, 21].

In Section 2, we present the Kirchhoff-Carrier equation, as well as its expansion into a first order system, and briefly review its energy conservation properties. In Section 3, we first present a Fourier decomposition of the solution to the system, and show how this leads to a system of ordinary differential equations, which may be discretized in such a way as to yield a conserved energy-like quantity. We then discuss the conditions under which this conserved quantity leads to a numerical stability guarantee, and conclude with a look at spurious oscillatory behavior and some implementation details. Numerical simulation results are presented in Section 4.

2. THE KIRCHHOFF-CARRIER EQUATION
The Kirchhoff-Carrier equation, as mentioned above, is a good first approximation to nonlinear behavior of a string, in a transverse...
polarization. It can be written as

$$\rho \frac{\partial^2 u}{\partial t^2} = \left( T_0 + \frac{EA}{2L} \int_0^L \left( \frac{\partial u}{\partial x} \right)^2 dx \right) \frac{\partial^2 u}{\partial x^2}$$  (1)

Here, \( u(x, t) \) is the transverse string displacement, \( t \geq 0 \) is a time variable, and \( x \in [0, L] \) is a space variable. The string is characterized by the parameters \( \rho \) (linear mass density), \( T_0 \) (applied tension), \( E \) (Young’s modulus), and \( A \) (cross-sectional area) \([17, 23, 24, 25]\). When the spatial derivative \( \frac{\partial u}{\partial x} \) is small, it approaches the wave equation. It is simple, as in the linear case, to introduce a term (proportional to linear mass density), \( \rho \), and to characterize by the parameters \( \rho \) and \( \frac{\partial u}{\partial x} \).

Here, \( \rho \) is a variable, \( x \in [0, L] \) is a space variable, and \( \rho \) is a time variable, and \( \rho \) is a parameter of the system.

2.1. First-order System

As discussed in [18], for energetic analysis purposes, it is useful to reduce (1) to a system in the new variables \( p \) and \( q \) defined by

$$p = \sqrt{\rho} \frac{\partial u}{\partial t} \quad q = \sqrt{T_0} \frac{\partial u}{\partial x}$$

in which case it can be written as

$$\begin{align*}
\frac{\partial p}{\partial t} &= c_0 \frac{\partial q}{\partial x} \\
\frac{\partial q}{\partial t} &= c_0 \frac{\partial p}{\partial x}
\end{align*}$$  (2a, 2b)

where we have introduced the quantities \( c_0 = \sqrt{T_0/\rho} \) and \( G \), as defined by

$$G \triangleq \left(1 + B \int_0^L q^2 dx\right)$$  (3)

with \( B \triangleq EA/2LT_0^2 \).

2.2. Energy Conservation

As also discussed in [18], the Kirchhoff-Carrier equation implies a conservation law, given by

$$E_{KC} = \frac{1}{2} \|p\|^2 + \frac{1}{2} \left(1 + \frac{B}{2} \|q\|^2\right) \|q\|^2$$  (4)

where \( \|f\| = \left( \int_0^L f^2 dx \right)^{1/2} \) for square-integrable functions \( f \in L^2(0, L) \). This further implies the bounds

$$\|p\| \leq \sqrt{2E_{KC}} \quad \|q\| \leq \sqrt{\frac{-1 + \sqrt{1 + 4BE_{KC}}}{B}}$$  (5)

In other words, the size of the state of the solution is bounded in terms of the initial energy present in the string.

3. FOURIER DECOMPOSITION AND A NUMERICAL SCHEME

Under fixed boundary conditions at either end of the string, i.e., \( u(0, t) = u(L, t) = 0 \), in terms of \( p \) and \( q \), we must have

$$p(0, t) = p(L, t) = 0 \quad \frac{\partial q}{\partial x} \bigg|_{0,t} = \frac{\partial q}{\partial x} \bigg|_{L,t} = 0$$

These conditions suggest the following sine and cosine decompositions of \( p \) and \( q \):

$$\begin{align*}
p(x, t) &= \sqrt{\frac{T_0}{L}} \sum_{n=1}^{\infty} p_n(t) \sin(\pi nx/L) \\
q(x, t) &= \sqrt{\frac{T_0}{L}} \sum_{n=1}^{\infty} q_n(t) \cos(\pi nx/L)
\end{align*}$$  (6a, 6b)

which satisfy the above boundary conditions automatically; for free terminations, one may proceed equally easily. (We note that, although in general, the expression for \( q(x, t) \) could contain a DC term \( Q_0(t) \); the identification of \( q(x, t) \) with \( \sqrt{T_0/\rho} \) means that \( q(x, t) \) will be zero mean for any differentiable initial condition \( \frac{\partial u}{\partial x} \).

Parseval’s relation implies that

$$\|P\| = \|p\| \quad \|Q\| = \|q\|$$

where \( \|F\| = (\sum_{n=1}^{\infty} F_n^2)^{1/2} \) for any square-summable sequence \( F_n \) (such as \( P_n \) or \( Q_n \) above). An equivalent form of the conserved energy (4) is then

$$E_{KC} = \frac{1}{2} \|P\|^2 + \frac{1}{2} \left(1 + \frac{B}{2} \|Q\|^2\right) \|Q\|^2 = \text{constant}$$  (7)

Upon substituting the expressions (6) into system (2), one obtains

$$\begin{align*}
dP_n &= -c_0 G \pi n \frac{\pi n}{L} Q_n \\
dQ_n &= c_0 \pi n M \frac{\pi n}{L} P_n
\end{align*}$$  (8a, 8b)

for \( n = 1, 2, \ldots \) which is an infinite set of coupled ordinary differential equations, with the coupling occurring through \( G = 1 + B \|Q\|^2 \). Such a Fourier decomposition for the Kirchhoff-Carrier equation was analyzed some time ago by Dickey [26].

3.1. Time Discretization

In order to numerically integrate the system (8), there are two approximations which must be made. First, we truncate the Fourier series representation of \( p \) and \( q \) to \( M \) terms. It is useful to introduce vectors containing the first \( M \) components, namely,

$$P(t) = [P_1, \ldots, P_M]^T \quad Q(t) = [Q_1, \ldots, Q_M]^T$$

Then, we approximate \( dP/dt \) and \( dQ/dt \) as

$$\begin{align*}
\frac{dP}{dt} \bigg|_{t=n \frac{k}{2}} &\approx \frac{1}{k} (P^n - P^{n-1}) \\
\frac{dQ}{dt} \bigg|_{t=n \frac{k}{2}} &\approx \frac{1}{k} (Q^{n+\frac{1}{2}} - Q^{n-\frac{1}{2}})
\end{align*}$$

The quantities \( P^n \) and \( Q^{n+\frac{1}{2}} \) are second-order approximations to \( P((n - \frac{1}{2})k) \) and \( Q(nk) \), respectively; \( k \) is the time step. Notice, in particular, that the approximations are interleaved, i.e., we calculate \( P^n \) and \( Q^{n+\frac{1}{2}} \) in alternation, at intervals of \( k/2 \) seconds.

The inner product of two \( M \)-component expansions \( F \) and \( G \) is defined in the usual way as

$$\langle F, G \rangle = F^T G = \sum_{m=1}^{M} F_m G_m$$  (9)
and the norm of any expansion $\mathbf{F}$ follows as
\[
\|\mathbf{F}\| = (\mathbf{F}^T \mathbf{F})^{1/2}
\] (10)

### 3.2. An Interleaved Finite Difference Scheme

System (8) then becomes
\[
\begin{align*}
\mathbf{P}^n - \mathbf{P}^{n-1} &= -c_0 k G^{n-\frac{1}{2}} \mathbf{D} \mathbf{Q}^{n-\frac{1}{2}} \\
\mathbf{Q}^{n+\frac{1}{2}} - \mathbf{Q}^{n-\frac{1}{2}} &= c_0 k \mathbf{D} \mathbf{P}^n
\end{align*}
\] (11a, 11b)

Here, $\mathbf{D}$, as defined by
\[
\mathbf{D} = \frac{\pi}{L} \text{diag}(1, \ldots, M)
\] (12)
is the spatial derivative operator, and $G^{n-\frac{1}{2}}$ is an approximation to $\mathbf{G}$ at time $t = (n - \frac{1}{2}) k$, to be specified shortly; in order that the scheme (11) remain second-order accurate, the approximation $G^{n-\frac{1}{2}}$ should be second-order accurate. In order that the scheme remain explicitly computable, it should also be a function only of $Q^{n-\frac{1}{2}}$.

### 3.3. Energy Conservation

In order to examine the energetic behavior of system (11), we may proceed in the following way: first, left-multiply (11a) by $\frac{1}{2} (\mathbf{P}^n + \mathbf{P}^{n-1})^T$ to get
\[
\frac{\|\mathbf{P}^n\|^2 - \|\mathbf{P}^{n-1}\|^2}{2} = -\frac{c_0 k}{2} G^{n-\frac{1}{2}} (\mathbf{P}^n + \mathbf{P}^{n-1}, \mathbf{D} Q^{n-\frac{1}{2}})
\] Noting, from (11b), that
\[
c_0 k \mathbf{D} (\mathbf{P}^n + \mathbf{P}^{n-1}) = \mathbf{Q}^{n+\frac{1}{2}} - \mathbf{Q}^{n-\frac{1}{2}}
\]
we then arrive at
\[
\frac{\|\mathbf{P}^n\|^2 - \|\mathbf{P}^{n-1}\|^2}{2} = -\frac{G^{n-\frac{1}{2}}}{2} ((\mathbf{Q}^{n+\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}) - (\mathbf{Q}^{n-\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}))
\]
At this point, we may make the following choice for $G^{n-\frac{1}{2}}$
\[
G^{n-\frac{1}{2}} = 1 + \frac{B}{2} ((\mathbf{Q}^{n+\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}) + (\mathbf{Q}^{n-\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}))
\] (13)
which is consistent with definition (3), and second order accurate. This then yields
\[
\frac{\|\mathbf{P}^n\|^2 - \|\mathbf{P}^{n-1}\|^2}{2} = \frac{1}{4} ((\mathbf{Q}^{n+\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}) - (\mathbf{Q}^{n-\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}))
\]
\[
-\frac{B}{4} ((\mathbf{Q}^{n+\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}})^2 - (\mathbf{Q}^{n-\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}})^2)
\]
from which we can extract a conserved quantity $E_{KC}$, defined by
\[
E_{KC} = \frac{1}{2} (\|\mathbf{P}^n\|^2 + (\mathbf{Q}^{n+\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}}) + \frac{B}{2} (\mathbf{Q}^{n+\frac{1}{2}}, \mathbf{Q}^{n-\frac{1}{2}})^2)
\] (14)
which is similar to the energy definition (4), but which is not necessarily positive. The determination of conditions on its positivity (so that it may be used as a numerical stability guarantee) follows in the next Section.

We also note that from definition (13), it would appear that $G^{n-\frac{1}{2}}$ is dependent on $Q^{n+\frac{1}{2}}$, thus rendering our difference scheme implicit. It is simple to show, however, that $G^{n-\frac{1}{2}}$ may be rewritten as
\[
G^{n-\frac{1}{2}} = \frac{1 + B\|\mathbf{Q}^{n+\frac{1}{2}}\|^2}{1 + \frac{B\|\mathbf{Q}^{n-\frac{1}{2}}\|^2}{2}}
\] (15)

Notice, in particular, that a choice of $G^{n-\frac{1}{2}} = 1 + B\|\mathbf{Q}^{n-\frac{1}{2}}\|^2$, perhaps the most straightforward choice, does not lead to a simple energy conservation property.

### 3.4. Numerical Stability

The conserved quantity $E_{KC}$ given by (14), unlike its continuous-time counterpart (7), is not necessarily positive. In this Section, we find the conditions under which it is positive, in which case it can be used to bound the size of the calculated solution, thus serving as a numerical stability guarantee. We first rewrite (14), using (11b), as
\[
E_{KC} = \frac{1}{2} \left( \|\mathbf{P}^n\|^2 + \|\mathbf{Q}^{n+\frac{1}{2}}\|^2 - c_0 k (\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}}) \right)
\] + \frac{B}{4} \left( \|\mathbf{Q}^{n+\frac{1}{2}}\|^2 - c_0 k (\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}}) \right)^2
\]

Examine now the term $(\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}})$ in the above expression. From the Cauchy-Schwarz inequality [27], we clearly have
\[
(\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}}) \leq \|\mathbf{D}\| \|\mathbf{P}^n\| \|\mathbf{Q}^{n+\frac{1}{2}}\|
\]
and, furthermore,
\[
(\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}}) \leq \|\mathbf{D}\| \|\mathbf{P}^n\| \|\mathbf{Q}^{n+\frac{1}{2}}\|
\]
where $\|\mathbf{D}\|$ is the induced matrix 2-norm of $\mathbf{D}$ [27]; as $\mathbf{D}$ is simply a scaled diagonal matrix, as given by (12), we have $\|\mathbf{D}\| = \frac{\pi M}{L}$, and thus
\[
(\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}}) \leq \frac{\pi M}{L} \|\mathbf{P}^n\| \|\mathbf{Q}^{n+\frac{1}{2}}\|
\]
It then follows that
\[
E_{KC} \geq \frac{1}{2} \left( \|\mathbf{P}^n\|^2 + \frac{c_0 k \pi M}{2L} \|\mathbf{Q}^{n+\frac{1}{2}}\|^2 \right)^2
\]
\[
\frac{1}{2} \left( \frac{1 - \left( \frac{c_0 k \pi M}{2L} \right)^2}{|\mathbf{Q}^{n+\frac{1}{2}}|^2} \right)^2
\]
\[
+ \frac{B}{4} \left( \|\mathbf{Q}^{n+\frac{1}{2}}\|^2 - c_0 k (\mathbf{D} \mathbf{P}^n, \mathbf{Q}^{n+\frac{1}{2}}) \right)^2
\]
The quantity on the right-hand side of the above inequality is clearly positive if
\[
k \leq \frac{2L}{c_0 \pi M}
\] (16)

If this condition is satisfied (and notice that it does not depend in any way on values of the solution), then we further have that
\[
\|\mathbf{Q}^{n+\frac{1}{2}}\| \leq \frac{2E_{KC}}{1 - \left( \frac{c_0 k M}{2L} \right)^2}
\] (17)

Thus the norm of $\mathbf{Q}^{n+\frac{1}{2}}$ is bounded in terms of the energy $E_{KC}$, which remains constant. An identical bound can be found for $\|\mathbf{P}^n\|$. (We note that it should be possible to find tighter bounds through further analysis.)

This bound implies a further bound on $G^{n-\frac{1}{2}}$, as defined by (15), namely
\[
G^{n-\frac{1}{2}} \leq 1 + \frac{2B E_{KC}}{1 - \left( \frac{c_0 k M}{2L} \right)^2}
\] (18)
which holds for all $n$.
3.5. Oscillatory Behavior

3.5.1. Stability Condition

We have found, above, a condition for numerical stability; it is not, however, sufficient to ensure that our calculated solution is acceptable from a physical point of view, as numerical oscillatory behavior may be present, even if the solution is stable. To this end, we rewrite system (11) in state-space form as

\[
\begin{bmatrix}
P^n_m \\
Q^n_m
\end{bmatrix} = \begin{bmatrix}
I_M - c_0kG^n_{-\frac{1}{2}}D_m \\
-I_M - c_0kG^n_{-\frac{1}{2}}D_m
\end{bmatrix} \begin{bmatrix}
P^{n-1}_m \\
Q^{n-1}_m
\end{bmatrix}
\]

where \(I_M\) is the \(M \times M\) identity matrix. The \(2M\) eigenvalues \(\lambda^n\) of the update matrix (which is dependent on the time index \(n\)) are given by

\[
\lambda^n = -\nu^n \pm \sqrt{(\nu^n)^2 - 1} \quad \nu^n = -1 + \frac{c_0k^2x^2m^2G^n_{-\frac{1}{2}}}{2L^2}
\]

for \(m = 1, \ldots, M\). For \(|\nu^n| \leq 1\), or, in other words, if

\[
k \leq \frac{2L}{c_0\pi M \sqrt{G^n_{-\frac{1}{2}}}}
\]

then all eigenvalues occur as \(M\) complex conjugate pairs, of unit magnitude; if, however, \(\nu^n > 1\), some eigenvalues are both real and negative, and in particular, one will be of magnitude greater than one. We thus expect, in this case, to find sign-flipping (accompanied by amplification) occurring in the modes whose eigenvalues violate this condition. It is thus important to ensure that this does not occur. Using (18) and (19), we may find another bound on \(k\),

\[
k \leq \frac{2L}{c_0\pi M \sqrt{1 + BE_{KC} - \sqrt{(1 + BE_{KC})^2 - 1}}}
\]

which is expressed in terms of the initial energy \(E_{KC}\).

3.6. Implementation Details

The algorithm itself, described by (11), operates entirely on the sinusoidal expansions coefficients. The most important practical consideration is the transformation between the coefficients of the sinusoidal expansions of \(p\) and \(q\) and the physical solution; this can be done rather simply using Fourier transforms, though it must be kept in mind that the expansions \(P\) and \(Q\) are sine and cosine series, not general Fourier series expansions of functions over an interval of length \(L\). Also, due to the incorporation of boundary conditions into the series, we have chosen to expand into sinusoids of wavelengths \(2L/m\), for \(m = 1, \ldots, M\). For example, for a sequence \(p_i\), for \(i = 0, \ldots, M - 1\), representing some approximation to \(p\) (here, \(p_0\) is constrained to be zero by the boundary conditions), we can generate the expansion coefficients by taking the Fourier transform of the sequence

\[/p_0, p_1, \ldots, p_{m-1}, 0, -p_{m-1}, \ldots, -p_1\]

and taking the imaginary parts of the first \(M\) values (in other words, we expand \(p_i\) to an odd sequence of length \(2M\) samples). Similarly, a set of values representing \(q\) can be expanded to a length \(2M\) even sequence, and the real parts of the first \(M\) values will represent the cosine series coefficients. Simplifications are certainly possible, given the various symmetries of the sequences; we do not enter into the details here. It is important to note that if one is merely interested in viewing the state of the string after an elapsed time, there is no need to perform a Fourier transform until this instant. Otherwise, for synthesis purposes, a Fourier transform must be taken at each sampling instant. For this reason, it will probably be of interest to constrain \(M\) to be a power of two, so that the FFT algorithm may be used.

Another question is that of initialization. As mentioned previously, the Kirchhoff-Carrier system (1) requires two initial conditions, \(u(x, 0)\) and \(\frac{\partial u}{\partial t}(x, 0)\). Scheme (11) also requires two initial conditions, \(P^0\) and \(Q^0\); due to interleaving, they do not occur at the same time instant. \(P^0\) may be rather simply set as \(\sqrt{p}\) times the Fourier expansion coefficients of \(\frac{u}{\partial t}\) at \(t = 0\), but \(Q^0\) requires a more delicate treatment. The simplest means of proceeding is simply to find the set \(Q^0\) to be \(\sqrt{Q}\) times the cosine expansion coefficients of the spatial derivative \(\frac{\partial u}{\partial t}\) at \(t = 0\) (perhaps through spectral differentiation of the initial displacement \(u(x, 0)\), or some other means). This means that we will have to accept a first-order error, due to the offset in the initial data. Alternatively, it is possible to develop a special scheme to be used only once, for initialization purposes [19].

4. Numerical Simulations

In this Section we present a few simulation results, using the algorithm (11). In Figure 1 we show snapshots of the time evolution of a steel string, under center-plucked conditions, for a variety of amplitudes (parameters as given in the caption to the figure). For low amplitudes (left column), the motion is, as expected, very similar to what one would expect from the wave equation, i.e., a triangular initial displacement gives rise to simple propagating “corners.” As the amplitude is increased (second and third columns), the truncated triangular shape becomes progressively more distorted; notice also that the corners propagate more rapidly for higher amplitudes. This is in line with what we expect of nonlinear plucked excitations, i.e., there should be an increased wave speed, leading to an increased perceived frequency of oscillation (if it can be called that). This point is made more clearly in the upper row of Figure 2, where we show the displacement of the string center as a function of time, for the same plucking conditions as in the columns of Figure 1. As a test of the energy conservation properties of the algorithm, we have also plotted the difference between string energy and the initial energy, normalized by the initial energy, as a function of time, in the bottom row of Figure 2, for the same set of excitations. Notice that the energy error is zero, to machine precision.

5. Conclusions

We have discussed here an extension of Fourier techniques to a nonlinear model of string vibration. The algorithm presented in this paper is a special case of what is known as a spectral method [28, 29]. In short, the spatial derivative operator has been approximated by frequency domain multiplication, and is thus exact (i.e.,
has no truncation error), at least over the range of solutions which may be expressed in terms of a fixed number of sinusoidal components. In particular, the spatial accuracy is far greater than that of a finite difference scheme for the same system [18]. Time discretization limits temporal accuracy to second order; it would, of course, be possible to use higher-order accurate methods (perhaps of the Runge-Kutta variety) for time integration, but we have chosen a simple interleaved scheme in order to highlight the special energy conservation property which is crucial for stability analysis. Indeed, a robust stability guarantee is of paramount importance for sound synthesis, especially for real-time applications; this can be difficult to achieve for nonlinear systems. We have also shown a means of controlling parasitic oscillations, which, interestingly, arise independently of numerical stability.

6. REFERENCES


WAVE FIELD SYNTHESIS – A PROMISING SPATIAL AUDIO RENDERING CONCEPT

Günther Theile

Institut für Rundfunktechnik (IRT)
Munich, Germany
theile@irt.de

ABSTRACT

Modern convolution technologies offer possibilities to overcome principle shortcomings of loudspeaker stereophony by exploiting the Wave Field Synthesis (WFS) concept for rendering virtual spatial characteristics of sound events. Based on the Huygens principle loudspeaker arrays are reproducing a synthetic sound field around the listener, whereby the dry audio signal is combined with measured or modelled information about the room and the source’s position to enable the accurate reproduction of the source within its acoustical environment. Not surprisingly, basic and practical constraints of WFS systems limit the rendering accuracy and the perceived spatial audio quality to a certain degree, dependent on characteristic features and technical parameters of the sound field synthesis. However, recent developments have shown already that a number of applications could be possible in the near future. An attractive example is the synthesis of WFS and stereophony offering enhanced freedom in sound design as well as improved quality and more flexibility in practical playback situations for multichannel sound mixes.

1. INTRODUCTION

Three psychoacoustic fundamentally different spatial audio imaging methods should be distinguished:

- (Multichannel) loudspeaker stereophony
- Binaural reconstruction of the ear input signals
- Syntheses of the sound field around the listener

All known spatial sound systems can be traced back to one of these methods or can contain mixed forms thereof, whereby certain advantages of the methods are being exploited, respectively its disadvantages are avoided, dependent on the intended application area.

1.1. Loudspeaker stereophony

This is in principle based on the characteristics of localization in the superimposed sound field, generated by two loudspeakers [7]. Directional imaging is done in the imaging area between two adjacent loudspeakers [2]. In the case of 3/2 stereophony, with the assistance of surround channels the imaging area between the front loudspeakers can be extended. Therefore possibilities are offered for the reproduction of early lateral sound for imaging of spatial depth as well as reverberation, in order to produce the spatial impression and the envelopment. Details are described in [3].

1.2. Binaural reconstruction of the ear input signals

The original employment of this method is the known dummy head stereophony. It is not intended to reproduce a suitable sound field at the reproduction location. Instead, the effective ear signals in the recording location are recorded with the assistance of a dummy head – and replayed in principle via headphones. Under ideal circumstances, the reproduced binaural signals are identical to the original ear signals that the listener received in the recording location. In practice it is possible to reproduce auditory events with excellent realism regarding spatial characteristics and sound color.

1.3. Synthesis of the sound field around the listener

The third approach has been pursued within the framework of the European Research Project “CARRUSO” [4]. It is based on the concept of Wave Field Synthesis (WFS, developed at the Technical University Delft, refer e.g. [3], [6]), i.e. the representation of a virtual source and a virtual room is achieved by rendering an acoustically correct sound field. The principle of WFS is based upon the assistance of loudspeaker arrays, when a complete sound field is generated in the listening zone which is identical to an appropriate real sound event (see Section 2). This acoustical counterpart to the optical holography is also described as “holophon”. The binaural ear input signals that are active for the auditory event thus arise in a natural way within the sound field, contrary to dummy head stereophony.

1.4. Combining stereophony and WFS

Further developments of spatial audio systems are based on useful combinations of basic methods, using sophisticated real time convolution algorithm. This paper concentrates on overcoming certain practical drawbacks of multichannel sound on the one hand and of WFS on the other. So called “Virtual Panning Spots” (VPS) are introduced to improve the WFS rendering quality of large complex sources (e.g. “choir”), to reduce the number of WFS transmission channels and to ensure compatibility and scalability. Useful combinations of VPS and conventional or sophisticated stereophonic panning and mixing techniques will provide advanced facilities for spatial sound design. A special VPS application allows play back of conventional multichannel mixes in a virtual high quality listening room rendered by means of WFS technologies, offering full backwards compatibility with usual loudspeaker stereophony, optimum multichannel format flexibility, as well as attractive practical benefits in the home, in the cinema, or in other applications.
2. WFS PRINCIPLES AND PROPERTIES

2.1. The “Huygens” principle

“If from a point S of a homogeneous isotropic medium a spherical wave is emitted, one can imagine the procedure of the individual wave reproduction in that a particle brought into oscillation by external forces, transfers its movements to its neighboring particles. This procedure then continues symmetrically in all directions and in this way gives cause to a spherical wave...” [7].

If a sound source S (the so-called “primary source”) emits spherical wave fronts, one can imagine in accordance with the “Huygens” Principle each emitted wave front (refer to Figure 1a blue), through the addition of all participating “secondary sources” (which also emit spherical waves) on the surface O. Due to the knowledge of the wave front on surface O (Figure 1a red), the state of the oscillation can be determined at an arbitrary point P of the sound field. The wave front through point P is constructed through the summation of all participating secondary source signals.

![Figure 1: The “Huygens” Principle: (a) Theoretical Model; (b) Application WFS.](image)

In principle, in the case of the WFS, one replaces the secondary point sources by loudspeakers and in this way again produces a spherical wave (refer to Figure 1b). The sound source S is virtual; the listener in point P receives the same wave front which is transmitted by the sound source S.

2.2. WFS – the application of the “Huygens” Principle

This applies correspondingly for a circular arrangement of the loudspeakers on a two dimensional level. Concerning a sound source S, (which emits a sine impulse and is located in an infinitely large plane without demarcation of walls), a wave front results as illustrated in Figure 2a. If one now places an array of n microphones (M) in this primary sound field and one reproduces the recorded microphone signals via an equally arranged array of n loudspeakers (L) – special equalization has to be included according to the relevant physical basics – in a reproduction room (Figure 2b), one obtains the synthesized wave front in the (red dotted) listening area. At any place in the listening area the listener perceives a virtual sound source S, as he can move around freely, whilst the virtual sound source remains correctly localized in terms of its direction (see [5] or [8]).

![Figure 2: Principle of WFS: (a) ideal source response and (b) typical output of a finite WFS array.](image)

2.3. Special Properties of WFS

2.3.1. Localization of virtual sources

Through WFS the sound engineer has a powerful tool to design a sound scene. One of the most important (with respect to conventional techniques) novel properties is its outstanding capability of providing a realistic localization of virtual sources. Typical problems and constraints of a stereophonic image vanish in a WFS sound scene.

In contrast to stereophony WFS is able to:

- produce virtual sources that are localized on the same position throughout the entire listening area, refer Figure 3. The red (dashed) and pink (dotted) arrows indicate the directions of the auditory events when the red and pink virtual point sources are reproduced.
- produce plane waves that are localized in the same di-
3. WFS PRACTICAL CONSTRAINTS

Not surprisingly, in practice it is not possible to match all theoretical requirements for a perfect result. The rendered WFS sound field differs from the desired sound field to some degree for a number of reasons (for details see [11]).

3.1. Discreteness of the array (spatial aliasing)

This effect produces spatial and spectral errors of the synthesized sound field due to the discretization of a continuous secondary source distribution. Above the spatial aliasing frequency $f_{\text{alias}}$, the time difference between two successive loudspeaker signals interferes at the listener’s position, depending on the spatial sampling interval, i.e. the loudspeaker / microphone inter-spacing.

3.2. Reflections of the reproduction room (spatial interference)

A WFS array cannot render the desired sound field perfectly if reflections of the reproduction room produce interference in spatial perception. In particular, perception of distance, depth and spatial impression are affected, because fragile distance cues of synthesised sources can be dominated by the stronger distance cues generated by the array speakers. They interfere with the desired reflection pattern of the synthesised source. Special room compensation algorithms being under investigation [12], [13] will perhaps be able to minimize this effect.

3.3. Restriction to the horizontal plane

Theory does not restrict WFS to the horizontal plane. However, the reduction of the array dimension to the horizontal plane is the practical approach, having a number of consequences. First, virtual sources can be synthesized only within the horizontal plane. This includes virtual reflections affecting the completeness of a natural reflection pattern and thus possibly resulting in impairments of perception of distance, depth, spatial impression and envelopment.

Another aspect is related to the measuring techniques used for capturing the room response. In practice there is some mismatch with respect to elevated reflections, because the measured room response includes elevated reflections although they are reproduced only in the horizontal plane. The effects of these types of
inaccurateness on spatial perception parameters are not well-known yet.

Furthermore, horizontal arrays do not generate real spherical waves, but cylindrical waves. In the case of imaging a plane wave for example there results an error with respect to the level roll-off (3dB doubling of distance), in comparison with the ideal plane wave (no roll-off) \( \left( 1 \right) \), \( \left( 4 \right) \).

### 3.4. Limitation of array dimensions (diffraction)

In practical applications the loudspeaker array will have a finite length. Due to a finite array so-called diffraction waves originate from the edges of the loudspeaker array \( \left( 11 \right) \). \( \left( 14 \right) \). These contributions appear as after-echoes (and pre-echoes respectively for focussed sources), and – depending on their level and time-offset at the receiver’s location – may give rise to colouration.

However, methods to reduce these truncation effects are known, e.g. by applying a tapering window to the array signals. This means that a decreasing weight is given to the loudspeakers near the edges of the array. In this way the amount of diffraction effects can be substantially reduced at the cost of a limitation of the listening area \( \left( 14 \right) \).

### 3.5. Effects on perception

Although a number of authors have suggested methods to deal with the practical limits of rendering accurateness or to minimize their effects, there is still a lack of knowledge (some details can be found e.g. in \( \left( 2 \right) \), \( \left( 3 \right) \), \( \left( 11 \right) \), \( \left( 12 \right) \), \( \left( 13 \right) \)). Several effects of the constraints on specific perceptual attributes are not known yet in detail. However, this knowledge is important for further developments of WFS systems in view of future applications.

Current psychoacoustic studies are concentrating on the subjective evaluation of principle characteristics of WFS systems in comparison with stereophonic or binaural systems. They are necessary to evaluate the resulting impacts on attributes of spatial perception not only with respect to the development of WFS systems for different applications but also in view of scientific knowledge. Particular attention should be turned to the perception of direction, distance, spatial depth, spatial perspective, spatial impression, reverberance, and envelopment, as well as sound colour.

### 4. WFS APPLICATIONS

#### 4.1. The European CARROUSO Project

The European CARROUSO Project ("Creating, Assessing and Rendering in Real Time of High Quality Audio-Visual Environments in MPEG-4 Context") has intended to break several limitations of these current commercial systems by merging the new WFS rendering technique with the flexible new coding technology MPEG-4 standard, allowing object-oriented and interactive sound manipulation.

By means of the MPEG-4 format the signal of the source ("Gestalt") and its spatial properties are transmitted separately. For reproduction, the dry source signal is convolved with the measured or modelled set of impulse responses (containing the spatial information), and emitted by a loudspeaker array. In contrast to stereophony WFS is able to

- produce virtual stable sources localized at the same position throughout the entire listening area,
- produce virtual sources in front of the loudspeaker array ("focused sources")
- produce plane waves that are localized in the same direction throughout the entire listening area,
- enhance the sense of depth, spatial impression and envelopment through a realistic reproduction of the original room response

The key objective of the project CARROUSO was to provide a new technology that enables to transfer a sound field, generated at a certain real or virtual space, to another usually remote located space, in a bit efficient way at highest perceived quality. The principle block diagram is illustrated in Figure 5 it shows three functional components:

- **Capturing**
  For recording of the sound field a microphone array technology is applied. Signal processing calculates the position of the sound sources which could be fixed or moving. The microphone array as well as the video cameras is used to gather relevant information on the acoustical conditions in the recording room. Acoustic models can be obtained to parameterize the acoustic data set, thus making it suitable for transmission.

- **Encoding**
  Encoding of audio objects is operated by the MPEG-4 standard and encapsulated into specific data streams. For broadcasting applications the transmission adopts digital video broadcasting (DVB) streams. The coding uses a subset of MPEG-4 components (no predefined profile for the given application in the standard).

- **Rendering**
  In the rendering process the transmitted data are demultiplexed, decoded and processed by a compositor. It enables the reproduction of a recorded or simulated sound field via WFS loudspeaker arrays, ensuring immersive sound perception in a wide listening area. The original acoustics of the reproduction room, which may negatively influence the obtained result, is optimally corrected for.

All components were combined to a demonstrator, as a basis for validation. This was done using perceptual experiments and field tests. The results of the project are a first step towards a new quality of high quality spatial audio imaging. CARROUSO has shown the possibility to capture, transmit and render sound sources and their related acoustic environment with more realism, compared to existing stereophonic methods.

This novel spatial audio technology was developed for applications in conjunction with moving pictures, using the recently introduced MPEG-4 standard. It is considered as a major milestone for immersive audio representation at public places and in private households. Two applications have been targeted within this project. The first one concerns high quality spatial audio with associated video for broadcasting. The second application is related to cooperative and interactive work on immersive audio objects. CARROUSO is expected to contribute to information, communication and media technology.

---

1 EU-Project IST-1999-20993 (Jan. 2001 – June 2003): \( \left( 4 \right) \).
4.2. Synthesis of WFS and stereophony

This paper concentrates on the synthesis of WFS and stereophony. Figure 6 illustrates the basic concept by means of music recording. Step one is always the room response measurement in the music hall, done e.g. with a stepwise rotating microphone. This measured spatial information is stored in the WFS processor.

For recording of orchestra and soloist closely spaced spot microphones are used. The stereophonic orchestra mix should be composed in a way that it contains as little room information (reverb, reflections, etc.) as possible; but it should contain the adequate spatial distribution of elements. This three channel stereophonic mix signals and the soloist signal are being convolved with the appropriate spatial impulse responses. As a result, the rendered WFS sound field represents stable virtual sources located in the concert hall. Listeners within the listening area perceive a three-channel stereophonic image of the orchestra and a point source image of the soloist, whereby the reproduced characteristics of the concert hall give a new sense of realism.

On this basis apparent advantages of established conventional stereophonic recording techniques on the one hand and of WFS technologies on the other can, in principle, can be utilized through a purposeful combination.

4.3. Virtual Panning Spots (VPS)

The key tool is use of so-called Virtual Panning Spots (VPS) [16], virtual point sources to be applied for panning across any stereophonic imaging plane in the virtual WFS imaging area. VPS can be understood as virtual “loudspeakers” which reproduce the stereophonic sound image of a spacious sound source (e.g. a choir) in the recording room (see also [6], [7], [8]). The suitable room impulse responses have to be measured in the original room or to be created artificially in a suitable way. In the example in terms of Figure 7, the orchestra is imaged with the assistance of six VPS, which are reproduced via WFS and are relatively freely configurable with regard to localization, expansion and distance.

The sound design advantage of this concept is self-explanatory: The stereophonic recording of the orchestra according to Figure 6 produces a spacious sound image of the sources as there is an image between the VPS in accordance with the principles of phantoms. Listeners perceive a three-channel stereophonic image of the orchestra and a point source image of the soloist, whereby the reproduced characteristics of the concert hall give a new sense of realism.

On this basis apparent advantages of established conventional stereophonic recording techniques on the one hand and of WFS technologies on the other can, in principle, can be utilized through a purposeful combination.
tionally stable in the listening area. The known disadvantages of phantom source localization, especially the low directional stability can be easily avoided by employing a sufficient number of VPS. The number of stereophonic imaging areas is in principle arbitrary. From an artistic point of view, one should orientate oneself towards the number of spacious instruments or instrument groups (in large ensembles, e.g. string groups, brass player groups, choir). The number and spatial distribution of the VPS depends on the following criteria:

- Size and shape of the homogeneous ensemble
- Circumstances of the production
- Artistic and sound balance-related intention of the sound engineer
- Available transmission capacity

Virtual Panning Spots, VPS, are selected points (“virtual loudspeakers”), which produce a stereophonic representation area. These can in principle be selected at choice in accordance with the recording situation and the desired sound image. As an example, Figure 7 shows two rear imaging planes and two front imaging planes, offering easy imaging of depth. The imaging area can be “spread out” by an arbitrary number of VPS in a random spatial expansion in accordance with the situation and intention.

The artistic arrangement of the ensemble upon the WFS transmission commences with the choice, dedication and positioning of the VPS. Three parameters should be mentioned, which lends the sound engineer to new possibilities of spatial sound design (refer to the example in Figure 8).

1. In the case that loudspeaker arrays are installed lateral to the listening area, there are in principal, no problems as far as directional stability is concerned, as a lateral stereophonic representation sector can be built up from a sufficient number of stable VPS. The same applies to the sector behind the listener.

2. A stereophonic imaging area does not only allow itself to be moved in all directions, stretched out or compressed, but can also be presented in an extensive range with different distances. The representation of depth is thus easily recognizable.

3. With certain constraints (see Section 2.3 and Figure 8), the VPS can be placed in a distance between the listener and the loudspeaker array and also with the stereophonic imaging field. Thus, the virtual imaging area theoretically reaches in dense closeness to the listener and allows for an expressive representation of depth.

The practical example shown in Figure 8 contains diverse new possibilities of the spatial sound design. A total of 13 VPS were employed, whereby those more remote have been arranged in terms of the 3/4 stereo ITU standard and act in the known fashion as stereophonic imaging areas (also see Figure 3). In the front (stage) area there are stereophonic representation areas in three different distances and also the lateral sectors have been prepared for stereophonic imaging. In addition a monophonic virtual source has been provided in the front between the listening area and the loudspeaker array, e.g. for a soloist.

The individual stereophonic presentations should contain the (dry) direct sound of spacious sound sources or ensembles, with as little room response as possible. The conventional tools for the representation of spatial depth (generation of suitable lateral reflections are superfluous within the concept of WFS and actually damaging, due to the fact that here by virtual sources (and VPS), the relevant distance information is available - in the form of room impulse responses.

In Figure 8 the parallel lines represent the plane waves for the reverberation synthesis. The reverberation is not produced with the assistance of VPS. Rather is the effective source to reproduce a plane wave infinitely far away.

4.3.1. Creation of VPS

The locations and number of VPSs should be found before the WFS recording starts. Suitable loudspeakers are positioned in the VPS areas of the recording room. The recording room is stimulated, the response is measured and the impulse response is calculated from this (see Figure 8). Of course, instead of a real measurement of the room impulse response, model based procedures
can be employed to create artificial room impulse responses.

In practice, for certain halls, such determination of the room impulse responses does not have to be done at each recording. Rather, this once determined data is permanently available in a data base. Also impulse responses from other halls or pseudo-realistic artificial (and sometimes better) characteristics can be desirable (comparable with the input of modern reverberators).

4.3.2. Reduction of transmission channels

Through the utilization of the VPS concept, the number of transmission channels as well as the computing capacity can be significantly reduced. The larger an ensemble, and also the number of individual participating instruments, the larger saving in transmission channels is. In the case of a small ensemble or individual instruments, however, the VPS concept does not save transmission channels or only saves a few (see [16]).

4.3.3. Handling aspects

A major gain in the utilization of the VPS concept is the improved handling of the recording techniques and the improvement in the spatial quality of the WFS reproduction. The handling of the VPS concept is based on the application of stereoponic techniques, which all sound engineers are familiar with. There are not any completely new microphone techniques; the mode of operation remains the same as during conventional productions (except for the representation of distance, which is automatically given within the configuration of VPS). Also the mix to the VPS positions requires no new modus operandi - conventional panning and mixing techniques are being used. The number of microphones remains approximately the same as in the case of an appropriate stereo recording.

The recording in accordance with the VPS concept thus has the significant advantage that the sound engineer can directly employ familiar processes of work. The improvement on spatial quality by way of WFS goes hand in hand with the incorporation of stereoponic techniques. The sound engineer has clearly more influence over important quality characteristics of a recording, he has more creative room for manipulating the spatial parameters and can more strongly influence the sound color.

4.4. Virtual loudspeaker reproduction

An important application of the VPS technique is a special preset of the VPS setup on the reproduction side, which enables the reproduction of conventional multichannel recordings in a virtual listening room [16]. For this purpose, two modifications are suggested for the WFS decoder, which can be activated in the event of need for application (see Figure 9).

1. The configuration of the VPS with regard to room impulse responses and spatial arrangement is done in accordance with the preset setup of virtual loudspeakers in a virtual listening room. Arbitrary arrangements of the virtual loudspeakers can be preset and be activated dependent on the stereoponic format to be reproduced.

2. The virtual source signal is not received via the transmission channel, but from the multichannel decoder on the reproduction side (e.g. that of a DVD player).

4.3.3. Handling aspects

The WPS reproduction unit operates completely detached from WFS transmission, and can principally offer three attractive advantages:

1. Diverse stereoponic multichannel formats can be easily reproduced optimal through the selection of a VPS preset, without having to appropriately adjust the loudspeaker arrangement within the living-room.

2. The virtual loudspeakers can also be placed outside the living-room, i.e. also in a confined area situation, the listening area for multi-channel stereophony is sufficiently large.

3. A future high quality WFS reproduction unit will allow for an electronic compensation of diverse defects in the reproduction room [12], especially the reduction of the effect of the early reflections and the balancing of asymmetrical arrangements of the speaker array.

From the technical and practical point of view the application of WFS for multichannel stereo reproduction could be the first step towards acceptance in the market place. In this regard, the development of the so-called MAP technology (see e.g. [20], [21]) is important. The flat panels, e.g. fed with glass fiber cables, can often be better integrated into the living-room and are more attractive than conventional loudspeakers. Thereby not only the application of virtual loudspeakers within the home is envisaged, but also the employment in cinemas, theaters as well as for high quality sound reinforcement.

5. ACKNOWLEDGMENT

I wish to thank my colleague Helmut Wittek for his basic contributions to this paper.
6. REFERENCES


Günther Theile has obtained his Ph.D. from the Techn. University of Berlin in 1980. In his thesis he introduced a general theory on sound source localisation. Based on this, he proposed the sphere stereo microphone and the diffuse field equalization, which has become ITU-R BS. 708 for studio monitor headphones. In 1984 he invented the masking threshold based subband coding technique and started the development of the MUSICAM system, which is now ISO/IEC Standard (MPEG 1 Audio Layer II). He was engaged in investigations on dynamic range control and on multichannel sound, including multichannel source coding (MPEG 2 Audio Layer II). Under his chairmanship ITU-R TG 10-1 has produced Recommendation BS. 775, known as 5.1 surround standard.

His current activities are concentrating on further developments in the field of multichannel sound and research on spatial audio systems based on binaural techniques and wave field synthesis.

In his fields of work Dr. Theile has published about one hundred papers, and he has been granted a number of International Patents. In 1992 he received the Lothar-Cremer-Medal for his basic work on perceptual source coding. He has received the AES Fellowship and AES Board of Governors Awards. He was Chairman of the 102nd AES Convention in Munich and of the AES 19th Conference on Surround Sound. He has chaired ITU-R Working Party 1Oc in the period 1997-1998. Currently he is serving as Chairman of the AES South German Section, as Vice-Chairman of the AES Technical Committee on Multichannel and Binaural Audio Technologies, as Chairman of the VDT specialist group “Research & Development” and “Surround Sound Forum”, as well as Chairman of the Programme Committee of the German Toneimiestertagung.
WAVE FIELD SYNTHESIS - GENERATION AND REPRODUCTION OF NATURAL SOUND ENVIRONMENTS

Thomas Sporer
Fraunhofer Institute for Digital Media Technology IDMT
Ilmenau, Germany
spo@idmt.fhg.de

ABSTRACT

Since the early days of stereo good spatial sound impression had been limited to a small region, the so-called sweet spot. About 15 years ago the concept of wave field synthesis (WFS) solving this problem has been invented at TU Delft, but due to its computational complexity it has not been used outside universities and research institutes. Today the progress of microelectronics makes a variety of applications of WFS possible, like themed environments, cinemas, and exhibition spaces. This paper will highlight the basics of WFS and discuss some of the solutions beyond the basics to make it work in applications.

1. INTRODUCTION

In 1931 Blumlein [1] filled a British patent which described the basics of stereo recording and reproduction, and which is up to now the basic of all stereo recording techniques. In 1934 researchers at AT&T Bell labs described two major configurations of spatial audio reproduction: binaural, that is two channels recorded at the ear position (dummy head) and multi-channel [2]. In experiments they proved three loudspeakers (left, center, right) provide superior quality to a larger audience compared to two loudspeakers. About twenty years later, after careful research on a variety of loudspeaker number and configurations, William Small concluded in [3] that “The number of channels will depend upon the size of the stage and listening rooms, and the precision in localization required.” “...for a use such as rendition of music in the home, where economy is required and accurate placement of sources is not of great importance if the feeling of separation of sources is preserved, two-channel reproduction is of real importance.” Due to limitations in physical delivery media (LP, CD) and broadcast formats two-channel stereo was and is dominant in most applications. After a commercially non-successful extension to four channels (quadrophony) in the seventies today 5-channel stereo, cations. After a commercially non-successful extension to four formats two-channel stereo was and is dominant in most appli-
sources form a wave front which is physically indistinguishable from the shape of the original wave front. This principle, originally described for water waves and optics, was first applied to acoustics at TU Delft. A large number of small and closely spaced loudspeakers form a so-called loudspeaker array (Figure 1). Each loudspeaker in the array is fed with a corresponding driving signal calculated by means of algorithms based on the Kirchhoff-Helmholtz integrals and Rayleigh’s representation theorems. (Equation 1)

$$P(x) = \frac{1}{4\pi} \oint \left[ p \frac{j k \Delta r \cos \phi \exp(-j k \Delta r)}{\Delta r} + j \omega \rho_0 V_n \frac{\exp(-j k \Delta r)}{\Delta r} \right] dS \quad (1)$$

The Kirchhoff-Helmholtz integral implies that an infinite number of monopoles and dipoles encircling the reproduction space is necessary to achieve perfect results. “Perfect results” includes the property that the reproduced sound field outside the listening space (behind speakers) is zero. Taking either monopoles or dipoles instead of both the sound field inside is the same and only the sound field outside is non-zero. Today most implementations of WFS are based on monopoles only.

A second step to simplify WFS is to reduce the sound-field from 3D to 2D, therefore all loudspeakers are located in one plane.

Reducing the number of loudspeakers to a finite number limits the frequency up to which WFS provides perfect reproduction. Above the alias frequency spatial alias terms occur. In practice it proved to be sufficient to locate a loudspeaker every 17 cm, giving an alias frequency of about 1 kHz. From an acoustic point of view this seems to be insufficient, but due to psychoacoustical effects a decrease of distance between loudspeakers has only marginal effects on audio quality.

Without room reflection of the reproduction room the sound field reproduced is perfect in nearly the whole space between the loudspeakers. However, the superposition of the individual loudspeaker signal does not work properly if the distance between listener and loudspeaker is smaller or similar to the distance between loudspeakers. Such close to the loudspeaker additional effects occur due to the near field of cone loudspeakers.

Three types of sound sources can be reproduced (Figure 2):

Point Sources are sound sources which are between or behind the loudspeakers. At each listening position the position of such sound sources is perceived to be the same.

Focused Sources are point sources in front of the loudspeaker array. While having similar properties as “normal” point sources for most of the listening positions, for positions between source and closest loudspeaker the sound field is inverted and there is no precise location.

Plane Waves behave similar to point sources which are in infinite distance: All listeners perceive the same direction of the source. For this type of signals the effect of distance dependent reduction of sound pressure level is negligible.

Figure 1: Principle of WFS: Superposition of secondary sound source recreates sound-field

Figure 2: Reproduction of point source, focused sources and plane waves with WFS

3. PROPERTIES AND REQUIREMENTS OF WFS

Reproduction of low Frequencies Due to the limited number of loudspeakers (sampling in the spatial domain) frequencies
below the alias frequency are amplified by 3 dB per octave. To compensate for this effect the input signals have to be pre-filtered. However the value 3 dB is only valid if the virtual sound source is far apart from the loudspeaker array: if the virtual source is placed exactly on a loudspeaker position there is only the frequency response of a single loudspeaker. The filtering therefore has to be position dependent. In practical applications prototype filters for regions are precomputed and stored. The size of the regions with the same correction filters is smallest near the loudspeaker array, at large distance the approximation of 3 dB per octave is sufficient. That way the computational complexity is reduced, which makes larger numbers of moving virtual sources feasible.

Source Position and Level The simplification of the Kirchhoff-Helmholtz integral as derived in [6] provide correct approximation for the horizontal plane for static (non-moving) sound sources. Due to the fact that these simplifications are done with the assumption of an infinite length of a (linear) loudspeaker array, the variation of the synthesized sound pressure level with position of the virtual source is not correct for smaller systems. This effect is clearly perceptible when audio objects are visible. Similar problems occur if the loudspeaker array is not complete due to limitations of reproduction room (windows, doors, curtains). In practical implementations the level has to be corrected. A simple way to achieve the correction value is to simulate the sound pressure level at a reference point in the listening area under the assumption of an ideal array and with the actual loudspeaker configuration. If the sound pressure level of the actual array would be too small the volume of all active loudspeakers is raised. For a given loudspeaker configuration the correction value only depends on the position of a virtual source. To reduce computational complexity and simplify parallel processing of rendering for different loudspeakers the correction value is precomputed and stored.

Doppler Shift For each position of a virtual sound source WFS creates a natural sound field. If sound sources are moved from one position to another a natural Doppler effect is created automatically. Being a benefit when applying to sound effects in movies this Doppler shift might be unwanted when moving musical instruments. A simple way to overcome this is to use panning between virtual objects as described in [9].

Loudspeaker Characteristics The basic theory of WFS starts with a combination of ideal loudspeakers being either pure monopoles or pure dipoles. Most installations done so far are using cone loudspeaker, which can be approximated as being monopoles up to the alias frequency of WFS. In [10] algorithms are shown to compensate for loudspeakers with directivity. Up to now these algorithms are hardly been used due to complexity.

Characteristics of the Reproduction Room Wave field synthesis is able to reproduce complete sound scenes including the room reflections of a natural or simulated (recording) environment. If the reproduction room has his own reflection pattern the reflection pattern of recording and reproduction room are combined, which often is regarded to sound unnatural. The ideal reproduction room therefore would be an anechoic room. Due to the fact that all walls of the reproduction room are equipped with loudspeakers electro-acoustic cancellation of the first room reflections are possible [11][12]. Experience from installations show that some diffuse reflections of the reproduction room improve the naturalness of simple simulations of room acoustics.

Recorded and Simulated Rooms Using WFS it is possible to process signals coming from the (primary) sound sources separately from the (secondary) signals coming from the recording room. Thus it is possible to manipulate the two signal classes independent of each other. If done properly it is even possible to change the (virtual) recording venue at the reproduction site. The information of the recording room inherent in a WFS file can be either the recording of the room signature of a real room or the simulated room signature of a room which even does not exist.

4. APPLICATIONS

WFS and mathematically related sound rendering methods like higher-order ambisonics [13] are feasible to all sound reproduction systems where ever it is possible to use more than just one or two loudspeakers. One significant advantage of WFS compared to classical multi-channel reproduction is the possibility to switch from reproduction based storage of audio (the format is defined by the number of loudspeaker channels) to a source based storage (each audio object is stored separately and can be rendered for the best possible audio quality given any reproduction setup). This paradigm shift will revolutionize both production and distribution of audio content, and might be even more important than the improvements of audio quality.

4.1. Application areas

Concert halls: A critical parameter for any kind of live performance is the input-output latency of the signal processing involved. Wave field synthesis has an intrinsic delay which is short enough. If the acoustics of the concert hall is sufficiently good, what should be a property of any concert hall, no room equalization filters, which might cause additional delay, are necessary. WFS can adapt the acoustical behavior of multifunctional venues to the requirements of any kind of music reproduction, and for other purposes, too, for instance for sport events or conferences. In contrast to the systems used today, where electro-acoustical amplification often destroys the sound scene, WFS can make it sound much more natural. Due to the fact that WFS preserves spatial angular and distance precision a much improved audio-visual coherence is achieved.

Open air events: An important requirement for open air concerts is to achieve a sufficiently high sound pressure level for the whole audience without creating dangerously high sound pressure levels near the stage. At the same time the audio quality should be as high as possible and there should be a good spatial coherence of sound and visual scene on stage. While line arrays of loudspeakers are satisfying the requirements concerning sound pressure level WFS additionally preserves the whole auditory scene. For very large venues line-arrays can only control the sound pressure level within (sub-)regions of the reproduction space with problems at the cross-sections of neighboring regions. WFS is based on
continuous sound fields. Therefor such problems can not occur with WFS. With WFS it is even possible to create an artificial room around the listening space with acoustical properties like indoors which is especially useful for classical concerts. Focused sound sources enable new creative possibilities for sound effects.

Cinema: For long time cinema sound systems have been screen centric: Most dialogs are just mixed to the center channel, because directors believed that sound which is not localized on screen distracts from the movie. Today almost all cinemas are equipped with sound systems with 5.1 rendering. The surround channels are used to create more immersion to the scene. Due to the fact that there are just two surround channels which are reproduced on the side via several loudspeakers the sound mixer has to fight several restrictions:

- at the back seats in the cinema surround channels sometimes are perceived as coming from the front side and might decrease the intelligibility of dialogs
- all sound sources mixed to surrounds are rather smeared over a large range
- moving sources from front (on screen) via side to the back change their sound color due to different frequency response of loudspeakers and phantom sources
- perception of sound very close to listeners is only possible by using very high sound pressure levels

Newer formats try to overcome some of these limitations by using additional channels, especially in the back.

WFS gives the possibility to render sound sources in their true spatial depth and direction for each seat in the cinema. In February 2003 the first cinema equipped with WFS system started daily service in Ilmenau, Germany (Fig. 5). Being a regular cinema in daily operation this cinema is also used for research, especially for the perceptual evaluation of loudspeaker configurations\[14\] algorithms in larger rooms. This cinema system uses 192 loudspeakers. In Figure 5 on top of the loudspeaker array the old 5.1 system can be seen. The theater is an amphitheater style cinema. To provide best audio experience for all seats the loudspeaker array is not in one plane but in two planes which are slightly folded.

A trailer, produced in a WFS compliant format, exhibits the potential of the new technology: It starts within an aquarium, with air bubbles modeled as focused sources within the cinema hall, shows slowly moving sound objects leaving the screen, moving around the hall and appearing on screen exactly in audiovisual coherence, virtual two channel stereo (loudspeakers visible and moving on screen) and WFS music reproduction where each instrument is placed on each own spatial position. In contrast to trailers for 5.1 formats this trailer does not need to be reproduced at a high sound pressure levels to create the sensation of immersion. The trailer is shown before every film show. All legacy format films benefit from the increased sweet spot offered by the compatible reproduction via the WFS system via virtual loudspeakers placed outside the cinema hall. The reproduction of the surround channels using plane waves improves the spatial sound quality and the intelligibility especially in the back rows. Perceptual experiments to study the performance of wave field synthesis audio reproduction combined with flat video reproductions have been conducted\[15\]. The results of these experiments already have been considered in the design of authoring tools. In July 2004 the first installation of a WFS system was done in Hollywood. This system, using 304 loudspeakers and a 8 channel WFS-subwoofer configuration, is integrated in a sound stage at Todd AO, Studio City. First experiences with remixing films and trailers show that the positioning of virtual sources anywhere at anytime brings more advantages than expected\[16\]:

- point sources are far less distracting than “old” surround channels
- dialogs moving slowly to the side of the screen are perceived as very natural, and do not draw the attention away from the screen
- no sound colorations appear with moving of sound sources
- Sound designers and mixers expect that the time and costs of mixing films using WFS are the same or lower compared to 5.1.

Home theater systems: Today WFS is regarded as too expensive and too space-consuming for the use in every home. However the installation in some higher rated home cinemas is already reasonable. The possibility to reproduce any content using virtual loudspeakers placed (far) outside the home theater enables to overcome the feeling of being in a small room, which is still inherent in home theaters today\[17\]. Problems for WFS in normal living rooms are the placement of the loudspeaker arrays and the acoustics of the room. For the later, a combination of acoustic treatment (e.g. curtains) and the application of room equalization techniques (e.g. compensation of a few early reflections) is probably the best solution. To solve the problem of loudspeaker placement new loudspeaker designs are necessary. Flat panel loudspeaker systems like DML panels\[18\] might play an important role in the home reproduction. Another possibility is to integrate them into furniture or even (far into the future) into wall paper.

Video Conferencing Today video conferencing is using mono or stereo reproduction of audio signals. If rooms with many persons are linked in a conference a lot of discipline is necessary to avoid interference between different persons talking, and to avoid background noise which reduces speech
intelligibility. Multi-channel sound systems have not been used because of the problem of acoustic echo cancellation. It has been proven that perfect echo cancellation is impossible for the general case of two or loudspeakers/microphones. However in [19] a method has been presented which takes into account that the loudspeaker array used for WFS turns out to be of benefit for acoustic echo cancellation, too.

4.2. Some remarks on large listening area sound reinforcement

The effect of distance dependent reduction of loudness is more expressed near a sound source. An even distribution of loudness across the whole listening area can be achieved by positioning sound sources far behind the loudspeaker array [8]. Infinite distance of sound sources relates to plain waves which do not have any distance dependent reduction (besides the damping in air).

Due to the expectations of the visitors for the applications listed above a high sound pressure level is essential. In average all loudspeakers contribute for that level, but in the worst case, where a sound object is placed close to the loudspeaker array only a few loudspeakers have to provide the whole power. For most applications it is possible to overcome this by just avoiding such positions of sound objects. For the compatible reproduction in the cinema the worst case are the front speakers: To avoid incoherence of visual and auditory image it is essential to place virtual loudspeakers for left, center and right channels rather close behind the screen.

The possibilities of placing and moving sound sources anywhere in the sound scenes provide new artistic possibilities like musicians acoustically flying thru the audience before appearing on the stage.

5. RECORDING AND DISTRIBUTION OF WAVE FIELD SYNTHESIS CONTENT

The best sound experience using WFS can be achieved when using specially prepared material. Such material consists of dry recordings of separate sound sources, their position in the room and information about the desired room acoustics (e.g. recording room). Using microphone array technique recording of sound sources requires subsequent signal processing. By means of signal processing, sound source signals can be separated and unwanted signals can be suppressed. In addition, information about the position of possibly moving signal sources is extracted [20]. Besides the microphone array technique conventional 5.1 recording technique (spots, main and room microphones) can be also applied.

Audio information (recorded or synthetic sources and/or room acoustics) and scene description are treated inside the WFS system on the reproduction side. The number of transmitted audio tracks (either point sources or plane waves) is related to the scene and independent from the number of loudspeakers at the reproduction site.

The necessary storage capacity for a 2 hours movie can be estimated as follows: During the mixing process all sound tracks are stored as PCM using 24 bit at 48 kHz resolution. A reasonable film might be composed of 130 sound tracks in the final mix.[1] This results in a total storage requirement of 125.5 GByte, an amount which easily can be stored on state of the art PC hard drives, but which is beyond the current capacity of cheap magneto-optical storage media (like DVD-ROM). For end-user applications a reduction is necessary. As a first attempt perceptual audio coding can be used: MPEG-4 AAC (Advanced Audio Coding) at comparably high bit-rates (2 bit/sample per channel) achieves a reduction of the combined audio data to about 10.5 GByte (data rate of 12 Mbit/s). By using just some more compression or a slightly lower number of independent sound tracks, current DVD-ROM technology is adequate to provide the audio and metadata (source position information) to control WFS rendering.

For the audio scene description the MPEG-4 standard is very suitable. MPEG-4 is actually the only standardized format that provides a high-level structured coding support to efficiently convey advanced 3D descriptions [21] as those required by WFS. Together with wide band transmission channels, like DVB or the upcoming wide-band Internet, the MPEG-4 3D Audio Profile permits a commercially feasible realization of WFS.

After decoding the final auralization processing is left to the WFS loudspeaker arrays.

5.1. Channel-oriented versus object-oriented

Current sound mixing is based on the channel or track paradigm, i.e. the coding format defines the reproduction setup. Any changes would mean doing the complete mix again. In the current mixing process there is a certain way of arranging tracks for a mix, following the requirements of a mixing desk, routing system and the format (like 5.1 or 7.1) in order to accelerate the work-flow. Looking at the example of mixing a helicopter flying around the listeners head, it would be necessary panning and placing each element from the track system individually or at least copying and pasting the related settings on the desk. This is a time-consuming task, but due to the 5.1 or 7.1 mixes this hardly happens because movements are quite rare in those formats.

The mixing process of the Wave Field Synthesis occurs in a sound object-oriented way. For this the sound source positions are needed. Tracks and channels, which are indirectly considered in the process, form an object and this can be moved in a Wave Field Synthesis authoring system. The position data can either be imported from a tracking system (virtual studio), rendering data (special effects) or manually imported using a pencil. The final WFS mix does not contain loudspeaker related material. Audio signals of all sound sources are transmitted from the final mix to the WFS render PCs, which calculate the signals for all loudspeakers.

The object oriented approach inherent in WFS and MPEG-4 enables additional functionalities: It is able to group sound objects and give (limited) access to some of the mixing parameters to the end user. Putting all dialogs in one group and all remaining objects in a second group enables hearing impaired people to improve the intelligibility by increasing the level of the dialogs only. Keeping the room information separately from the musical objects enables the listener to put an orchestra into another concert hall. Putting sound tracks of different dubbed version in one bit stream enables multilingual listeners to hear all characters in a movie in their original language, and to replace only unknown languages with the preferred dubbed version.

---

[1] The number of raw tracks is much higher. The normal work flow of film mixing involves the reduction to so-called “stems”. The number 130 mentioned above is equivalent to the number of all channels in all stems used for the final mix. However the work flow for reduction of tracks for WFS is different.
6. CONCLUSIONS

The next generation of audio formats will be object oriented, and Wave Field Synthesis will be used for the reproduction replacing stereo and multi-channel systems. It will find its way into many applications, like concerts, cinemas, theme parks and, eventually, into the home. After long years of research, computational complexity is no longer an obstacle for widespread adoption of WFS. First professional installations have been already made, and more will come in the next year.

7. ACKNOWLEDGMENTS

The author would like to thank all the many contributors to this work within Fraunhofer, at Delft Technical University and the other CARROUSO partners. The work at Fraunhofer IDMT was partly funded by the Thuringian Minister of Science, Research and Culture TMWFK, the German Minister of Education and Research BMBF and by the European Commission in the context of the CARROUSO project.

8. REFERENCES

Spatial Impulse Response Rendering (SIRR) is a recent technique for reproduction of room acoustics with a multichannel loudspeaker system. SIRR analyzes the direction of arrival and diffuseness of measured room responses within frequency bands. Based on the analysis data, a multichannel response suitable for reproduction with any chosen surround loudspeaker setup is synthesized. When loaded to a convolving reverberator, the synthesized responses create a very natural perception of space corresponding to the measured room. In this paper, the SIRR method is described and listening test results are reviewed. The sound intensity analysis is refined, and improvements for the synthesis of diffuse sound components are discussed.

1. INTRODUCTION

In recent years, multichannel loudspeaker reproduction systems have become increasingly common. A standard 5.1 setup is able to produce a surrounding sound field with fair directional accuracy especially in front of the listener. By adding more channels, the precision can be further enhanced, or the reproduction can be extended to 3-D. However, due to limitations of microphone technology, current recording systems cannot achieve as high directional resolution as that available for the playback. Furthermore, with most recording techniques the loudspeaker setup needs to be known already at the time of recording, and conversion for other setups is very difficult if not impossible. Spatial Impulse Response Rendering (SIRR) has been designed to overcome some of these problems.

In a typical recording scenario several spot microphones are placed close to sound sources to yield fairly “dry” source signals with ideally no audible room effect. An artificial scene is then constructed by positioning these signals in desired directions using, for instance, amplitude panning. Spatial impression is created by adding the signals of additional microphones placed further away from the sources in the recording room, or with the help of reverberators. When convolving reverberators it has recently become possible to use actual measured room responses to simulate a chosen acoustical environment. However, the problem is—as in any surround sound recording application—how to capture the responses so that the perceived spatial impression of the measured room or hall is accurately reproduced.

SIRR is primarily targeted for processing room responses to be used in convolving reverberators. The responses can be measured with commercially available SoundField or Microflown systems or with a suitable custom microphone array. The method yields multichannel impulse responses that can be tailored for an arbitrary surround loudspeaker system in the postprocessing phase. SIRR can also be applied to continuous sound but this part is still under development.

In this paper, the SIRR method and some refinements are described, and earlier listening test results are reviewed. The paper is organized as follows. Secs. 2 and 3 provide background related to conventional multichannel recording techniques and psychoacoustics of spatial hearing. Sec. 4 with description of the SIRR method forms the main part of the paper. Listening test results are reviewed in Sec. 5 and the paper is summarized in Sec. 6.

2. PROBLEMS WITH CONVENTIONAL TECHNIQUES

Spatial audio or multichannel impulse responses have been typically recorded using one microphone per loudspeaker. Several different microphone configurations have been proposed in the literature. It has been shown that coincident microphone techniques are able to produce sharpest virtual sources. In coincident microphone setups, directive microphones are positioned as close to each other as possible. The sound signal from a single sound source is thus captured in the same phase with all microphones. The microphones should have orientations and directivities corresponding to the loudspeaker configuration, so that sound from any specific direction would only be picked up by few microphones. Using more loudspeakers requires thus narrower directional patterns. However, with existing microphone technology, narrow enough broad band patterns cannot be achieved. Consequently, the sound from any direction is always picked up by several microphones, which results in a blurred and colored reproduction due to the crosstalk between loudspeaker channels.

Ambisonics solves the directivity problem by employing a spherical harmonic decomposition of the sound field. In theory it can accurately reproduce a directional sound field in a small sweet spot by the sympathetic operation of all loudspeakers in an arbitrary surround setup. In practice, however, microphone technology limits the order and thus the directional resolution of Ambisonics. The authors are only aware of first order commercial implementations, although higher order microphone systems have been recently proposed. Furthermore, the presence of the head of the listener further disrupts the ideal operation, and consequently the technique reduces to using a set of virtual coincident microphones that can be adjusted during playback. The problems are also similar to those discussed in the previous paragraph.

In contrast to coincident techniques, spaced microphones are positioned at a considerable distance between each other. The sound signal from a single sound source is thus captured in different phases by different microphones. In a reverberant environment the resulting microphone signals will also be to a certain degree decorrelated. The noncoincident techniques are often said to cre-
egrate a better feeling of “airiness” and “ambience”, and the reproduction is less sensitive to the location of the listener. However, the directional accuracy is even lower than what can be achieved with a coincident microphone setup.

3. PSYCHOACOUSTICAL BACKGROUND

The goal of sound recording and reproduction is normally to relay a perception. However, in order to recreate the perceived spatial impression of an existing room or a hall, it is not necessary to perfectly reconstruct the original soundfield. Human sound localization is based on four frequency-dependent cues: (1) the interaural time difference (ITD) and (2) the interaural level difference (ILD), which resolve the left/right direction of a sound source, (3) monaural spectral cues, and (4) the effect of head rotation on the previous cues [9]. Additionally, human listeners are sensitive to the coherence of the left and right ear input signals (e.g. [10] and references therein), which has been proposed to be an important cue for localization in reverberant environments and multi-source scenarios [11]. In a room, reflections from different surfaces can affect these cues. For any nonstationary source signal, the summation of sound in different phase at the ears of the listener lowers the coherence and produces time-varying fluctuations in ITD, ILD, and the spectral cues. In SIRR it is assumed that these time and frequency dependent cues are what needs to be reproduced.

The limited resolution of human hearing has been studied extensively for monaural conditions (e.g. [12]). The frequency resolution of binaural hearing appears to be equal to that of monaural hearing [13][14], although slightly larger analysis bandwidths have been found for some test signals [15]. This suggests that the monaurally derived ERB frequency resolution [16] is also appropriate for the analysis and synthesis of binaural cues. Determining the time resolution of binaural hearing is a little more complicated.

A human listener is only capable of tracking in detail the spatial movements of sound sources corresponding to fluctuations of the ITD and ILD cues up to 2.4 and 3.1 Hz, respectively [17]. However, Grantham and Wightman [18] observed that listeners were able to detect ITD fluctuations up to 500 Hz, not based on movement but on perceptual widening of the sound sources.

As already mentioned, the interaural coherence and the time-variance of the other localization cues are related. Depending on the length of an analysis window, high frequency fluctuations transform into lowered coherence. If the fluctuations cannot be exactly recreated, it is important to reproduce the lowered coherence not only due to human sensitivity to it but also due to its stabilizing effect on the spatial sound image.

4. SPATIAL IMPULSE RESPONSE RENDERING

As discussed earlier, the current technology has shortcomings in recording and reproduction of spatial sound. The problems could be alleviated by designing microphones with higher directivity, but this is not an easy task. However, the previous psychoacoustical considerations suggest a different solution. In SIRR the direction of arrival and diffuseness of sound are analyzed at narrow frequency bands within short time windows. Based on an omnidirectional microphone signal and the analysis data, a perceptually similar sound field is then synthesized using a chosen reproduction system. The direction of arrival determines the localization cues appearing at each analysis band, and the diffuseness estimate is related to the interaural coherence. Hence, excluding the limitations of reproduction systems we assume that, if the frequency bands are narrow enough and the time windows are short enough, the reproduced spatial impression is very close to that of the recording room.

The analysis and synthesis can be implemented in several different ways. For the time-frequency processing we have adopted a short-time Fourier transform (STFT) based scheme common in audio coding applications. Similar processing could also be realized using an analysis-synthesis implementation of an auditory filter bank. However, Baumgarte and Faller [19] found the computationally more efficient FFT implementation to perform equally well with an auditory filter bank in their experiments with the Binaural Cue Coding (BCC) algorithm sharing some features with SIRR.

The analysis and synthesis parts of SIRR are illustrated in Figs. 1 and 2, respectively. The directional analysis discussed in this paper is based on the concept of sound intensity as analyzed from SoundField microphone recordings. The synthesis for a multi-channel loudspeaker system consists of spatialization of a recorded omnidirectional signal using amplitude panning and decorrelation techniques. The analysis will be described in more detail in Sec. 4.1 and the synthesis in Sec. 4.2.

4.1. Directional analysis based on sound intensity

The analysis data needed for SIRR consists of direction of arrival and diffuseness estimates as a function of time and frequency. Energetic analysis of a sound field can be used to obtain both of these estimates. In this Section, the sound intensity analysis is first introduced, followed by derivation of the required quantities from the B-format SoundField microphone signals.

The instantaneous sound intensity is defined as the product of the sound pressure \( p(t) \) and the particle velocity vector \( \mathbf{u}(t) \)

\[
I(t) = p(t) \mathbf{u}(t)
\]  

(1)

The intensity describes the transfer of energy in the sound field, and the direction of arrival can be estimated simply as the opposite of the direction \( \mathbf{u}(t) \). Depending on the sound field, the direction of the instantaneous intensity may vary as a function of time, which means that part of the sound energy oscillates locally and only part of it constitutes a net flow. The net flow can be characterized with the active intensity (or radiating intensity [20]) defined as the time average of the instantaneous intensity.

The proportion of sound energy contributing to the net transport of energy can be used to characterize the diffuseness of the sound field. In earlier papers, we derived the total energy from the sound pressure signal as the active intensity of an ideal plane wave having the same sound pressure. However, this relation is valid only in monochromatic sound fields. The instantaneous energy density of a general sound field can instead be calculated as

\[
w(t) = \frac{1}{2} \rho \left[ z^{-2} p^2(t) + \mathbf{u}^2(t) \right],
\]

(2)

where \( \rho \) is the mean density and \( z = \rho c \) is the impedance of the medium, where \( c \) denotes the speed of sound [21]. An average diffuseness estimate can now be written in the form

\[
\psi = \frac{\| (I(t)/c) \|}{\| (w(t)) \|} = \frac{2 z \| (p(t) \mathbf{u}(t)) \|}{\| (p(t)) + z^2 \| (\mathbf{u}(t)) \|},
\]

(3)

where \( \| \cdot \| \) denotes the norm of a vector and \( \langle \cdot \rangle \) denotes time averaging. This estimate equals the speed of energy transfer divided
by the speed of sound, and it can be shown to be bound to values between $[0,1]$, where 0 indicates an ideally diffuse sound field (no net transport of energy), whereas 1 signifies the absence of any locally oscillating sound energy \[^2\]\. Note that an instantaneous value $\psi(t)$ could also be defined but it would be of little use for synthesis purposes. A sound field with exact instantaneous properties according to Eqs. (1) and (2) is very difficult to synthesize, but a sound field with approximately similar time averages can, however, be created with the help of $\psi$.

The time-frequency analysis can be realized either by feeding the sound pressure and particle velocity signals through a filter bank and applying the equations above, or with a short-time Fourier transform (STFT) based scheme. In STFT implementation, the single-sided frequency distribution of the active intensity in an analysis window can be written as

$$ L_\omega = 2 \text{Re} \{ P^*(\omega) U(\omega) \} , $$

where $P(\omega)$ and $U(\omega)$ are the Fourier transforms of the time windowed sound pressure and particle velocity, respectively, and $^*$ denotes complex conjugation \[^2\]\. Furthermore, the single-sided frequency distribution of the diffuseness estimate is given by

$$ \psi(\omega) = \frac{\| L_\omega / |\omega| \|}{|P(\omega)|^2 + |U(\omega)|^2} , $$

where $|\cdot|$ denotes the absolute value of a complex number.

The intensity and diffuseness estimates can be derived from the B-format output signals $W$, $X$, $Y$, and $Z$ of an ideal SoundField microphone system as follows. The ideal omnidirectional signal $W$ is proportional to the sound pressure $p$ at the measurement position. Since we are not interested in the absolute values of the sound intensity and energy density, we define

$$ p = W , $$

disregarding the sensitivity of the microphone. Furthermore, the orthogonal figure-of-eight signals $X$, $Y$, and $Z$ are proportional to the components of the particle velocity in the corresponding directions of a cartesian coordinate system. $X$, $Y$, and $Z$ are normalized such that a plane wave propagating in the direction of the corresponding coordinate axis yields twice the signal power of $W$.

For a plane wave, the pressure $p$ and the particle velocity $u$ have the relation

$$ |u| = \frac{p}{z} . $$

The particle velocity is thus

$$ u = \frac{1}{\sqrt{2z}} X', $$

where

$$ X' = (X e_x + Y e_y + Z e_z) , $$

and $e_x$, $e_y$, and $e_z$ represent unit vectors in the directions of the corresponding cartesian coordinate axes. By substituting (6) and (8) in (4) and (5) we now have the frequency distributions of the active intensity and the diffuseness estimate

$$ I_\omega = \frac{\sqrt{2}}{z} \text{Re} \{ W^*(\omega) X'(\omega) \} , $$

and

$$ \psi(\omega) = \frac{\sqrt{2} \| \text{Re} \{ W^*(\omega) X'(\omega) \} \|}{|W(\omega)|^2 + |X'(\omega)|^2 / |x|^2} . $$

### 4.2. Synthesis with a multichannel loudspeaker system

Based on the analysis data, a sound field with similar energetic properties needs to be created. In SIRR, the synthesis is based on processing the omnidirectional signal $W$, which is analyzed with STFT using the same time-frequency resolution as in the directional analysis. Different spatialization methods are applied to the diffuse and non-diffuse parts of a room response.

An obvious method for synthesizing the non-diffuse part of a response with a multichannel loudspeaker system is to reproduce it as sharply as possible from the correct direction, for instance, with Vector Base Amplitude Panning (VBAP) \[^2\]\. Based on the proportion of non-diffuse sound energy, the frequency components within a time window are weighted by $\sqrt{\psi(\omega)}$ and panned to the direction opposite to the frequency-dependent intensity vector $I(\omega)$. This step corresponds to deriving different linear phase filtered versions of the omnidirectional signal for each loudspeaker, with the filters changing from one time window to another.

For the diffuse part of sound a different method is required. The total diffuse energy $|W(\omega)|^2 \| 1 - \psi(\omega) \|$ is distributed uniformly around the listener by reproducing frequency weighted de-
correlated versions of the current time window from all loudspeakers. Several methods can be used to implement the decorrelation. In earlier work, the phases were randomized by computing continuous uncorrelated noise for each loudspeaker, and by setting the magnitude spectrum of each channel in each time window equal to the magnitude spectrum of the omnidirectional microphone signal in the time window. This method can create highly decorrelated signals. However, the energy is spread over the whole analysis time window, which may produce audible pre-echo with long analysis windows. Furthermore, the frequency domain equalization of the magnitudes increases the time spreading and, if care is not taken, signal wrapping may occur even with a large amounts of zero padding. The latter problem can be alleviated by trading off the excessive time spreading to deviations in the magnitude response, which can be realized by windowing in the time domain or by smoothing the frequency response of the equalization filter. An alternative technique would be to design specific decorrelation filters, which would allow more precise control of the time spreading and the amplitude deviations [24]. We will return to this topic in future work.

### 4.3. Comparison of SIRR with existing techniques

Processing measured room responses with SIRR can be characterized as follows. In a large concert hall, the direct sound and early reflections are relatively sparse in time and they can usually be individually analyzed and synthesized. As non-diffuse sound, they are synthesized as point-like virtual sources using amplitude panning. The reproduction resembles coincident microphone techniques where SIRR can be thought to adaptively narrow the microphone beams in order to get the best possible directional accuracy. On the other hand, the late reverberant part of a room response is reproduced largely as decorrelated sound emanating from all loudspeakers. This is close to spaced microphone techniques and the pleasant “airiness” or “ambience” of the room should be preserved. In a smaller room the reflections are more dense, which means that fewer reflections can be individually processed and some of the directional resolution is thus lost. However, as will be seen in Sec. 5 the results are still preferred to Ambisonics.

The motivation of SIRR starts from the psychoacoustical principles discussed in Sec. 5. Interestingly, Farina and Ugolotti [25] have independently proposed an almost identical method based on theoretical considerations of sound energy analysis and the principles of Ambisonics. The difference is that Farina and Ugolotti did not divide the SoundField microphone signals into frequency bands, which has proven to be an important part of SIRR.

### 5. LISTENING TESTS

The perceptual quality of reproduction of room responses with SIRR has so far been evaluated in two formal listening tests [3][4]. This Section reviews the results reported in [3].

#### 5.1. Stimuli

Since a perfect spatial sound recording and reproduction method does not exist, it is not possible to directly compare the perception in a real reference space to the perception of the reproduced sound in a controlled listening room. For this reason, a different approach was chosen. The evaluation was done by first creating as naturally-sounding virtual reality as possible. Recording of the virtual impulse responses with a SoundField microphone system was subsequently simulated, and the recordings were reproduced with the investigated techniques. In other words, the purpose of the test was to evaluate how close the reproduction can get to the (virtual) reference.

The virtual reference rooms were created with the DIVA software [26], which models the direct sound and early reflections with the image-source method, and late reverberation statistically. Two different room geometries were applied: a large room with a reverberation time of 1.5 s, and a class room with a reverberation time of 0.6 s. The direct sound and early reflections were applied to the nearest single loudspeakers, since using any spatialization method would have produced abnormal responses in recording the virtual environment. The same 16-channel 3-D loudspeaker system was used for reproduction of both the reference and the test samples.

Three different reproduction methods were tested: SIRR, diffusion, and Ambisonics. The loudness of each system in the reference listening position was equalized by monitoring the samples by ear. The SIRR method was implemented with 2.5 ms Hann windows with 2.5 ms zero padding. The reproduction of the diffuse time-frequency components utilized random noise equalized to have approximately the same magnitude spectrum as the omnidirectional signal in each time window, as described in Sec. 4 (for details see [3]). Note that the diffuseness estimate was not based on the expression of energy density for general sound fields and could have thus slightly lowered the quality of the reproduction.

The diffusion method was equivalent to SIRR, where the diffuseness indicator was set to a constant value of 0. Thus, the whole
The results for Ambisonics are almost identical in the two virtual rooms. In the reference listening position, the MOS values are 2.6 for the drum sample, and 3.4 for the speech sample. However, in the worst case position, the MOS values are close to 2.0 with both sound samples, which means that the listeners have perceived the samples annoyingly different from the reference. According to their comments, the sound was localized to the nearest loudspeaker, which completely changed the perception of directions and the envelopment created by the virtual room.

In the large room, the MOS of the diffusion method has a nearly constant value of approximately 3.0 for both stimuli and listening positions. In the class room, the MOS drops with the drum sample to 2.5. The listeners reported that with the diffusion method the sound was not colored and that the room size remained the same as in the reference. However, in the reference listening position, the localization of the sound sources was lost, and in the worst case position the sources were localized mainly to the nearest loudspeaker. Nevertheless, the envelopment by the virtual room did somewhat remain.

Figure 3: MOS and 95% confidence intervals for the investigated spatial room impulse response reproduction methods for different stimuli and listening positions.
6. SUMMARY

The Spatial Impulse Response Rendering (SIRR) method for reproduction of room acoustics was described. SIRR is motivated by psychoacoustics, and it utilizes energy analysis of sound fields to obtain the necessary data to synthesize room responses suitable for reproduction with arbitrary surround loudspeaker systems. More specifically, the direction of arrival and diffuseness of the sound field are analyzed within frequency bands. The discussed synthesis method spatializes an omnidirectional response using amplitude panning and a decorrelation technique. The reviewed listening test data indicate that the perceptual quality of SIRR is superior compared to Ambisonics and the tested diffusion method, providing at best almost transparent reproduction of the spatial impression of a measured room or hall.

7. ACKNOWLEDGMENTS

The authors would like to thank Angelo Farina for pointing out the correct form for energy density of general sound fields and Tapio Lokki for his contribution in the evaluation of SIRR. The work of Juha Merimaa has been supported by the research training network for Hearing Organisation And Recognition of Speech in Europe (HOARSE, HPRN-CP-2002-00276). Ville Pulkki has received funding from the Academy of Finland (101339).

8. REFERENCES


BINAURAL SOURCE LOCALIZATION

Harald Viste
Audiovisual Communications Lab
Swiss Federal Institute of Technology Lausanne
Switzerland
harald.viste@epfl.ch

Gianpaolo Evangelista
Dept. of Physical Sciences
Federico II University of Naples
Italy
gianpaolo.evangelista@na.infn.it

ABSTRACT

In binaural signals, interaural time differences (ITDs) and interaural level differences (ILDs) are two of the most important cues for the estimation of source azimuths, i.e., the localization of sources in the horizontal plane. For narrow band signals, according to the duplex theory, ITD is dominant at low frequencies and ILD is dominant at higher frequencies.

Based on the STFT spectra of binaural signals, a method is proposed for the combined evaluation of ITD and ILD for each individual spectral coefficient. ITD and ILD are related to the azimuth through lookup models. Azimuth estimates based on ITD are more accurate but ambiguous at higher frequencies due to phase wrapping. The less accurate but unambiguous azimuth estimates based on ILDs are used in order to select the closest candidate azimuth estimates based on ITDs, effectively improving the azimuth estimation. The method corresponds well with the duplex theory and also handles the transition from low to high frequencies gracefully.

The relations between the ITD and ILD and the azimuth are computed from a measured set of head related transfer functions (HRTFs), yielding azimuth lookup models. Based on a study of these models for different subjects, parametric azimuth lookup models are proposed. The parameters of these models can be optimized for an individual subject whose HRTFs have been measured. In addition, subject independent lookup models are proposed, parametrized only by the distance between the ears, effectively enabling source localization for subjects whose HRTFs have not been measured.

1. INTRODUCTION

Binaural source localization is the problem of estimating the location of a sound source based on the signals observed at the entrances of the two ears. For localization of sources in the horizontal plane, i.e., the estimation of azimuth angles, the differences between the two ear signals are most important. These are described by the relation between the HRTFs for the two ears. In this paper we consider the problem of estimating source azimuth angles.

1.1. Background

Over the past decades, several computational models for the estimation of source azimuths in binaural signals have been proposed. Many of these models are based on the coincidence model proposed by Jeffress in 1948. This is a model of the neural system where nerve impulses from each of the two ears propagate along delay lines in opposite directions. At the position along the delay lines where the impulses coincide a nerve cell is excited, effectively transforming time information into spatial information. This model corresponds to evaluating ITDs by means of a running short-time cross-correlation function between the ear input signals. Based on Jeffress model, several extensions have been proposed that take into account ILDs, such as the models by Lindemann [3] and Gaik [4]. An overview of these and other models of binaural perception is given in [5]. Most of these models work by decomposing the ear input signals into perceptual bands and estimate the interaural cues in these bands. When several sources at different locations have significant energy within a given perceptual band, the resulting azimuth estimates for that band will not, in general, correspond to any of the actual azimuths of the sources. In some cases, therefore, it can be advantageous not to be limited by the frequency resolution of the human auditory system, but rather to estimate the azimuths in individual narrow frequency bands.

1.2. Contribution

When HRTFs have been measured at several different azimuth angles for each of the two ears, the differences between these two sets of HRTFs describe the ILDs and ITDs as functions of azimuth (and frequency). This means that, in an observed signal, an ILD estimate can be compared with the HRTF data sets in order to obtain an estimate of the source azimuth. This is referred to as HRTF data lookup, yielding azimuth estimates based on ILD only. Similarly, the ITDs can be used for HRTF data lookup of azimuths based on ITD only.

In this paper we propose a method for the estimation of source azimuths through the joint evaluation of ILDs and ITDs. The method is based on the short-time Fourier transform (STFT) spectra of the input signals, and the ILD and ITD is estimated for each spectral coefficient. On one hand, the azimuth estimates based on ILDs have a relatively large standard deviation. On the other hand, the azimuth estimates based on ITD have smaller standard deviation, but are ambiguous. By jointly evaluating these quantities the ILDs are used in order to resolve the ITD ambiguities, effectively improving the azimuth estimates. Since the method is based on the STFT it is computationally fast and has a simple reconstruction scheme that is highly useful, e.g., in source separation applications.

We also propose a parametric model for the relation between azimuth angle and interaural cues (ILD and ITD). In this model the parameters are optimized with respect to an individual head for which the HRTFs have been measured. Similarly to the azimuth estimation by HRTF data lookup, it is possible to lookup the azimuths by use of this model. This is referred to as individual
model lookup. This method has the advantage that the azimuth lookup is faster. However, the azimuth estimates are not as accurate as those obtained by HRTF data lookup. In addition, it still requires the HRTFs to be measured for the individual head in order to determine the parameters.

Based on the study of the parameters of the individual model for each of the 45 subjects in the CIPIC database of HRTFs [6] we propose a generic model for the relation between azimuth angle and ILDs and ITDs that only depends on one parameter, namely the distance between the ears. This model can be used with any head and does not require the measurement of HRTFs. Estimation of azimuths by average model lookup gives results comparable to those obtained by individual model lookup.

In Section 2 the estimation of ILDs and ITDs for individual spectral coefficients in the STFT spectra is discussed. In Section 2 the parameters of the individual model are studied and the average model is proposed. The application to source localization is studied in Section 2. In Section 2 we draw the conclusions.

2. CUE ESTIMATION

From the two observed ear entrance signals, the STFT spectra are computed. These are denoted by $X_{\text{left}}(k, q)$ and $X_{\text{right}}(k, q)$, where $k$ and $q$ are the time and frequency indexes, respectively. In this time-frequency representation, the spatial cues can be easily estimated for each spectral coefficient (left/right pair) individually. The ILDs in dB are given by

$$\Delta L(k, q) = 20 \log_{10} \left| \frac{X_{\text{right}}(k, q)}{X_{\text{left}}(k, q)} \right|.$$  

This is simply the ratio, measured in dB, of the STFT magnitudes of the right and left ear signals. Similarly, the ITDs are estimated as

$$\Delta T_p(k, q) = -\frac{\Delta P_p(k, q)}{q} \frac{L}{2\pi},$$

where $L$ is the window length in the STFT. The interaural phase differences $P_p(k, q)$ are given by

$$\Delta P_p(k, q) = \arg \frac{X_{\text{right}}(k, q)}{X_{\text{left}}(k, q)} + 2\pi p.$$  

Each spectral coefficient represents a periodic and narrow band signal. The phase of a periodic signal can only be estimated up to an integer multiple of $2\pi$. This is reflected by the integer parameter $p$ in the estimates of ITD and interaural phase difference. The practical significance of this is that, for a given frequency, several different source locations yield the same phase difference between the two ear input signals. This is equivalent to the spatial aliasing seen in beamforming techniques. The parameter $p$ indexes these positions, with $p = 0$ corresponding to the source position closest to zero azimuth, $\theta = 0$. A negative value of $p$ corresponds to a position on the left side (negative $\theta$). Positive $p$ corresponds to positions on the right side. In this case, possible values of $p$ depend on the physical layout of the sources and sensors. The frequency whose period equals twice the largest possible delay between the two ears corresponds to the highest frequency for which the phase can be estimated without ambiguity. Below this frequency only $p = 0 \, \text{is physically realizable.}$ For an average head size the phase ambiguities occur for frequencies above approximately 1500 Hz.

3. ESTIMATION OF AZIMUTH ANGLES

In order to relate the ILDs to the ITDs, a common reference frame is needed such as the azimuth angle. Using measured HRTFs, the azimuth can be estimated from ILDs and ITDs by HRTF data lookup.

3.1. HRTF data lookup

Based on the HRTFs measured at different azimuth angles, the ILD and ITD can be described as function of azimuth and frequency.

Since the HRTFs are assumed to be time-invariant, there is no dependency on the time index $k$. Instead, the HRTFs depend on the azimuth angle $\theta$. By changing the role of the time index $k$ with that of the azimuth angle $\theta$ and by using the left and right HRTFs as functions of azimuth and frequency, $H_{\text{left}}(\theta, q)$ and $H_{\text{right}}(\theta, q)$, as the signal spectra in Equations (1) and (2), we obtain the HRTF data lookup models for level difference, $\Delta L(\theta, q)$, and time difference, $\Delta T(\theta, q)$, as functions of azimuth angle $\theta$ and frequency index $q$. In the computation of the ILD lookup model special care must be taken to “unwrap” the phase, i.e. to determine the correct choice of $p$ for all frequencies and azimuths.

Figure 1: Interaural level and time differences as functions of azimuth angle and frequency. (a): HRTF data lookup. (b): HRTF data lookup smoothed across azimuth.

The ILD and ITD as functions of azimuth and frequency for one particular head are shown in Figure 1. The panels in the left column show the ILD and ITD as computed from the measured HRTFs. In the right column, the same functions are shown after smoothing in the azimuth direction. No processing across frequency was performed.

When a sound source is located on one side of the head, the source signal will arrive first at the ear on the same side. Also, the signal level will normally be stronger at the ear facing the source than at the ear on the opposite side. Intuitively, the farther to the side a source is located, the larger the level and time differences between the ears should be. The smoothed HRTF data confirm this intuitive facts. Both functions are relatively monotonic in azimuth.

Based on the ILD estimate $\Delta L(\theta, q)$ for a given left/right pair of spectral coefficients, $X_{\text{left}}(k, q)$ and $X_{\text{right}}(k, q)$, the azimuth angle can be looked up in the smoothed HRTF data $\Delta L(\theta, q)$. This yields azimuth estimates based on ILD only, denoted $\theta_L(k, q)$. 

---

— DAFx’04 Proceedings —

146

146
Similarly, each ITD estimate \( \theta \) can be looked up in \( \Delta T(\theta, q) \). Due to the phase ambiguity, this results in multiple possible azimuth estimates, denoted \( \theta_T(k, q) \), indexed by \( p \).

The ITD is usually a relatively smooth function of azimuth, as seen in Figure 1. This means that the standard deviation of the azimuth estimates based on ITD is relatively small. However, there may be several possible azimuth candidates due to the phase ambiguity. The ILD is a more complex function of azimuth and must be smoothed across azimuth in order to become useful for azimuth lookup. Consequently, the azimuth estimates based on ILDs have a much larger standard deviation than those based on ITD. In addition, the ILDs as function of azimuth are not, in general, monotonic for all frequencies. When this is the case, the azimuth lookup is non-unique, yielding multiple possible azimuth estimates. For the examples in this paper the azimuth estimates closest to 0 degrees were chosen whenever this was the case.

### 3.2. Joint evaluation of ILD and ITD

Since both, the ILD and the ITD, are related to the azimuth, they can also be related to each other. We propose a method for the joint evaluation of these quantities in order to improve the azimuth estimates. Briefly explained, the noisy \( \theta_T(k, q) \) provides a rough estimate of the azimuth for each left/right spectral coefficient pair. Then, this estimate is refined by choosing the \( \theta_T(k, q) \) that lies closest. The combined azimuth estimate is then given by

\[
\theta(k, q) = \theta_T(k, q) \left| \arg\min_p \left( |\theta_L(k, q) - \theta_T(k, q)| \right) \right.
\]

Effectively, the ILD estimate is used in order to choose the “correct” parameter \( p \) in the ITD estimate. The azimuth estimate based on ITD is chosen since this estimate is “more precise”, i.e. the standard deviation of these estimates are smaller. The general processing is illustrated in Figure 2.

![Figure 2: Combined evaluation of ILDs and ITDs for the estimation of azimuths. (a): The interaural cues are estimated by use of the STFT. (b): The relation between these cues and azimuth angle is described by the ILD and ITD in the smoothed HRTF data. (c): Azimuth estimates are found by lookup of the interaural cues in the HRTF data models. (d): Final azimuth estimates are obtained by combined evaluation of the azimuths estimated from the ILDs and ITDs.](image)

Figure 3 shows some experimental results to further illustrate the processing. In order to assess the performance of the proposed method for different frequencies, a white noise signal was chosen as source position. A window length of about 10 ms was chosen in the computation of the STFT spectra, and the interaural cues were estimated by means of (1) and (2). The five different situations are shown in the different columns in the figure, with azimuth angles of 0, 15, 30, 45 and 65 degrees (on the right side of the head), respectively. The panels in the first row show two-dimensional histograms as function of azimuth and frequency, based on azimuths estimated from ILD only. These histograms imply, as mentioned in Section 3.1, that the azimuth estimates based on ILD have a larger standard deviation than those based on ITD. The precision decreases with increasing azimuth. In addition, at low frequencies, the ITDs are small and are virtually useless for the estimation of the azimuth. The panels in the second row of Figure 3 show similar histograms based on azimuths estimated from ITDs. Above approximately 1–2 kHz, these azimuth estimates are ambiguous. For a given frequency this is seen as several equally strong peaks at different azimuths. These correspond to different choices of \( p \) in (2). As the frequency increases, more values of \( p \) are possible. Additionally, the distance between the peaks decreases with increasing frequency, making the ITD estimates less useful at higher frequencies. In the third row the results that were obtained by applying the proposed method are shown. Visually explained, the estimates based on ILDs (first row) are used in order to select the right \( p \) in the estimates based on ITDs (second row). Note that this is done independently for each spectral coefficient in the STFT spectra: no processing is performed across frequency.

The last row of panels shows the one-dimensional marginal

![Figure 3: Histograms of azimuth estimates for 5 different heads and azimuth angles, at 0, 15, 30, 45 and 65 degrees. (a)–(e), respectively. First row: based on ILD only. Second row: based on ITD only. Third row: based on our method for combined evaluation of ILD and ITD. Bottom row: marginal histograms for our method.](image)
histograms as function of azimuth (i.e. summing over all frequencies), based on the azimuths computed by the proposed method. The strongest peaks are found at the true azimuth angles of 0, 15, 30, 45 and 65 degrees, respectively.

\[ \Delta T(\theta, q) = \alpha_q \frac{r \sin(\theta + \theta)}{c}, \]

where \( r \) is the “head radius”, and \( c \) is the wave propagation speed. In reality, the ITD is slightly larger than this due to the fact that the head is not perfectly spherical. In addition, the time difference is also somewhat larger at low frequencies, as has been observed in [8]. In order to take this into account, we propose to use a frequency dependent scaling factor \( \alpha_q \).

4.2. Interaural level differences

As implied by the data shown in Figure 1 the ILD is a much more complex function of azimuth and frequency. Based on a study of the HRTFs in the CIPIC database, we propose the following model:

\[ \Delta L(\theta, q) = \beta_q \frac{\sin \theta}{c}, \]

with frequency dependent scaling factor \( \beta_q \). The effect of the source distance can be largely neglected, as indicated in [9, 10]. Based on the observation that the ILD is periodic in \( \theta \), a Fourier series expansion of the ILD was proposed in [11]. Our model is similar to this (single-term expansion), with the exception that we only consider the range \(-90^\circ \leq \theta \leq 90^\circ\).

4.3. Frequency dependent scaling factors

The ILD and ITD models in (7) and (6) can be optimized for a given head by finding the scaling factors \( \alpha_q \) and \( \beta_q \) that give the closest match to the smoothed HRTF data. For the head whose HRTF data is shown in Figure 1, the best matching parametric models were found. The individual parametric models and the model errors are shown in Figure 5.

For the estimation of azimuth based on ILDs and ITDs, (Figure 2(b)), the HRTF data can be replaced by the parametric model. The experiment shown in Figure 4 was repeated, but with the use of parametric models. For each of the 45 heads, the best matching parametric model was found and used for azimuth lookup. In particular, the estimation errors are significant at higher frequencies. The estimation error is mainly due to the model error in the ILD model, as shown in Figure 5. For most heads, however,
the model is useful up to about 6 kHz. In any case, the use of the proposed method for the joint evaluation of ILD and ITD yields sharper peaks in the azimuth histograms.

Figure 6: Energy weighted histograms of azimuth estimates for 45 subjects in 4 different frequency bands; using individual model lookup.

5. AVERAGE ILD AND ITD MODELS

In order to obtain ILD and ITD models for a given head HRTFs must be measured for a range of different azimuth angles. This can be a tedious task. Also the parametric models that were proposed rely on the HRTF data for estimating the frequency dependent scaling factors. In this respect, the parametric models do not present an advantage over the HRTF data. In fact, the parametric models are less accurate.

The scaling factors for the ILD model, \( \beta_q \), and ITD model, \( \alpha_q \), were computed as functions of frequency for the 45 subjects in the CIPIC database. These are shown in gray in Figure 7. Qualitatively, all these quantities follow the same trend, at least up to about 7 kHz. Above this frequency, the ILD scaling factors vary highly among the different heads.

The black lines in Figure 7 show the scaling factors averaged over all heads. If the average scaling factors are used in the parametric ILD and ITD models, these models only depend on one parameter, namely the head radius \( r \). This can easily be measured, and consequently these average parameter models can be employed for any head without the need to measuring the specific HRTFs.

In order to compare the accuracy of the individual parametric models and the average models, the model errors were computed for each head (relative to the smoothed HRTF data). The absolute value of these errors were then averaged over all azimuths and heads. Figure 8 shows these errors as functions of frequency. The ITD models are shown in the top row, and the ILD models in the bottom row. Columns (a) and (b) correspond to the individual parametric models and the average parameter models, respectively. The average models (b) are almost as good as the individual models (a), and only a slight increase in error can be observed. However, above about 6-8 kHz the accuracy of all the models is significantly worse than that attained by the smoothed HRTF data.

The performance of the average models for estimation of azimuths is shown in Figure 9. The accuracy of azimuth estimates based on the average models are comparable to those based on the individual parametric models, as shown in Figure 6.

6. SOURCE LOCALIZATION

So far, only a single source was considered and white noise was used as the source signal in order to study the accuracy of azimuth estimates at different frequencies. Since the proposed method for joint evaluation of ILDs and ITDs is based on interaural cues in narrow bands independently it is also applicable in situations where several sources are located at different locations.

In the following experiment, three different harmonic tones were chosen as source signals, i.e. three consecutive half-notes played by an alto trombone. The binaural signal was obtained by filtering these sources with the HRTFs at different azimuth angles. Figure 10 shows histograms of the estimated azimuth angles in this mixture. The histograms have been energy weighted, i.e. each azimuth estimate is weighted by the energy of the corresponding spectral coefficient. The three columns show the results obtained by use of HRTF data, individual models, and average models, respectively. The panels in the first row show the results...
for azimuth estimates based on ILD only. Since the histograms are weighted, the different harmonics can be observed as strong peaks (dark horizontal lines). However, these are quite wide and do not provide very accurate estimates of the azimuth. In the second row, the estimates based on ITD are shown. The peaks corresponding to the harmonics are much narrower, but ambiguous above about 1500 Hz. In the third column the results obtained by the proposed method for joint evaluation of ILD and ITD are shown. For all three models, the different harmonics are well aligned about the true source azimuths of –30, 15 and 45 degrees, respectively. The marginal histograms are shown in the bottom row.

Figure 9: Energy weighted histograms of azimuth estimates for 45 subjects in 4 different frequency bands: using average model lookup.

Figure 10: Histograms of azimuth estimates in different frequency bands, comparing the three different models for azimuth lookup. (a): HRTF data. (b): Individual parametric model. (c): Average parameter model.

7. CONCLUSIONS

We have presented a method for localization of source in the horizontal plane based on binaural signals. The method is narrow-band and short-time since it works with each spectral coefficient independently, effectively enabling the tracking of sources when the sources or sensors move. We have also proposed a parametric head model and an average parameter model that can be used for a head whose HRTFs have not been measured.

8. REFERENCES

PARAMETRIC CODING OF SPATIAL AUDIO

Christof Faller

Agere Systems, Mobile Terminals Division
Allentown, PA, USA
cfaller@agere.com

ABSTRACT

Recently, there has been a renewed interest in techniques for coding of stereo and multi-channel audio signals. Stereo and multi-channel audio signals evoke an auditory spatial image in a listener. Thus, in addition to pure redundancy reduction, a receiver model which considers properties of spatial hearing may be used for reducing the bitrate. This has been done in previous techniques by considering the importance of interaural difference cues at high frequencies and by considering the binaural masking level difference when computing the masked threshold for multiple audio channels. Recently, a number of more systematic and parameterized such techniques were introduced.

In this paper an overview over a technique, denoted binaural cue coding (BCC), is given. BCC represents stereo or multi-channel audio signals as a single or more downmixed audio channels plus side information. The side information contains the inter-channel cues inherent in the original audio signal that are relevant for the perception of the properties of the auditory spatial image. The relation between the inter-channel cues and attributes of the auditory spatial image is discussed. Other applications of BCC are discussed, such as joint-coding of independent audio signals providing flexibility at the decoder to mix arbitrary stereo, multi-channel, and binaural signals.

1. INTRODUCTION

Generally speaking, audio coding is a process for changing the representation of an audio signal to make it more suitable for transmission or storage. Although high capacity channels, networks, and storage systems have become more easily accessible, audio coding has retained its importance. Motivations for reducing the bitrate necessary for representing audio signals are the need to minimize transmission costs or to provide cost-efficient storage, the demand to transmit over channels with limited capacity such as mobile radio channels, and to support variable-rate coding in packet-oriented networks.

The audio coding techniques discussed in this paper, binaural cue coding (BCC) [1, 2, 3, 4, 5] and related techniques (limited to stereo) [6, 7, 8, 9, 10], enable higher compression ratios for stereo and multi-channel audio signals. This is achieved by transmitting only the waveform of one single audio channel. This single audio channel contains all signal components (disregarding spatial aspects) which are present in the original stereo or multi-channel audio signal. In addition, parameters describing “perceptually relevant differences” (in terms of spatial hearing) between the original audio channels are estimated. These parameters contain about two orders of magnitude less information than the waveforms themselves and thus the bitrate is significantly reduced by transmitting them as opposed to transmitting all the audio channels. In the decoder, the transmitted audio channel is processed such that the “perceptually relevant differences” of the synthesized channels approximate those of the original audio channels.

Figure 1 shows a BCC scheme. As indicated in the figure, the input audio channels \( x_c(n) \) (1 ≤ \( c \) ≤ \( C \)) are downmixed to one single audio channel \( s(n) \), denoted sum signal. As “perceptually relevant differences” between the audio channels, inter-channel time difference (ICTD), inter-channel level difference (ICLD), and inter-channel coherence (ICC), are estimated as a function of frequency and time and transmitted as side information to the decoder. The decoder generates its output channels \( \hat{x}_c(n) \) (1 ≤ \( c \) ≤ \( C \)) such that ICTD, ICLD, and ICC between the channels approximate those of the original audio signal.

BCC can also be used with two \([11]\) or more \([12]\) transmitted audio channels for stereo backwards compatible coding of multi-channel surround and scalable bitrate and audio quality, respectively. Another variation of BCC, denoted BCC for flexible rendering \([1, 3, 5]\), provides flexibility at the decoder to freely mix binaural, stereo, or multi-channel audio signals.

The paper is organized as follows. Section 2 discusses spatial hearing and spatial audio playback. Based on this, BCC is motivated in Section 3. BCC for flexible rendering is described in Section 4. The results of a subjective evaluation using BCC for coding of multi-channel surround signals are described in Section 5. Finally, conclusions are presented in Section 6.

2. SPATIAL HEARING AND SPATIAL AUDIO PLAYBACK

Similarly to the way humans perceive a visual image, humans are also able to perceive an auditory spatial image. The different objects which are part of the auditory spatial image are denoted...
auditory events. When stereo or multi-channel audio signals are played back over headphones or loudspeakers they evoke an auditory spatial image in the listener. In the following, spatial hearing is discussed with emphasis on phenomena relevant for spatial audio playback.

2.1. Spatial hearing with one sound source

The simplest listening scenario is when there is one sound source in free-field. In this case, the ear input signals can be viewed as being filtered versions of the source signal. The filters modeling the path of sound from a source to the left and right ear entrances are commonly referred to as head related transfer functions (HRTFs) [13]. For each source direction different HRTFs need to be used for modeling the ear entrance signals.

A more intuitive but only approximately valid view for the relation between the source angle $\phi$ and the ear entrance signals considers the difference in length of the paths from the source to the two ear entrances as a function of the source angle $\phi$ [13]. As a result of the different path lengths, there is a difference in arrival time between both ear entrances. Due to this path length difference, there is a difference in arrival times of sound at the left and right ears, denoted interaural time difference (ITD). Additionally, the shadowing of the head results in an intensity difference of the left and right ear entrance signals, denoted interaural level difference (ILD). For example, a source to the left of a listener results in a higher intensity of the signal at the left ear than at the right ear.

The following measures are used for ITD and ILD relative to the ear entrance signals $\tilde{x}_1(n)$ and $\tilde{x}_2(n)$:

- **ITD [samples]:**
  \[
  \tau_{12}(n) = \arg \max_d \{ \Phi_{12}(d, n) \},
  \]
  with a short-time estimate of the normalized cross-correlation function
  \[
  \Phi_{12}(d, n) = \frac{E[\tilde{x}_1(n - d_1)\tilde{x}_2(n - d_2)]}{\sqrt{E[\tilde{x}_1^2(n - d_1)] E[\tilde{x}_2^2(n - d_2)]}},
  \]
  where
  \[
  d_1 = \max\{-d, 0\},
  d_2 = \max\{d, 0\},
  \]
  and $E[\cdot]$ denotes expectation.

- **ILD [dB]:**
  \[
  \Delta L_{12}(n) = 10 \log_{10} \left( \frac{E[\tilde{x}_2^2(n - d_2)]}{E[\tilde{x}_1^2(n - d_1)]} \right).
  \]

Diffraction, reflection, and resonance effects caused by the head, torso, and the external ears of the listener result in that ITD and ILD not only depend on the source angle $\phi$ but also on the source signal. Nevertheless, if ITD and ILD are considered as a function of frequency, it is a reasonable approximation to say that the source angle solely determines ITD and ILD as implied by data shown in [14]. When only considering frontal directions ($-90^\circ \leq \phi \leq 90^\circ$) the source angle $\phi$ approximately causally determines ITD and ILD. However, for each frontal direction there is a corresponding direction in the back of the listener resulting in a similar ITD-ILD pair. Thus, the auditory system needs to rely on other cues for resolving this front/back ambiguity. Examples of such cues are head movement cues, visual cues, and spectral cues (different frequencies are emphasized or attenuated when a source is in the front or back) [13]. The following discussion does not cover these other cues, since these are not considered explicitly in BCC. For audio playback systems with loudspeakers these other cues are automatically inherent in the ear entrance signals due to the physical location of the loudspeakers.

2.2. Ear entrance signal properties and lateralization

Figure 2(a) illustrates perceived auditory events for different ITD and ILD [13] for two coherent left and right headphone signals. When left and right headphone signals are coherent, have the same level (ILD = 0), and no delay difference (ITD = 0), an auditory event appears in the center between the left and right ears of a listener. More specifically, the auditory event appears in the center of the frontal section of the upper head of a listener, as illustrated by Region 1 in Figure 2(a). By increasing the level on one side, e.g. right, the auditory event moves to that side as illustrated by Region 2 in Figure 2(a). In the extreme case, when only the signal on the left is active, the auditory event appears at the left side as illustrated by Region 3 in Figure 2(a). ITD can be used similarly to control the position of the auditory event.

Another ear entrance signal property that is considered in this discussion is a measure for the degree of “similarity” between the left and right ear entrance signals, denoted interaural coherence (IC). IC here is defined as the maximum absolute value of the normalized cross-correlation function,

\[
C_{12}(n) = \max_d |\Phi_{12}(d, n)|,
\]

where delays $d$ corresponding to a range of $\pm 1$ ms are considered. IC as defined has a range between zero and one. IC = 1 means that two signals are coherent (signals are equal with possibly a different scaling and delay) and IC = 0 means that the signals are independent.

When two identical signals (IC = 1) are emitted by the two transducers of the headphones, a relatively compact auditory event is perceived. For noise the width of the auditory event increases
as the IC between the headphone signals decreases until two distinct auditory events are perceived at the sides, as illustrated in Figure 2(b) [15].

2.3. Two sound sources: Summing localization

For two sources at a distance (e.g. loudspeaker pair), ITD, ILD, and IC are determined by the HRTFs of both sources and by the specific source signals. Nevertheless, it is interesting to assess the effect of cues similar to ITD, ILD, and IC, but relative to the source signals and not ear entrance signals. To distinguish between these same properties considered either between the two ear entrance signals or two source signals, respectively, the latter are denoted ICTD, ICLD, and ICC. For headphone playback, ITD, ILD, and IC are (ideally) the same as ICTD, ICLD, and ICC. In the following a few phenomena related to ICTD, ICLD, and ICC are reviewed for two sources located in the front of a listener.

Figure 3(a) illustrates the location of the perceived auditory events for different ICLD for two coherent source signals [13]. When left and right source signals are coherent (ICC = 1), have the same level (ICLD = 0), and no delay difference (ICTD = 0), an auditory event appears in the center between the two sources as illustrated by Region 1 in Figure 3(a). By increasing the level on one side, e.g. right, the auditory event moves to that side as illustrated by Region 2 in Figure 3(a). In the extreme case, when only the signal on the left is active, the auditory event appears at the left source position as is illustrated by Region 3 in Figure 3(b). ICTD can be used similarly to control the position of the auditory event. This principle of controlling the location of an auditory event between a source pair is also applicable when the source pair is not in the front of the listener. However, some restrictions apply for sources to the sides of a listener [16, 17].

Figure 3: (a): ICTD and ICLD between a pair of coherent source signals determine the location of the auditory event which appears between the two sources. (b): The width of the auditory event increases (1-3) as the IC between left and right source signals decreases.

When coherent wideband noise signals (ICC = 1) are simultaneously emitted by a pair of sources, a relatively compact auditory event is perceived. When the ICC is reduced between these signals, the width of the auditory event increases [13], as illustrated in Figure 3(b).

The insight that when signals with specific properties are emitted by two sources the direction of the auditory event can be controlled is of high relevance for applications. It is this property, which makes stereo audio playback possible. With two appropriately placed loudspeakers, the illusion of auditory events at any direction between the two loudspeakers can be generated.

Another relevance of the described phenomena is that for loudspeaker playback and headphone playback similar cues can be used for controlling the location of an auditory event. This is the basis, which makes it possible to generate signal pairs which evoke related illusions in terms of relative auditory event location for both loudspeaker and headphone playback. If this were not the case, there would be a need for different signals depending on whether a listener uses loudspeakers or headphones.

2.4. Other spatial attributes

So far the discussion mostly focused on the attribute of perceived direction or lateralization of auditory events. One exception was the discussion of the role IC and ICC play for noise signals in determining the extent of the auditory event. In the following, other attributes related to auditory events and the auditory spatial image are briefly discussed. These attributes mostly depend on the properties of reflections relative to the direct sound.

Spatial impression is defined as the impression a listener spontaneously gets about type, size, and other properties of an actual or simulated space [13]. Spatial impression is largely determined by the relation between direct sounds and reflections, and number, strength, and directions of reflections. In the following, attributes related to spatial impression are briefly reviewed. More complete reviews are given in [13, 18].

Coloration:
The first early reflections up to about 20 ms later than the direct sound can cause timbral colorization due to a “comb filter” effect which attenuates and amplifies frequency components in a frequency-periodic pattern.

Distance of auditory event:
In free-field, the following two ear entrance signal attributes change as a function of source distance: Power of signal reaching the ears and high frequency content (air absorption). For a source for which a listener knows its likely level of emitted sound, such as speech, the overall sound level at the ear entrances provides an absolute distance cue [19, 20]. However, in situations when a listener does not expect a source to have a certain emitting level, overall sound level at the ear entrances can not be used for judging absolute distance [21].

On the other hand, in a reverberant environment there is more information available to the auditory system. The reverberation time and the timing of the first reflections contain information about the size of a space and the distance to the surfaces, thus giving an indication about the expected range of source distances. For relatively distant sources the ratio of the power of direct to reflected sound is a reliable distance cue, see e.g. [19, 20, 22].

Width of auditory events and envelopment:
As implied by the results presented in Sections 2.2 and 2.3 IC and ICC are related to the width of auditory events. IC can be related to the width of auditory events and listener envelopment [1, 22] by computing it for the early and late part of binaural room impulse responses (BRIRs) (e.g. up to 80 ms and later part). These two measures are often denoted early and late interaural cross-correlation coefficient (IACC) [25, 26]. A thorough review of IACC and related measures is given in [18].

Since IC and ICC are in many cases directly related, i.e. lower ICC between a loudspeaker pair results in lower IC between the ear entrance signals [27], also ICC can be related to the width of auditory events and listener envelopment.
3. SYNTHESIZING STEREO AND MULTI-CHANNEL AUDIO SIGNALS GIVEN A SINGLE AUDIO CHANNEL

Given the sum signal, BCC synthesizes a stereo or multi-channel audio signal such that ICTD, ICLD, and ICC approximate the corresponding cues of the original audio signal. In the following, the role of ICTD, ICLD, and ICC in relation to auditory spatial image attributes is discussed.

The discussion in Section 2 implies that for one auditory event ICTD and ICLD are related to perceived direction. When considering binaural room impulse responses (BRIRs) of one source, there is a relationship between the width of the auditory event and listener envelopment and IC estimated for the early and late parts of the BRIRs. However, the relationship between IC (or ICC) and these properties for general signals (and not just the BRIRs) is not straightforward.

Stereo and multi-channel audio signals usually contain a complex mix of concurrently active source signals superimposed by reflected signal components resulting from recording in enclosed spaces or added by the recording engineer for artificially creating a spatial impression. Different source signals and their reflections occupy different regions in the time-frequency plane. This is reflected by ICTD, ICLD, and ICC which vary as a function of time and frequency. In this case, the relation between instantaneous ICTD, ICLD, and ICC and auditory event directions and spatial impression is not obvious. The strategy of BCC is to blindly synthesize these cues such that they approximate the corresponding cues of the original audio signal.

BCC usually uses filterbanks with subbands of bandwidths equal to two times the equivalent rectangular bandwidth (ERB) [28]. Informal listening revealed that the audio quality of BCC did not improve notably when choosing a higher frequency resolution. A lower frequency resolution is favorable since it results in less ICTD, ICLD, and ICC values that need to be transmitted to the decoder and thus in a lower bitrate.

Regarding time-resolution, ICTD, ICLD, and ICC are considered at regular time intervals. Best performance is obtained when ICTD, ICLD, and ICC are considered about every 4 – 16 ms. Other schemes have also used time varying rates for cue synthesis [6, 7, 9]. Note that by considering the cues at regular time intervals, the precedence effect [13, 29] is not directly considered. Assuming a classical lead-lag pair of sound stimuli, when the lead and lag fall into a time interval where only one set of cues is synthesized, localization dominance of the lead is not considered. Despite of this, BCC achieves good audio quality on average and up to nearly transparent quality for certain audio signals.

The often achieved perceptually small difference between reference signal and synthesized signal implies that cues related to a wide range of auditory spatial image attributes are implicitly considered by synthesizing ICTD, ICLD, and ICC at regular time intervals. In the following, some arguments are given on how ICTD, ICLD, and ICC may relate to a range of auditory spatial image attributes.

Early reflections up to about 20 ms result in coloration of sources’ signals. This coloration effect is different for each audio channel determined by the timing of the early reflections contained in the channel. BCC does not attempt to retrieve the corresponding early reflected sound for each audio channel (which is a source separation problem). However, frequency dependent ICLD synthesis imposes on each output channel the spectral envelope of the original audio signal and thus is able to mimic coloration effects caused by early reflections.

Most perceptual phenomena related to spatial impression seem to be related directly to the nature of reflections that occur following the direct sound. This includes the nature of early reflections up to 80 ms and late reflections beyond 80 ms. Thus it is crucial that the effect of these reflections is mimicked by the synthesized signal. ICTD and ICLD synthesis ideally result in that each channel of the synthesized output signal has the same temporal and spectral envelope as the original signal. This includes the decay of reverberation (the sum of all reflections is preserved in the transmitted sum signal and ICLD synthesis imposes the desired decay for each audio channel individually). ICC synthesis de-correlates signal components that were originally de-correlated by lateral reflections. Also, there is no need of considering reverberation time explicitly. Blindly synthesizing ICC at each time instant to approximate ICC of the original signal has the desired effect of mimicking different reverberation times, since ICLD synthesis imposes the desired rate of decay.

The most important cues for auditory event distance are overall sound level and direct sound to total reflected sound ratio [30]. Since BCC generates level information and reverberation such that it approaches that of the original signal, also auditory event distance cues are represented by considering ICTD, ICLD, and ICC cues.

4. BCC FOR FLEXIBLE RENDERING

Flexible rendering means that the decoder can determine the auditory spatial image of its output signal. A number of discrete source signals (e.g. separately recorded instruments) are encoded and transmitted jointly. The decoder generates stereo or multi-channel audio signals with an artificial auditory spatial image determined by the user at the decoder. Note that this includes not only determining the auditory spatial image at the decoder, but also the number of playback channels and the rendering method (rendering with ICTD and ICLD, rendering with HRTFs or BRIRs).

For providing flexible rendering capability at the decoder with conventional techniques, the source signal of each source to be rendered has to be transmitted to the decoder. Thus the bitrate scales with the number of sources.

BCC for flexible rendering offers a similar capability at a bitrate nearly as low as a mono audio coding bitrate. It transmits only a single channel, the sum of all source signals, to the decoder plus side information. The decoder can still freely render binaural, stereo, and multi-channel audio signals [5] as if the sound sources were coded separately.

Regular BCC relies on a perceptually motivated synthesis technique for generating stereo or multi-channel audio signals at the decoder given the sum signal. BCC for flexible rendering relies on the same synthesis technique. The difference lies in how the sum signal is computed and the nature of side information that is transmitted.

Encoder processing

The top of Figure 4 schematically shows a time-frequency representation of the sum signal. In this example, there are three source signals mixed into the sum signal. As indicated, these sources dominate in different regions of the time-frequency plane. In the area between the regions where one source dominates, there is either vanishing signal power or a mix of power of various sources. BCC for flexible rendering transmits the structure of such regions
Figure 4: Different source signals dominate in different regions of the time-frequency plane of the sum signal (top). For each subband at each time $k$ the source index of the strongest source (bottom) is transmitted to the decoder.

Decoder processing
At the decoder, the sum signal plus the source indices of the dominating source in each subband at each time is given. Whereas regular BCC transmits the spatial cues (ICTD, ICLD, and ICC), BCC for flexible rendering obtains the spatial cues from a local table which stores one set of spatial cues for each source. For each subband the spatial cues are chosen according to the transmitted source index. Then the multi-channel output signal is generated by applying ICTD/ICLD/ICC synthesis. The ICTD and ICLD stored in the table for each source determine the direction, whereas the ICC determines the width of the auditory event. Time adaptive flexible rendering is implemented by (smoothly) modifying the spatial cues in the table in real-time.

BCC for flexible rendering can also support other rendering methods for generating its output audio signal. For example, HRTFs or BRIRs can be used to generate signals for binaural audio playback. An example how to implement this is given in [5].

5. SUBJECTIVE EVALUATION

A test was conducted to assess the quality of multi-channel BCC synthesized items relative to the non-coded reference items.

Subjects and playback setup
Nine adults with an age range of 22-29 participated as subjects in the listening test. Seven subjects are experienced listeners and two are non-experienced. During the test, the subjects were sitting on a chair that was placed in the sweetspot of a standard 5.1 listening setup [31] in a sound insulated room. High quality D/A converters and active loudspeakers were used.

Stimuli
Different kinds of reference 5-channel audio material was selected: Classical recordings mimicking a concert hall experience and movie soundtrack style items with auditory events occurring in all directions. We chose audio material that we consider critical for multi-channel BCC coding (e.g. applause). The reference items were compared to BCC synthesized items. The sum signal was not coded to avoid affecting the test results due to coding artifacts.

Test method
The test method used was the hidden reference method, used according to [32]. The reference item is played, followed by the reference item and the degraded item in random order. A 5-grade impairment scale was used for comparing the degraded item to the reference. After the three items were initially played, the listener could selectively listen to the items again while switching between the items at any time. This method is suitable for subjective assessment of small impairments. We decided to use this method, after informal listening revealed that for the considered items the degree of impairment is fairly small.

Results
Figure 5 shows the results for the individual items averaged for all subjects and the overall average. BCC achieves an overall grading between “perceptible but not annoying” and “imperceptible”. The items with the best quality in Figure 5 are a classical recording, movie soundtracks, and scenes with auditory events all around the subject. The most critical item is the applause signal (a). Item e also contains critical applause and a talker at the side. Item f is a classical recording with very tonal components, where BCC synthesis introduces some distortions. Item g is a movie soundtrack signal.

6. CONCLUSIONS
Binaural cue coding (BCC) and related techniques were reviewed, motivated, and described. Spatial hearing phenomena explored by spatial audio playback systems and BCC were discussed. The use of level difference, time difference, and coherence cues for synthesizing audio signals with desired attributes of the spatial image that is evoked during playback was motivated. Also a variation of BCC was discussed, denoted BCC for flexible rendering, which provides flexibility at the decoder for determining the auditory spatial image of its output signal. Results of a subjective test were
presented, assessing the quality of BCC for multi-channel audio coding. The results indicate that BCC achieves good audio quality and thus enables low bitrate coding of multi-channel audio signals.

7. REFERENCES


FROM JOINT STEREO TO SPATIAL AUDIO CODING - RECENT PROGRESS AND STANDARDIZATION

Jürgen Herre
Fraunhofer IIS
Erlangen, Germany
hrr@iis.fraunhofer.de

ABSTRACT
Within the evolution of perceptual audio coding, there is a long history of exploiting techniques for joint coding of several audio channels of an audio program which are presented simultaneously. The paper describes how such techniques have progressed over time into the recent concept of spatial audio coding, as it is under standardization currently within the ISO/MPEG group. As a significant improvement over conventional techniques, this approach allows the representation of high quality multi-channel audio at bitrates of only 64kbit/s and below.

1. INTRODUCTION
During the recent decades, low bitrate audio coding has made significant progress and found its way into many multimedia applications. A number of relevant international standards, most prominently the ones originating from the well-known ISO/MPEG standardization group, have been developed ranging from the world’s first generic audio coding standard (MPEG-1 Audio [1] [2]) and its extensions (MPEG-2 Audio [3]) to recent audio coding technology (MPEG-2 Advanced Audio Coding, AAC, [4] [5] and MPEG-4 Audio [6]) and its latest extensions for perceptual coding [7] [8].

Even the very first of these standards acknowledged the importance of efficient representation on two-channel stereo audio material by including several provisions which take advantage of joint coding of the audio channels, i.e. so-called joint stereo coding techniques. Since then, significant progress has been achieved in the area of joint stereo coding of two or more audio channels. This paper reviews some of the well-known approaches for joint stereo coding and discusses more recent techniques which overcome many of the limitations inherent in previous schemes. Special focus will be given to the recent idea of spatial audio coding, as it is currently under standardization within the ISO/MPEG audio group. Some indication for the expected performance of such spatial audio coding schemes is provided together with a range of attractive applications of this type of technology.

2. JOINT STEREO CODING
The goals of joint coding of stereo and multi-channel audio material can generally be expressed as follows:

- Firstly, it should enable efficient coding of several audio channels by exploiting inter-channel redundancy / irrelevancy. Practically speaking, this means that a joint encoding of N audio channels should result in significantly less than N times the bitrate required for encoding a single audio channel.
- Secondly, there are cases for which good quality separate encoding of the individual audio channels does not at all lead to an unimpaired reproduction when all audio channels are presented simultaneously, i.e. for regular presentation of the audio material. This originates from perceptual phenomena relating to human spatial hearing and thus becomes relevant when encoding non-monophonic audio material. Proper joint coding of the involved audio channels addresses such perceptual phenomena and enables unimpaired coding of such audio material.

Starting around 1990/91, the first generations of joint stereo coding algorithms were designed with the intention of not significantly increasing the computational (and structural) complexity of traditional audio coders. Thus, these algorithms operate on the spectral values available within the audio coder rather than employing an additional dedicated filterbank for their purpose.

Figure 1: Generic model of joint stereo encoding.
spectral coefficients of the channel signals, plus an appropriate perceptually based joint stereo coding control.

Historically, two approaches to joint stereo coding have been used extensively, namely M/S stereo coding and intensity stereo coding. Even though other concepts have been proposed over time (e.g. [9]), M/S stereo and intensity stereo have been predominant for around ten years of audio coding history. They are briefly outlined in the following.

2.1. M/S Stereo Coding

M/S stereo coding was introduced to low bitrate audio coding in 1991 [10]. A matrixing operation similar to the approach used in FM stereo transmission is used in the coder with the appropriate dematrixing in the decoder. Rather than transmitting the left and right signal, the normalized sum and difference signals are used which are referred to as the middle (M) and the side (S) channel. The matrixing (i.e. sum/difference) operation is carried out on the spectral coefficients of the channel signals and can be thus performed in a frequency selective fashion. M/S stereo coding can be seen as a special case of a main axis transform of the input signal, rotating the input signals by a fixed angle of 45 degrees (see [11]). The main features of M/S stereo processing can be described as follows:

- **Redundancy vs. irrelevance removal**: M/S stereo coding provides considerable coding gain for near-monophonic signals that often turn out to be critical for dual mono perceptual coders due to stereo unmasking effects (binaural masking level differences [12] [13] [14]). Accordingly, M/S stereo coding is activated/deactivated dynamically depending on the input signal. At the same time, such adaptive M/S stereo coding exploits irrelevance by ensuring proper spatial masking of the generated coding noise.

- **Perfect reconstruction**: The sum/difference matrixing used in M/S joint stereo coding is invertible. In the absence of quantization and coding of the matrix output, the joint stereo processing is completely transparent and can thus be applied also at high coder bitrates / audio quality levels without introducing artifacts.

- **Signal dependent saving**: The coding gain of M/S stereo coding heavily depends on the actual signal. It varies from a maximum of nearly 50% in the case where the left and right channel signals are equal (or exactly out of phase) to situations where M/S must not be used because it would be more expensive than separate coding.

- **Full range application**: Because M/S matrixing basically preserves the full spatial information, it may be applied to the full audio spectral range without the danger of introducing severe artifacts.

Within the family of ISO/MPEG Audio coders, M/S stereo coding has been used extensively within the well-known MPEG-1/2 Layer 3 (“mp3”) (full band on/off switching) and within the MPEG-2/4 Advanced Audio Coder [5] in an enhanced fashion (individual switching for each scalefactor band). For use with multi-channel audio, M/S stereo coding is applied to channel pairs that are symmetric to the listener (front/back) axis.

2.2. Intensity Stereo Coding

A second important joint stereo coding strategy for exploiting inter-channel irrelevance is the well-known generic concept of “intensity stereo coding” [11] [15]. This idea has been widely utilized in the past for stereo and multi-channel coding under various names (“dynamic crosstalk”, “channel coupling”).

Intensity stereo exploits the fact that the perception of high frequency sound components mainly relies on the analysis of their energy-time envelopes [12]. Thus, it is possible for certain types of signals to transmit a single set of spectral values that is shared among several audio channels with virtually no loss in sound quality. The original energy-time envelopes of the coded channels are preserved approximately by means of scaling the transmitted signal to a desired target level which needs to be carried out individually for each frequency (scalefactor) band. In the sense of Figure 1, the stage for joint stereo processing would then consist of computing a single signal for transmission (e.g. by summing left and right hand channel) and associated scaling/angle data for each frequency band. The main features of intensity stereo coding can be described as follows:

- **Emphasis on irrelevance reduction**: Even though specific signals with a large correlation of left versus right time domain signal (such as pan-pot stereo mixed signals) can be represented well by using intensity stereo coding, the main emphasis of this technique is on the exploitation of irrelevancy at high frequencies.

- **Not perfectly reconstructing**: While intensity stereo coding of pan-pot type stereo signals may lead to perfect reconstruction, this is not the case for general audio signals including uncorrelated signal components. Frequently, the potential loss of spatial information is considered to be less annoying than other coding artifacts. Therefore, intensity stereo coding is mainly used at low bitrates in order to prevent annoying coding artifacts.

- **Significant data rate saving**: For the frequency range where intensity stereo coding is applied, only one channel of the sub-band data has to be transmitted. If we assume that intensity stereo coding is applied for half of the spectrum, we can expect a saving of about 20% of the net bit-rate. In practise, the maximum saving is about 40%.

- **Useful only for the high frequency range**: As explained above, intensity stereo encoding is used only for part of the spectrum. Extending intensity stereo processing towards low frequencies can cause severe artefacts, especially for signals with a wide stereo image composed of decorrelated components, such as applause [13]. Application of this technique thus has to be done in a carefully controlled way [15].

Within the family of ISO/MPEG Audio coders, intensity stereo coding has been used both for all MPEG-1/2 coders as well as within the MPEG-2/4 Advanced Audio Coder [5]. For multi-channel audio coding, intensity stereo can be generalized by combining the spectral coefficients of several audio channels into a single set of spectral coefficients plus scaling information for each channel.

3. PARAMETRIC STEREO

As a next step in the evolution of joint stereo perceptual audio coding, parametric stereo coding techniques have been proposed recently [16] [17] which further develop the basic idea of intensity stereo coding to overcome many of its original limitations:

- Rather than the coder’s own filterbank, a dedicated (complex-valued, not critically-sampled) filterbank is used to re-
synthesize two channel stereo output from a transmitted mono channel. This avoids artefacts due to imperfect time domain alias cancellation, e.g. by time-varying scaling of spectral channels.

• Besides level differences, also time differences between output channels can be re-created, thus also capturing time-delay stereophony, as it results from use of non-coincident microphones.

• In order to represent stereo content with a wide stereo image consisting of uncorrelated sound components, inter-channel coherence has been found to be an important perceptual cue [16] [21]. Use of this parameter enables parametric stereo schemes to reproduce wide sound images which led to image collapse with traditional intensity stereo schemes.

As a consequence of these enhancements, parametric stereo schemes can operate on the full audio bandwidth and thus convert a monophonic signal transmitted by a base coder into a stereo signal. While development of such technology has originally been pursued in the context of the MPEG-4 parametric audio coder [8], the parametric stereo tool defined in this standard may also be applied in the context of the MPEG-4 HE AAC coder [7]. Since a detailed description of the parametric stereo tool is outside the scope of our discussion, it is left to a dedicated paper [19].

4. BINAURAL CUE CODING

Although predating parametric stereo in publication history, the Binaural Cue Coding (BCC) approach [18] [20] [21] can be considered a generalization of the parametric stereo idea, delivering multi-channel output (with an arbitrary number of channels) from a single audio channel plus some side information. Figure 2 illustrates this concept. Several input audio channels are combined into a single output (“sum”) signal by a downmix process. In parallel, the most salient inter-channel cues describing the multi-channel sound image are extracted from the input channels and coded compactly as BCC side information. Both sum signal and side information are then transmitted to the receiver side, possibly using an appropriate low bitrate audio coding scheme for coding the sum signal. Finally, the BCC decoder generates a multi-channel output signal from the transmitted sum signal and the spatial cue information by re-synthesizing channel output signals which carry the relevant inter-channel cues, such as Inter-channel Time Difference (ICTD), Inter-channel Level Difference (ICLD) and Inter-channel Coherence (ICC).

Figure 3 shows the general structure of a BCC synthesis scheme. The transmitted (“sum”) signal is mapped to a spectral representation by a filterbank. For each output channel to be generated, individual time delays and level differences are imposed on the spectral coefficients, followed by a coherence synthesis process which re-introduces the most relevant aspects of coherence and de-correlation between the synthesized audio channels. Finally, all synthesized output channels are converted back into a time domain representation by inverse filterbanks.

Since a detailed description of the BCC approach is beyond the scope of this paper, the reader should be referred to [22] for a recent treatment of this technology.

Similar to parametric stereo, Binaural Cue Coding exhibits a number of marked advantages of simple intensity stereo coding by being able to recreate output signals with time differences and a wide sound stage consisting of uncorrelated components. Consequently, BCC can be applied to the full audio frequency range without unacceptable signal distortion. Conversely, the traditional intensity stereo processing can be interpreted as a BCC type processing which is limited to ILD synthesis only and is subject to imperfect reconstruction due to the use of critically subsampled coder filterbanks.

An alternative type of BCC has also been used to enable bit-rate-efficient transmission and flexible rendering of multiple audio sources which are represented by a single transmitted audio channel plus some cue side information [20] [21].

5. TOWARDS MPEG SPATIAL AUDIO CODING

This Section discusses the recent evolution of the previously described concepts into a new generation of compatible multi-channel representations, as it is currently under investigation within the ISO/MPEG standardization group. It includes a review of backward compatibility issues to non-multi-channel transmission, a snapshot of the current outline of the MPEG standardization activities in this field and a discussion of the projected performance of such schemes.

5.1. Backward Compatible Multi-channel Representation

From a functional perspective, the Binaural Cue Coding approach, as described in the preceding Section, offers two main features:

• Most obviously, it enables a bit-rate-efficient representation of multi-channel audio signals due to the fact that only one audio signal has to be sent to the decoder together with a compact set of spatial side information. Compared to a transmission of $N$ discrete audio channel signals, this results in impressive bitrate savings (e.g. up to almost 80% for material in the common 5 (3/2) channel audio format).
BCC offers a bridging function between monophonic and multi-channel representation: The transmitted sum signal corresponds to a mono downmix of the multi-channel material and can be presented by receiving devices that do not support multi-channel sound reproduction. This enables listening to the transmitted signal on low-profile monophonic reproduction setups in a fully compatible way, i.e. without any change in transmission format. Conversely, BCC can also be used to enhance existing services involving the delivery of monophonic audio content towards multi-channel audio.

In this sense, BCC can be also considered a method for representation of multi-channel audio which is fully backward compatible to a monophonic audio transmission (assuming that sending the spatial cues can be done in a compatible way). Since today’s consumer electronics equipment is based on 2-channel stereophony rather than on monophonic audio, a good concept for a backward compatible representation of multi-channel on such devices needs to adopt two-channel stereo as its basic compatibility layer. This motivates the use of a stereo sound representation as the basis for a BCC-type algorithm which then could scale up the information contained in these channels towards a multi-channel sound image. This concept can be summarized as follows:

• Two audio channels are transmitted from the encoder to the decoder side forming a compatible stereo downmix of the multi-channel sound to be represented.

• A BCC-type algorithm produces multi-channel sound at the decoder end by making best possible use of the information contained in the transmitted stereo downmix signal.

• For systems using a low bitrate audio coder, the compact spatial cue information can be embedded into the basic stereo bitstream in a compatible way, such that a standard stereo decoder is not affected.

A first commercial application of this idea has recently been described under the name MP3 Surround and is based on the well-known MPEG-1/2 Layer 3 algorithm as an audio coder for transmission [23]. Figures 4 and 5 illustrate the general structure of MP3 Surround encoding / decoding for the case of a 3/2 multi-channel signal (L, R, C, Ls, Rs). As a first step, a two-channel compatible stereo downmix (Lc, Rc) is generated from the multi-channel material by a downmixing processor or other suitable means. The resulting stereo signal is encoded by a conventional MP3 encoder in a fully standards compliant way. At the same time, a set of spatial parameters (ICLD, ICTD, ICC) is extracted from the multi-channel signal, encoded and embedded as surround enhancement data into the ancillary data field of the MP3 bitstream. On the decoder side, the MP3 Surround bitstream is decoded into a compatible stereo downmix signal that is ready for presentation over a conventional 2-channel reproduction setup (speakers or headphones). Since this step is based on a compliant MPEG-1 Audio bitstream, any existing MP3 decoding device can perform this step and thus produce stereo output. MP3 Surround enabled decoders will detect the presence of the embedded surround enhancement information and, if available, expand the compatible stereo signal into a full multi-channel audio signal using a BCC-type decoder.

5.2. Recent Standardization Activities

The general idea of applying BCC-type processing to expand a compatible mono or stereo signal into multi-channel sound does not rely on the use of a particular type of audio coder. In fact, even PCM transmission of the compatible channels may be used to represent the downmix channel(s). Demonstrations of the proposed paradigm with a number of well-known audio coders indicated the practical viability of the approach, including the following configurations: MPEG-1/2 Layer 3 + BCC (MP3 Surround at the 115th AES Convention New York 10/2003, 5 channel surround at 192 kbit/s), MPEG-2/4 AAC + BCC (67th MPEG meeting, Hawaii 12/2003, 5.1 multi-channel at ca. 140 kbit/s) and MPEG-4 High- Efficiency AAC + BCC (EBU workshop, Geneva 2/2004, and NAB, Las Vegas, 3/2004; 5.1 multi-channel at about 64 kbit/s).

In the area of international standardization, the ISO/MPEG Audio group has noted these recent advances and their market potential and started a new work item on Spatial Audio Coding. This process aims at complementing the existing MPEG-4 AAC-based general audio coding schemes with a tool for efficient and compatible representation of multi-channel audio. It addresses both technology that expands stereo signals into multi-channel sound (called “2-to-n” scheme) and the more traditional mono variant (called “1-to-n” scheme). The key requirements are [24]:

• Best possible approximation of original perceived multi-channel sound image
• Minimal bitrate overhead compared to conventional transmission of 1 or 2 audio channels
• Backward compatibility of transmitted audio signal with existing mono or stereo reproduction systems, i.e. the transmitted audio channels shall represent a compatible (mono or stereo) audio signal representing all parts of the multi-channel sound image
• Independence from specific audio coding technology (among other transmission schemes the technology is expected to also support MPEG-4 AAC and HE-AAC profile coders)
• Single unified architecture for both “1-to-n” and “2-to-n” processing

As of the time of writing of this paper, the MPEG group has issued a “Call for Proposals” (CfP) at its 68th meeting in March 2004 [24]. Submissions in response to this call are collected at the July meeting, and the selection of the first Reference Model (RM) is scheduled for October 2004. The final result of the standardization process can be expected to be available after a work period of ca. 2 years following these initial activities.

5.3. Performance Expectations

Looking at the underlying technological approach it becomes clear that Spatial Audio Coding avoids an expensive discrete transmission of the multi-channel signal by extensively relying on perceptual principles to “multiplex” the presented audio channels into a significantly lower number of transmitted channels. This certainly raises the question about the subjective quality which can be achieved by the new approach. Generally, for the case of the stereo-compatible (“2-to-n”) systems, some performance expectation of the Spatial Audio Coding schemes can be derived from the following considerations:

• In Spatial Audio Coding with two transmission channels, the multi-channel sound image is multiplexed (downmixed) into a stereo signal and expanded again at the decoder side. This is analogous to the well-known formats for matrixed surround, such as Prologic, Logic 7 etc.

• Contrary to such matrixed surround formats, however, the Spatial Audio Coding approach has access to some side information in order to support the reconstruction of the multi-channel sound image at the decoder side. Usage of this side information potentially results in a significant improvement over matrixed surround systems and removes the need for manipulating phase information for successful multi-channel encoding.

Consequently, the expected performance ranges between that of matrixed surround systems and a fully discrete transmission (at a significantly higher bitrate). Ideally, the improvement due to the transmission of side information might deliver a system performance approaching that of a fully independent transmission of multi-channel material, i.e. a discrete surround format.

First test results for the subjective quality of a Spatial Audio Coding scheme can be found in [23]. Carried out with a test methodology that allows a critical comparison of sound characteristics among several test conditions (MUSHRA [25]), the tests show promising results indicating the viability of the Spatial Audio Coding approach. Using a consumer grade MP3 codec running at a bitrate of 192 kbit/s stereo including spatial side information, the test outcome can be summarized as follows: The Spatial Audio Coding system achieved an overall quality that is significantly higher that a widely used system for matrixed surround coding (Dolby Prologic 2, without any low-bitrate coding). The quality of the spatial audio encoded/decoded signals was mostly rated within the “excellent” range of the grading scale. While test listeners frequently reported clear changes in the perceived sound stage for the matrixed surround format, such degradation was not noted for the spatial audio system.

In comparison to stereo compatible spatial audio coding, for which part of the original sound stage information is still conveyed in the downmix channels, mono compatible spatial audio coding (“1-to-n”) systems have to reconstruct the entire spatial sound image only from the compact spatial cue information.

While current research aims at further improving the spatial synthesis process, it will be interesting to learn about the level of fidelity that ultimately can be reached by such techniques. At this time, results of rigorous tests for “1-to-n” schemes are available only for the special case of parametric stereo coding (i.e. “1-to-2”) in the context of a particular application and coder [26].

Besides a discussion of achieved subjective audio quality, it is also instructive to look at the long term development / evolution of bitrates required for “good quality” coding (i.e. an audio quality that would be acceptable to average users for day-to-day use) of 5.1 channel material over time. An estimate of these bitrates is depicted in Figure 6 for different multi-channel audio coding algorithms as they emerged over time. Firstly, the multi-channel Layer 3 codec defined within the MPEG-2 Audio specification provided such a quality at a rate of ca. 320 kbit/s when run in non-matrixed mode (this codec was never brought into broad commercial application). Around six years later, a similar performance is shown by the MPEG-4 Advanced Audio Coding (AAC) scheme at rates around 220 kbit/s owing to its numerous refinements compared to the original Layer 3 scheme. Another significant step forward in compression performance for “good quality” audio coding is marked by the adoption of the “Spectral Band Replication” (SBR) technology by MPEG [7], leading to the so-called High-Efficiency AAC (HE-AAC) coder which demonstrated 5.1 sound at rates of 128-160 kbit/s. Recent demonstrations (e.g. at the NAB 2004) have shown that a combination of this most efficient audio codec with Spatial Audio Coding technology provides another break-through by enabling 5.1 multi-channel sound at bitrates of 64 kbit/s and even lower. This is certainly a level of compression performance that could not have been conceived a few years ago and will enable many applications for multi-channel distribution even over severely bandwidth limited channels.

6. APPLICATIONS OF SPATIAL AUDIO CODING

The main application areas of spatial audio coding schemes are related to the two most prominent features of this approach: Firstly, efficient representation of multi-channel audio with a compression efficiency significantly beyond that of discrete multi-channel coding opens the door for introducing surround sound also for applications with clear limitations in available bandwidth.
Secondly, the backward compatibility of spatial audio coding schemes allows existing (mono or stereo) audio distribution infrastructures to be seamlessly extended to surround sound without disrupting the operation of existing receivers. Examples for promising application areas include music download services, streaming music services / Internet radios, Digital Audio Broadcasting, multi-channel teleconferencing and audio for games.

7. CONCLUSIONS

Even after almost two decades of active research and development work in the area of perceptual audio coding, progress continues. This paper reviewed the evolution of techniques for joint coding of several audio channels from well-known simple joint stereo coding techniques to the recent trend on compact representation of multi-channel audio. Based on a generalized Binaural Cue Coding approach, such schemes for the first time offer high quality multi-channel sound at bit-rates of only 64 kbit/s and below when combined with state-of-the-art audio coding schemes. At the same time, the backward compatibility inherent in this approach promises to bring surround sound to existing applications without disruption of regular service. Standardization of this type of technology is on its way.

8. REFERENCES

[22] C. Faller, “Parametric Coding of Spatial Audio,” 7th Int. Conf. on Digital Audio Effects (DAFx-04), Naples, Italy, October 2004.
ABSTRACT

Parametric stereo coding in combination with a State-of-the-Art coder for the underlying monaural audio signal results in the most efficient coding scheme for stereo signals at very low bit rates available today. This paper reviews those aspects of the parametric stereo paradigm that are important for audio coding applications. A complete parametric stereo coding system is presented, which was recently standardized in MPEG-4 Audio. Using complex modulated filter banks, it allows implementation with low computational complexity. The system is backward compatible and enables high quality stereo coding at total bit rate of 24 kbit/s when used in combination with aacPlus.

1. INTRODUCTION

The performance of low bit rate audio coding systems can be significantly improved for stereo signals when a parametric stereo (PS) coding tool is employed. In such a system, a mono signal is conveyed using a State-of-the-Art audio coder and stereo parameters are estimated in the encoder and added as side information to the bit stream. In the decoder, the stereo signal is reconstructed from the decoded mono signal with help of the stereo parameters.

Techniques for joint stereo coding [11, Section 11.2.8], like intensity stereo (IS) and mid/side (M/S) coding, have long been used in audio coding systems. Intensity stereo coding can be seen as a simple form of parametric stereo, where the lateral localization of a mono signal is controlled by pan parameters. Because IS re-uses the time-to-frequency mapping of the underlying audio coder to achieve frequency selective processing, it is prone to aliasing artifacts and typically only used for higher frequency bands. Furthermore, IS is unable to reconstruct the stereophonic ambience that might have been present in the original signal.

Stereo ambience can be reconstructed using appropriate techniques to decorrelate the two channels of a stereo signal in the decoder. Further research studied reconstruction of the time or phase differences between the stereo channels.

This paper reviews the progress in parametric stereo coding, considering the PS tool recently standardized in Extension 2 of MPEG-4 Audio [5] as an example. This PS tool combines low computational complexity with advanced means for recreating stereo ambience in the decoder. To assess the performance of a complete parametric stereo coding system, the combination of the PS tool with aacPlus will be studied here. The aacPlus coder, which is used to convey the full bandwidth mono signal is this system, is the combination of Spectral Band Replication (SBR) [6] and Advanced Audio Coding (AAC) [7] and was standardized as High Ef ciency AAC (HE-AAC) in Extension 1 of MPEG-4 Audio [8].

This paper is organized as follows. Section 2 reviews the parametric stereo coding paradigm and presents the fundamental concepts of stereo parameter estimation in the encoder and stereo reconstruction in the decoder. Section 3 studies a low complexity implementation of a parametric stereo system. Optimized techniques to generate synthetic ambience are discussed in Section 4. Section 5 presents the combination of aacPlus and PS in MPEG-4 and reports the results of listening tests. Finally, conclusions are drawn in Section 6.

2. THE PARAMETRIC STEREO CODING PARADIGM

Parametric stereo coding is a technique to efficiently code a stereo audio signal as a monaural signal plus a small amount side information for stereo parameters. The monaural signal can be encoded using any audio coder. The stereo parameters can be embedded in the auxiliary part of the mono bit stream, thus achieving full forward and backward compatibility. In the decoder, the monaural signal is decoded after which the stereo signal is reconstructed with help of the stereo parameters. Figures 1 and 2 show the generalized block diagram of an encoder and decoder, respectively, that employ parametric stereo coding.

Figure 1: Generalized block diagram of PS encoder.

Three types of parameters can be employed in a parametric stereo system to describe the stereo image [9, 10, 11].

- Inter-channel Intensity Difference (IID), describing the intensity difference between the channels.
- Inter-channel Cross-Correlation (ICC), describing the cross-correlation or coherence between the channels. The coherence is measured as the maximum of the cross-correlation as a function of time or phase.
- Inter-channel Phase Difference (IPD), describing the phase difference between the channels. This can be augmented by

Heiko Purnhagen

Coding Technologies
Döbelns gatan 64, 11352 Stockholm, Sweden
hp@codingtechnologies.com
an additional Overall Phase Difference (OPD) parameter, describing how the phase difference is distributed between the channels. The Inter-channel Time Difference (ITD) can be considered as an alternative to IPD.

These stereo parameters vary over time and frequency. Psychoacoustics indicate that a Bark or ERB like frequency scale is appropriate for stereo parameters. Hence, in the PS systems discussed here, the audio bandwidth of 20 kHz is non-uniformly divided into 10, 20, or 34 stereo bands according to such a perceptual frequency scale. The temporal resolution of the stereo parameters is in the order of 10 to 50 ms.

To enable frequency-selective stereo analysis in the decoder and stereo reconstruction in the decoder, respectively, an appropriate time-to-frequency mapping is needed. To avoid artifacts caused by signal modification in the decoder, the frequency domain representation simplifies time- or phase-related analysis and modification. Both bank and transform based approaches have been used successfully.

The remainder of this paper focuses on a system using only IID and ICC parameters. It is the baseline subset of the generalized parametric stereo tool (including IPD/OPD parameters) developed by Philips and Coding Technologies that was recently standardized in Extension 2 of MPEG-4 Audio [12], [13].

2.1. Encoder

In order to enable time and frequency selective analysis and processing, the channels \( l[n] \) and \( r[n] \) of the stereo input signal are transformed into an oversampled complex frequency domain representation \( L[k,i] \) and \( R[i,k] \). Estimation of stereo parameters is carried out individually for each tile in the time-frequency plane. Such a tile is defined as the intersection of a short segment (frame) with \( i \in I \) and a frequency region (stereo band) \( b \) with \( k \in K_b \). For each tile, the IID and ICC parameters are estimated as follows.

\[
I^D \{|b| = 10 \log_{10} \left( \sum_{k \in K_b} \sum_{i \in I} |L[k,i]|^2 \right)
\]

\[
I^C \{|b| = \text{Re} \left( \sum_{k \in K_b} \sum_{i \in I} L[k,i] \right) \left( \sum_{k \in K_b} \sum_{i \in I} R[k,i] \right) \left( \sum_{k \in K_b} \sum_{i \in I} L[i,k] \right) \right) \left( \sum_{k \in K_b} \sum_{i \in I} R[k,i] \right) \right) \left( \sum_{k \in K_b} \sum_{i \in I} R[i,k] \right) \right)
\]

A suitable mono downmix is obtained as a linear combination of both input signals.

\[
M[k,i] = g_l L[k,i] + g_r R[k,i]
\]

where \( g_l \) and \( g_r \) are appropriate weights. With \( g_l = g_r = 1/2 \) a normal mono downmix is obtained.

2.2. Decoder

Based on the decoded mono signal \( m \), both channels \( \hat{l}, \hat{r} \) of the stereo signal are reconstructed with help of the stereo parameters IID and ICC in the decoder. Also for this stereo synthesis process, an oversampled complex frequency domain representation \( M[k,i], \hat{L}[k,i], \hat{R}[k,i] \) is employed and the same tiling of the time-frequency plane as in the encoder is used.

In order to be able to reconstruct stereo ambience as characterized by the ICC parameter, a decorrelated version \( D \) of the decoded mono signal \( m \) with \( E\{M^*D\} \approx 0 \) is generated by means of an appropriate all-pass filter (see Section 4). The decorrelated signal (approximately) the same spectral and temporal energy distribution as the mono signal, i.e., \( E\{D^*D\} = E\{M^*M\} \).

Based on \( M[k,i] \) and \( D[k,i] \), the stereo signal \( \hat{L}[k,i], \hat{R}[k,i] \) can now be reconstructed as

\[
\begin{bmatrix}
\hat{L}[k,i] \\
\hat{R}[k,i]
\end{bmatrix} = H[k,i] \begin{bmatrix}
M[k,i] \\
D[k,i]
\end{bmatrix}
\]

using the up-mix matrix \( H \).

\[
H = \begin{bmatrix}
c_l \cos(\beta + \alpha) & c_l \sin(\beta + \alpha) \\
c_r \cos(\beta - \alpha) & c_r \sin(\beta - \alpha)
\end{bmatrix}
\]

with \( c = 10^{10M/20} \), \( c_l = \sqrt{2c}/\sqrt{1+c^2} \), \( c_r = \sqrt{2c}/\sqrt{1+c^2} \), and \( \alpha = \arccos(\text{ICc})/2 \) reconstructs \( \hat{L}, \hat{R} \) such that they fulfill

\[
E\{\hat{L}^*\hat{L}\} + E\{\hat{R}^*\hat{R}\} = 2E\{M^*M\}
\]

and the spatial characteristics as described by the stereo parameters IID and ICC estimated in the encoder. To find an appropriate value for the remaining free parameter \( \beta \), the amount of the (undecorrelated) mono signal \( M \) in \( \hat{L} + \hat{R} \) is maximized, which gives

\[
\beta = \arctan \left( \frac{\tan(\alpha) c_r - c_l}{c_r + c_l} \right)
\]

This process can be considered as a rotation in the \( M, D \)-plane and is shown in Figure 3. Equation 7 can be simplified using the approximation

\[
\beta = \alpha \frac{c_r - c_l}{\sqrt{2c}}.
\]

3. LOW COMPLEXITY IMPLEMENTATION

For typical DSP-based applications like mobile devices, the computational complexity and memory usage of the decoder should be minimized in order to achieve e.g. maximum battery operation time. In an earlier FFF-based version [13] of the PS decoder considered here, the complexity, both computationally as well as in terms of memory, is dominated by the time-to-frequency (t/f) and frequency-to-time (f/t) transforms that are applied [12].
Recently, the SBR tool [9] for bandwidth extension of audio coding has been introduced. Similar to the PS tool, also SBR is a parametric audio coding enhancement tool that operates as post-processing in the decoder. Moreover, the structure of the SBR tool and the PS tool in the decoder are similar. Both rst apply a t/f transform to obtain a frequency domain representation, then carry out processing in this domain, and nally apply an t/f transform to convert the processed signal back to the time domain.

The fact that both tools are post-processing algorithms means their complexity adds to that of the underlying audio decoder. However, due to the fact that the underlying decoder operates at either a reduced sampling rate in the case of SBR, or in mono in the case of PS, it is less complex that in conventional full-bandwidth or stereo operation. This effect compensates for most of the complexity added by the SBR or PS tools.

The SBR algorithm makes use of complex-exponential modulated (Pseudo) Quadrature Mirror Filter (QMF) banks as t/f and t/f transforms, enabling e xible signal modulation with high computational e ciency [14]. Therefore, it is a suitable alternative to transforms, enabling e xible signal modi cation with high complexity added by the SBR or PS tools.

### 3.2. Hybrid filter bank for improved frequency resolution

For a typical sampling rate of 48 kHz, the 64 band QMF bank results in an effective bandwidth of 375 Hz. However, a PS system with, e.g., 20 stereo bands following a perceptual frequency scale, asks for a ner frequency resolution than 375 Hz at low frequencies.

In order to capture the perceptually relevant cues at a suf cient frequency resolution, the QMF bank is extended. For the lower sub-bands, an additional sub-band itering is carried out by means of oddly-modulated M band iter banks. The analysis itering for sub-band k is described by

\[
q_{k,m}[n] = \sum_{k=0}^{K-1} q_k[n - \lambda] g_k[\lambda] e^{j 2\pi (m+1)(k-\lambda)^{-\frac{1}{2}}} \tag{11}
\]

with \(k\) the prototype iter length, \(g_k[\lambda]\) the prototype iter, \(K\) the number of frequency bands, and \(m = 0, 1, ..., K-1\) the frequency index of the resulting sub-sub-band signals \(q_{k,m}[n]\).

Similarly to the signals \(q_k[n]\), the sub-sub-band signals \(q_{k,m}[n]\) can be processed resulting in \(\hat{q}_{k,m}[n]\). Using the simple synthesis operation

\[
\hat{s}_k[n] = \sum_{m=0}^{M-1} \hat{q}_{k,m}[n] \tag{12}
\]

this additional sub-band itering achieves perfect reconstruction [10]. Also the complex-valued sub-sub-band signals \(q_{k,m}[n]\) are approximately analytic signals.

For the sub-band signals that are not decomposed in separate sub-sub-bands, delay compensation is applied

\[
q_{k,0}[n] = s_k[n - \frac{K-1}{2}] \tag{13}
\]

This combination of the QMF bank with additional sub-band itering results in a hybrid iter bank. The PS system considered here typically uses 20 stereo bands and applies additional sub-band itering to the rst 3 QMF bands with \(M_0 = 8, M_1 = 4, M_2 = 4\). For all \(k = 3, \ldots, 63\) delay compensation is applied according to Equation 13. In order to further reduce the complexity of this conguration, some of the iter bank outputs have been summed. For \(k = 2, 3\) this leads to 3 iter with a real-valued impulse response.

As illustrated in Figure 4, this conguration results in a total of 71 (sub-)sub-bands. An alternative hybrid iter bank conguration, suited for 34 stereo bands, employs \(M_0 = 12, M_1 = 8, M_2 = 4\) for \(k = 2, 3, 4\) [10].

### 3.3. Stereo synthesis in (sub-)sub-band domain

In the decoder, rst a (sub-)sub-band representation of the decoded mono signal \(m\) is obtained by means of the hybrid analysis iter bank presented above. The decorrelated signal \(d\) generated directly in the (sub-)sub-band domain as described in Section 3. Then, the (sub-)sub-band representation of both channels of the stereo signal
are obtained according to Equation [4]. Finally, two instances of the hybrid synthesis iter bank are used to convert the reconstructed stereo signal into the time domain.

The stereo parameters are typically updated every 16 or 32 QMF samples, i.e., every 21.3 or 42.7 ms at 48 kHz sampling rate. In order to achieve smooth transitions, linear interpolation is applied to the elements $h_{i,j}$ of the matrix $H$ in Equation [4] for the QMF samples located between two stereo parameter updates.

Complexity estimates for the earlier FFT-based PS decoder and low complexity QMF-based PS decoder indicate that the complexity of the FFT-based system is dominated by the $t/f$ and $b/t$ transforms. The QMF-based decoder for 20 stereo bands reduced the computational complexity by 42% and the RAM requirements by 80% when compared to the FFT-based PS decoder.

4. GENERATION OF SYNTHETIC AMBIENCE

There are various ways of generating a decorrelated signal $d$ from a mono signal $m$ to enable the stereo reconstruction outlined in Section 4.3. The common approach is to apply an all-pass iter to $m$. An obvious such all-pass iter is a constant delay, and typically a delay time like 10 ms is used. However, when $m$ and $d$ are combined in the mixing process of Equation [4], the reconstructed signals $l, r$ can exhibit a strong comb-iter like characteristic.

To reduce this problem, a frequency dependent delay can be used, which corresponds to a iter with a chirp-like impulse response [13]. A better and more advanced approach to the decorrelation problem utilizes principles known from articial reverberation systems and will be presented below. A more detailed discussion of synthetic ambience in parametric stereo coding can be found in [4].

4.1. The QMF/IIR all-pass approach

To achieve computational efficiency artificial reverberation of an audio signal, algorithms combining delay lines of different lengths with feedback and all-pass filters are commonly used. Such an algorithm, called a reverberation time of a few 10 ms, can also be seen as an IIR all-pass filter and constitutes a powerful decorrelation iter.

When used in combination with QMF-based PS tool discussed in this paper, it is advantageous to implement the decorrelation for the QMF sub-band signals. This allows to easily change the decorrelator characteristics over frequency. To reduce the computational complexity, it is also possible to use a simple frequency dependent delay for QMF sub-bands at high frequencies. In the system discussed here, a reverberation-based decorrelation is employed only for the first 23 QMF bands.

Choosing delay line lengths is a crucial part in reverberator design. Best results are usually obtained by using lengths that are large numbers and are mutually prime. This is a problem in the QMF domain because of the low sampling rate of the QMF sub-band signals (750 Hz for a 48 kHz original signal). To access net time resolution, delay by fractions of the sampling interval was employed. Such a fractional delay can easily be approximated by rotating the phase of the complex (approximately analytic) QMF sub-band signals by the angle that corresponds to the desired fraction of a sample at the center frequency of the QMF band in question.

The resulting QMF/IIR decorrelator consists of a chain of three delay lines with the length of 3, 4, and 5 QMF samples. Each delay line includes also a fractional delay and the feedback is implemented in an all-pass like manner with appropriately chosen iter coefficients. Figure 5 (b) indicates the impulse response of the complete QMF/IIR decorrelator in the time domain, i.e., after the QMF synthesis bank.

4.2. Improving transient behavior

A major problem when incorporating delays or all-pass filters that include long decays into a decorrelator is the performance at transients due to the risk of generating audible post-echoes or unnatural colorization. An efficient approach to overcome this problem is to detect such transients in the decoder, i.e., within the PS synthesis, and thereafter reduce the level of the decorrelated signal using a soft decision.

Figure 5 shows the behavior of the transient reducing process. Waveform (a) shows the original input signal $m$ used in this experiment. It is a short impulse that is followed by a long tail of reverb (room response) present in the original downmixed signal. Waveforms (b) and (c) show the sum $m + d$ of the original and the decorrelated signals (i.e., synthetic ambience), without and with the transient reduction, respectively. It can be seen that the impulse response of the decorrelator is strongly attenuated when applying the transient reduction process. Hence, the reverberation characteristics (e.g., perceived room size) of the original signal remains practically unchanged.

5. COMBINING PS WITH AACPLUS IN MPEG-4

The combination of MPEG-2/4 AAC with the SBR bandwidth extension tool is known as aacPlus and was standardized in MPEG-4 as HE-AAC [8]. The basic principle of SBR has been elaborated...
on in several papers [6, 16, 13]. Figure 6 (a) depicts a mono aacPlus decoder. The output of the AAC decoder, which operates at half the sampling rate of the full bandwidth audio signal, is first analyzed with a 32 band QMF bank. Then a HF generator recreates the highband by patching QMF sub-bands from the lowband to the highband. An envelope adjuster modifies the spectral envelope of the regenerated highband and can add additional components such as noise and sinusoids, all according to the SBR guidance information in the bit stream. Finally the lowband and highband are combined and a 64 band QMF synthesis bank is employed to obtain full bandwidth decoder output signal at the original sampling rate.

5.1. Overview of the complete AAC+SBR+PS system

When the PS tool presented in this paper is combined with aacPlus, this results in a coder that achieves a significantly increased coding efficiency for stereo signals at very low bit rates when compared to aacPlus operating in normal stereo mode. Figure 6 (b) shows a simplified block diagram of the resulting decoder, which is referred to as aacPlus v2. Since the SBR tool already operates in the QMF domain, the PS tool can be included in such a decoder in a computationally very efficient manner directly prior to the nal QMF synthesis filter bank. Comparing Figures 6 (a) and (b), it is evident that only the parametric stereo decoding and synthesis, including its hybrid filter bank, have to be added to a mono aacPlus decoder, plus of course a second QMF synthesis bank. The computational complexity of such a decoder is approximately the same as that of a aacPlus decoder operating in normal stereo mode, where AAC decoding, QMF analysis, filtering and SBR processing have to be carried out for both channels of a stereo signal.

The PS tool allows for flexible configuration of the time and frequency resolution of the stereo parameters and supports different quantization accuracies. It is also possible to omit transmission of selected parameters completely. All this, in combination with time or frequency differential parameter coding and Huffman codebooks, makes it possible to operate this PS system over a large range of bit rates.

When an aacPlus v2 coder is operated at target bit rate of 24 kbit/s, the PS parameters require an average side information bit rate of 2 to 2.5 kbit/s, assuming 20 stereo bands for IID and ICC. For lower target bit rates, the PS frequency resolution can be decreased to 10 bands, reducing the PS side information bit rate accordingly. On the other hand, the PS tool permits to increase time resolution and to transmit IPD/OPD parameters, which improves the quality of the stereo reconstruction at the cost of 10 kbit/s or more PS side information.

5.2. Subjective test results

Figure 7 shows subjective results from a listening test comparing aacPlus v1 using normal stereo coding at 24 and 32 kbit/s with aacPlus v2 utilizing the PS tool at 24 kbit/s [13]. Two sites (indicated in black and gray) participated in this test, with 8 or 10 subjects per site, respectively. The 10 items from the MPEG-4 aacPlus stereo verification test [17] were used as test material and playback was done using headphones. The test employed MUSHRA methodology and included a hidden reference and low-pass filtered anchors with 3.5 and 7 kHz bandwidth.

At both test sites, it was found that aacPlus v2 at 24 kbit/s achieves an average subjective quality that is equal to aacPlus v1 stereo at 32 kbit/s and that is significantly better than aacPlus v1 stereo at 24 kbit/s. It is of interest to relate these results to the MPEG-4 verification test [17]. There, it was found that aacPlus v1 stereo at 32 kbit/s achieved a subjective quality that was significantly better than AAC stereo at 48 kbit/s and was similar to or slightly worse than AAC stereo at 64 kbit/s. This shows that aacPlus v2 achieves more than twice the coding efficiency of AAC for stereo signals. Further MUSHRA tests have shown that aacPlus v2 achieves a significantly better subjective quality than aacPlus v1 stereo also for 18 and 32 kbit/s.

Figure 5: Impulse fed to a conventional synthetic reverberator. (a) Original mono downmixed signal (no decorrelation). (b) Original signal + QMF/IIR decorrelation without transient reduction. (c) Original signal + QMF/IIR decorrelation with transient reduction.
Figure 7: MUSHRA listening test results for two sites (black and gray) showing mean grading and 95% confidence interval (from [13]).

6. CONCLUSIONS

A low complexity parametric stereo coding tool has been presented. It was shown that this parametric stereo coding tool significantly enhances the coding efficiency of existing audio coders. The presented tool is particularly interesting in combination with audio coders using SBR bandwidth extension, since the resulting coder has approximately the same computational complexity as in a normal stereo configuration. The combination of AAC, SBR, and the parametric stereo tool presented here was included in the MPEG-4 Audio standard and is referred to as aacPlus v2. It enables coding of stereo signals at bit rates that are less than 50% of those required by AAC to achieve the same subjective quality and was recently adopted as recommended coder for audio streaming in Release 6 of the 3GPP standard for mobile services.

7. ACKNOWLEDGMENTS

The author would like to express his gratitude to Erik Schuijers, Jeroen Breebaart, and Jonas Engdegard for providing some of the material that was used for preparing this paper.

8. REFERENCES


168 — DAFx’04 Proceedings — 168
COMPUTATIONAL REAL-TIME SOUND SYNTHESIS OF RAIN

Stanley J. Miklavcic, Andreas Zita and Per Arvidsson

Department of Science and Technology
University of Linköping,
S-601 74, Norrköping Sweden
stami@itn.liu.se

ABSTRACT
Real time sound synthesis in computer games using physical modeling is an area of great potential. To date, most sounds are pre-recorded to match a certain event. Instead, by using a model to describe the sound producing event, a number of problems encountered when using pre-recorded sounds can be avoided. This paper deals with the application of physical modeling to the sound synthesis of rainfall. The implementation of a real-time simulation and a graphics interface allowing an interactive control of rainfall sound are discussed.

1. INTRODUCTION
The goal of sound synthesis is the artificial production of acoustic information mimicking actual sound events. One area of application is the real-time sound synthesis for computer game environments. To date, most computer game sounds are pre-recorded for use in certain events, engines, collisions and other interactions, weapons fire, weather effects, etc. However, as computer games are becoming increasingly more realistic, allowing the user to interact with the virtual world in many conceivable ways, sounds that match any possible situation are more difficult to produce using sampling methods. At best, pre-recorded sounds can be filtered or conformed in some way to match a given situation, but this is often an inefficient and time-consuming task. Furthermore, a given sound source can behave in a vast variety of ways depending on the variables specific to each occurrence. For example, a physical event such as a collision can be very different depending on a large number of factors, such as the amount of energy released, the velocities, masses and angles involved. Obviously, too great a scope of possible scenarios exist to be treated with pre-sampled sounds. Sounds generated procedurally require precisely the type of input dictated by physical parameters. In such cases, real-time generated sound based on a physical model can represent a more desirable alternative. This is our point of departure.

In this paper we report on our efforts concerning the synthetic generation of the sound of rainfall using physical modeling of the sound generation process [1]. Emphasis is placed on rainfall onto solid surfaces. Based on this model, a real-time simulation of large clusters of raindrops has been implemented, relying on the superposition principle and a random time and space distribution of impact points and raindrop sizes. Important physical parameters such as rain intensity, drop volume and impact speed contribute to the overall audio simulation.

We have also been involved with the modeling of sound generated by drops falling specifically onto a liquid surface [2]. The overall sound event is schematically represented in Figure 1 [3, 4].

Naturally, to synthetically produce a realistic drop sound as heard above the surface of the water using a physical model requires a correct understanding of the mechanism(s) behind the acoustic process. Although a consensus in the literature on this subject is doubtful, what is generally accepted is that the familiar, characteristic “plopping” or “plunk” sound is strongly correlated with the appearance of an entrained bubble under the water surface [3, 4]. Furthermore, it is generally accepted that not all falling raindrops generate entrained bubbles nor produce the characteristic sound. Clearly, these facts too must also be taken into account in a computational sound simulation model. We outline how these features can be incorporated and give an expression for the sound function to be implemented in a simulation.

2. SOUNDS OF IMPACTING RAINDROPS
2.1. Raindrop impacting on a hard surface
A mathematical model for sound produced by a falling drop onto an arbitrary flat surface can be specified in terms of the acoustic pressure, \( p = p(\mathbf{x}, t) \). This function satisfies the wave equation

\[ \nabla^2 p - \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} = 0, \]

subject to appropriate boundary conditions. For a rigid surface these are the no fluid penetration condition, \( \mathbf{n} \cdot \mathbf{u} = u_x = 0 \), and, sufficiently far away, the Sommerfeld radiation condition [5, 6]. With help of the Green’s function for a rigid surface (response from a point source in the form of a direct wave and a surface
reflection), the acoustic pressure registered at position $\mathbf{x}$ at time $t$, can be expressed as the boundary integral

$$p(\mathbf{x}, t) = \frac{\rho_0}{2\pi} \int_{\Gamma} \frac{\hat{v}(x_s, y_s, t - R/c)}{R} dx_s dy_s,$$

over a source distribution found on the horizontal water surface, $z = 0$, denoted $\Gamma$. Here, $R = \sqrt{(x - x_s)^2 + (y - y_s)^2 + z^2}$ is the geometrical distance from the observer position $\mathbf{x} = (x, y, z)$ to the source position $\mathbf{x}_s = (x_s, y_s, 0)$. $\rho_0$ is the air density and $c$ is the air speed of sound. The pressure is completely determined once the surface velocity is specified. For simplicity, it is sufficient to assume that a raindrop imparts an impulse velocity

$$v(x_s, y_s, t) = A(v_{\text{term}}) H(t)$$

uniformly over a circular area symmetrically positioned around the impact center. The factor $A(v_{\text{term}})$ is related to the kinetic energy of the drop with terminal velocity, $v_{\text{term}}$, just prior to impact. $H(t)$ is the Heaviside step function. Under these assumptions $v_s$ takes values within a circle of radius $a$ (the drop radius) centered at the impact center $(x_0, 0, 0)$ (see Figure 2). Taking the listener position to be $\mathbf{x} = (0, 0, H)$, the integration in (1) can be performed [1, 2] giving

$$p(0, 0, H, t) = \frac{\rho_0 c A(v_{\text{term}}) \cos^{-1} \left( \frac{c^2 t^2 - H^2 + x_0^2 - a^2}{2x_0 \sqrt{c^2 t^2 - H^2}} \right)}{2\pi}$$

(2)

for $t_s = R_s/c \leq t \leq R_t/c = t$, and $p = 0$ for all other $t$. Here $R_s(t) = \sqrt{(x_0 + a)^2 + H^2}$ is the shortest (longest) distance from the impact zone to the listener position. This analytic result is an accurate representation of the initial contribution to the sound schematically represented in Figure 1. In terms of sound simulation, two advantages are evident in (2). Firstly, the result is analytic, which can make computation rendering very efficient. Secondly, physical characteristics of the drop such as size, terminal velocity and impact location are naturally incorporated.

2.2. Raindrop impacting on a water surface

As rain continues to fall onto an impermeable surface one expects pools of water to form. The sound of the rainfall changes character which must be included for realistic effects. Elsewhere we have presented a description of the physical processes involved and a mathematical model based on these. However, the main feature is that the impact drives the fluid interface to form a crater in the shape of a cone [7]. At the apex of this cone, an entrained bubble is pinched off. The pinching-off gives rise to oscillations in the cone tip which lead to sound generation at the cone/crater opening [2].

As mentioned, not all drops satisfy conditions on size and terminal speed to create an entrained bubble and thus produce the characteristically familiar sound effect. However, assuming that these conditions are met and an impact crater in the shape of a truncated cone of length, $l$, splay constant, $\lambda$, and area, $A$, of opening at $z = 0$ is formed, the sound produced can be represented by the fluid particle velocity at the surface, $z = 0$ for $t \geq 1/c$ is [2]

$$u(t, \mathbf{x}) = v_0 \left( \frac{t - \frac{1}{c}}{1 - \frac{l}{s}} - \frac{c}{s} \frac{l}{s^2} \right) \times \int_0^{t - l/c} \int_0^{t - l/c} \exp \left( -\frac{c}{s} \frac{l}{s^2} \frac{t - \frac{1}{c} - \tau}{s - l} \right) d\tau. \quad (3)$$

Here, $s = \sqrt{A/\pi/\lambda} > l$ and $v_0(t)$ for $t \geq 0$ (zero otherwise) is a velocity representing the pinch-off effect triggering acoustic vibrations in the cone.

To pursue the issue of the sound heard above the level of the water it is sufficient to insert this expression into the boundary integral expression, (1), for the acoustic pressure. As in Section 2.1, the integration is over the $z = 0$ plane, effectively restricted to the cone opening. The end result depends very much on the nature of the apex motion, $v_0$. Simple examples are considered elsewhere [2].

3. IMPLEMENTATION AND SOUND REPRODUCTION

3.1. Rendering of the computational sound

Using superposition, the above models of single impact sounds have been implemented in an algorithm based on clusters of drops falling in an area bounded by two circles centered at the listener position. The drop source positions are distributed randomly and uniformly in this area, with each source receiving a direction, $\phi$, and a distance, $x_0$, relative to the listener. Naturally, the radius, $a$, of each drop source must either be user-specified or randomly allocated a value. We have taken the latter approach.

Some care must be exercised when distributing each drop source in the simulated time interval. The most straightforward approach of randomly producing, for each drop, a value of a global impact time $t_0 = r \times L$ where $r$ is a random value between 0 and 1 and $L$ is the simulation length, can lead to problems. Randomizing $t_0$ would only define the impact time of each drop, not the arrival time. Since each drop also receives a random impact distance, the arriving sound impulses are distributed quite differently in the simulation period. In a real-time algorithm this is not a suitable solution. For example, assuming a continual rate of drop impact, since the sound originated initially comes from the drops lying closest to the listener, the intensity will artificially grow until the sound from

![Figure 2. Schematic showing the geometric relation between the circular source region \((x_s, y_s, 0)\) and the listener position \((0, 0, H)\), and distances involved.](image-url)
drops lying furthest away reach the listener. From this point on the intensity will stay at a constant level. This effect prevails each time a change in conditions is implemented. The problem arises from the difference between the impact time and the initial arrival time, and this depends on the distance between the impact point and the listener.

Secondly, to design a system that handles simultaneous production and playback\(^2\) is non-trivial. However, as our algorithm is implemented in C++, we are able to take advantage of available C++ routines for playback. The Sound Toolkit (STK) [8] is a free set of pre-implemented C++ routines to handle various useful sound synthesizing and playback tasks. Among other things, it handles real-time playback of synthesized sound information on a wide variety of platforms. In this project, STK was used for these tasks with very good results. The code is very easy to use and well documented.

3.2. Multi-speaker sound reproduction

As the listener is modeled as a single point, binaural effects are not incorporated in the present version of the audio simulation. However, equivalent stereophonic or, more generally, multiphonic sound can be produced using a simple alternative method. The impact zone symmetrically located around the listener is divided up into circle sectors defined according to the number of output speakers at one’s disposal. For example, two half-discs in the case of stereo, or quadrants in the case of quadraphonic sound, etc. Note that these need not be equal area sectors, as in Figure 4.

When a drop impact with the ground is simulated using the random distribution function its sector is determined by its relation to the placement of the two nearest speakers. The sound pressure calculated using either of the two methods described in previous Sections is distributed between the pair of speakers, weighted according to a simple linear function of the angle between the drop direction and the speaker directions, relative to a zero-angle in the listener’s direction. A drop falling in the direction of one of the speakers results in a weighting of 100% distributed to this speaker. This method not only allows for a stereo speaker setup, it also allows for any arrangement and number of speakers, as in Figure 4.

This problem can be solved if one instead judiciously randomizes the arrival time, \(t_s\). This results in a direct response to user implemented changes. Further considerations are necessary to take into account due to the coupling between distance and sound arrival time [1].

In a real-time simulation we restrict the length of the simulation rendering stage described above and loop over this interval during the entire simulation. This iteration loop has a length of only a few hundredths of a second\(^1\). Hence, the algorithm prepares these so-called sound buffers (Figure 3), which are handed over to the playback device upon request. This technique allows for continuously playing sound. As fast as a buffer has been played, a new buffer will be taken on. Note also that this process is not sequential; the buffer production stage does not need to wait for the playback of a previous buffer to finish. It can safely continue preparing the upcoming buffer while the current buffer is being played.

This non sequential, semi-parallel process has many benefits. It is naturally faster and also allows for the possibility of user-interaction. Since new buffers are prepared many times a second, one can change the properties defining the rain system smoothly during the simulation. This is precisely what is desired in a real-time algorithm. Naturally, it is more complex and cumbersome to implement. Firstly, it is important to have an algorithm which produces sound pressure buffers at a fast enough rate. If the algorithm is too slow, it will not be able to produce buffers at the same rate as they are being played and there could be a delay in the process in which the playback routine waits for follow up buffers. When this happens repeated buffers will be played, which is undesirable.

\(^1\)The algorithm uses a simulation period, or buffer, of 512 samples. With a sample rate of 44100 smp/sec we have a buffer size \(\approx 0.0116\) seconds. Longer or shorter buffer sizes can be experimented with. However, 512 was found suitable in this case.

\(^2\)A system such as this is called a multi-threaded system. In it, simultaneous tasks can be calculated or processed.
been found to greatly enhance the surround sound effect as well as the depth of produced sound. This method also fits for further extensions of the simulation, using localized source types and sound events in a 3D environment.

A simple user interface application was implemented in Macromedia Director to wrap the functionality of the real-time algorithm and playback system (not documented). This was made possible by producing a so-called script-Xtras for Macromedia Director to control the real-time algorithm and indicate the desired or destination values of properties such as blue slider positions. The kinematics of the animation (the red sliders move to the left) are set by the user to simulate different rain conditions.

Methods such as granular synthesis rely on extensive psycho-acoustic surveys in order to establish parameters that affect the sound in the desired way [10]. Such a process is time consuming and often results in parameters that do not have the flexibility needed for simulation of real rain. Compared with real rain sound, the physical model produced sound that was very similar in nature to the bare sound of rain, i.e., under ideal circumstances. In practice it is very unusual to hear only the rain sound we have modeled. More often than not a number of other events produce sounds we are accustomed to hearing and associate with rainfall: thunder, water flowing and splashing, rain impacting on vibrating surfaces (rooftops, etc.), rustling leaves, howling winds, etc. Only when a combination of these other sound events is arranged will it be possible to fully appreciate the end result of our synthesized sound.

An important consideration is the computational effort involved in the simulation. Depending on simulated rain intensity (number of drops) and rain source radius (larger sources result in more samples to evaluate), the program proved to run smoothly on an ordinary PC setup (>1GHz) with all ranges of simulated rain sound.

In summary, we find that a physical model is a good option for use in virtual environments such as game worlds. It is likely that as more physical scenarios are successfully modeled, the computer games industry will be offering more complete solutions of real-time sound engines based on physical models for use in real-time game environments.

5. REFERENCES


A STRATEGY FOR MODULAR IMPLEMENTATION OF PHYSICS-BASED MODELS

Matthias Rath
Dipartimento di Informatica
Università degli studi di Verona, Italy
rath@sci.univr.it

ABSTRACT
For reasons of practical handling as well as optimization of the processes of development and implementation, it is desirable to realize realtime models of sound emitting physical processes in a modular fashion that reflects an intuitively understandable structure of the underlying scenario. At the same time, in discrete-time algorithms based on physical descriptions, the occurrence of non-computable instantaneous feedback loops has to be avoided. The latter obstacle prohibits the naive cross-connection of input-output signal processing blocks. The following paper presents an approach to gain modularity in the implementation of physics-based models, while preventing non-computable loops, that can be applied to a wide class of systems. The strategy has been realized practically in the development of realtime sound models in the course of the Sounding Object [1] European research project.

1. INTRODUCTION
One common question in the development and implementation of sound generating algorithms based on models of physical processes is that of possible modularity. It is often desirable to represent distinct independent objects in a physical scenario to be modelled, a string, a hammer, a bow,..., in accordance distinct algorithms that might be developed independently and combined in a second step. Ideally, implementational representations of physical objects or processes should be combinable, i.e. interconnectable, finally on a higher level of implementation, e.g. at runtime through a graphical interface, according to the intended overall physical scenario. E.g., in mechanical reality solid objects such as strings and hammers may be combined in various ways, "a hammer striking a string", "a hammer sliding along a string", "two hammers colliding"..., while their inner structure and properties stay generally unchanged (assumed that involved forces are not excessive, not causing lasting deformation). For practical use (in particular for musicians or non-experts) it would be of strong value to preserve this "modularity" of physical reality, throughout the process of modelling and implementation in such a way that software objects representing physical units (strings, hammers, interaction processes...) might be connected in a way analogous to mechanical combination. Furthermore, development and implementation may be optimized when algorithms representing physical objects can be reused in various combinations. Usually, in implementations of sound synthesis based on physical models, specific closed physical systems are handled and changes in the underlying physical system require the repetition of the whole process of development (discretization of differential equations) and implementation.

The obstacle that is responsible for the difficulty to achieve modularity in implementations of physical models is the need to avoid the occurrence of non-computable instantaneous loops. Such non-computabilities occur e.g. when naively connecting two signal processing blocks with no additional delay in such a way that the input, internal state and output vector of one block instantaneously depends on the contemporaneous values of the other block and vice versa. The most simple reaction to the problem, the insertion of an ad-hoc delay somewhere in the non-computable chain necessarily introduces errors that are hard to control. Thus, unproblematic modularity is usually only achieved when cross-dependencies of input/output blocks are excluded. The latter is e.g. the case when fixed force profiles are assumed for the excitation of resonators, such as for contacts of solid objects (e.g. [2], [3]); this however, is a rather restricted model of physical reality where in fact interaction forces and the contemporaneous states of interacting objects instantaneously mutually depend. During periods of interaction indeed, generally distinct, independent physical objects form a common system. The occurrence of non-computable, instantaneous loops is usually avoided during the process of discretization. One approach that allows the handling also of non-linearities (and which forms a basis for the strategy presented in this paper) has been developed by Borin et al. under the name of "K-method" [4]. As a side effect of such strategies to avoid instantaneous loops in discrete-time algorithms already at continuous-time level, before and during the process of discretization, eventual structurization of an initial physical scenario, e.g. into distinct solid objects, is generally not passed on to the final algorithm since describing continuous-time equations are merged in a first step. The whole development process then has to be repeated for varied physical scenarios, e.g. exchanges of objects or different modes of interaction.

In the following I present an approach to combine independent discrete-time algorithms in a modular fashion without introducing non-computable instantaneous loops or artificial ad-hoc delays. The key ideas are inspired by the K-method as described extensively in [4] and some final technical details are taken over identically and not handled here again; I instead refer to [4] at respective points. The following approach however attacks the problem directly on discrete-time level; it thus allows the integration also of algorithms that were derived from continuous-time differential equations through different methods of discretization or modelling techniques (finite-elements approximations, digital waveguides).

2. INITIAL SETTING, FORMULATION AND SCOPE
The implementational strategy described in this paper was developed during the work on realtime physics-based sound models of scenarios of contacting solid objects that was done as part of the Sounding object (SOb) European research project [1]. Behind the
goal of modularity in implementation lies the central consideration, that the solid objects in the intended scenarios (“hitting”, “bouncing”, “rolling”, “rubbing”) show a characteristic, individual inner behavior which can be described independently from the different ways in which these objects may interact — impact and friction are at the core of the cases looked at in the course of SOh project. The same decomposition into objects of fixed independent characteristics and a specific way of interaction can surely be applied to many other sound-emitting processes, also such of non-mechanical, e.g. electro-magnetic nature. For this reason, the developed technique is assumed to be of potential wide use in the field of modelling of physical systems for sound synthesis.

Relevant for the interaction of the distinct objects in our cases of interest, is usually only a limited configuration, not their complete internal state. Furthermore, internal properties of interacting (i.e. here: contacting) objects can be described in different ways (such as in terms of resonant modes), independently from the interaction, that does not induce permanent changes to the objects. On the other hand, information is exchanged only via the objects’ configurations in the areas of contact; the entire state of the objects can generally not be deduced from their behavior in one, or some, limited contact areas. A structure of implementation is therefore of interest that allows to develop independently, computational algorithms representing distinct objects and processes of interaction, and to freely connect such algorithms without the need of further adaptation. In the following, I refer to representations — of whatever nature, discrete–time (mostly) or continuous–time, of independent objects in the explained sense as “resonators”, and representations of processes of interactions as “interactors”. The term resonator here aims solely at the presence of some sort of memory, i.e. some internal state that is relevant for the subsequent, future behavior. This notion of an internal state is reflected through a differential operator in continuous–time representations, while we have some temporally changing state vector, \( w \), in the discrete–time algorithms, with a “state–update” algorithm, \( w(n) \rightarrow w(n + 1) \); no general a priori specifications, e.g. concerning linearity, are given at this point. For simplicity, interactors are here assumed to be memory–less, i.e. instantaneous relations; this assumption may be bypassed \(^{1}\) but is unproblematic and simplifies the description.

Resonators and interactors are connected through input and output vectors that can most easily be thought of as forces \( f \) (coming from the interactors) and spatial position–velocity configurations \( x \), as in our concrete case: the complete state \( w \) of the resonator is generally not passed to the interactors. Figure 1 gives a sketch of the intended modular structure as described, and the connected problem; only one resonator is depicted here, which has no influence on the validity of the following arguments. In the most strict sense, “modularity” would mean here, that discrete–time realizations of resonators and interactors formulas can be exchanged and “plugged” at this discrete–time level without any further knowledge about the internal algorithms or their origins, such as an underlying continuous–time model or the used technique of discretization (such as bilinear transform, Euler backward differenting . . .) whatsoever. Discrete resonators should be handable as “black boxes” that produce output vectors at every time step depending on their contemporaneous input vector and the hidden state-vector. It is seen that this goal conflicts with the instantaneous cross–relationship given by the interactors: in Figure 1, \( f(n) \) would be computed from \( x(n) \), which in turn depends on \( f(n) \); this loop cannot be resolved without additional information about the internal structure of the resonator, i.e. without “breaking the black box”. A non-computable instantaneous feedback-loop occurs.

3. MODULARITY USING “LABELED BLACK BOXES”

The described problem is solved and modularity is reached in the development and interconnection of the resonators and interactors through a “labeled–black–box” approach. It is clear that discrete–time resonators, in the situation of Figure 1, cannot be handled as strict black boxes, in the sense of passing input to output vectors without additional information. However, as we will see now, it is on the other hand not necessary to reveal the origin and complete internal structure of the resonator algorithm, nor to reconstruct the whole computational structure for each change of objects or interaction. Under certain presumptions on the resonator algorithm we are able to resolve the instantaneous feedback loop, with the help of a “label” attached to the black box, representing minimum information about its hidden internal structure. The important point here is the exact specification of these presumptions on the resonator and of the “minimum information necessary”, and the derivation of a uniform representation and interconnection procedure. The developed solution, that is now presented in detail, is inspired by, and closely related to, the K-method; we however work directly and only on the discrete–time level without referring back to (possible) continuous–time origins of the discrete algorithms. I will finally apply the techniques inherited from the K-method, that are not explained in detail again; at the point I refer to \(^{2}\).

In discrete time, the most general resonator consists of a discrete–time state vector \( w \) and some “time-step” or “update” function \( R \) such that
\[
 w(n) = R(w(n-1), e(n), f(n)) ,
\]
where \( f \) is the output (“force”) vector of the interactors (see Figure 1) and the vector \( e \) represents some external influence on the resonator, that is independent from the interactors. \(^{2}\) In plain words, with each time step, the resonator state is updated from the previous state vector and the contemporaneous external input vectors \( f \) and \( e \), coming from the interactors resp. some independent external source. Further on, the resonator shows a representing configuration vector \( x(n) \) to the “outside world”, on which in turn \( f(n) \) depends. In this application here, a vibrating solid object is accessed through its configuration, position and velocity, in a certain contact point (or area). \( x(n) \) “represents” (to the outside, in particular the interactors) the resonator in its state \( w \) via some function \( S \):
\[
 x(n) = S(w(n)) .
\]
Combining equations (1) and (2), we can see \( x(n) \) as a function of \( f(n) \) and the vectors \( w(n-1) \) and \( e(n) \), that are known from the previous time step resp. an external input:
\[
 x(n) = S(R(w(n-1), e(n), f(n))) .
\]

1A model of friction interaction has been developed and implemented, that makes use of the structure presented here, and, indeed, a friction interactors with internal memory.

2For clarity of the picture, \( e \) is not depicted in Figure 1 as not relevant for the general idea and unproblematic.
The condition that is now imposed on the resonator, is for the concatenation $S \circ R$ of the functions $R$ and $S$ to split into two summands, one of which depends only on the known vectors $w(n-1)$ and $e(n)$, and another depending linearly only on $f(n)$. (The symbol “$\triangleq$” is used in the following equation to indicate that as a necessary precondition for the applicability of the following approach this relation with some $T$ and $L$ as above has to exist):

$$(S \circ R)(w(n-1), e(n), f(n))) = S(R(w(n-1), e(n), f(n))) \triangleq T(w(n-1), e(n)) + L(f(n)), \quad L \text{ linear}. \quad (4)$$

This condition is fulfilled in particular if both $R$ and $S$ are linear, as in our case of modal description with “pick up” points, or if both functions split in the described way. It is however thinkable that the condition holds neither for $R$ nor $S$, but for the concatenation $S \circ R$, i.e. that non-linearities “cancel out”. $L$ is a linear mapping between finite-dimensional vectors can also be seen as a matrix $L$ whose dimensions are the dimensions of $\mathcal{L}$, i.e. the second main goal behind the term “modularity”.

In plain words, $p(n)$ is equal to the output vector of the resonator under some fictitious “pseudo-update” with zero input (force). As a result, we finally see that the non-computable loop in Figure 1, $f(n) = F'(?) = F'(f(n))$, can be turned into a resolvable implicit relation, equation (7), if the black box of the resonator is equipped with

1. a label containing $\mathcal{L}$ and
2. a pseudo-update functionality, that delivers the “simulated” resonator output with zero input, without de-facto updating the internal state.

The dimensions of $\mathcal{L}$ have already been mentioned as being of a similar order as those of $f$ and $x$; exactly, $\mathcal{L}$ contains $\dim(f) \times \dim(x)$ elements. If the vectors $f$ and $x$ are of orders below 10 — we most often deal with force and position/velocity vectors in one- to three-dimensional space — passing $\mathcal{L}$ whenever necessary, i.e. when resonator or interactor or any of their attributes (such as modal parameters or the point of interaction for impact or friction) are exchanged, is thus a negligible overhead in comparison with the processing of the in- and output vectors $f$ and $x$ that have to be passed with each time step, i.e. usually 44100 times per second. In particular is the size of $\mathcal{L}$ often small compared to the state vector of the resonator: the internal state vector of a digital waveguide e.g., can easily reach dimensions of the order of $3^{10}0000$ while its representing external configuration would usually be of dimension 2 (position and velocity in a point...).

Summing up, the update-cycle at each time-step $n$ for the complete discrete–time system consists of the schedule given in Figure 2.

---

A simple two-directional waveguide with a minimal frequency of 10 Hz at a sample-rate of 44100 Hz, contains at least two delay lines of 4410 samples each.
1. Read in external variables to the resonator(s), such as additional external forces or related signals (e(n) in the notation above).

2. Pseudo-update of the resonator(s) from previous state w(n − 1) and e(n), without de-facto update of the internal resonator(s) state. p(n) is passed to the interactor.

3. Calculation of f(n) from p(n). The mathematical technique for this step depends on the interactor function F. In the concrete cases here of impact an explicit formulation can be used in the (piece-wise) linear case, while the non-linear relation is solved through Newton–Raphson approximation [5].

4. After f(n) has been computed and passed to the resonator(s), the internal resonator states are updated, w(n − 1) → w(n).

Figure 2: The update schedule at each time-step (sample cycle).

4. EXAMPLE APPLICATIONS

The previously described strategy has been applied successfully in the implementation of several models of contacting solid objects. Combined can be resonators in general modal description, realized in discrete time using discretization by bilinear transform, and the simple case of an inertial point mass. A digital waveguide resonator is being developed. These representations of solid objects are integrated through interactors modelling interaction in contacts based on impact or friction. The resulting models have been implemented in C as modules for the realtime sound software pd [6]. Due to the surrounding architecture of pd, the different resonators and interactors are linked statically at compilation time, but dynamical linkage would be straightforward in a suitable software environment. The modular approach presented above strongly minimized development efforts. All details of the development process, from the physical description to computational algorithms and practical handling can be found in [7] (more exactly [8], [9] and [10]).

5. CONCLUSION

In this paper, an approach for the modular implementation and development of computational algorithms based on physical models for sound generation has been described. The approach, that has been derived and practically realized during the work of implementation in the course of the European project The Sounding Object [1], has been presented in a general form and can be applied to a wide range of sound-producing scenarios.

6. ACKNOWLEDGEMENTS

This work has been supported by the European Commission under contract IST-2000-25287.

7. REFERENCES


IMPLEMENTING LOUDNESS MODELS IN MATLAB

Joseph Timoney, Thomas Lysaght
Dept. of Computer Science
NUI Maynooth, Maynooth, Co. Kildare, Ireland
{jtimoney|tlysaght}@cs.may.ie

Marc Schoenwiesner
AG Neurobiologie Institut
Leipzig, Germany
marcs@rz.uni-leipzig.de

Lorcán Mac Manus
Dept. of Elec. Engineering
DIT, Dublin, Ireland
lorcan.macmanus@dit.ie

ABSTRACT

In the field of psychoacoustic analysis the goal is to construct a transformation that will map a time waveform into a domain that best captures the response of a human perceiving sound. A key element of such transformations is the mapping between the sound intensity in decibels and its actual perceived loudness. A number of different loudness models exist to achieve this mapping. This paper examines implementation strategies for some of the more well-known models in the Matlab software environment.

1. INTRODUCTION

The primary tool in the field of audio for the time-frequency analysis of sound is the Spectrogram. It is popular because it is computationally fast and its output is well understood. However, since the 1990’s much work has been carried out on the development of better tools for sound analysis that make more efforts to take psychoacoustic properties into consideration. This has been driven by the availability of the technology to fully implement the results of psychoacoustic research that had been published over the previous decades, combined with the desire for significant advances in the coding of speech and audio signals. The MP3 standard is a good example of this. Thus nowadays, many algorithms designed for speech and audio processing will make reference to psychoacoustic transformations. An important limitation of the spectrogram in this regard is the manner in which the signal intensity is displayed, generally in Decibels SPL. While this provides a measure of objective sound intensity, it does not properly capture the subjective impression a sound creates on the listener in terms of its loudness. To achieve this the sensitivity of the ear to the various sound levels of the frequency components contained in the sound must be accounted for. This is the kind of information contained in equal loudness curves for the human ear [1]. These curves show that the ear is less sensitive to low frequency sounds, having a maximum sensitivity in the region of 3–4kHz. Employing these curves to modify the dB SPL intensity display of the sound transforms the intensity to the Phon scale, where different frequency components having the same Phon value will have the same loudness but will have different dB SPL intensities. One disadvantage with the Phon scale is that it is not directly proportional to perceived loudness, and thus a doubling of loudness value in Phons does not mean a doubling of the sound loudness [2]. To this end, the Sone scale was introduced to provide a linear scale of loudness. The Sone scale can be related to the phon scale by the equation [3]:

\[ L(i) = \begin{cases} \frac{D(i)}{40}, & \text{if } D(i) < 40 \\ 2^{\frac{1}{3}(60(i) - 40)}, & \text{if } D(i) \geq 40 \end{cases} \]  

where \( L(i) \) is the perceived loudness of the critical band \( i \), and \( D(i) \) is the spread critical spectrum in terms of phons in band \( i \).

The conversion from a time domain signal to a representation that describes its loudness in terms of Sone is outlined in Figure 1.

![Figure 1: Block Diagram of loudness modeling procedure [4].](image)

There are various approaches to implementing the different stages of the Loudness model in Figure 1. The basic procedure is to first transform the signal into the time-frequency domain. The frequency analysis points specified will have a relation to the critical band resolution of the ear. Time and Frequency masking may be accounted for and compensation carried out for components below the threshold of audibility. This stage is followed by a conversion from the intensity levels of each time-frequency slice to specific
loudness levels for each frequency band. These are then summed to give the overall loudness for each time slice.

In this paper, three implementation strategies are examined:

1. A direct implementation based on a time-frequency decomposition, a mapping from dB SPL to Phon followed by a direct implementation of equation (1);
2. Three implementations of Zwicker’s model;
3. The Moore and Glasberg Loudness model.

The sources for some of the implementations discussed are speech quality measurement strategies. Specifically, the time-frequency decompositions and loudness conversion from the EMBSD [3], PSQM [10] and PEAQ [12] measures are investigated.

2. MODEL IMPLEMENTATIONS

2.1. Calibration

In all implementations the first, and possibly most crucial, stage is the calibration of the input signal. In Matlab sounds read in from wav files are normalized to have amplitude levels lying between 1 and –1. However, this will neither reflect the true recording or playback levels of the sound. The amplitude of sound can be scaled to give the sound a desired level of dB SPL. When using dB SPL to set a sound level, a value for the reference level must be chosen. For air, the reference level is usually chosen as 20 micropascals [5]. If the actual dB SPL used when recording the sound is unknown, in the case of speech, if it is at a normal level, it is reasonable to assume a conversational level of between 65 and 70 dB SPL. Thus, to scale the signal vector \( y \) to a level of 70dB in Matlab [6]:

\[
\text{SPLmeas}=70; \\
\text{Pref} = 20 \times 10^{-6}; \\
y_{\text{refscaled}}= (y./\text{Pref}); \\
\text{RMS} = \sqrt{\text{mean}(y_{\text{refscaled}}.^2)}; \\
\text{SPLmat}=20 \times \log_{10}(\text{RMS}); \quad \% \text{dB SPL in Matlab} \\
y_{\text{cal}}=c*(y_{\text{refscaled}}); \\
\text{if the HUTear toolbox is installed, it is also possible to use the function [7]:} \\
\text{ret=Pascalize}(y,70); \\
\]

2.2. Direct Implementation of Loudness Representation

The algorithm for the direct implementation is taken from the EMBSD speech quality measure. The signal is separated into frames, each one windowed with a Hanning function and the power spectral density obtained. Each power spectrum is partitioned into critical bands of width one bark, with an upper frequency limit of 3.4 kHz. In [3], Schroeder’s spreading function model is applied to include the effects of frequency masking across the critical bands. The loudness level of each critical band in units of phon is obtained using a set of equal-loudness contours taken from the literature and dB intensity values that lie in between the published contours are interpolated to get the correct loudness level [3]. These loudness levels are then converted to some using equation (1).

\[
\text{FFT}=1024; \text{NOVERLAP}=0; \\
\text{Bf}=1:18; \\
[y_{\text{xx}},f] = \text{psd}(y_{\text{cal}},\text{FFT},fs,\text{FFT},0); \\
\text{Yxx}_\text{scale}=(2.*y_{\text{xx}})/\text{FFT}; \\
[B_{\text{xx}},\text{bark}]=\text{bk_{frq02}}(\text{Bf},f,\text{Yxx}_\text{scale}); \\
C_{\text{XX}}=\text{spread_new}(Bf,B_{\text{xx}}); \\
P_{\text{XX}}=\text{dbtophon}(C_{\text{XX}}); \\
S_{\text{XX}}=\text{photon}(P_{\text{XX}}); \\
N_{\text{mbsd}}(l)=\text{sum}(S_{\text{XX}}); \\
\]

The Matlab programs are as given in [3]. However, it was found that it was necessary to make adjustments to the program dbtophon.m. First of all, in the program code a file named equal.mat is called as it holds the transcribed equal loudness contours. However, the C program version in the thesis also contains the contour values in an array, which can be copied for use with Matlab. Furthermore, the lines below, which were found to cause errors on occasion:

\[
j=1; \\
\text{while } T(i) >= \text{equalcon}(j,i) \\
j=j+1; \\
\text{if } j == 16, \text{fprintf(1,'ERROR\n')}, \text{end} \\
\text{end} \\
\text{if } j == 1, P_{\text{XX}}(i) = \text{phons}(1), \text{end} \\
\]

can be replaced with:

\[
[I]=\text{find}(T(i)<=\text{equalcon}(:,i)); \\
\text{if } \text{min}(I)==1, P_{\text{XX}}(i) = \text{phons}(1), \text{end} \\
\]

2.3. Implementations based on Zwicker’s model

Possibly the most well-known and popular model of loudness is the one proposed by Zwicker. It has formed part of an international standard [8], and has been adopted for use in a number of ITU standards on speech and audio quality. However, differences exist in the implementations.

2.3.1. Implementation of DIN 45631/ISO532B Loudness Model

This Matlab program was a direct conversion from the basic program provided in [6]. This implementation uses a filterbank of one-third-octave filters for the spectral decomposition of the signal. However, a drawback is that this yields only a rough approximation to the shape of the auditory filters and the location of their centre frequencies. The equation for the specific loudness \( N^* \) in Sone/Bark of the dB SPL sound level \( L_g \) in a one-third-octave band is given by [9]:

\[
N^* = 0.00641 \times 10^{4 \times \text{Log}_{10} \left[ \left( \frac{11 + 1.4 \times 10^8 (f_f - f_0)}{4} \right) - 1 \right]} \\
(2)
\]

The transmission of freefield sound to our hearing system through the head and the outer ear is described as attenuation \( a_g \). The excitation threshold in quiet is \( L_{E(Q)} \). Values for these two parameters are given in [8]. To run the implementation given in [6], the sequence of Matlab commands is:

\[
[y_{\text{xx}},f]=\text{PowSpec}(\text{Pref}.*y_{\text{cal}},fs,df); \\
[Y_{\text{dB}}, \text{err}]=\text{Convert2dB}(y_{\text{xx}},1); \\
\]

In this implementation, the power spectral density of the signal \( y_{\text{xx}} \) is found without the factor Pref taken into account. Furthermore, this quantity is only included after it has been converted to dB, i.e. the following line is required in Convert2dB.m:
The third-octave-band filters are generated using the code below.

```matlab
N_pseq=sum(Lx);%total loudness

for ink=1:index_last
    H=0.94833723551160*H;
end

index_first=index(1);
indice=(Ffreqs(i-1) <= f & f < Ffreqs(i));
deltaf=Ffreqs(i)-Ffreqs(i-1);
deltaz=deltaz/deltaf;
scal=(deltaf./deltaz);
PPX=PPX.*scal;

N_tot(l)=Ntot;
[Ntot]=PQLoud(E,'Basic','FFT',Fs);

E=PQspreadCB(Eb,'Basic',Fs);
X2=POQDFTFrame(ycal,len);
Eb=PQgroupCB(X2,'Basic',len,Fs);
E=PQspreadCB(Eb,'Basic',Fs);
```

2.3.3. Zwicker’s Loudness model as used in PEAQ

This is the most sophisticated of the psycho-acoustic decompositions [12]. The power spectrum of each frame is weighted by the frequency response of the outer and middle ear derived from a model. The power spectral energies are then grouped into Critical bands, spaced at 0.25 Bark. An offset is then added to the Critical band energies to compensate for internal noise generated in the ear. A triangular (in dB) spreading function is used to implement spreading in the frequency domain. The spread excitation pattern is given by \( E_{\text{sc}} \). Unlike the PSQM algorithm the values for the excitation threshold in PEAQ were computed using a model description [12], with \( \alpha \) a constant that is set to 1.07664:

\[
LX(f) = \alpha(1 - s(f) + s(f)E_{\text{sc}}(f))^{-1} - 1
\]

where, in terms of dB, the threshold index is given by:

\[
s_{\text{th}}(f) = -2 \cdot 2.05 \tan^{-1}\left( \frac{f}{4000} \right) - 0.75 \tan^{-1}\left( \frac{f}{1600} \right)
\]

and the excitation threshold is:

\[
E_{\text{sc}}(f) = 3.64 \left( \frac{f}{1000} \right)^{0.8}
\]

A complete implementation of this function is given in [13]. The variables \( X2 \) are the power spectrum of the frame, \( E_b \) is the Bark warped spectrum, \( E \) is the spectrum following the application of spreading, and \( LX \) and \( N_{\text{tot}} \) are the specific loudness and total loudness respectively. The functions named in the code below are the same as described in [13] but with the additional input parameters of signal length \( \text{len} \) and sampling frequency \( Fs \).
For each frame the excitation pattern of the signal was generated using the `AudMod` function from the HUTear toolbox [7]:

```matlab
Esig = AudMod(y_cal, model);
```

Similarly, to generate the excitation pattern at the hearing threshold:

```matlab
[CrtcLinPwrs, frqNpts, CrtcdB] = OutMidCrt2('MAP', 128, Fs); MAP = interp1(frqNpts, CrtcdB, CentFreq, 'linear', 'extrap');
for i = 1:length(CentFreq)
tones(i, :) = pascalize(sin(2 * pi * [0:frame_len-1].*CentFreqs(i)./Fs), MAP(i));
end;
```

```matlab
Ethq = AudMod(sum(tones), model);
```

Once the excitation patterns are known, the specific and total loudness can be calculated. To find the total loudness from the specific loudness, scaling is applied based on the bandwidth of the ERB filters before summing:

```matlab
N = C.* (Esig.^alpha - Ethq.^alpha); N(find(N<0))=0; EarQ = 1/0.107939; minBW = 24.7; order = 1; b = 1.019; ERBwidth = ([CentFreq/EarQ].^order + minBW^order).^(1/order); totalLoudness = sum((N.*ERBwidth).^2);```

### 3. OUTPUT CALIBRATION AND TESTING

In the cases of the MBSD loudness model, the PSQM loudness model and the Moore and Glasberg loudness model, calibration was found to be necessary. The Matlab function `lsqcurvefit.m` from the optimization toolbox was used. One of its requirements is a function name within its input arguments; using the MBSD loudness model as an example, it can be written in the form:

```matlab
function [l] = mbsd_cal(coef, S_XX)
    l = diag(sqrt(coef.*S_XX)*sqrt(coef.*S_XX)');
end;
```

where `coef` is the calibration parameter, `S_XX` is excitation used to compute the specific loudness and `l` is the total loudness.

Sinewaves of frequency 1000 Hz, 512 points in length, sampled at 16 KHz and calibrated to be [40, 50, 60, 70, 80] dB SPL were used. These should have a total loudness of \([1, 2, 4, 8, 16]\) respectively.

In the case of the MBSD loudness model `S_XX` needs to be scaled by a factor of 0.2567. For the PSQM model, `S_0` = 7.63 × 10^{-4} and \(\gamma = 0.2941\). For the Moore and Glasberg model `C = 0.0002` and `\alpha = 0.8885`.

The total loudness in Sone produced by each model for these sinewaves is given in Table 1. It can be seen from the table that none of the measures produce the exact figure for total loudness but that all are approximately close to the expected value.

**Table 1: Input Sinusoid SPL Values and Models Outputs in Sones.**

<table>
<thead>
<tr>
<th>Model</th>
<th>40dB</th>
<th>50dB</th>
<th>60dB</th>
<th>70dB</th>
<th>80dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBSD</td>
<td>1.985</td>
<td>2.551</td>
<td>4.001</td>
<td>5.933</td>
<td>7.97</td>
</tr>
<tr>
<td>DIN 45631</td>
<td>0.987</td>
<td>1.974</td>
<td>2.551</td>
<td>3.935</td>
<td>4.93</td>
</tr>
<tr>
<td>PSQM</td>
<td>0.975</td>
<td>1.985</td>
<td>4.001</td>
<td>8.007</td>
<td>15.98</td>
</tr>
<tr>
<td>PEQ</td>
<td>1.312</td>
<td>2.551</td>
<td>4.701</td>
<td>8.363</td>
<td>14.486</td>
</tr>
<tr>
<td>MooreGlas</td>
<td>0.9364</td>
<td>2.06</td>
<td>3.978</td>
<td>7.23</td>
<td>13.1</td>
</tr>
</tbody>
</table>

### 4. CONCLUSIONS

This paper has presented Matlab implementations of a number of loudness models. Furthermore, where necessary the issue of model calibration was addressed. Finally, results were presented to demonstrate the model output for sinewaves of various dB SPL levels.

### 5. REFERENCES


[2] [http://hyperphysics.phyastr.gsu.edu/hbase/sound/power.pdf](http://hyperphysics.phyastr.gsu.edu/hbase/sound/power.pdf)


[5] [http://www.nd.edu/~atassi/Teaching/ame553/Notes/Processing/index.html#KabalR2002](http://www.nd.edu/~atassi/Teaching/ame553/Notes/Processing/index.html#KabalR2002)


[7] [http://www.nd.edu/~atassi/Teaching/ame553/Notes/Processing/Sound_power.pdf](http://www.nd.edu/~atassi/Teaching/ame553/Notes/Processing/Sound_power.pdf)


[10] ITU Recommendation, P.861 *Objective Measurement of Telephone Band (300-3400Hz) Speech Codecs (PSQM)*


TIMBRAL ATTRIBUTES FOR OBJECTIVE QUALITY ASSESSMENT OF THE IRISH TIN WHISTLE

Joseph Timoney and Thomas Lysaght
Dept. of Computer Science, NUI Maynooth
Maynooth, Co. Kildare, Ireland
*tlysaght@cs.may.ie

Lorcán Mac Manus* and A. Schwarzbacher
School of Electrical and Electronic Engineering
Dublin Inst. of Tech., Kevin St, D.8, Ireland
*lorcan.macmanus@dit.ie

ABSTRACT
In this paper we extract various timbral attributes for a variety of Irish tin whistles, and use these attributes to form an objective quality assessment of the instruments. This assessment is compared with the subjective experiences of a number of professional musicians. The timbral attributes are drawn from those developed in the Timbre Model [1].

1. INTRODUCTION
The Irish tin whistle is an instrument most often associated with Irish folk music. It is an interesting instrument in that it breaks many of the norms of musical instruments. In particular, there is no consensus among musicians on the merits of more finely crafted, expensive instruments. Many of the great tin whistle players prefer to play inexpensive whistles, believing the sound to be more in keeping with the traditions of folk music rather than the ‘sweeter’ tones of the more expensive varieties [3].

In its basic form, the whistle consists of a tube (usually metal), which is either cylindrical or conical in shape. A whistle is attached to one end. Different notes can be sounded by blowing through the whistle, and using a different finger pattern on the six holes on the shaft of the instrument. The most common whistle type is in the key of D, which is also the key of most Irish folk music. Air is passed through the whistle opening and strikes the lip of the whistle as it exits the fipple-hole (See Figure 1) [4]. The air jet then rapidly switches between one of two paths: through the bore of the instrument and passing out to the atmosphere, thus setting up a series of acoustic oscillations in the shaft. The effective length of the shaft can be altered by appropriate fingering, as shown in Figure 2.

Alter the effective length of the instrument in this way changes the resonance frequency. This ensures that only the relevant fundamental tone and partials are amplified, to produce the required note. Figure 3 below shows a typical spectral magnitude plot of a D note played on a Generation D tin whistle. It shows a strong first partial, followed by a series of harmonics that roll-off at a rate of approximately 8dB/octave. As can be seen, the tin whistle has only a small number of distinct harmonics, relative to other instruments.

2. CHARACTERISING MUSICAL INSTRUMENTS
Various approaches towards the characterisation of wind instruments are possible. Acoustic impedance measurements have been applied by [5] to classical and modern flutes. In this approach, an experiment is carried out that characterises the ratio between the air pressure and (the complex) airflow rate, to give a complex impedance measure, which is a function of frequency. This acous-
tic impedance is a characteristic of the instrument that is independent of the musician. Other approaches focus on analysis/synthesis techniques which seek to parameterise the pertinent perceptual attributes of the instrument and/or its expressive qualities [1,2]. Because of advantages of cost and efficiency, in this paper, a signal processing approach is taken using the recently developed Timbre Model [1].

2.1. The Timbre Model

The Timbre Model proposes a number of different measures to characterise the envelopes of thepartials of a sound. This characterisation takes the form of a number of perceptually significant parameters, such as, spectral envelope, amplitude envelope, and noise parameters. The spectral envelope alone can be used to characterise a particular instrument and is associated with brightness and resonances of the sound [1]. For this research, in which we compare different makes of the one instrument, we choose measures associated with the spectral envelope to assess the qualities of these instruments. We are not concerned here with the expressive qualities evident in the sounds which are mainly due to the performer such as vibrato and tremolo.

Once the partial envelopes have been extracted, a number of spectral measures based on ratios of the partial envelopes are proposed. These include brightness, tristimulus1, tristimulus2, tristimulus3, irregularity and odd/even relation of the partials. As explained in [1], only two out of three of the tristimulus ratios are required. If each partial amplitude is denoted as \( a_k \), and the sum of all \( N \) partial amplitudes is given as:

\[
S_N = \sum_{k=1}^{N} a_k
\]

Subsequently, the various ratios may be calculated as follows:

\[
brightness = \frac{\sum_{k=1}^{N} k a_k}{S_N}
\]

\[
\text{tristimulus}_1 = \frac{a_1}{S_N}
\]

\[
\text{tristimulus}_2 = \frac{a_1 + a_2 + a_3}{S_N}
\]

\[
\text{tristimulus}_3 = \sum_{k=1}^{N} \frac{a_k}{S_N}
\]

\[
\text{irregularity} = \sum_{i=2}^{N-1} \left| a_i - \frac{a_{i-1} + a_i + a_{i+1}}{3} \right|
\]

Extraction of the partial envelopes is itself a complex matter. We use a dynamic time warping method to clearly identify the Attack and Release sections of each partial, thus enabling identification of the Attack, Sustain and Release phase of each note. The technique used is explained in [6].

The brightness function is the spectral centroid of the note. Generally speaking, brighter notes will be perceived to have a sweeter sound. The tristimulus measures are an auditory equivalent to the three visual primary colours. It may be seen from equations (3)–(5) that these three values add to unity; therefore one of the three is redundant. Certain instruments favour either the odd or even harmonics, giving rise to an even or jagged spectrum and this is the basis for the measures of equation (6).

2.2. Application of the Timbre Model to the Tin Whistle

Being a simple instrument, the tin whistle has relatively few partials. Indeed, our experiments have found from examination of the spectrograms of each note, that only low partials up to about the fifth or sixth are most significant. This is unsurprising, as it is commonly known that tin whistles are usually only played with other instruments that have richer tones, e.g. the fiddle, to add depth. For this reason, when calculating the tristimulus measure for the tin whistle, we use only tristimulus1 and tristimulus2. Furthermore, we include irregularity (equation (6)) rather than the odd/even measure [1] due to the small number of partials. Experiments supported this view, as measures for tristimulus3 and odd were low for all instruments and so did not illuminate significant differences in their timbres.

2.3. The Sounds

The sounds were recorded in uncompressed format in a quiet room using a standard microphone with a mini-disc recording system at 44.1 kHz (CD quality). The instruments were played by a professional Irish traditional musician. Five different tin whistles were used, the Generation D, Feádóg D, Clarks D, Hohner C, and Shaws D. A one octave scale was recorded in the key of each instrument. Four of the whistles are relatively cheap to purchase (around €5) whereas the Shaws whistle is about four times this cost.

3. RESULTS

In all of these results, the brightness, tristimulus, and irregularity parameters are calculated for each note, and then averaged over all notes in the octave for each whistle. There are therefore five plots in each section, one for each of the whistles used. It should be noted that the plots proceed from the Start of Attack to the End of Release for all notes.

3.1. Brightness Measures

Figures 4 and 5 display plots of the average brightness function for each of the 5 whistles. The brightness value during the attack is high for all whistles. However, the Generation D and Feádóg whistles display a value of above 2 for the duration of their notes. The Hohner whistle also fluctuates around 2 for its duration while the Clarks and Shaws whistles drop to between 1.5 and 2. The Generation D and Feádóg whistles, therefore, display a stronger brightness value for the duration of their notes.

3.2. Tristimulus1 Measures

The timbre of the tin-whistle is characterised by a fundamental with only a few harmonics. For this reason, we compare the tristimulus1 and tristimulus2 values for each whistle only. Tristimulus1 is plotted against tristimulus2 to show the relative strength of the fundamental compared with the first three partials for each whistle. Figures 6 and 7 illustrate these results. In Figure 6 (a), for the Generation D whistle, it can be seen that there is a good balance between the fundamental and low harmonics for the duration of the tone. This is also true of the Feádóg whistle as shown in Figure 7. The Clarks D (Figure 6 (b)) and Shaws D (Figure 6 (d)), on the other hand, show a much stronger fundamental with tris-
timulus1 dominating the plots. Finally, the Hohner C (Figure 6 (c)) exhibits a better balance but is not as good as the Generation D and Feadóg whistles.

The mean and standard deviation of the tristimulus values are shown in Table 1 for the various instruments. For this calculation, only tristimulus values in the sustain portion of each note were considered. The standard deviation results for both tristimulus measures are not markedly different for any of the instruments, indicating that this may be a characteristic of the tin whistles. The average value of tristimulus1 is higher in the Clarks and the Shaws whistle and these whistles also suffer from a lower value of tristimulus2. This indicates that there is more energy concentrated in the fundamental, relative to the other partials, and this indicates a poorer balance across harmonic amplitudes for these two whistles.

### Table 1: This table shows the mean and standard deviation of the tristimulus values for each of the tin whistles studied.

<table>
<thead>
<tr>
<th>Tin Whistle</th>
<th>$\overline{T}_1$</th>
<th>$\overline{T}_2$</th>
<th>$\overline{T}_1$</th>
<th>$\overline{T}_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation D</td>
<td>0.52166</td>
<td>0.38794</td>
<td>0.16259</td>
<td>0.15098</td>
</tr>
<tr>
<td>Clarks D</td>
<td>0.67097</td>
<td>0.25806</td>
<td>0.16074</td>
<td>0.15224</td>
</tr>
<tr>
<td>Hohner C</td>
<td>0.53509</td>
<td>0.37605</td>
<td>0.2013</td>
<td>0.1869</td>
</tr>
<tr>
<td>Shaws D</td>
<td>0.7557</td>
<td>0.14476</td>
<td>0.14867</td>
<td>0.094288</td>
</tr>
<tr>
<td>Feadóg D</td>
<td>0.45373</td>
<td>0.13269</td>
<td>0.1518</td>
<td>0.13269</td>
</tr>
</tbody>
</table>

3.3. Irregularity Measures

The average irregularity for the 5 tin whistles is shown in Figure 8. This measures the average spectral irregularity or ‘jaggedness’,
for each whistle, a feature of timbre found to be one of the most perceptually salient for instrumental tones [2].

The main observation from the plots is that both the Generation D and Feadóg whistles have a decreasing irregularity after an initial peak during the attack phase, after which the irregularity settles at around 0.5. The other three whistles have less irregularity in their attacks after which it rises to around 0.5 in the Clarks and Hohner whistles. The Shaws whistle shows a very high irregularity for the sustain portion of the note.

4. CONCLUSIONS

The timbral qualities of 5 Irish tin whistles have been analysed using spectral measures or brightness, tristimulus, and irregularity developed in the Timbre Model. The Generation D and Feadóg whistles showed the most similar results with both being distinctive for their high values for brightness, balance for tristimulus, and similar curve shapes and values for irregularity. The tristimulus results for these whistles reconfirm the results for brightness, where the influence of the low partials contributes to a centroid calculation for a brighter sound. The results also explain the duller timbre of the Clarks and Shaws whistles while the Hohner whistle is closer in brightness to Generation D and Feadóg. The similarity in irregularity results for the Generation D and Feadóg whistles would possibly relate to the stronger presence of the low harmonics whereas the results for the other three whistles possibly reflect the absence of low harmonics. Moreover, the high value in these three whistles is due to the relationship of a strong fundamental to weak harmonics especially as the tone progresses. We can conclude, therefore, that the Generation D and Feadóg whistles are the brightest and contain stronger low harmonics. These results would appear to agree with the subjective assessment of most professional tin whistle players, the majority of whom choose the Generation D whistle for performances in order to produce a clear bright sound. This would appear to indicate that the Timbre Model is a useful predictor of subjective quality. Future work intends to explore more rigorously potential applications of the Timbre Model to other instrument families.

5. ACKNOWLEDGEMENTS

The authors wish to thank Mr. Roy Galvin, a former All Ireland champion tin whistle player, for his professional assessment and playing of the tin whistle sounds recorded for this research.

6. REFERENCES


SINGLE-NOTE ORNAMENTS TRANSCRIPTION FOR THE IRISH TIN WHISTLE BASED ON ONSET DETECTION

Mikel Gainza, Eugene Coyle
Dublin Institute of Technology, Ireland
{mikel.gainza|eugene.coyle}@dit.ie

Bob Lawlor
National University of Ireland, Maynooth, Ireland.
rlawlor@eeng.may.ie

ABSTRACT
Ornamentation plays a very important role in Irish Traditional music, giving more expression to the music by altering or embellishing small pieces of a melody. Single-note ornamentation, such as cuts and strikes, are the most common type in Irish Traditional music and are played by articulating the note pitch during the onset stage. A technique for transcribing single note ornamentation for the tin whistle based on onset detection is presented. This method focuses on the characteristics of the tin whistle within Irish traditional music, customising a time-frequency based representation for detecting the instant when new notes played using single-note ornamentation start and release.

1. INTRODUCTION
A musical onset is defined as the precise time when a new note is produced by an instrument, and a musical offset is the progressive release of the note.

The onset of a note is very important in instrument recognition, as the timbre of a note with a removed onset could be very difficult to recognise. Masri [1] stated that in traditional instruments, an onset is the phase during which resonances are built up, before the steady state of the signal. Other applications use separate onset detectors in their systems, like in rhythm and beat tracking systems [2], music transcriptors [3, 4, 5], time stretching [6], or music instrument separators [7, 5].

The onset detectors encounter problems in notes that fade-in, in fast passages, in ornamentations such as grace notes, trills and fast arpeggios and in glissando (fast transition between notes) or cuts and strikes in traditional music, which are discussed in Section 3. Also, the physics of the instruments and recording environments can produce artefacts, resulting in a detection of spurious onsets. Amplitude and frequency modulations that take place in the steady part of the signal can also result in inaccurate detections.

Section 2 focuses on the existing approaches that have dealt with the onset detection problem. In Section 3 we describe the main characteristics of the Irish tin whistle and we present a method for transcribing single-note ornamentation based on onset detection, which takes those characteristics into consideration. Sections 4 and 5 present some results and a conclusion, respectively.

2. EXISTING APPROACHES
There are many different types of onsets. However, the two most common are:

1) A fast onset, which is a small zone of short duration of the signal with an abrupt change in the energy profile, appearing as a wide band noise burst in the spectrogram (see Figure 1). This change manifests itself particularly in the high frequencies and is typical in percussive instruments.
2) Slow onsets which occur in wind instruments like the flute or the whistle are more difficult to detect. In this case, the onset takes a much longer time to reach the maximum onset value and has no noticeable change in the high frequencies (Figure 2).

Figure 1: Spectrogram of Piano playing C4.

Figure 2: Spectrogram of a tin whistle playing E4.
A significant amount of research on onset detection and analysis has been undertaken. However, accurate detection and analysis of slow onsets remains unsolved.

Early work which dealt with the problem took the amplitude envelope of the entire input signal for onset detection [8]. However, this approach only works for signals that have a very prominent onset, which led to the development of multi-band approaches for giving information on specific frequency regions where the onset occurs. This was first suggested by Bilmes [9], who computed the short time energy of a high frequency band using a sliding window, and by Masri in [1], who gave more weight to the high frequency content (HFC) of the signal. However, these two methods only work well for sharp onsets.

Scheirer in [2], presents a system for estimating the beat and tempo of acoustic signals requiring onset detection. A filterbank divides the incoming signal into six frequency bands, each one covering one octave, the amplitude envelope is then extracted, and the peaks are then detected in every band. The system produced good results, however, the amount of band amplitude envelopes are not enough for resolving fast transitions between notes in non percussive onsets.

Klapuri [10], developed an onset detector system based on Scheirer’s model. He used a bank of 21 non-overlapping filters covering the critical bands of the human auditory system, incorporating Moore’s psychoacoustic loudness perception model [11] into his system. Klapuri obtained the loudness of every band peak, to combine all peaks together sorted in time, and calculated a new peak value for every onset candidate by summing the peak values within a 50 ms time window centered in the onset candidate. This approach is not appropriate for onsets that have energy in a few harmonics, because it would only produce peaks in a few bands.

Other approaches [12, 13] use phase based onset detection based on phase vocoder theory to calculate the difference between the expected and detected phase.

3. PROPOSED APPROACH

This Section is subdivided into two parts: Section 3.1 describes the most important aspects of the characteristics of the Irish tin whistle, and this knowledge is then used to develop an appropriate onset based ornamentation transcribing system.

3.1. Tin Whistle Theory

Tin whistles come in a variety of different keys. However, the most common is the small D whistle, which is used in more than 80% of Irish traditional tunes. This whistle is a "transposing instrument", which means that when it is played, the note that is heard differs from the written musical notation. For example, for the small D whistle, if a D4 note is written on the score, a D5 note sounds (one octave higher). To refer to a given note, this score notation will be used in this paper.

The small D key whistle is capable of playing in many different modes. Some of them require a half hole covering, which is not practical in many musical situations. Without half covering, the following modes that are very common in Irish Traditional Music can be played with the small D Whistle [14]: D Ionian (major scale) and D Mixolydian, E Dorian and E Aeolian (natural minor), G Ionian (major), A Mixolydian and A Dorian, and B Aeolian (natural minor).

If the tune is played in a key that requires half covering, like the F note in D Dorian, the player will change to a tin whistle that can play the mode without using half covering, like a C key Whistle. Therefore, only the following notes shown in Table 1 are considered in the presented algorithm:

![Table 1: Full covering notes for the D tin whistle.](image)

Ornamentation is very important in the Irish Traditional sound. However, it is understood in a different manner than in classical music. Ornamentation in traditional music is utilised for giving great emphasis and emotive expression to the tune by altering small pieces of a melody. On the other hand, classical music adds music expression by adding notes to the melody.

There are many different types of ornamentation in Irish traditional music: cut, strike, slide, rolls, trill, etc [14], but cuts and strikes are the ornamentation types most commonly used in Irish traditional music.

Cuts and strikes are pitch articulations: the cut is a subtle and quick lift of the finger covering its hole followed by an immediate replacement, which increases the pitch, and the strike is a rapid impact of an uncovered hole that momentarily lowers the pitch. The sound of both is very brief, and not perceived as having a discernible pitch, note or duration [14]. Therefore, they are not considered to be notes, nor grace notes, but rather are just part of the onset.

3.2. System Overview

This Section describes the different parts of the proposed single note ornamentation system. A time - frequency analysis is first required, which splits the signal into 14 frequency bands, one band per note shown in Table 1. The energy envelope is calculated for every band, which is used then to obtain the first derivative function of the envelope. To investigate the existence of onset and offset candidates, positive and negative energy changes are separated into two different functions. Then, onsets and offsets are matched together to form audio segments, and are classified in note and ornamentation segment candidates. Finally, all band segments are combined, and by applying music theory rules, the ornamentation is transcribed.

3.2.1. Time-Frequency Analysis

The audio signal is first sampled at 44100 Hz. Then, the frequency evolution over time is obtained using the Short Time Fourier Trans-
form (STFT), which is calculated using a 1024 sample Hanning window (23 ms), 50% overlap between adjacent frames and 4096 FFT length. These parameters interpolate the spectrum by a factor of 4, which is required for accuracy purposes.

\[ X(n,k) = \sum_{m=0}^{\infty} x(m+nH)w(m)e^{-j2\pi f_{\text{hop}} n} \]  

(1)

where \( w(m) \) is the window that selects a \( L \) length block from the input signal \( x(m) \), \( n \) is the frame number and \( H \) is the hop length in samples.

Every frame is filtered using a bank of 14 band pass filters. Each band covers a logarithmic note range centered at the frequency of the notes shown in Table 1.

3.2.2. Energy Envelope Extraction

The average energy is calculated in each band for each frame:

\[ E_{\text{av}}(i,n) = \frac{1}{l_i} \sum_{k_i} |X(k_i,n)| \]  

(2)

where \( X(k_i,n) \) is the filter output of band \( i \), \( k_i \) is \( i \)th frequency bin number and \( l_i \) is the band \( i \)th length in frequency bins.

This operation smooths the subband signal, limiting the effect of signal discontinuities. However, additional smoothing is still required, which is obtained by convolving the average energy signal with a 46 ms Half Hanning window. This operation performs a similar operation to the human auditory system, masking fast amplitude modulations but emphasizing the most recent inputs [2]. The smoothed signal after being convolved is denoted as \( E_{\text{av}}(i,n) \).

3.2.3. Audio Segmentation

The first order difference of the energy envelope is calculated for each band and then, the energy increases and decreases are separated into two different vectors, \( D_{\text{inc}}(i,n) \) and \( D_{\text{dec}}(i,n) \), which are then searched for onset and offset candidate peaks respectively, that reach a predetermined threshold.

Figure 4 illustrates a G\(_4\) note played with a cut when moving from note E\(_4\) (top plot). Middle and bottom left plots show the \( D_{\text{inc}}(i,n) \) of the E\(_4\) and G\(_4\) bands respectively. Middle and bottom right plots show the \( D_{\text{dec}}(i,n) \) of the E\(_4\) and G\(_4\) bands respectively.

Other multi-band energy based approaches [2, 10] used the same threshold for every band. However, this is not adequate for wind instruments such as the tin whistle, where strong amplitude modulations in high bands can have similar peak values as onset peaks in low bands.

Each note of a wind instrument has a different pressure range within which the note will sound satisfactory; this range increases with the frequency. Martin [15] stated that usual practice for recorder players is to use a blowing pressure proportional to the note frequency, thus the pressure increases by a factor 2 for an octave jump. We can then conclude that as with the note frequency, the general blowing pressure for different notes is spaced logarithmically. This also applies to the tin whistle, due to its acoustic similarity with the recorder.

In both cases, the threshold should also be proportional to the frequency and will have a logarithmic spacing. Then, the threshold for a band \( i \) will be:

\[ T_i = T \cdot 2^{\frac{x}{277}} \]  

(3)

where \( T \) is the threshold required for the band of a given note \( x \), and \( s \) is the semitone separation between the note in the \( i \)th band and the reference note \( x \).

An onset candidate is detected if:

\[ D_{\text{inc}}(i,n) = E_{\text{av}}(i,n) - E_{\text{av}}(i,n-1) \geq T_i \]  

(4)

An offset candidate is detected if:

\[ D_{\text{dec}}(i,n) = E_{\text{av}}(i,n) - E_{\text{av}}(i,n+1) \leq -T_i \]  

(5)

Then, every onset candidate \( t_{\text{on}} \) is matched to the closest offset candidate in time \( t_{\text{off}} \), where \( t_{\text{off}} > t_{\text{on}} \), to form audio segments \( S_g = [t_{\text{on}}, t_{\text{off}}] \). Next, according to time duration, the audio segments are split into note and ornamentation segments as follows:

\[ S_g = S_{\text{orn}} \text{ if } t_{\text{off}} - t_{\text{on}} < T_e \]  

(6)

\[ S_g = S_{\text{note}} \text{ if } t_{\text{off}} - t_{\text{on}} > T_e \]  

(7)

where \( T_e \) is the longest expected ornamentation time.

As can be appreciated in Figure 4, for \( T_e = 44 \text{ ms} \), \( T_2 = 100 \) and \( T_3 = 119 \), a note segment will be formed in band E\(_4\) (\( i = 2 \)):

\[ S_{\text{note}} = [D_{\text{inc}}(2,49), D_{\text{dec}}(2,49)] \]

And an ornamentation segment will be formed in band A\(_4\) (\( i = 5 \)):

\[ S_{\text{orn}} = [D_{\text{inc}}(5,7), D_{\text{dec}}(5,10)] \]

Note segments in every band are combined and sorted in onset segment time order. Next, for every onset segment, only the segment that has the strongest \( D_{\text{inc}}(i,n) \) value between the onset and offset segment time, \( t_{\text{on}} \) and \( t_{\text{off}} \) is kept. The same process is repeated for the ornamentation segments.

3.2.4. Ornamentation transcription

To decide whether a note represented in \( S_{\text{orn}} \) was played with the ornamentation represented in \( S_{\text{orn}} \) some music theory is considered:

- Single-note ornamentation in Irish Traditional music is played right on the beat, providing an accurate time for when a new note starts. Thus an ornamentation and a note segment have to come consecutively, \( t_{\text{off}}(S_{\text{orn}}) = t_{\text{on}}(S_{\text{note}}) \).
- As explained in Section 3.2, a cut increases the pitch. So, if a cut is played, the ornamentation is located in a higher

\[ T_{\text{on}}(S_{\text{orn}}) \]

\[ T_{\text{off}}(S_{\text{orn}}) \]

\[ T_{\text{on}}(S_{\text{note}}) \]

\[ T_{\text{off}}(S_{\text{note}}) \]

Figure 4: Cut ascending from note E\(_4\) to note G\(_4\) (top plot). Middle and bottom plots show the first order difference function in G\(_4\) and A\(_4\) frequency band respectively: energy increases on the left plot and the absolute value of the energy decreases on the right plot.
band than the note pitch: \( \hat{i}(S_{\text{gorn}}) > \hat{i}(S_{\text{note}}) \)
- On the other hand, the strike lowers the pitch. Thus, if a strike is played, the ornamentation is located in a lower band than the note pitch: \( \hat{i}(S_{\text{gorn}}) < \hat{i}(S_{\text{note}}) \)
- Ascending strikes to notes that are in the same register (see Table 1) cannot be played. E.g.: from E4 to G4 note.
- If the ascending strike breaks through the second register, ascending steps greater than an octave cannot be played. E.g.: from E4 to F#5 note.

In Figure 4, a cut ascending from E4 to G4 can be appreciated, where \( t_{\text{off}}(S_{\text{gorn}}) = t_{\text{on}}(S_{\text{note}}) = 7 \), and \( \hat{i}(S_{\text{gorn}}) = 5 > \hat{i}(S_{\text{note}}) = 2 \).

4. RESULTS

Three excerpts of Irish traditional music tunes were used for evaluating the performance of the presented system for detecting and transcribing single-note ornamentation (cuts and strikes). These tunes come from Grey Larsen’s book [14] with the corresponding music notation (including ornamentation), which was useful for verifying the results. In [16], an onset detector was presented and compared against the widely cited energy based onset detector approach of Klapuri in [10], thus consolidating the approach. The presented system is more challenging, since a wrong onset or offset detection in a note or ornamentation segment, would result in wrong ornamentation transcription.

The percentage of correct ornamentation detections was calculated using the following equation [10]:

\[
\text{correct} = \frac{\text{total - undetected - spurious}}{\text{total}} \times 100\%
\]  

Results are shown in Table 1, the first tune used (Tune 1 in Table 2) is a 17 seconds excerpt of “The boys of Ballisadore” [14] (p. 134), the second (Tune 2 in Table 2) is a 11 seconds excerpt of “Willy Coleman’s gig” [14] (p. 344), and the third (Tune3 in Table 2) is a 39 seconds excerpt of “Bantry Bay” [14] (p. 152).

The percentage of correct ornamentation detections was very high. However, the system encountered problems in Tune 3 detecting quick strikes used for articulating repeated notes, since the ornamentation segment was detected but not the note segment onset.

<table>
<thead>
<tr>
<th>Tune</th>
<th>Orm./notes played</th>
<th>Undetected Orm. (%)</th>
<th>Spurious</th>
<th>Correct orn. transcription (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17/42</td>
<td>0/17 = 0%</td>
<td>2</td>
<td>88.2 %</td>
</tr>
<tr>
<td>2</td>
<td>9/46</td>
<td>1/9 = 11.1%</td>
<td>0</td>
<td>88.9 %</td>
</tr>
<tr>
<td>3</td>
<td>31/135</td>
<td>5/31 = 16.1%</td>
<td>1</td>
<td>80.6 %</td>
</tr>
</tbody>
</table>

Table 2: Ornamentation transcription results

5. CONCLUSIONS

A system that transcribes single-note ornamentation for D key tin Whistle audio signals was presented. Previously, a summary of onset detector literature review was presented and the onset detector system was customised to the D key tin whistle. The algorithm was tested for real audio signals, and the results show the strength of approach addressing this challenging problem. Transcribing slides and multi-note ornamentation should be considered as an area for future research.

6. REFERENCES

EFFICIENTLY COMPUTABLE SIMILARITY MEASURES FOR QUERY BY TAPPING SYSTEMS

Gunnar Eisenberg, Jan-Mark Batke, and Thomas Sikora

Communication Systems Group
Technical University of Berlin
{eisenberg|batke|sikora}@nue.tu-berlin.de

ABSTRACT
A Query by Tapping system is a database which contains metadata descriptions of songs. The database can be scanned by tapping the melody line’s rhythm of a song requested on a MIDI keyboard or an e-drum. For the processing of queries the system computes the similarity of the query and the content inside the database by applying a similarity measure. Due to the high number of comparison processes in large databases efficiently computable similarity measures are needed. This paper presents two efficiently computable similarity measures which evaluate rhythmic properties of monophonic melodies represented in an MPEG-7 compliant manner. The usage and effectiveness is presented and evaluated with the real time capable Query by Tapping system BeatBank.

1. INTRODUCTION
Well known Music Information Retrieval (MIR) systems in the context of meta-information processing are Query by Humming (QBH) systems. QBH systems are used to search songs by humming their melody into a microphone.

Less known MIR systems are Query by Tapping (QBT) systems which allow users to formulate a query by tapping the rhythm of the song’s melody. In QBT-systems pitch information is normally not taken into account in any way. The tapping of the rhythm is performed on a MIDI keyboard or on an e-drum. Some systems even allow users to tap the melody’s rhythm as an acoustic process like clapping or knocking which is recorded and processed.

An integral component of every QBH or QBT system is the similarity measure which is an algorithm for the computation of the similarity of two database entries. This paper focuses on algorithms for the comparison of melody’s rhythms presented in an MPEG-7 compliant manner. These algorithms are primarily needed for QBT systems since they deal with rhythmic properties only. Two efficiently computable similarity measures are presented and discussed in detail in the following. Due to their efficiency they can be applied in a real-time capable QBT system.

The QBT system BeatBank is which is used for evaluation purposes operates in real-time and online. This means that after every tap made by the user, the system presents the search’s actual result list. The content of the database is saved in MPEG-7 XML documents.

2. PREVIOUS APPROACHES
Theoretical aspects regarding the comparison of two symbol strings have been discussed in several publications. These theories can be applied to the comparison of two rhythms represented in an MPEG-7 compliant manner. A very common method used to measure the similarity of two strings is the so-called Approximate String Matching method which is an application of Dynamic Programming. In addition to these theoretical approaches there are publications which present implementations of QBT systems.

McNab, Smith, and Lloyd carried out experiments on the similarity of melodies using a large database of 9600 songs. One of their goals was to find out which musical errors occur when persons are singing or humming melodies well known to them. They report that test persons generally tend to fill in extra notes or to drop notes when reproducing melodies. Their results can be generalized to the reproduction of rhythms. It is shown that simpler similarity measures lead to longer queries. Consequently, a similarity measure, which only takes rhythmic properties into account, needs longer search queries than a more complex similarity measure. Therefore the similarity measure should be efficiently computable.

Uitdenbogerd and Zobel presented different matching strategies. Some of the experiments were performed with automatically generated queries, whereas others are performed with manually generated queries. The manually generated queries were played on a MIDI keyboard and recorded by a MIDI sequencer. Their experimental setup is similar to the setup presented in this paper. One of their main results is that similarity measures which perform well with automatically generated queries do not necessarily yield good results with manually entered queries.

Kim, Chai, and Garcia carried out experiments with different melody representations. Besides the time signature the beat vector was taken into account as a rhythmic property. The proposed comparison process computes the similarity for every single beat. The experiments carried out used automatically generated random queries from the database. It could be shown that the usage of additional rhythmic information allows shorter queries for search processes.

Chen and Chen presented a string matching method which divides rhythm-strings into smaller sub-strings. With their method a tree structure is generated which is searched in greater detail. They present different matching strategies and different definitions of similarity measures.

Jang, Lee, and Yeh presented a QBT system which allows a user to clap or tap the rhythm of the melody requested and record it with a microphone. Queries are processed by an offline process in which the system extracts the notes’ durations. The similarity measure which is applied is based on Dynamic Programming.

Due to the fact that the presented algorithms use different test scenarios and different databases it is hard to compare their effi-
ciency in terms of computational effort as well as their ability to find similar patterns.

3. THE BEATBANK SYSTEM

3.1. Overview

The BeatBank system is a QBT system which is implemented as a Virtual Studio Technology plug-in instrument (VST). With an appropriate VST host, the system operates in real-time and online. This means that a new search result list is computed and presented by the system after every note entered. BeatBank is a free system which could help to define a uniform test scenario at least for the comparison of rhythms. The latest version for Windows as well as the used database can be downloaded for free at our department’s website.

3.2. MPEG-7 Description of Rhythms

The database content is represented in an MPEG-7 compliant manner, by using the Description Scheme (DS) MelodyContour. MelodyContour can be used for a loose description of monophonic melodies. It contains the Descriptors Contour and Beat.

The Descriptor Contour describes pitch information by a five-level contour and is not evaluated by the BeatBank system. The Descriptor Beat contains a vector of integers, describing the melody’s rhythm. It is formed by numbering every note with the integer number of the last full beat. The beats are being counted continuously, starting with the first beat in the first bar of the melody (see Figure 2). If the first bar is an upbeat the vector’s first entry carries implicit information on the length of the upbeat. This results from the beats which passed before the first note occurs in the upbeat.

More Information on MPEG-7 can be obtained from [11] and [12].

4. SIMILARITY MEASURES

4.1. Overview

A similarity measure represents the similarity of two rhythms as a decimal number between 0 and 1 with 1 meaning identity.

When comparing two MPEG-7 Beat vectors the goal is to find pairs of elements for matching notes. This can be achieved by means of Dynamic Programming and Dynamic Time Warping. However, a computation by these methods (e.g. the Dot-Plot) can be costly and not applicable for real-time capable systems. Therefore efficiently computable similarity measures are needed. Two efficiently computable similarity measures named Direct Measure and Wring Measure are presented in the following. Both utilize certain limitations of the MPEG-7 Beat representation.

4.2. Direct Measure

All elements of MPEG-7 Beat vectors are positive integers and every element is equal or bigger than its predecessor. These limitations enable a simplified computation of matching elements. This leads to the Direct Measure which is robust against single note failures. For two Vectors \( \mathbf{U} \) and \( \mathbf{V} \) it can be computed by the following iterative process:

- If \( u_i = v_j \) the comparison is considered a match. Increment the indices \( i \) and \( j \) and proceed the comparison.
- If \( u_i \neq v_j \) the comparison is considered a miss. Increment only the index of the vector whose element has been smaller for the next comparison.
- Continue the comparison until the last element of one of the vectors has been detected as a match or the last element in both vectors is reached.

![Flowchart of the System](image)

**Figure 1:** Flowchart of the System. The MIDI input gets transformed into an MPEG-7 compliant representation. Then a comparison with the database content is performed. The results are presented to the user.

The user interfaces the system by a MIDI-input device like an e-drum or a MIDI keyboard. While the input query is played, the taps can be acoustically monitored by loudspeakers. The search process’ results are presented continuously i.e. the result list is updated automatically after every note played by the user. The entered query gets transcribed into an MPEG-7 compliant representation and compared with the content of the database (see Figure 1). The comparison process uses the similarity measures which are described in Section 4. The rhythms inside the database originate from MPEG-7 XML files which are uploaded into the memory during the system’s initialisation.

**Figure 2:** First bars of the song “O Tannenbaum” which would be represented by a Beat vector of \([3 4 4 5]\).

**Figure 3:** Comparison path for two vectors compared by the Direct Measure. The circles mark matching elements.

[www.steinberg.com](www.steinberg.com)
[www.nue.tu-berlin.de/wer/eisenberg/beatbank.html]
An example for the pair matching is shown in Figure 3. The similarity $A$ is then computed as the following ratio with $T$ being the number of matches and $V$ being the number of comparisons:

$$A = T / V$$  \hspace{1cm} (1)

The maximum number of iterations for two vectors of length $N$ and length $M$ is equal to the sum of the lengths ($N+M$). This is highly more efficient than a computation with classic methods like the Dot-Plot which needs at least $N \cdot M$ operations [6].

### 4.3. Wring Measure

With the usage of MIR systems and especially OBT systems several predominant errors occur while users tap queries [3] [7]. Besides single note failures, especially unskilled users sometimes lose their measure and stop tapping. They restart tapping at the start of one of the succeeding bars. The test persons sometimes start tapping one or more bars too soon or too late. The Wring Measure tries to compensate these effects by focusing on the evaluation of transitions from one integer value to the next bigger one in the two vectors compliant to the MPEG-7 Beat Descriptor the Wring Measure can be computed by the following three step process:

- Transform both Beat vectors $V$ into auxiliary vectors $V'$. The elements of $V'$ are 1 if the corresponding element in $V$ is bigger than its predecessor and 0 if it is smaller. The first element of the auxiliary vector is set to 1 by default.
- Build two new Beat vectors $V^*$ from both auxiliary vectors $V'$ by accumulating all values of the auxiliary vectors’ elements whose index is less than or equal to the index of the element to be computed.
- Compare the two new Beat vectors $V^*$ with the Direct Measure.

The following equations give an example for the transformation of a Beat vector $V$ to a new Beat vector $V^*$.

$$V = [1 \ 3 \ 3 \ 3 \ 5 \ 5 \ 6 \ 7 \ 7 \ 8] \hspace{1cm} (2)$$

$$V' = [1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1] \hspace{1cm} (3)$$

$$V^* = [1 \ 2 \ 2 \ 2 \ 3 \ 3 \ 4 \ 5 \ 5 \ 5 \ 6] \hspace{1cm} (4)$$

The result of a complete comparison process using the Wring Measure can be seen in Figure 4. In this figure the same vectors as in Figure 3 are compared. Vector 2 is transformed by the Wring Measure as shown in equations (2) to (4).

5. **EVALUATION**

### 5.1. Setup

Two major experiments were carried out with the BeatBank system whose database contained 56 pop songs. The songs which were used are the same 47 songs as used by Kim, Chai, and Garcia [11] for their experiment plus nine pop songs which have already been used for prior experiments [3]. These nine songs were used to formulate queries.

The first major experiment was performed with musicians as test persons to test the similarity measures under real world conditions. Four musicians tried to tap the first four bars of each of the nine additional melodies. The persons had to listen twice to the melodies played in cycle mode. Then they had to tap the rhythm of the melody on a MIDI e-drum. The experiment had been repeated four times with different tapping devices: one hand, two hands, one drum stick, two drumsticks. Thus a total number of 144 queries was recorded and evaluated with both similarity measures.

The second major experiment was performed to test the robustness of the two similarity measures against the two sorts of errors which predominantly occur while unskilled users tap queries (loss of measure and early/late start). Therefore the 144 recorded queries were manually transformed into new queries which contain both errors described. The first two recorded bars were delayed by one bar, the second two bars were delayed by two bars thus producing a gap of one bar between the two pieces (Figure 5). This transformation injects errors which mess up the queries quite heavily. The 144 transformed queries were also evaluated with both similarity measures. With both experiments summed up both similarity measures were tested with 288 queries.

### 5.2. Results

The first major experiment shows that when the queries are evaluated with the Direct Measure 66.7% of the queries determined the correct melody as a best match (see Figure 6). For only 15.3% the requested song was listed worse than the 10th place, which is a very good result. This shows that the Direct Measure is very robust against single note failures.

When evaluated with the Wring Measure 46.5% of the queries determined the correct melody as a best match. For 29.2% the requested song was listed worse than the 10th place. Due to the amount of blur added by the Wring Measure, some correctly played melodies are evaluated wrong. This results in the worse matching outcome compared to the Direct Measure.

![Transformation of a query to reproduce the two typical sorts of errors which occur while formulating queries (loss of measure and early/late start).](http://www.media.mit.edu/~chaiwei/mpeg7/midi.zip)
The results of the second experiment differ quite much from the results of the first experiment. When evaluated with the Direct Measure 17.4% of the queries determined the correct melody as a best match. For only 32.6% the requested song was listed worse than the 10th place, which is a quite poor result. The Direct Measure is very robust against single note failures but it is vulnerable to the sorts of errors which had been manually inserted for the experiment (loss of measure and early/late start). This leads to the poor matching results.

When evaluated with the Wring Measure 38.2% of the queries determined the correct melody as a best match. For only 32.6% the requested song was listed worse than the 10th place. This is a quite good result with regards to the heavy error injection which had been performed for the experiment. The Wring Measure is more robust against the inserted errors than the direct Measure.

During the experiments it could be seen that single note failures frequently occur in several situations. Users tend to drum along and try to enhance the rhythm with additional strokes, being only poorly similar with the original. When melodies contain long sustained notes users tend to reproduce these by tapping more than just one stroke. This happened especially when they were allowed to use both hands or two sticks for the tapping. People tap more accurately stroke. This happened especially when they were allowed to use both hands or two sticks for the tapping. People tap more accurately.

Note users tend to reproduce these by tapping more than just one note. Frequently, this occurs in several situations. Users tend to drum along and try to enhance the rhythm with additional strokes, being only poorly similar with the original. When melodies contain long sustained notes users tend to reproduce these by tapping more than just one stroke. This happened especially when they were allowed to use both hands or two sticks for the tapping. People tap more accurately stroke. This happened especially when they were allowed to use both hands or two sticks for the tapping. People tap more accurately.

Frequently, this occurs in several situations. Users tend to drum along and try to enhance the rhythm with additional strokes, being only poorly similar with the original. When melodies contain long sustained notes users tend to reproduce these by tapping more than just one stroke. This happened especially when they were allowed to use both hands or two sticks for the tapping. People tap more accurately stroke. This happened especially when they were allowed to use both hands or two sticks for the tapping. People tap more accurately.

Figure 6: Mean search results of the BeatBank system for the Direct Measure and the Wring Measure for both major experiments.

The experiments carried out show that rhythm can be a powerful feature for distinguishing melodies. The introduced efficiently computable similarity measures Direct Measure and Wring Measure yield good results for comparing MPEG-7 compliant rhythms. Each measure has its strength and weaknesses but together they produce good results with linear computational effort. They can be directly used for QBH systems but also help reducing the search duration in other MIR systems. For example, they can be used in QBH systems for the fast setup of reasonable data subsets for further comparison processes with more complex similarity measures.

6. FUTURE WORK

The two presented similarity measures both produce good results in a certain scenario. When used in parallel they produce good overall search results. One probably nonlinear similarity measure needs to be developed that combines the strengths of both measures.

The limitations of rhythm as a feature for distinguishing melodies need to be further investigated with a larger database. Two melodies with a similar rhythm will always be both determined as good match candidates when one of them is played as a query. The mean query’s length which is needed for the distinction needs to be determined. This could be done in experiments like those presented by Kim, Chai, and Garcia. It will affect in which situations the melody’s rhythm can be used as a single powerful feature.

One of the next versions of BeatBank will have a simple open Application Programmers Interface (API) so that it can use third party similarity measure algorithms compiled into a dynamic link library (dll). This should help to define a uniform test scenario and allows other research groups to test their own similarity measures.

7. REFERENCES

AN OPEN SOURCE TOOL FOR SEMI-AUTOMATIC RHYTHMIC ANNOTATION

Fabien Gouyon, Nicolas Wack
Universitat Pompeu Fabra
Music Technology Group
{fgouyon|nwack}@iua.upf.es

Simon Dixon
Austrian Research Institute for Artificial Intelligence
simon@oeai.at

ABSTRACT

We present a plugin implementation for the multi-platform WaveSurfer sound editor. Added functionalities are the semi-automatic extraction of beats at diverse levels of the metrical hierarchy as well as uploading and downloading functionalities to a music metadata database. It is built upon existing open source (GPL-licenced) audio processing tools, namely WaveSurfer, BeatRoot and CLAM, in the intent to expand the scope of those softwares. It is therefore also provided as GPL code with the explicit goal that researchers in the audio processing community can freely use and improve it. We provide technical details of the implementation as well as practical use cases. We also motivate the use of rhythmic metadata in Music Information Retrieval scenarios.

1. INTRODUCTION

Rhythm is a fundamental musical feature. Anyone perceives rhythm while enjoying music listening. One can represent rhythm explicitly (i.e. write it down) in many ways, with diverse degrees of detail [1] and by different means, manually or automatically. For instance, a trained listener can transcribe a musical piece into score notation while listening repeatedly to it. He can also assign a single value for the basic tempo (in BPM). The level of detail in the representation depends on the purpose of annotation. That is, different applications require different representations [1].

In any case, it is clear that the task of associating such metadata to musical pieces would be eased by the use of additional software tools. For instance, a simple sound editor plotting waveform and spectrogram would be highly informative to a potential user. Also, a system that would compute automatically the desired metadata would obviously be relevant. However, in this case, subsequent human corrections are a must. Further, as it is clear that no automatic rhythm description system is perfect, nor human annotations are error-free, interactive systems are highly desirable. In such systems, either the user or an algorithm does a first rough analysis of (part of) the data, then the other uses the results of this analysis to orient its own analysis; the process can be iterated several times.

Very few beat annotation systems exist. In [2], Goto refers to a “beat-position editor.” This is a manual beat annotation tool that provides waveform visualisation and, for accurate annotations, audio feedback in the form of short bursts of noise added at beat times. To our knowledge, the only publically available (and open-source) beat annotation software is BeatRoot [3]. To lower the annotation effort, an automatic beat tracking algorithm is available. Interactivity resides in that the user’s corrections to the algorithm output (the beat times) are fed back as inputs to the very algorithm.

In this paper, we report on a system built upon BeatRoot as well as other open source audio processing tools, namely WaveSurfer [4] and CLAM (both part of the AGNULA GPL distribution of Linux sound software). The intent is to “take the best of several worlds”, that is, group useful functionalities of those different softwares in a single application as well as expand their scope and capabilities.

We focus on a particular kind of rhythmic annotations, the metrical structure, as it has been formalised by Lerdahl and Jackendoff in the Generative Theory of Tonal Music [5]. That is, the metadata we propose to associate to musical signals are particular time points: the beats, at several metrical levels.

Annotations can be stored locally and, when correct, they can easily be uploaded to a distant repository, e.g. a structured musical metadata database such as the MTG database [6], via the SOAP protocol.

2. APPLICATIONS

The knowledge of beats at different levels of the metrical hierarchy can be useful in many applications.

In Music Information Retrieval research, metadata associated to musical data are very useful. First of all because a database of “ground truth” metadata greatly facilitates the design of automatic algorithms for audio content description. In addition, some recent work in this field includes rhythmic information as input to systems that compute other types of metadata. For instance, beats at a metrical level can be used to determine other metrical levels [7], [8], [9]. They can also be useful as audio segment boundaries for instrument classification, such as percussion [10], [11], [12]. Other examples are the use of the metrical structure for long-term segmentations and rhythmic complexity computation. However, reliable determination of such information from automatic systems is itself a challenge. It is therefore clear that in this type of research, semi-automatic systems would be desirable.

Performers’ choices in tempo and expressive timing with respect to position in the metrical structure are very relevant to Musical Performance research. Software tools that ease the annotation of the whole metrical structure, and that generate tempo or timing deviation curves are clearly useful in this field [13].

Finally, other applications are the synchronisation or the sequencing of several musical excerpts, the determination of “looping points” for cut-and-paste operations, the application of tempo-synchronous audio effects (or visual animations), music identification, rhythmic expressiveness transformations.
3. ANNOTATING METRICAL LEVELS SEMI-AUTOMATICALLY

This Section details diverse use cases the diverse functionalities offered by the system. Some functionalities are available in WaveSurfer (common sound editing) and others have been added (beat tracking and database connections). Figure 1 gives an illustration of one possible configuration for the system and the result of the annotation of three metrical levels.

3.1. WaveSurfer functionalities

WaveSurfer [4] was initially developed as an open source software for speech research at the Department of Speech, Music and Hearing at the Royal Institute of Technology in Sweden.1 We found diverse reasons to build a rhythmic annotation system on top of it.

First of all, WaveSurfer’s typical applications are sound analysis and annotation/transcription. It therefore offers many useful functionalities such as visualisation of waveform, spectrogram, power plots, pitch contour, formant plots, etc. Panes (for transcription, data visualisation, etc.) can be dynamically added or removed, they are all time-aligned and display a running cursor while playing. Collection of panes can be saved as a configuration, that can be applied later to any other sound. This allows users to easily customize the interface. The complete sound waveform is displayed at the bottom while (optionally) a separate pane displays solely part of the waveform with additional zooming functionalities. This greatly simplifies working with large sound files. Data plots are also available, opening the way to easily visualise any relevant data, e.g. tempo curves, deviation curves. WaveSurfer has been designed in agreement with common user interface standards and it also provides intuitive keyboard shortcuts for playing, stopping, selecting regions, looping them etc. It is also possible to easily customise the look-and-feel to personal tastes.

Then, importantly, this is a multi-platform software and it handles many standard audio file formats. Last, but obviously not least, it is extensible through a simple plug-in architecture (note that it is widely used and diverse functionalities are regularly added by researchers in the audio processing community).

3.2. Diverse annotation modes

3.2.1. Manual annotations

The user can set manually the time indexes of every beat by simple left-clicks on the computer mouse. However this task is very time-consuming and error-prone. Therefore we added the possibility to specify beat times in real time while listening to the sound by simply tapping any of the keyboard’s key.

Individual beats can be subsequently adjusted with the help of the computer mouse by selecting and finely shifting them.

3.2.2. Interactive annotations

The system presents several menus in addition to WaveSurfer’s usual ones as found in transcription panes:

- ‘Compute beats’
- ‘Erase beats from cursor’

---

1http://www.speech.kth.se/WaveSurfer/
3.4. Differences with BeatRoot

The graphical interface shows important differences with BeatRoot. For instance, it is our belief that WaveSurfer’s built-in visualisation functionalities (e.g. running cursor and scrolling panes synchronized with audio playback), and its intuitive keyboard shortcuts and general look-and-feel are an enhancement of BeatRoot’s interface. Also, an important point is that graphical configurations can be defined by the user. Other relevant differentiating features are the capture of keyboard messages while listening to the audio, and the possibility to instantiate diverse transcription panes in order to annotate several metrical levels. However, it must be noted that BeatRoot also permits to annotate beats of MIDI data, which the system reported here does not permit.

4. IMPLEMENTATION

4.1. General architecture

WaveSurfer is built using Snack, a sound manipulation extension to the Tcl/Tk scripting language. Snack can be used as a scripting language, from a command prompt, WaveSurfer offers a graphical interface to it. See [4] for details. The important point here is that functionalities can be added to WaveSurfer by creating new Snack (or Tcl) commands and calling them through plugins.

In order to add beat tracking functionalities to WaveSurfer, two components are needed: an external library embedding those functionalities (with an appropriate C interface) and a plugin script to call them from WaveSurfer.

The plugin script is written in Snack and logically accounts for a command that loads the library, typically:

```
load C:/WaveSurfer/1.6/plugins/libBeatSnack.dll
```

Additional commands (calling library functions) are directly accessible from WaveSurfer transcription panes, as e.g. ‘Compute beats’, ‘Retrack’, ‘Erase beats from cursor’, etc.

On the other hand, the external library is written in C and C++ (see Figure 2). A first part of the library creates the actual Snack function called in the plugin e.g. from the command ‘Compute beats’. This part is written in C. The second part, the core beat tracking algorithm, is written in standard C++ and CLAM.

As shown in Figure 2, Interface2CLAM.h is the interface between these two parts of the library. This interface is a simple function whose input parameters are the sound samples and optionally a list of beats. Its output is a beat list (and optionally an onset list). The function definition is written in Interface2CLAM.h while its implementation is in Interface2BeatRoot.cxx.

![Figure 2: External library architecture.](http://www.oefai.at/~simon/beatroot/index.html)
to present differences with the public version of BeatRoot (the set of default parameter may differ though).

In order to account for the diverse usage modes specified above, the input to the beat tracking algorithm can be the following:

- Case 1: The sound samples.
- Case 2: The samples and some (correct) beat times.
- Case 3: The samples and a (correct) tempo value.

In all cases, outputs are beat times of the whole audio.

4.3. Database connection

In case the audio signal under annotation is part of the MTG repository (see [6]), interesting functionalities are provided.

The audio repository can be browsed from any remote web browser. It is possible to launch WaveSurfer directly from the web interface. The audio is loaded automatically together with annotations that may be available on the repository. There is a connection to the MTG database server via web services (SOAP) which allows to upload new segmentations. SOAP8 is a lightweight protocol based on XML-RPC calls. The service interface is described in a WSDL file (a superset of XML) indicating methods calls, objects, and exceptions that will be sent across the net. As all the exchange of information is made with XML (both data and control messages), SOAP allows the interaction between programs on different platforms or in different languages running anywhere on the Internet.

5. SUMMARY

In this paper, we presented a multi-platform software for semi-automatic beat annotation of audio signals. Being open source, it can be used and modified at will. It has been implemented as a plugin for the WaveSurfer sound editor and, in addition to this software’s functionality, it embeds useful functionalities from CLAM and BeatRoot. Source code and binaries are available at http://www.iua.upf.es/~fgouyon/BeatTrackingPlugin.html.

Applications for this software are manifold. Particularly in the field of Music Information Retrieval.

At the time of writing, additional functionalities are being added to this software. For instance the handling of diverse audio file formats (WaveSurfer can handle many standard format (WAV, AIFF, MP3, etc.), so does CLAM, however, we still did not add this functionality to the beat tracking algorithm). We are also working on the integration of the free “aubio” library developed by Paul Brossier at the Queen Mary University of London,1 this library provides a C implementation of powerful and fast onset detection algorithms [14]. Visualization and interactive onset detection will also be a useful feature. Finally, we are working on the addition of diverse data plots such as tempo curves.

6. ACKNOWLEDGMENTS

This work reported in this paper was partially funded by the EU-FP6-IST-507142 project SIMAC (Semantic Interaction with Music Audio Contents). More information can be found at the project website http://www.semanticaudio.org.

Thanks to Miguel Ramirez and David Garcia for support in dynamic library implementation within CLAM. Thanks also to Georgios Emmanouil, Günter Geiger, Pedro Cano, Jose Pedro Garcia, Vadim Tarasov, Markus Koppenberger, Paul Brossier and the SIMAC team in the MTG.

7. REFERENCES


A SPECTRAL-FILTERING APPROACH TO MUSIC SIGNAL SEPARATION

Mark R. Every, John E. Szymanski

Media Engineering Group, Department of Electronics
University of York, York, U.K.
mre104@york.ac.uk, jes1@ohm.york.ac.uk

ABSTRACT

The task of separating a mix of several inter-weaving melodies from a mono recording into multiple tracks is attempted by filtering in the spectral domain. The transcribed score is provided in MIDI format a priori. In each time frame a filter is constructed for each instrument in the mix, whose effect is to filter out all harmonics of that instrument from the DFT spectrum. The complication of overlapping harmonics arising from separate notes is discussed and two filter shapes that were found to be fairly successful at separating overlapping harmonics are presented. In comparing the separated audio tracks to the original instrumental parts, signal-to-residual ratios (SRR's) in excess of 20 dB have been achieved. Audio demonstrations are on the internet [1].

1. INTRODUCTION

Music separation, or more specifically, separating a number of instruments playing inter-weaving melodic lines from a mono recording, is nearly impossible to perform perfectly as mixing audio signals almost always results in a loss of information. A more achievable aim is to obtain separation of adequate quality to be useful in a number of applications. These include: audio restoration; de-mixing of old mono recordings before cleaning-up the separated instruments individually and re-mixing; re-mixing mono recordings in stereo/surround sound; structured audio coding; and some creative applications, for example an effects processor that applies an effect to a structured component of a sound rather than the whole.

The task is considered here to be a two-stage process: transcribing the mix into separate instrumental parts for which the pitch and timing of each note are found, and then performing the separation. It is conceivable that the separated results could conversely aid the transcription process, but this is not part of this implementation. As the first stage, automatic music transcription (AMT), is a demanding task in itself and the reader is referred to [2] for an account of some approaches to AMT; this research instead focuses on achieving good separation performance given the score in advance. The score is provided in MIDI format, such that a transcription of each instrumental part is available on a separate MIDI track.

To begin with, the mixed waveform is split into overlapping time frames and the DFT of the signal is computed in each frame. The pitch of a note can vary considerably over its duration, whereas a transcription of a note will most likely assign the note to a constant and discrete pitch. It was also observed that high fidelity separation was only achieved when the variation in pitch over the duration of each note was estimated accurately. Thus, for every time frame, a refinement is made of the MIDI pitches of all notes sounding in this frame. Following this, a filter is designed in the frequency domain for each instrument, whose purpose is to remove the harmonics assigned to that instrument from the spectrum. It is possible that an instrument could be playing more than one note concurrently, and in this case a filter is designed that filters from the spectrum the harmonics of each note played by this instrument. Re-synthesized separated waveforms are produced by calculating the DFT−1 of each filtered spectrum, and interpolating between time frames using an overlap-add technique.

Another approach to separating musical instruments [3] also notes the need for an accurate time-varying pitch estimate of each note, but instead takes an additive approach to re-synthesis, whereby the harmonics of each note are synthesized using oscillators whose time-varying frequency, amplitude and phase have been previously estimated in a least-squares sense. Similarly, additive synthesis has been used for the separation of harmonic sounds in [4]. Whilst these approaches may be able to produce fairly realistic synthesized sounds, some difficulty was encountered in the preliminary stages of our research in obtaining a realistic sounding residual using this method. The residual in this respect is the original mix minus the sum of separated sounds produced by additive synthesis. During this time domain subtraction, harmonics are liable to bleed into the residual unless highly accurate phase matching is achieved between the sinusoidal components of the additive synthesis model and corresponding components in the original mix.

Alternatively, one could consider the spectrum of a harmonic sound in a single time frame to consist of a sum of scaled and translated Fourier transforms of the window function centred at the harmonic frequencies, plus a residual component. This type of model is discussed for example in [5]. The separation of harmonic sources could then be achieved by removing the harmonics of each source from the spectrum by subtracting from the mixed spectrum a sum of ideal window shapes, whose amplitude, phase and centre frequencies had all been optimally calculated. On the contrary, in the approach described here, assuming for the moment that a spectral peak we are investigating contains a single harmonic, the harmonic is separated from the mixed spectrum by constructing a filter of unit amplitude across the main lobe of the spectral peak between the troughs in the amplitude spectrum on either side of the peak. Thus if the shape of the spectral peak was indeed the Fourier transform of the window function, this method would not remove the window’s side-lobes, but for example in the case of the Hamming window, the largest side-lobes are 43 dB lower than the main lobe, so they may be sufficiently small for this not to be a concern. It is fairly common to observe spectral peaks significantly higher than the noise level that do not closely resemble the shape of the DFT of the window function, even if one takes into account the distortion of their shape due to noise or residual com-
components, the average envelope of which could be interpolated from the surrounding spectrum. A possible explanation is that these distortions arise from frequency and amplitude modulations of a sinusoidal component within the time frame, or alternatively, that the modelling of instrument harmonics as slowly time-varying single sinusoids may not always be very accurate. The filter of unit amplitude removes the majority of the energy attributed to the instrument harmonic, without assuming that the harmonic conforms to a precise shape in the spectral domain. Signal-to-residual ratios of more than 30 dB have been achieved [6] when separating mixes of two simultaneous notes, and this provides some validation for using this filtering approach.

2. PRE-PROCESSING

All instrumental note samples used were in .wav format, mono, sampled at 44.1 kHz, 16 bit resolution, and all except the piano samples were recorded in an anechoic chamber. Mixed samples of 5-20 seconds in length consisting of multiple inter-weaving melodies were produced within a software sequencer such that audio and MIDI tracks for each instrument were recorded in parallel. MIDI note messages were used to trigger real audio samples, and it was possible for each instrument to be playing more than one note concurrently. The original mix was then split into overlapping time frames of length $N_{win} = 8192$ samples (186 ms), with an overlap of 87.5%. In each frame, after time weighting the signal with a Hamming window, an FFT was used to transform to the spectral domain. In each time frame, the number of simultaneously sounding notes in the original mix was found from the MIDI data. As the transcribed pitches of these notes in the MIDI data were restricted to the notes of a keyboard, and considerable pitch variations over the duration of a note are not uncommon, a pitch-refinement process was used to accurately estimate all pitches present in each frame. Each refined pitch estimate was taken to be the mean of $\{f_{jp}^1; j = 1 \ldots J\}$, where $f_{jp}^1$ is the frequency of the $j$th harmonic of pitch $p$. The harmonic frequencies $\{f_{jp}^1\}$ were found using an iterative process starting with the identification of the fundamental frequency spectral component and then searching for spectral peaks at successively higher harmonics.

An effective method for detecting prominent spectral peaks was necessary both during pitch refinement and later in the filter design. The aim was to detect all local peaks in the amplitude spectrum significantly higher than the noise floor. A frequency-dependent threshold is usually necessary to detect all harmonics, whilst keeping the number of spurious spectral peaks or noise components above the threshold to a minimum. This frequency-dependent thresholding was implemented by dividing the spectrum by $Env(f)$ where $c$ was chosen to be between 0 and 1, and $Env(f)$ is the convolution of the amplitude spectrum with a Hamming window of length $1 + N_{win}/0.4$. Local peaks were found above the threshold using a neighbourhood search. Harmonics right up to the Nyquist frequency were detected effectively using this method.

Finally, the baricentric interpolator [7] was used to interpolate the spectral peak centre frequencies to sub-bin frequency resolution. This interpolator was compared with others such as Grandke’s, Quinn’s and the parabolic interpolator, and found to be quite effective for Hamming windowed data.

3. FILTER DESIGN

The basic idea in this spectral-filtering approach to separation is that if the pitches are known of all notes present during a particular time frame, and the number of notes is not too large, then it is possible to identify most of the prominent spectral peaks uniquely with single harmonics, and to construct filters notches of unit amplitude across the width of each peak to remove the corresponding harmonic from the spectrum. A separate filter is designed for each pitch whose effect is to remove all the harmonics of this pitch from the spectrum, and the width of the notches are taken to be between the troughs in the amplitude spectrum on either side of the peak maxima. A difficulty arises when harmonics of more than one pitch are overlapping in the spectrum. This problem was resolved in [8] for combinations of two overlapping partials in a stereo mix. In our case, the sum of the filter amplitudes for all pitches, is set to unity across the width of this peak, and the shape of each filter notch is designed so that a suitable division of the energy in the spectral peak is achieved. This will be discussed in more detail below.

To begin with, it was necessary to ascertain whether each prominent spectral peak was attributable to a single harmonic or multiple harmonics. For the former case, we will refer to the spectral peak as a single-component peak and in the latter, a multi-component peak. A peak was matched to the $j$th harmonic of note $p$ if its centre frequency $f_k$ was within a fixed range $\delta$ of the predicted harmonic frequency $f_{jp}^p$, where $f_{jp}^p \approx j \cdot f_p^0$, and $f_p^0$ is the pitch of note $p$. The values of the $f_p^0$ were all deviated from exact harmonicity ($f_{jp}^p = j \cdot f_p^0$), such that if a single-component peak at $f_k$ was found to be very close to a predicted harmonic $f_{jp}^p$, then $f_{jp}^p$ would be set equal to $f_k$. In either case, the next predicted harmonic would be at $f_{jp+1}^p = f_{jp}^p + f_p^0$. This modification improved separation performance, probably due to the fact that instruments whose harmonics are slightly de-tuned are treated more appropriately, and also that any slight pitch errors would not necessarily be compounded when multiplying by $j$ to find the $j$th harmonic.

When a spectral peak was matched to more than one harmonic from separate notes, then corresponding to each note $p$ contributing to that peak, a filter notch was designed that depended on the predicted frequency and predicted amplitude of its harmonic within the peak: $f_{jp}^p$, $A_{jp}^p$, where it is implicit that $j \equiv j(p)$. The prediction of harmonic frequencies was discussed previously, and the predicted harmonic amplitudes were obtained by linear interpolation between the amplitudes of the nearest harmonics of this pitch, above and below $f_{jp}^p$, that were matched to single-component peaks. Two similar filter notch designs were tested, both achieving comparable performance after fine-tuning their parameters. The filter notches $H^p(f)$ were defined for frequencies $f$ between the troughs on either side of the peak: $f_k$ and $f_k'$. For the first design, the filters obeyed equation (1a).

$$H^p(f) = A_{jp}^p \cdot \exp \left[ -\frac{|f - f_{jp}^p|}{\sigma} \right], \forall p \in Q \quad (1a)$$

followed by a normalisation:

$$H^p(f) = \frac{\hat{H}(f)}{\sum_{q \in Q} H(q)} \quad (1b)$$

where

$$Q = \{ p ; \exists j(p) \text{ s.t. } |f_k - f_{jp}^p| < \delta, p = 1 \ldots P \} \quad (1c)$$

198 — DAFx’04 Proceedings — 198
and a suitable value for \( \sigma \) was found to be about \( 0.02 \cdot (f^0_x - f^0_y)^2 \).

For the second filter notch design, if \( \mathcal{F}_\text{win}(f) \) is the DFT of the window function truncated to frequencies between zero and the Nyquist limit, then the filters notches were designed according to:

\[
\hat{H}^n(f) = A^n_y \cdot |\mathcal{F}_\text{win}(\epsilon \cdot [f - f^n_y])|
\]

(2)

where \( 0.5 < \epsilon < 1 \), and again normalised using equation (1b) to obtain \( \hat{H}^n(f) \).

The shape of the filters designed using equation (2) is illustrated in Figure 1 for a peak composed of two overlapping harmonics, and the two resulting filtered peaks are compared with the original spectra of the individual harmonics in Figure 1.

### 4. RESULTS

The signal-to-residual ratio (SRR) has been used as a quantifiable measure of separation performance. The residual in this case is the difference between the original \( x \) and separated \( x' \) waveforms of each instrumental part. Explicitly,

\[
\text{SRR}_x(x') [\text{dB}] = 10 \log \left( \frac{\sum x_n^2}{\sum (x'_n - x'_n)^2} \right)
\]

(3)

Another measure of separation performance is the average increase in the sum of SRR’s for the \( M \) instrumental parts:

\[
\frac{\pi(x_i, x'_i, y)}{M} = \frac{1}{M} \cdot \sum_{m=1}^{M} (\text{SRR}_x(x'_m) - \text{SRR}_x(y))
\]

(4)

where \( y = \sum x_m \) is the mixed original signal, and larger values of \( \pi/M \) correspond to better separation performance.

The average SRR’s and \( \pi/M \) are presented in Table 1 for some sample mixes of two to four instrumental parts. The waveforms of the original mixes were between 5 and 20 seconds in length. The dual polyphony sample was a mix of two harmonising flute melodies, the polyphony of three corresponded to a few upbeat bars in a major key played by a mix of flute, clarinet and French horn, and the example with a polyphony of four was a rough rendition of a few bars of Barber’s ‘Adagio For Strings’ played on flute, French horn and two soprano saxophones. The audio files corresponding to these test cases have been put on the internet [1] for comparison.

A visual representation of the original, separated and residual time waveforms of each instrumental part in the mix, for the sample consisting of a mix of two flute melodies in Table 1 is given in Figure 2. For the same sample mix, the spectrograms of the original mixed sound and the separated flute parts after filtering are shown in Figure 3. One can see from this last figure a clear separation of the set of harmonics belonging to each instrument, and also note that the noise level in the separated spectrograms is of lower amplitude than that of the original mix, i.e. the noise components of the original mix have mostly gone into the residual waveform.

### 5. DISCUSSION

Although the results describe only a small selection of test cases, both the quantitative results given in Table 1 and direct comparison by listening to the original and separated audio files, show that this
A fairly straight-forward approach to music signal separation is quite successful. Mean SRR’s of between 10.4 and 23.2 were obtained in Table 1 which represents a factor of about 11 to 210 times more energy in the original un-mixed sounds $x_{in}$ than in the residuals $x_{m} - x_{in}$. These can be compared with the mean SRR’s achieved for separating mixtures of single notes in [6]. In [6], mean SRR’s of 26.0 and 18.8 dB were obtained for polyphonic of 2 and 4 respectively, as an average over many almost random sample mixes. These samples mixes were chosen by randomly selecting an instrument out of a group of 10 orchestral instrument types and then randomly choosing a pitch out of each instrument’s pitch range. In this paper, the sound examples studied consisted of instrumental parts that harmonised with each other; i.e. notes intervals such as octaves, fifths and thirds were common, making separation considerably more difficult than in random note mixtures due to the fact that many more harmonics would be overlapping in the spectral domain. This is believed to be one of the main reasons that higher SRR’s where achieved in [6]. Hence, the issue of how to separate overlapping harmonics is relevant to separating typical musical signals. It is also worth considering that notes usually contain a significant noise or inharmonic component, and given that these algorithms only attempt to remove prominent spectral peaks from the mixed spectrum, even if the harmonics of each note were perfectly subtracted from the spectrum, the maximum SRR’s achievable using this approach would be limited by the amount of inharmonic content produced by each instrument.

During listening, as expected the most noticeable differences between the original and separated sounds occur at note onsets. This is partly due to the fact that there is usually a larger inharmonic component of a note at its onset than during the sustained section. Also, the accuracy of the note timing information is an important factor in separation performance. If for example, a note actually starts sounding slightly later than the note onset time provided in the MIDI data, then it is possible that the filter corresponding to this instrument will be filtering content from the mixed spectrum in the few time frames preceding the first time frame that the note is actually present.

Lastly, we have found that the separation algorithms tend to produce interesting sounding residuals that seem to preserve the inharmonic characteristics of each instrument, for example the ‘breathiness’ of a flute or percussiveness of a piano note. There is potential for further research in finding ways of separating the mixed residual into instrumental parts and recombining these with the separated harmonic components in such as way as to produce more natural sounding results. Furthermore, these residuals may be useful in creative applications such as adding natural sounding, inharmonic instrument characteristics to synthesized sounds.

6. ACKNOWLEDGEMENTS

We would like to show our appreciation to the University of Iowa, Electronic Music Studios for providing all the instrumental samples that were used in this research.

7. REFERENCES


ABSTRACT

The software for most today’s applications including signal processing applications is written in imperative languages. Imperative programs are fast because they are designed close to the architecture of the widespread computers, but they don’t match the structure of signal processing very well. In contrast to that, functional programming and especially lazy evaluation perfectly models many common operations on signals.

Haskell is a statically typed, lazy functional programming language which allow for a very elegant and concise programming style. We want to sketch how to process signals, how to improve safety by the use of physical units, and how to compose music using this language.

1. INTRODUCTION

Imperative programming languages are the usual choice for today’s software. The currently popular CPUs conform to the imperative programming paradigm and allow a fast execution of imperative programs.

Nevertheless functional programming languages like Haskell [1,2] became valuable alternatives in the recent past. The term Functional Programming [3,4] denotes a kind of program flow that is different from the imperative one. The program flow is independent from a particular type system and from whether programs can be compiled or not. In fact Haskell can be both interpreted [5] and compiled [6].

Today functional conceptions are integrated into almost every imperative programming language. They allow for structured and safe programming. A function is a part of the program with some declared input and output. Most imperative languages allow for bypassing the input/output interface by the use of global variables and by manipulation (update) of input objects. In contrast to that Haskell disallows or at least strongly discourages that. This makes things more deterministic: If you apply a function to the same argument values it will always result in the same value. Because of this strong determination Haskell is suited for interactive computations (see the interactive mode of Glasgow Haskell [6]) but you do not have to resign static type safety as in scripting languages.

The functional approach allows features that can be imagined hardly for imperative programming languages: Lazy evaluation means that function arguments and parts of data structures are only computed if they are needed. Thus a list may contain infinitely many elements. This poses no problem since a terminating algorithm will be able to access only finitely many of them. If you represent a certain signal by a list you don’t need to worry about its length – make it infinite and it will be constructed only as far as needed for the final application.

The functional approach allows for working with functions like with other kind of data. Functions can be argument and value of other functions, so called higher-order functions. Thus, loop structures need not to be hard-wired into the language but the user can create loop structures as higher-order functions that take the loop body as argument. Now you have the combinatorial power of functions that traverse through a data structure and functions that process atomic data.

This allows for a very compact notation which reduces the need for specialised library functions or library functions that do very different things depending on type and number of the passed arguments. By the way, the latter would conflict with Haskell’s static typing and the partial application of functions.

This article shows how the features of functional programming and especially that of Haskell permit elegant programming of audio processing algorithms.

The second Section describes some of the basics of Haskell, e.g. syntax, type variables, data structures. The third Section describes some routines for plain signal processing. The fourth Section sketches the design of computations with physical quantities which improves the safety on using signal processing routines. The fifth Section presents Haskore, a system for programming music, where music can be rendered into an audio or MIDI stream eventually.

2. Haskell Basics

To become familiar with Haskell let’s get some impressions of its syntax:

> zero :: Int
> zero = 0

The first line is the signature of the constant zero. It declares the type Int for that constant, where Int denotes machine size integers. The second line defines the value of the constant.

> doubleInt :: Int -> Int
> doubleInt x = x+x

The type Int -> Int states that doubleInt is a function that maps an integer to an integer. This can be more generally formulated for any numeric type:

> double :: Num a => a -> a
> double x = x+x
The identifier `Num` is a type class and `a` is a type variable. The expression `Num a` is the context of the type declaration and is separated from it by `=>`. The signature tells us that `double` can be used for all types `a` that support certain numerical operations (to be more precise: operations of algebraic rings). If you omit the type context (i.e. `double :: a -> a`) the compiler would refuse to compile because in this example it can’t assert that the type `a` supports the `+` operator. This is a bit more structured than the overloading of operator symbols in C++.

How can one define functions with more than one argument? Strictly speaking it is not possible, but it can be simulated easily.

```haskell
add :: Num a => a -> a -> a
add x y = x + y
```

The signature may look strange but it is pure logic: The arrow `->` is a constant with value 2. One value the result is a function of one argument. E.g. the result of `(add 1)` is the increment function. Consequently, `((add 1) 2)` is a constant with value 3. Since function application is left-associative the above example can be abbreviated to `add 1 2`.

A function application that doesn’t result in a constant is called partial application. Partial applications are no magic. Essentially they defer computation until all function arguments are known. They are one of the reasons of Haskell’s concise programming style. Usually you will note that in subsequent applications of a function some arguments receive the same value all the time and others receive always different values. You should sort the arguments in the function definition according to increasing expected variability.

Data structures also follow the philosophy of deferred computation. Here the term is lazy evaluation. A list structure is defined roughly this way:

```haskell
data List a = Empty | Prepend a (List a)
```

This is a recursive definition of a singly linked list meaning: A list over type `a` is either `Empty` or a `Prepend` of type `a` with a single element of type `a` prepended. The identifiers `Empty` and `Prepend` are called constructors. The list of the numbers 1, 2, 3 would be written as

```haskell
Prepend 1 (Prepend 2 (Prepend 3 Empty))
```

The designers of Haskell decided to use `[]` instead of `Empty` and the infix constructor instead of the prefix `Prepend`. Thus a standard Haskell list can be written as `1:2:3:[]` or even shorter `1:2:3:[]`, since :: is right-associative. The common notation `[1,2,3]` is available, too. Finally, the syntax of the list type is `[a]` instead of `List a`.

With lazy evaluation we can process even infinite data structures. Let’s have a look at some infinite list like `repeat 0` which is a list consisting of infinitely many zeros. When the Haskell interpreter is asked to print the list it will actually start printing – but it will never stop. Here is another example that may convince you that lazy evaluation works:

```haskell
infiniteLoop :: Integer
infiniteLoop = 1 + infiniteLoop
bypassInfinity :: [Integer]
bypassInfinity = tail [infiniteLoop, 1, 2]
```

Calling `infiniteLoop` leads to an infinite loop. The function `tail` removes the first element from a list. Since it ignores the value of the first element it isn’t even computed and thus `bypassInfinity` results in `[1,2]` rather than an infinite loop.

A language using strict evaluation would compute the list `[infiniteLoop, 1, 2]` including its elements first and then it would apply tail to the result. Lazy evaluation works different: When you call `bypassInfinity` the run-time system starts thinking about how to obtain the first element of the resulting list. It will encounter that due to `tail` the first element of the original list is not required.

### 3. SIGNAL PROCESSING

There is already a library [7] containing many routines related to signal processing. For now let’s start with some simple examples. How to superpose two signals represented by lists?

```haskell
superpose :: Num a => [a] -> [a] -> [a]
superpose = zipWith (+)
```

When implementing `superpose` we omitted the arguments. This is a short notation that means roughly that a function is expressed in terms of another function. Here we use the function `zipWith` partially applied to the operation `+`. The function `zipWith` applies an operation to the corresponding elements of two lists. The expression `(+)` denotes the binary addition operator. The functional definition can be interpreted as: "Every occurrence of `superpose x y` is expanded to `zipWith (+) x y"."

The function `zipWith` is a function from the standard Haskell library "Prelude" as most other functions presented here. It can be used analogously for the amplification of a signal with an arbitrary envelope: `zipWith (*)`.

The static amplification is also no problem:

```haskell
amplify :: Num a => a -> [a] -> [a]
amplify v = map (*v)
```

The function `map` applies an operation to each element of the list. The notation `*v` denotes the infix operator `*` applied to only one argument, i.e. while `*` is a function with two arguments, in `*v` the first argument is fixed and thus it denotes a function with only one argument.

The function `iterate` from the standard library creates a list from interim results of an iteration:

```haskell
exponential :: Num a => a -> [a]
exponential decay = iterate (decay*) 1
```

The function `iterate` accumulates values from a list using an arbitrary accumulation function. Thus the standard library defines `sum` as `foldl (+) 0` and `maximum` similarly. We can use it to determine several volume measures of a finite signal:

```haskell
amplitude :: (Num a, Ord a) => [a] -> a
amplitude x = foldl max 0 (map abs x)
```

The function `foldl` is similar but needs no initial value for the accumulator and requires a non-empty list. The accumulator
doesn’t need to be a scalar value, any other type is also fine. That allows for a compact definition of the superposition of an arbitrary number of signals:

```haskell
> superposeMulti :: Num a => [[a]] -> [a]
> superposeMulti = foldl1 superpose
```

We defined `superposeMulti` according to its meaning, i.e. the first signal is superposed with the second one, then with the third one and so on. But the actual execution is very different: If you compute the result you will start asking for the first sample of the signal, which in turn requires Haskell to evaluate the first samples of the input signals. Accordingly, subsequent samples of the output are computed.

Now we know enough about Haskell to create a first simple instrument sound:

```haskell
> {- an oscillator
> with ‘freq’ waves per sample -}
> oscillator :: Floating a => a -> [a]
> oscillator freq = map sin (iterate (2*pi*freq +) 0)
> {- a bell sound is a sine oscillator
> enveloped by an exponential -}
> bell :: Floating a => a -> [a]
> bell decay freq = zipWith (*) (exponential decay) (oscillator freq)
```

We have realised the power of general list functions such as `iterate`, `map`, `zipWith`, `foldl1`. Note that we never cared about indices! These functions operate on lists in quite a linear manner. What about signal processing including feedback?

Haskell’s answer to feedback is recursion:

```haskell
> {- an oscillator
> with ‘freq’ waves per sample -}
> oscillator :: Floating a => a -> [a]
> oscillator freq = map sin (iterate (2*pi*freq +) 0)
> {- a bell sound is a sine oscillator
> enveloped by an exponential -}
> bell :: Floating a => a -> [a]
> bell decay freq = zipWith (*) (exponential decay) (oscillator freq)
```

By defining lists recursively we can write the solution of difference equations with the common notation of differential equations. (cf. [8]) The standard function `scanl` accumulates values from a list using an arbitrary accumulation operation, but in contrast to `foldl1` it returns a list of the intermediate results. Thus the definition

```haskell
> integrate :: Num a => a -> [a] -> [a]
> integrate = scanl (+)
```

is straight-forward. The second argument of `scanl`, which is the initial value of the accumulator, turns into the first argument of `integrate` and represents the integration constant. Using it we can numerically solve the inhomogeneous oscillation equation $y'' + c_1 y' + c_0 y = u$ with the driving force $u$ and the initial values $y(0)$ and $y'(0)$.

```haskell
> osciODE :: Num a => (a,a) -> a -> (a,a) -> [a]
> osciODE (c0,y0) (c1,y'0) u = let
> integrate x y = integrate y0 y'
> y = integrate y0 y'
> y' = integrate y'0 y''
> y'' = u .- (c0 * y .+ c1 * y')
> in y
```

Note that the apostrophe has no special meaning and is part of the identifiers. The infix operators `.+-` are introduced just for visual convenience.

At the first glance this example looks a bit like magic. It’s not obvious how the program actually solves the equation but you can verify that it computes something (i.e. there is an order of magnitude). The equation is solved by using a simple recursive function, which calls itself in each step.

Recursive filters (notion taken from [9] instead of IIR) could be implemented either as solution of difference equations or, in the tradition of imperative languages, using states. That is, the value samples of the signal are not processed independently but while scanning the signal an internal state is stored and updated. The functional programming paradigm forbids update operations. They must be implemented by inputting the current state and returning the updated state. The compiler is responsible to turn this back into update operations if possible.

We like to demonstrate this technique for a first order lowpass filter.

```haskell
> lowpass1Update :: Num a => a -> a -> a -> (a,a)
> lowpass1Update k u y1 = let
> y0 = u0+k*(y1-u0) in (y0,y0)
> lowpass1 :: Num a => a -> a -> a -> (a,a)
> lowpass1 = foldl1 lowpass1Update
> in x : lowpass1 news k us
```

The definition should be read as: An infinite echo is a superposition of the original signal and an attenuated and delayed version of itself. This is a kind of recursion which describes the data structure recursively. It relies on the lazy evaluation of data and is very common in Haskell though it is quite uncommon in other languages.

Instead of simply attenuating the signal on feedback any other processing can be applied, say a lowpass or highpass filter:

```haskell
> echoProc :: Num a => a -> a -> a -> [a] -> [a]
> echoProc time feedback x = let
> y = superpose x (delay time (feedback y))
> in y
```
Here \texttt{lowpass1Aupdate} takes the filter feedback \(k\), the current input signal value \(u_0\). The state is the previous output value \(y_0\). The function returns the new output value \(y_0\) and the updated state which is \(y_0, u_0\). The function \texttt{lowpass1A} applies the filtering process to a signal. The call to \texttt{lowpass1Aupdate} can be easily replaced by a call to every state updating function of this type.

Note that constructors like \(\text{::}\) are sort of two-way: You can not only use the colon to prepend an element to a list but by using \textit{pattern matching} in an argument like \((u:ub)\) you can also split a list into the head element \(u\) and the rest of the list \(ub\).

Because states are a common programming technique there is a data type \texttt{State}. An object of type \texttt{State s a} is essentially a function with signature \(s \to (a, s)\), i.e. a function that receives the current state and outputs some data and the updated state. Since functions can be easily constructed on the fly (in fact that is the way multi-argument functions are implemented) it is also possible to feed the update function with additional data.

\[
\begin{align*}
\texttt{lowpass1Bupdate} :& \text{:: } \text{Num a} \Rightarrow \text{a} \\
& \text{a} \to \text{a} \to \text{State a a} \\
\texttt{lowpass1Bupdate} \ k \ u_0 =
\text{let } y_1 =
\text{let } y_0 = u_0 + k*(y_1-u_0) \in (y_0,y_0) \\
& \text{in } \text{State update} \\
\texttt{lowpass1B} :& \text{:: } \text{Num a} \Rightarrow \text{a} \to \text{a} \to \text{[a]} \to \text{[a]} \\
\texttt{lowpass1B} \ s \ k \ u = \text{evalState} \\
& \text{(mapM } \text{lowpass1Bupdate } k \ u) \ s
\end{align*}
\]

The expression \texttt{lowpass1Bupdate} \(k\) is of type \texttt{a \to State a a}, i.e. a function that maps an input value to a state updata function. The function \texttt{mapM} applies this map to each input value and glues together the resulting update functions. Eventually \texttt{evalState} executes the actions beginning with state \(s\).

\section{Physical Units}

The key tool to describe natural sound phenomena is physics. So it is an obvious question if one can use physical units rather than scalar values in Haskell. Physical units provide more details and allow for more consistency checks. Certainly one can argue that units are for physics what types are for informatics.

Imagine you want to simulate an echo where the sound has to cover a distance \(s\) for returning by the acoustic velocity \(v\) sampled at a rate of \(r\). The number \(n\) of samples for the delay can be obtain from

\[
\begin{align*}
\frac{s}{v} &= 100 \text{ m} \\
v &= 330 \text{ m/s} \\
r &= 44100 \text{ samples/s} \\
n &= \frac{s \cdot r}{v} \\
&\approx 13363 \text{ samples}
\end{align*}
\]

and the correct unit of \(n\) verifies that our computation was not totally wrong.

Because of Haskell’s polymorphic type system numbers equipped with physical units can be nicely integrated into the collection of numeric types \([10]\). The type \texttt{classes} of Haskell allow the usage of infix operators like \(\times\) and \(\ast\) for custom types. Though it should be mentioned that infix operators are pure syntactic sugar making computer formulas similar to mathematical notation. The price to be paid are lots of precedence and associativity rules, more complicated syntax checking and more difficulties in understanding syntax error messages.

As in most other languages it is not possible to generate custom compiler errors. E.g. a comparison like ‘\texttt{a}’ < \(i\) is rejected by the compiler due to the type mismatch. But the compiler can’t be advised to reject expressions like \(1 \text{ m} < 2 \text{ s}\). By unfolding the function calls the compiler may even realize that the expression will always result in an error but it will translate it into a permanent runtime error rather than a compilation error.

So, what’s a physical quantity? A physical quantity is essentially a number equipped with a vector of the exponents of some base units. E.g. a force of \(14\) \(\text{N}\) can be expressed by \(14\) and the vector \((1, -2, 1, 0)\), where the vector contains the exponents of meter, second, kilogramme and coulomb respectively.

To stay independent from a specific unit system we define

\[
\begin{align*}
\text{type Unit i = FiniteMap i Int}
\end{align*}
\]

where the standard data type \texttt{FiniteMap} represents a dictionary with keys of type \(i\) and values of type \texttt{Int}. \texttt{FiniteMap} perfectly reflects the sparse structure of the exponent vector though it might seem to be somewhat overkill. For specific unit systems like the one of SI \([11]\) one would choose the type \texttt{i} to be \texttt{Int} or some enumeration. Now one can define some operations on the exponent vectors like add, subtract.

The next step is to combine a numerical value with a unit:

\[
\begin{align*}
\text{data PhysValue i a = PV a (Unit i)}
\end{align*}
\]

It means that an object of type \(\text{PhysValue i a}\) is composed of a number of type \(a\) and a unit exponent vector of type \(\text{Unit i}\). As an example let’s look at the definition of the equality relation \(==\) for this type:

\[
\begin{align*}
\text{instance (Eq i, Eq a) =>} \\
\text{Eq (PhysValue i a) where} \\
\text{(PV x xu) == (PV y yu) = x=y }\&\& xu=yu
\end{align*}
\]

This reads as: If both the type \texttt{i} and the type \texttt{a} are comparable then physical quantities constructed from them are comparable, as well. Two physical quantities are equal if and only if the numerical value and the unit matches.

The next step is to provide a type for a specific unit system. Here we gear towards the SI system of units. We define

\[
\begin{align*}
\text{data SIDim =} \\
\text{Length | Time | Mass | Charge |} \\
\text{Angle | Temperature | Information |} \\
\text{deriving (Eq, Ord, Enum, Show)}
\end{align*}
\]

and then \texttt{Unit SIDim} is the type that represents the composed units in the SI system. Further on we require a set of constants for prefixes like \texttt{kilo}, \texttt{milli}, a set of basic units like \texttt{meter}, \texttt{second}, some physical constants like \texttt{mach} (sonic velocity).

This would be enough for plain computation but it is more convenient if physical values could be also converted to strings. Decomposing a unit into common SI units requires some heuristics but it can be done in a satisfactory manner. With such a system interactive computations with physical quantities look like...
5. MUSIC COMPOSITION

In the past special purpose languages for composing music were developed \[12\]. Haskell’s syntax is so concise that there is hardly a need for a special markup language. The most famous approach for creating music with Haskell is Haskore \[13,14\]. Haskore turns Haskell into a fully programmable statically safe music description language. No extra interpreter is required.

Haskore is organised as follows:

1. The front-end is a data structure for abstractly describing music.
2. A performer function turns this data structure into a sequence of musical events.
3. Several back-ends exist that convert such a sequence into a MIDI stream \[15\], a CSound orchestra file \[17\] or into an audio stream.

This Section provides some guidance on how to set up a Music data structure. Creating a piece of music at the level of the core data structure looks like

```haskell
> cMajMelodic0, cMajHarmonic0 :: Music
> cMajMelodic0 = Note (C,0) qn [] :+:
>     Note (E,0) qn [] :+:
>     Note (G,0) qn [] :+:
>     Note (C,1) qn []
> cMajHarmonic0 = Note (C,0) qn [] :+:
>     Note (E,0) qn [] :+:
>     Note (G,0) qn [](':',
>     Note (C,1) qn []
```

where `qn` denotes the duration of a quarter note and `[]` is an empty list that could be filled with additional note attributes.

Some assisting functions may simplify writing. The functions `c, d` and so on create a Note for given octave, duration, attributes.

```
> stdNote :: t -> (t -> [a] -> m) -> m
> cMajList :: [Music]
> cMajList = map (stdNote qn) [c 0, e 0, g 0, c 1]
> cMajMelodic1, cMajHarmonic1 :: Music
> cMajMelodic1 = line cMajList
> cMajHarmonic1 = chord cMajList
```

Now you can use all programming features for the creation of music. The simplest of them is probably an infinite melody loop:

```
> cMajLoop :: Music
> cMajLoop = repeatM cMajMelodic1
```

How about an infinite loop of notes that are randomly chosen from a given set of notes?

```
> randomChoiceLoop :: RandomGen g =>
>     [Pitch] -> Dur -> g -> Music
> randomChoiceLoop ps d g =
>     let indexToNote i = Note (ps!!i) d []
>     in line (map indexToNote
>             (randomRs (0,(length ps)-1) g))
> cMajRandomLoop :: RandomGen g =>
>     g -> Music
> cMajRandomLoop = randomChoiceLoop
>     [(C,0), (E,0), (G,0), (C,1)] qn
```
The example uses the ps!!i operation which selects the ith element of the list ps and the function randomRs which generates an infinite list of random numbers of the specified range and the random number generator g.

Here is another example of programming music: We like to loop a pattern where the number of notes played increases over the time.

\[
\text{guitarChord} \text{ :: [PitchClass]} \rightarrow \text{[Pitch]} \rightarrow \text{Music}
\]

The last example is a bit less abstract: A function that computes a chord on a guitar for a chord given as list of tones. To each string of the guitar we assign the tone of the chord that is closest to the base tone of this string.

\[
\text{choosePitchForString} \text{ :: [PitchClass]} \rightarrow \text{Pitch} \rightarrow \text{Pitch}
\]

The function pitchClass converts a pitch value like C into an integer and the function trans translates an absolute pitch value.

6. OUTLOOK

Haskell’s strengths are the concise style of programming combined with static typechecking. Haskell’s weaknesses today is the low performance. On the one hand Haskell programs are a pleasure for optimisers because the compiler can clearly realise the flow of data. There are no hidden flows that can confuse the optimiser. On the other hand it’s difficult to detect input/output values that can be turned into efficient update operations. The big flexibility makes it difficult to generate efficient code for a specific application.

Even today there are several possibilities for tuning Haskell programs for efficiency but the big challenge is to achieve both elegance and efficiency. Progresses in compiler technique combined with programmers assistance may make Haskell a very valuable tool for signal processing in future.

7. REFERENCES

A COMPARISON BETWEEN FIXED AND MULTiresolution ANALYSIS FOR ONSET DETECTION IN MUSICAL SIGNALS

Chris Duxbury, Juan Pablo Bello, Mark Sandler and Mike Davies

Centre for Digital Music
Queen Mary University of London, London, UK
{chris.duxbury|juan.bello-correa}@elec.qmul.ac.uk

ABSTRACT

A study is presented for the use of multiresolution analysis-based onset detection in the complex domain. It shows that using variable time-resolution across frequency bands generates sharper detection functions for higher bands and more accurate detection functions for lower bands. The resulting method improves the localization of onsets on fixed-resolution schemes, by favouring the increased time precision of higher subbands during the combination of results.

1. INTRODUCTION

In [1] we showed that onset detection in the complex domain can offer significant improvements on basic energy-based onset detection methods. In this paper, we consider the effects of using a fixed resolution spectral analysis when compared with a multiresolution subband approach.

Subband schemes, such as those discussed in [2, 3], were proposed because different onsets may be stronger in different subbands, as well as the argument that spectral analysis in subbands more closely represents the non-linearities of human hearing. These previous subband based onset detection schemes were essentially energy based. As such, they were effective at selecting strong percussive transients, but were not as effective in detecting softer onsets, particularly at low frequencies. In [4], we presented a hybrid scheme which went some way to solving this issue by using energy in the upper subbands, and the spectral distance measure in lower subbands. This idea was extended in [5], where a Kullback-Lieber function was used in the lower subbands, which is similar in principle to the spectral distance measure. However, this work preceded the development of the complex detection function, which effectively combines the energy, and frequency (now measured using phase information) approaches. In previous results, we have shown how robust the complex domain detection function is to onset type. We shall now discuss the effects of extending the complex detection approach to a subband structure.

2. MULTiresolution MUSIC SIGNAL ANALYSIS

Window length selection is a key consideration when using an FFT based analysis. Good time resolution, implying a short window length, is needed for detecting fast changes within the signal. On the other hand, good frequency resolution, and therefore a long window length, is required for accurate location of sinusoidal components, particularly at low frequencies where small changes in frequency are heard by the listener.

There are two main approaches to solving this fixed-resolution problem. The first we refer to as multiresolution-in-time based on window switching, and the second multiresolution-in-frequency based on subband analysis. Naturally, resolutions in time and frequency are inversely proportional. Hence, in reality, both approaches are multiresolution in both time and frequency.

In time-varying multiresolution signal analysis, as described in [6], window switching techniques are used such that short analysis windows and hop sizes are used at transient frames whilst longer windows are used at more steady state regions. This is used in many audio coding schemes, as well as for transient preservation in sinusoidal modelling. The key problem with this approach is that the sharp change in frequency resolution at window boundaries makes the matching of sinusoidal components across frames problematical, suggesting it is impractical for many applications.

In the frequency-varying multi-resolution scheme, the frequency spectrum is split into subbands, and processing is performed independently on each. This allows shorter analysis windows to be used at higher frequencies whilst lower frequencies can still have the required frequency resolution to separate closely spaced sinusoids. The advantages of the basic principles are described in [7]. In the following we will briefly discuss a number of proposed methods for achieving multiresolution in frequency.

2.1. Multiresolution in frequency

In [8], an approach is proposed that uses multiple different length FFT analyses on the signal. The longer window analysis bins are used for low frequency component values, whilst the short window FFT bins are used for high frequencies. Complexity in an algorithm such as this increases linearly with the number of different length FFT analyses performed. In essence, this is a redundant analysis, although certain bins are ignored at the processing stage.

Another approach, proposed in [9], proposes the use of a twice-oversampled Laplacian pyramid structure, which is traditionally used for image compression. This has the advantage that the subbands are approximately alias-free. However, the oversampling required still leads to an increase in computation.

The simplest approach (as discussed in [8]) to multiresolution in frequency, is the use of a critically sampled constant-Q filterbank of quadrature mirror filters (QMF). QMF filters are pairs of filters, G_0 and G_1, given by:

\[ G_1(z) = G_0(-z) \]  
\[ G_1(\omega) = G_0(\omega - \pi) \]

Hence, G_0 is G_1 modulated by \( \pi \). If we rearrange this we get:

\[ G_1(0.5\pi + \omega) = G_0(0.5\pi - \omega) \]
From this we can see that the filters are a mirror image about 0.5π. A constant Q filterbank is obtained by cascading several QMF pairs, and downsampling the outputs of each pair prior to analysis (see Figure 1). If re-synthesis is required, a reconstruction filterbank must be used (due to aliasing between subbands). A variant of this scheme is proposed in [10], where a bandpass constant Q filterbank is used. However, the lack of downsampling in this scheme leads to a highly redundant representation of the signal.

A variant of this scheme is proposed in [10], where a bandpass constant Q filterbank is obtained by cascading several QMF pairs, and downsampling the outputs of each pair prior to re-synthesis. If re-synthesis is required, a reconstruction filterbank must be used (due to aliasing between subbands). A variant of this scheme is proposed in [10], where a bandpass constant Q filterbank is used. However, the lack of downsampling in this scheme leads to a highly redundant representation of the signal.

In this work we favour the use of a simple constant Q filterbank of perfect reconstruction QMF filters. This choice keeps the algorithm simple and efficient. However, any of the above algorithms could be applied equally well (depending on the particular application).

2.2. Subband onset detection

There are several examples in the literature of subband analysis for onset detection. In [2] an scheme is proposed based on individual energy analysis of subbands. This aims to mimic onset detection by human hearing. After re-scaling, the signal is filtered in a filterbank with 21 overlapping “nearly-critical” bands. The lowest 3 filters are octave spaced, whilst the remaining 18 are third-octave spaced. For each subband, onsets are characterised using the log-amplitude difference detection function. After peak-picking on each subband, the detected onsets are recombined using a psychoacoustic model, based on the loudness model of [11].

In [3], a similar scheme is proposed using the linearly spaced subbands of the FFT. During a short window around the time of an attack, a triangular shape is fitted to the energy profile of each frequency channel using a least-squares method. A detection function is derived from the maximum peak of the triangle, and its mean amplitude before and after the attack. The individual results are then aggregated across frequencies and along an uncertainty interval in time.

Both of the above onset schemes are still essentially energy based schemes, and as such, suffer from poor detection of softer note transitions, such as those of bowed strings. Further to this, they tend to under-detect softer low frequency notes, due to the slower increase in energies found at these frequencies. In the following Sections we discuss an alternative method for onset detection in the complex domain and its implementation as a subband scheme.

3. Onset detection in the complex domain

There are a number of reasons that justify combining phase and energy information for onset detection: while energy-based approaches favour strong percussive onsets and are more reliable when using high-frequency information (where energy changes are not masked by the overall energy of a sound), phase-based approaches emphasize soft, “tonal” onsets and are more robust in the lower end of the spectrum, where those tonal changes occur. In [1] a fully combined approach in the complex domain is presented and successfully tested. We will briefly explain its theory in the following.

For locally steady state regions in audio signals, we can assume that frequency and amplitude values remain approximately constant. Therefore it is clear that by inspecting changes in either frequency and amplitude, onset transients can be located. Furthermore, by predicting values in the complex domain, the effect of both variables can be considered.

Let us assume that, in its polar form, the target value for the $k^{th}$ bin of the STFT of a signal $s(n)$ is given by:

$$\hat{S}_k(m) = R_k(m)e^{j\phi_k(m)}$$

where the target amplitude $R_k(m)$ corresponds to the magnitude of the previous frame $|S_k(m-1)|$, and the target phase $\phi_k(m)$ can be calculated as the sum of the previous phase and the phase difference between preceding frames:

$$\hat{\phi}_k(m) = \text{princarg}[2\hat{\phi}_k(m-1) - \hat{\phi}_k(m-2)]$$

We may then consider the measured value in the complex domain from the STFT $\hat{S}_k(m) = R_k(m)e^{j\phi_k(m)}$, where $R_k$ and $\phi_k$ are the magnitude and phase of the current STFT frame. By rotating target and current phasors, such that $\hat{S}_k(m)$ is mapped onto the real axis, and by measuring the Euclidean distance between them, we can quantify the stationarity for the $k^{th}$ bin as:

$$\Gamma_k(m) = \left\{ R_k(m)^2 + R_k(m)^2 - 2R_k(m)R_k(m)\cos(d_{\phi,k}(m)) \right\}^{\frac{1}{2}}$$

where $d_{\phi,k}(m) = \text{princarg}[(\hat{\phi}_k(m) - 2\hat{\phi}_k(m-1) + \hat{\phi}_k(m-2))]$ is the phase deviation between target and observed phase values in a given frame. Summing these stationarity measures across all $k$, we can construct a frame-by-frame detection function as:

$$\eta(m) = \sum_{k=1}^{K} \Gamma_k(m)$$

In [1] it was shown that $\eta(m)$ is an adequate detection function showing sharp peaks at points of low stationarity while returning a smoother profile than those obtained with energy or phase-based methods (consistently outperformed on the experimental results). Also, results showed robustness for a wide range of musical signals.
4. COMPLEX-DOMAIN DETECTION IN SUBBANDS

A simple 3-level constant-Q filterbank of QMF filters is used, splitting the signal into four frequency subbands for individual onset detection analysis (as shown in Table 1).

Figure 2 shows the complex detection function calculated separately for each subband (lower subbands area at the bottom of the figure). In this case, the hop size and window size are set such that they have the same resolution across all subbands. It can be seen that a greater number of onsets are detected in the lowest subband. This is because some of the softer notes have weak energy in the higher subbands. In this case, thresholding of the lowest subband would achieve high accuracy in onsets detected.

However, the localisation of detected onsets is often as important as the number of detections. We know that at higher frequencies, energy changes are sharper, particularly at hard (percussive) onsets. Therefore, the higher subbands are useful for onset localisation, suggesting the use of a shorter analysis window. This is shown in Figure 3, where the multiresolution scheme is such that the hop and window sizes are fixed in terms of samples, but the downsampling in the filterbank leads to a multiresolution analysis in time.

From these examples we can conclude that the lower subbands have a tendency to be robust to noise, and therefore produce accurate results in terms of numbers of detected onsets. Unfortunately, the long windows needed for analysis at these frequencies lead to poor resolution, and therefore poor onset localisation. Conversely, higher subbands tend to be more prone to noise and miss-detection of very low notes, but produce better localised results. In the following we will try use this observations to appropriately combine information from all subbands.

5. COMBINING SUBBAND INFORMATION

The proposed scheme for combining onset information across subbands, is a re-implementation of our previous work in the area [4].

One possible solution to the combination problem, is to generate an overall detection function as the sum of all subband functions, such that:

\[
\eta_{all}(m) = \eta_1(m) + \eta_2(t) + \eta_3(t) + \eta_4(t)
\]  

where the sub-index corresponds to subband numbers as shown in Table 1. Applying peak-picking to this function does not solve the problem of weaker onsets remaining undetected, even though they may be strong within a certain frequency band (due to masking by stronger onsets in the overall function). An alternative is to peak-pick onsets on each subband, and then combine the results.

5.1. Peak-picking

For the detection of onsets, first our algorithm normalises and DC-removes the obtained subband detection functions. This is to facilitate the thresholding of the detection functions by emphasising the characteristics of similar peaks and making them more uniform, not only within a signal, but between a number of different signals.

Then, the median filter is used to obtain an adaptive threshold curve \( \delta_t(m) \). This is calculated as the weighted median of an \( H \)-length section of the detection function around the corresponding frame, such that:

\[
\delta_t(m) = \delta + \text{median} \eta(k_m), k_m \in [m - \frac{H}{2}, m + \frac{H}{2}]
\]  

\( \delta \) is a constant value with a large influence on the number of good and false onset detections. [12] demonstrated the effectiveness of the median filter for the thresholding of peaks in detection functions generated from music.

Finally, local maxima above the calculated threshold are simply selected as the onsets of a particular subband.

5.2. Combining detected onsets

After peak-picking, each output of the subband scheme produces a list of onset positions and magnitudes. In many cases, onsets
are present in multiple subbands. Let us define $P(m)$ as a signal-length vector containing one at onsets and zeros elsewhere, such that:

$$P_{all}(m) = P_{n1}(m) + P_{n2}(m) + P_{n3}(m) + P_{n4}(m)$$  (11)

where the sub-index refers to the subband detection functions or to the combination of them “all”. When onsets appear in more than one subband, they will tend to be slightly misaligned (due to the multiresolution nature of the scheme). Thus, a short window $k_m$ of 50ms is taken, such that only one onset is allowed per window. Selection of onsets within this window is done by weighting the output of each subband:

$$P_{all}(k_m) = \alpha P_{n1}(k_m) + \beta P_{n2}(k_m) + \gamma P_{n3}(k_m) + P_{n4}(k_m)$$  (12)

such that $k_m \in [m - 25ms, m + 25ms]$. $\alpha$, $\beta$, and $\gamma$ are weighting terms such that:

$$\alpha > \beta > \gamma > 1$$  (13)

This approach is adopted rather than position averaging to favour the use of onsets from higher-frequency subbands (thus improving time resolution), while increasing the number of correct detections with information from the lower subbands (used when detections are not obtained in the higher bands). A second reason for using a weighting scheme of this nature is that it may be tuned so that only ‘hard’ (percussive) or ‘soft’ (tonal) onsets are selected.

### 6. COMPLEX SUBBAND DETECTION RESULTS

In order to test the effectiveness of the subband based complex domain detection compared to the fixed resolution scheme, detection results were tested on recordings of a MIDI-controlled acoustic piano. This test data was preferred over hand-labelled music files (as used in [1]), as localisation of onsets is at the core of our discussion, and the hand marked data proved to lack the accuracy for this. Let us define a measure of onset detection accuracy as given by:

$$\text{Accuracy} = \frac{\text{Total} - \text{Missed} - \text{Bad}}{\text{Total}} \times 100$$  (14)

where Total represents the total number of correct onsets, Missed is the number of missed detections, and Bad is the number of bad detections. We measured the accuracy of onsets detected as the correct detection analysis (CDA) window varies in size. This window is the acceptable distance between the measured onset, and the true onset. Hence, as the window is short, only well localised onsets will be labelled as good detections. As the window increases, less accurate onsets will be accepted as good detections.

From the results in Figures 4 and 5, it can be seen that the multiresolution scheme outperforms the fixed window complex domain onset detection for short CDA window lengths. This is in line with the hypothesis that a multiresolution scheme will improve onset localisation. However, for longer CDA window lengths, the multiresolution scheme under-performs. This is due to a higher number of bad detections, as illustrated by the right side of Figure 6. This increase in bad detections is brought about by the noise introduced by the upper subband detection functions, as can be seen in the upper subbands of Figure 3.

![Figure 3: Multiresolution window-length subband complex detection functions. The hop size corresponds, in (e) and (d), to 11.6 ms (256 samples at 44100 kHz sampling frequency) for the lower subbands - (e) and (d) - , 5.8 ms (128 samples at 44100) for the second subband (c), and 2.9 ms (64 samples at 44100) for the highest subband (b). Window lengths are such that in all subbands a 50% overlap is used.](image)

![Figure 4: Percentage accuracy comparison of localisation of detected onsets complex detection and multiresolution complex detection.](image)

### 7. CONCLUSIONS

Fixed-scale complex-domain onset detection is a robust and efficient method that successfully incorporates energy and phase
In this paper we present a simple subband scheme for complex-domain onset detection. The approach uses a constant-Q filterbank of QMF filters followed by complex-domain onset detection on each of the resulting bands. Experiments show that by using variable time resolution across frequency bands we can improve on the localisation of onsets at higher subbands while relying on the high detection rates returned by lower subbands.

The algorithm for the combination of subband onset detections favours the use of the onset times from higher subbands. Results show that for short comparison windows (CDAs), the proposed scheme improves detections on the fixed resolution complex-domain onset detection. Conversely, for longer comparison windows, the fixed-resolution approach shows more robustness, mostly due to the over-detections introduced by high-frequency noise into the subband scheme. Therefore, the multiresolution method proves an alternative for a range of music applications where the precision of the detection is of great importance.

8. REFERENCES

PIANO TRANSCRIPTION USING PATTERN RECOGNITION:
ASPECTS ON PARAMETER EXTRACTION

Luis I. Ortiz-Berenguer
Universidad Politécnica de Madrid, Spain
lortiz@diac.upm.es

F. Javier Casajús-Quirós, Marisol Torres-Guijarro,
J. A. Beracoechea
Dept. Signals, Systems and Radiocommunications
Universidad Politécnica de Madrid, Spain

ABSTRACT
A method for chord recognition for piano transcription has been previously presented by the authors. The method presents some limitations due to errors in parameter extraction carried out during the training process. Parameter extraction of piano notes is not as straightforward as sometimes can be thought. Spectral components detection is necessary but not enough to obtain accurately some note parameters. The inharmonicity coefficient B is one of the parameters that are difficult to evaluate. The obtained value of B is different for every partial used to calculate it, and sometimes, these differences are high. Tuning with respect to tempered scale is another important note parameter. The problems arise when we try to measure the tuning of a note belonging to octaves 0 or 1, because the fundamental is radiated by the soundboard with a very low level and, therefore, it is not captured by the recording microphone and cannot be measured.

A method to avoid these drawbacks is presented in this paper, including an explanation of the basis.

1. INTRODUCTION
A method for chord recognition for piano transcription has been previously presented by the authors. The method is based on spectral pattern matching using a set of spectral patterns generated by a physical model of the piano. The identification of several notes belonging to a chord is performed using iterative note detection with spectral subtraction. Both the spectral patterns and the subtraction masks are generated using a physical model of the piano. The proposed model makes use of several parameters in order to calculate patterns and masks. These parameters are obtained by a training process, using only a few notes [1].

Some of the limitations of the chord recognition algorithm have to do with the fact that subtraction masks do not fit well the actual spectrum of the piano note. The reason for this drawback is related with some degree of imperfection during parameter extraction that is carried out during training process.

Piano notes have two main characteristics which are unusual in typical musical modeling. They are inharmonic (its partials are not harmonically related) and they are not exactly tuned to tempered scale.

Moreover, the lower octave notes have not significant radiation of its fundamental frequency, so it is not present on recorded signals using microphones.

This paper will present some aspects and solutions for extracting, the more precisely possible, the following parameters: inharmonicity coefficient B and fundamental tuning. Both parameters are essential for note identification and for spectral subtraction used in piano chord recognition [2].

2. INHARMONICITY COEFFICIENT
Inharmonicity appears due to string stiffness. As a result, every partial has a frequency that is higher than the corresponding harmonic value. Moreover, the higher the partial order, the higher the separation from the harmonic value.

The frequencies are calculated using [3]:

\[ f_n = n f_0 \sqrt{1 + n^2 B} \]  \hspace{1cm} (1)

where \( f_0 \) is the fundamental frequency for a flexible string (without stiffness) with hinged ends and the “inharmonicity coefficient” \( B \) is defined as [3]:

\[ B = \frac{E \pi^4 d^4}{64 L T} \]  \hspace{1cm} (2)

where \( E \) is the Young’s Module, \( d \) is the string diameter, \( L \) is the string length and \( T \) is the string tension.

It can be seen that the fundamental frequency of stiff strings differs from that of flexible strings. We can obtain:

\[ f_n = n f_0 \sqrt{1 + n^2 B} \]  \hspace{1cm} (3)

which is very useful since \( f_0 \) cannot be directly measured in actual strings, because they always have stiffness and we do not know the a priori value of \( B \).

Apparently, extracting the value of \( B \) during the training process is as simple as measuring a pair of partial (e.g. \( f_1 \) and \( f_0 \)) and calculating:

\[ B = \frac{\delta - 1}{n^2 - \delta} \]  \hspace{1cm} (4)

where

\[ \delta = \left( \frac{f_1}{f_0} \right)^2 \]

However, the results are different depending on the selected partials. This is due to the fact that frequency values of partials are not only affected by string stiffness, but are also affected by bridge impedance (i.e. soundboard impedance).
2.1. Soundboard induced inharmonicity: $I_{SB}$

The previous equation from Fletcher was obtained considering both ends of the string hinged. Hinged boundary conditions allow the end of the string to have slope but not to move, so the impedance of the string support is infinite. But actual bridges present finite impedance controlled by the soundboard impedance. The soundboard impedance is a function varying with frequency, controlled by the joint effect of the modes. The effect of the soundboard impedance on the string vibration frequency is such that if the string tries to vibrate with a frequency above a resonant frequency of the soundboard, the resulting frequency is even higher. If string tries to vibrate with a frequency below a resonant frequency of the soundboard, the resulting frequency is even lower [4][5].

So, the effect of the soundboard seems to be to prevent the string to vibrate with a frequency equal to a resonance of the soundboard. The exception is when the string tries to vibrate with a frequency coincident with a soundboard resonance. In that case, due to the high resistive value of the soundboard impedance, the soundboard has no effect on the string frequency. Actually, the partial frequencies are modified by the effect of the soundboard impedance. We have called this variation “Soundboard induced Inharmonicity” $I_{SB}$ so the frequency of any partial must be modeled as:

$$f_n = f_n I_{SB}(\tilde{f}_n)$$

(5)

where

$$\tilde{f}_n = n f_0 \sqrt{1 + n^2 B}$$

and then, substituting $f_0$:

$$f_n = n f_0 \sqrt{1 + n^2 B I_{SB}(\tilde{f}_n) + 1 I_{SB}(f_0)}$$

(6)

The difficult task of modeling soundboard impedance as well as the impossibility of knowing the soundboard characteristics from piano recordings, make it impossible to evaluate $I_{SB}$ corresponding to any partial of any note.

Previous studies [6][7] show that the value of $I_{SB}$ decreases with partial order and note(for the same kind of impedance behavior, i.e. above or below a soundboard resonance), and is not just a function of frequency (see Figure 1). So, the fundamental of lower notes are very much affected by $I_{SB}$ whereas the higher partials are less affected.

2.2. Modified calculation of $B$

Taking into account equation (6), the value of $B$ that is actually calculated from the measured partial’s frequencies is:

$$B = \frac{\delta - \varepsilon}{\varepsilon n^2 - \delta^2}$$

(7)

where

$$\delta = \left( \frac{f_0}{n f_n} \right)^2$$

and

$$\varepsilon = \left( \frac{I_{SB}(\tilde{f}_n)}{I_{SB}(f_0)} \right)^2$$

As $\varepsilon$ cannot be evaluated, the value of $B$ has error, except if $\varepsilon$ is near 1 in which case the equation (3) can be still used. From Figure 1 it is evident that $\varepsilon$ cannot have a value near 1 if we use the fundamental as a reference for the calculation of $B$. If we want to use as a reference one partial different from the first, we have to rewrite equation (6) as:

$$f_n = n f_0 \sqrt{1 + n^2 B I_{SB}(\tilde{f}_n) + 1 I_{SB}(f_0)}$$

(8)

where $m$ is the order of the lower partial used in calculation (previously $m$ was 1). Equation (7) becomes:

$$B = \frac{\delta - \varepsilon}{\varepsilon n^2 - \delta^2}$$

(9)

where

$$\delta = \left( \frac{mf_n}{n f_m} \right)^2$$

and

$$\varepsilon = \left( \frac{I_{SB}(\tilde{f}_n)}{I_{SB}(f_m)} \right)^2$$

If partials $m$ and $n$ are correctly selected, $\varepsilon$ can be considered nearly 1 and $B$ can be calculated with a low error using the equation:

$$B = \frac{\delta - 1}{n^2 - m^2 \delta}$$

(10)

where

$$\delta = \left( \frac{mf_n}{n f_m} \right)^2$$
Figure 2 shows the calculated values of $B$, using equation (10), for note A0 from a Steinway & Sons grand piano (approx. 2.9 m. long). The actual value is about $2 \times 10^{-4}$. It can be seen that some of the obtained values of $B$ are negative, which is evidently erroneous, because it is physically impossible. It is evident that $\varepsilon$ is not always near 1. It is important to note that the use of two consecutive partials (i.e. $m = n - 1$) must be avoided. It also must be avoided the use of nearest partials except if they are very high order partials. After several tests, using different notes, we have concluded that for lower octaves, $n$ must be selected between 14 and 20 and $m$ must be selected between 1/2 and 2/3 of $n$ (that is an intermediate partial order).

This selection of calculation partials is a bit more problematic in octaves 3 to 5 where the number of available partials decreases. And it is especially critical in the case of the higher octaves 6 to 8.

2.3. $B$ calculation on octaves 6 to 8.

Notes belonging to octaves 6 to 8 have two problematic characteristics: only two partials have enough level to be accurately measured and the notes always present a high degree of non-linearity. The first issue is not very important due to the fact that $I_{\delta \theta}$ tends to be very little noticeable at those frequencies above C6 with little difference between fundamental and second partial. The non-linearity is the main problem, because the second harmonic of the fundamental appears very close to the second partial, so they almost cannot be distinguished, except if values of $B$ are high enough.

Figure 3 shows the second partial “zone” of note C7, for which the $B$ value allows us to distinguish the second harmonic from the second partial. It can be seen that some of the spectral peaks are positioned at twice the fundamental frequency, so they are not the second partial but an IM (InterModulation) product (i.e. 2nd harmonic).

It is also very interesting to notice that the second harmonic has higher level than second partial. This adds even more problems to parameter extraction. The non-linearity effects have been modeled using InterModulation products (IM) [8]. Two IM products appear around the second partial. Those products are separated from the 2nd partial an amount that can be calculated depending on the value of $B$ coefficient (Figure 4). As the $B$ value is not known, but is being measured, a bounded value of the separation must be approximated in order to carry out the correct parameter extraction.

It is necessary to measure precisely the second partial in order to calculate B coefficient, but this is not an easy task. The several
spectral peaks must be measured and the second harmonic values have to be discarded.

3. TUNING OF THE FUNDAMENTAL

Musical instruments tend to be tuned according to tempered scale. However, in the case of piano, inharmonicity establishes that the second partial of a note is slightly above the fundamental of the note one octave higher. If tempered tuning is performed, an audible beating that is not comfortable will be produced. This effect makes it necessary to tune the piano so that the fundamentals of higher notes are slightly above their tempered values, in order to be coincident with the second partial of the note one octave below [9]. In this case, beating is avoided. This can be expressed using:

\[ f_{i, i+12} = 2 f_{i, i} \frac{\sqrt{1 + 4B_i}}{\sqrt{1 + B_i}}. \]  

(11)

where \( i \) is the number of the note and \( i+12 \) is the number of the note one octave above.

As tempered tuning is not used, the tuning of every note can be expressed as:

\[ f_{i, i} = A \cdot f_{1, i}, \]  

(12)

where \( f_{1, i} \) is the tempered value of the fundamental of note \( i \), and \( A \) is the “Tuning Factor”. This tuning factor used to be called simply “Tuning”.

The actual tuning process begins with note A0, which is tuned to 440 Hz (sometimes is tuned to 442 or even 444 Hz). The remaining notes of octave 4 are tuned according to intervals, but for simplicity, we are going to consider that all the notes of octave 4 are tuned to their tempered value. With this assumption, Figure 5 shows the value of “tuning factor” calculated using equation (11). It can be seen that the values calculated for octaves above octave 4 are very coincident with the values of the Railsback curve [10].

The Railsback curve is an average of several measures of actual piano tunings (Figure 6). It can be noticed that the Railsback curve for octaves below octave 2 differs from the results obtained considering the \( B \) coefficient. In conclusion, inharmonicity does not justify the tuning factor for lower octaves. The effect of \( I_{SB} \) must be considered again to explain this difference.

3.1. Effect of \( I_{SB} \) on tuning

The condition for tuning can be rewritten as:

\[ f_{i, i+12} = 2 f_{i, i} \frac{\sqrt{1 + 4B_i} I_{SB}(f_{i, i})}{\sqrt{1 + B_i} I_{SB}(f_{i, i})}, \]  

(13)

where

\[ f_{i, i} = f_{1, i} \sqrt{1 + B_i}, \]  

\[ f_{i, i+12} = 2 f_{1, i} \sqrt{1 + 4B_i}. \]  

This introduces some changes in the tuning values for octaves above octave 4, but they are not very important. But for octaves below octave 4, the tuning process is the reverse, so note \( i+12 \) has been previously tuned and after, the note \( i \) has to be tuned. Then, the equation for tuning lower octaves is:

\[ f_{i, i} = f_{i, i+12} \frac{\sqrt{1 + B_i} I_{SB}(f_{i, i})}{\sqrt{1 + 4B_i} I_{SB}(f_{i, i})} \]  

(14)

For octaves 0 to 2, where the values of \( B \) range between \( 10^{-8} \) and \( 10^{-7} \), the term depending on \( B \) only justifies a tuning factor of up to \( -3 \) cents. However, for those octaves, the soundboard presents only a few resonances that are separated and then, the values of the term depending on \( I_{SB} \) can be very high. This term can justify tuning factors of up to \( -40 \) cents. These results are coincident with Railsback curve.

Figure 5: Calculated tuning considering inharmonicity coefficient \( B \).

Figure 6: Railsback curve and some piano tuning measured by Martin and Ward [10]. Note the great values of tuning factor (deviation respect to tempered tuning) at higher octaves (with high \( B \) values) and also at lower octaves (with low \( B \) values).
4. CONCLUSIONS

Evaluation of the inharmonicity coefficient $B$ requires to measure the frequency of the partials and to select the correct partials to be used in the calculation. Lower partials are very affected by the soundboard and their use must be avoided. Due to the fact that the frequency of the partials may be increased or decreased by the effect of soundboard, several values of $B$ must be measured using several partials and the mean value of them can be considered to be the string $B$ value.

Measuring $B$ in higher octaves can be done using only the second and first partials. Soundboard effect is almost negligible but care must be taken in order to avoid the error of measure the second harmonic (non-linear product) instead of the second partial.

Tuning measure is almost straightforward for octaves above 3, but it can be difficult for lower octaves where the fundamental has very low level due to radiation limitations. In these cases, the second partial may be used as reference and the fundamental can be calculated using the $B$ value previously determined and some approximation to the value of the ratio of $I_{SB}$ in equation (6).

5. REFERENCES


ON FINDING MELODIC LINES IN AUDIO RECORDINGS

Matija Marolt

Faculty of Computer and Information Science
University of Ljubljana, Slovenia
matija.marolt@fri.uni-lj.si

ABSTRACT

The paper presents our approach to the problem of finding melodic line(s) in polyphonic audio recordings. The approach is composed of two different stages, partially rooted in psychoacoustic theories of music perception: the first stage is dedicated to finding regions with strong and stable pitch (melodic fragments), while in the second stage, these fragments are grouped according to their properties (pitch, loudness...) into clusters which represent melodic lines of the piece. Expectation Maximization algorithm is used in both stages to find the dominant pitch in a region, and to train Gaussian Mixture Models that group fragments into melodies. The paper presents the entire process in more detail and provides some initial results.

1. INTRODUCTION

With the recent explosion of researches in computer music and especially in the field of music information retrieval, one of the problems that remain largely unsolved is the extraction of perceptually meaningful features from audio signals. By perceptually meaningful, we denote features that a typical listener can perceive while listening to a piece of music, and these include tempo and rhythm, melody, some form of harmonic structure, as well as the overall organisation of a piece.

It is clear that a set of tools that could handle these tasks well would be useful in a variety of applications that currently rely on symbolic (i.e. MIDI) as opposed to audio data. Such tools would bridge the gap between a large number of researches made on parametric (MIDI) data that amongst other include similarity measures, estimation of rhythm, GTTM decomposition and also query by example searching systems, where large musical databases could be made available, tagged with information extracted from audio. Audio analysis, learning and compositional systems could also make use of such information.

An overview of past researches shows that techniques for tempo tracking in audio signals are quite mature; several tools (i.e. [1]) are available for use, some of them work in real-time. Most have little problems with modern pop styles with small variations in tempo, while tracking an expressive piano performance usually still causes headaches to algorithms or their authors. Rhythmic organisation is already a harder problem, as it has more to do with higher level musical concepts, which are harder to represent [2]. A promising approach to finding harmonic structure in audio signals has been presented by Sheh and Ellis [3].

Our paper deals with extraction of melodic lines from audio recordings. The field has been extensively studied for monophonic signals, where many approaches exist (i.e. [4, 5]). For polyphonic signals, the work of several groups is dedicated to complete transcription of audio signals, with the final result being a score that represents the original audio ([6, 7, 8]). Algorithms for simplified transcriptions, like extraction of melody, have been studied by few, with the notable exception of the work done by Goto [9].

Our work builds on ideas proposed by Goto with the goal of producing a tool for extraction of melodic lines from audio recordings. The approach includes extraction of sinusoidal components from the original audio signal, EM estimation of predominant pitches, their grouping into melodic fragments and final clustering of melodic fragments into melodic lines. The paper briefly describes each of these stages and presents some preliminary results.

2. DISCOVERING MELODIC FRAGMENTS

Our approach to finding melodic lines begins with discovery of fragments that a melodic line is composed of – melodic fragments. Melodic fragments are defined as regions of the signal, that exhibit a strong and stable pitch. Pitch is the main attribute according to which fragments are discovered; other features, such as loudness or timbre, are not taken into consideration. They come into picture when fragments are merged into melodic lines according to their similarity.

2.1. SMS analysis

To locate melodic fragments, we initially need to estimate the predominant pitch(es) in the input signal. To achieve that, we first separate the slowly-varying sinusoidal components (partials) of the signal from the rest (transients and noise) by the well known spectral modelling synthesis approach (SMS, [10]). SMS analysis transforms the signal into a set of sinusoidal components with time-varying frequencies and amplitudes, and a residual signal, obtained by subtracting the sines from the original signal. We used the publicly available SMSTools software (http://www.iua.upf.es/mtg/clam) to analyse our songs with a 100 ms Blackman-Harris window, 10 ms hop size. Non-harmonic style of analysis was chosen, as our signals are generally polyphonic and not necessary harmonic (drums...).

2.2. Masking

The obtained sinusoidal components are subjected to a psychoacoustic masking model that eliminates the components masked by other, stronger ones. Only simultaneous masking within critical bands is taken into consideration – temporal masking is ignored. Tonal and noise maskers are calculated from the set of sinusoidal components and the residual signal, as described in [11], and
components that fall below the global masking threshold removed. The mask- ing procedure is mainly used to reduce the computational load of predominant pitch estimation, as it on average halves the maximal number of sinusoidal components (to approx. 60 per frame).

2.3. Predominant pitch estimation

After the sinusoid components have been extracted, and masking applied, we estimate the predominant pitch(es) in short (50 ms) segments of the signal. Our pitch estimating procedure is based on the PrefEst approach introduced by Goto [9], with some modifications. The method is based on the Expectation-Maximisation (EM) algorithm, which treats the set of sinusoidal components at each time instant as a probability density function (observed PDF), which is considered to be generated from a weighted mixture of tone models of all possible pitches at this time instant. A tone model is defined as a PDF, corresponding to a typical structure of a harmonic tone (fundamental frequency + overtones). The EM algorithm iteratively estimates the weights of all tone models, while searching for one that maximizes the observed PDF. Consequently, each tone model weight represents the dominance of the tone model and thereby the dominance of the tone model’s pitch occurring when several strong components fall within the width of a Gaussian, representing a tone model component. In this case, the function limits the sum of contributions of all components, which in a simplified way mimics the effects that distance between frequency components plays in the perception of loudness [12].

The process is illustrated in Fig. 1, where a tone model with pitch 529 Hz is applied to a series of partials found by the SMS algorithm. The model acts as a sieve, picking and summing up contributions of individual partials that would fit into a tone with a pitch of 529 Hz. Only the first six tone model partials are shown.

![Figure 1: Applying a tone model on a set of partials](image)

The weights \( w^{(t)}(g) \) of all possible tone models (eq. 2) and amplitudes of their harmonics \( (c) \), are iteratively calculated by the EM algorithm:

\[
\begin{align*}
\bar{w}^{(t)}(g) & = \sum_{\{f,a\} \in G^{(t)}} M\left(f,g,c^{(t)}(g)\right) \\
\bar{c}^{(t)}(h,g) & = \sum_{\{f,a\} \in G^{(t)}} M\left(f,g,c^{(t)}(g)\right)
\end{align*}
\]

When the iterative algorithm converges, the pitch of the tone model with the highest weight \( w \) is taken to be the predominant pitch. We use early stopping to stop the convergence prematurely and take the first few highest weights to represent the predominant pitches in the time window under consideration. These are later tracked and grouped into melodic fragments.

In the beginning, all tone model weights and amplitudes are initialized to the same value. Tone models contain a maximum of 20 harmonics, values of \( \sigma \) range between 50 cents (1st harmonic) to 100 cents (20th harmonic). After some experiments, the value of \( n \), representing the width of the analysis window, was set to 5, thereby encompassing a time interval of 50 ms. This significantly reduced the effects of “noisy” partials, found by SMS analysis, on estimation of predominant pitch.

The effect can be seen in Fig. 2, representing the outcome of the EM algorithm on a short fragment from Aretha Franklin’s inter-
2.4. Forming melodic fragments

Weights produced by the EM algorithm indicate the pitches that are dominant at each time instance. Melodic fragments are formed by tracking dominant pitches through time and thereby forming fragments that have continuous pitch contours. The first part of the procedure is similar to pitch salience calculation as described by Goto [13]. For each pitch with weight greater than a dynamically adjusted threshold, salience is calculated according to its dominance in a 50 ms look-ahead window. The procedure tolerates pitch deviations of up to 100 cents per 10 ms window and also tolerates individual noisy frames that might corrupt pitch tracks by looking at the contents of the entire 50 ms window.

After saliences are calculated, grouping into melodic fragments is performed by continuously tracking the top three salient peaks and producing fragments along the way as follows:

- the procedure ignores all time instances, where total loudness of the signal, calculated according to Zwicker’s loudness model [12] falls below a set threshold;
- the initial set of melodic fragments \( F \) is empty; the initial set of candidate melodic fragments \( C \) is empty;
- the following operations are repeated:
  - in each time instance \( t \), select the top three salient peaks that differ from each other by more than 200 cents and find their exact frequencies \( f_i \), according to the largest weight \( w_i \) in the neighbourhood:
    - in the set of candidate fragments \( C \), find a fragment \( c \) with average frequency closest to \( f_i \);
    - if the difference in frequencies between \( c \) and \( f_i \) is smaller than 200 cents, add \( f_i \) to the current candidate fragment;
    - otherwise, start a new candidate fragment
  - after the top three pitches at time \( t \) have been processed, find all candidate fragments, that have not been extended during the last 50 ms. If their length exceeds 50 ms, add them to the set of melodic fragments \( F \) and remove them from the set of candidates \( C \). If their length is shorter than 50 ms, remove them from \( C \).
  - after the signal has been processed, merge harmonically related melodic fragments, appearing at the same time (only 1st and 2nd overtones are taken into consideration) and join continuous fragments (in time and frequency).

The final result of this simple procedure is a set of melodic fragments, which may overlap in time, are at least 50 ms long and may have a slowly changing pitch. Parameters of each fragment are its start and end time, its time-varying pitch and its time-varying loudness. The fragments obtained provide a reasonable segmentation of the input signal into regions with stable dominant pitch. An example is given in Fig. 3, which shows segmentation obtained on a 5.5 seconds excerpt from Aretha Franklin's interpretation of the song "Respect". 25 fragments were obtained; six belong to the melody sung by the singer, while the majority of others belong to different parts of the arrangement, which become dominant when lead vocals are out of the picture. Additionally, three noisy fragments were found, which were either due to consonants or drum parts. These can usually be dealt with in the last part of the procedure, where fragments are merged into melodic lines.

We performed informal subjective listening tests by resynthesizing the fragments (on the basis of their pitch and amplitude) and comparing these resynthesized versions with the original signal covering the same time spans. Most of the fragments perfectly captured the dominant pitch in the areas, even if, while listening to the entire original signal, some of the fragments found were not immediately obvious to the listener (i.e. organ parts in the given example). We carried out such tests on a set of excerpts from 10 different songs, covering a variety of styles, from jazz, pop/rock to dance, and the overall performance of the algorithm for finding melodic fragments was found to be satisfying; it discovered a large majority of fragments belonging to the lead melody, which is the main point of interest in this study.

Figure 2: Effect of window size \( n \) on the EM algorithm for predominant pitch estimation

Figure 3: Segmentation into melodic fragments of an excerpt from Otis Redding’s song “Respect” sung by Aretha Franklin
3. FORMING MELODIC LINES

The goal of our project is to extract one or more melodic lines from an audio recording. How is a melodic line, or melody, defined? There are many definitions; Levitin describes melody as an auditory object that maintains its identity under certain transformations along the six dimensions of pitch, tempo, timbre, loudness, spatial location, and reverberant environment; sometimes with changes in rhythm; but rarely with changes in contour [14]. Not only that melodies maintain their identity under such transformations, or rather because of that, melodies themselves are usually (at least locally in time) composed of events that themselves are similar in pitch, tempo, timbre, loudness, etc.

The fact becomes useful when we need to group melodic fragments, like the ones obtained by the procedure described before, into melodic lines. In fact, the process of discovering melodic lines becomes one of grouping melodic fragments through time into melodies. Fragments are grouped according to their properties. Ideally, one would make use of properties, which accurately describe the six dimensions mentioned before, especially pitch, timbre, loudness and tempo. Out of these, timbre is the most difficult to model; we are not aware of studies that would reliably determine the timbre of predominant voices in polyphonic audio recordings. Many studies, however, make use of timbre related features, when comparing pieces according to their similarity, classifying music according to genre, identifying the singer, etc. (i.e. [15], [16]). The features used in these studies could be applied to our problem, but so far we have not yet made such attempts. To group fragments into melodies, we currently make use of only four features, which represent:

- **pitch** as the centroid of fragment’s frequency with regard to its dominance;
- **loudness** as the mean value of the product of dominance and loudness. Loudness is calculated according to Zwicker’s loudness model [12] for partials belonging to the fragment. The product of dominance and loudness seems to give better results than if loudness alone would be taken;
- **pitch stability** as the average change of pitch over successive time instances. This could be classified as the only timbral feature used and mostly separates vocal parts from stable instruments;
- **onset steepness** as the steepness of overall loudness change during the first 50 ms of the fragment’s start. The feature penalizes fragments that come into picture when a louder sound stops.

To group melodic fragments into melodies, we use a modified Gaussian mixture model estimation procedure, which makes use of equivalence constraints during the EM phase of model estimation [17]. Gaussian Mixture Models (GMMs) are one of the more widely used methods for unsupervised clustering of data, where clusters are approximated by Gaussian distributions, fitted on the provided data. Equivalence constraints are prior knowledge concerning pairs of data points, indicating if the points arise from the same source (belong to the same cluster - positive constraint) or from different sources (different clusters - negative constraint). They provide additional information to the GMM training algorithm, and are very useful in our domain. We use GMMs to cluster melodic fragments into melodies according to their properties. Additionally, we make use of two facts to automatically construct positive and negative equivalence constraints between fragments.

Fragments may overlap in time, as can be seen in Fig. 2. We treat melody as a succession of single notes ( pitches). Therefore, we can put negative equivalence constraints on all pairs of fragments that overlap in time. This forbids the training algorithm to put two overlapping fragments into the same cluster and thus the same melodic line. We also give special treatment to the bass line, which may appear quite often in melodic fragments (Fig. 2). To help the training algorithm with bass line clustering, we also put positive equivalence constraints on all fragments with pitch lower than 170 Hz. This does not mean that the training algorithm will not add additional fragments to this cluster; it just causes all low pitched fragments to be grouped together.

The clustering procedure currently only works on entire song fragments (or entire songs), and we are still working on a version that will work within an approx. 5 second long sliding window and dynamically add new fragments to existing clusters or form new clusters as it progresses through a given piece.

We have not yet made any extensive tests of the accuracy of our melody extracting procedure. This is mainly due to the lack of a larger annotated collection of songs that could be used to automatically measure the accuracy of the approach. We have tested the algorithm on a number of examples and are overall satisfied with the performance of the fragment-extracting procedure, and less so with the performance of GMM clustering. GMMs may work perfectly in some cases, like Aretha Franklin’s example used for this paper, while for others, problems may occur mainly because fragments belonging to accompanying instruments, which appear close to the lead melodic line are taken to be part of the line.

Results of clustering on a 30 second excerpt of Otis Redding’s song “Respect”, as sung by Aretha Franklin, are given in Table 1.

<table>
<thead>
<tr>
<th>lead vocal</th>
<th>back vocals</th>
<th>bass</th>
<th>guitar</th>
<th>brass</th>
<th>keys</th>
<th>noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.03</td>
<td>0.24</td>
<td>0.03</td>
<td>0</td>
<td>0.1</td>
<td>0.33</td>
</tr>
<tr>
<td>C2</td>
<td><strong>0.93</strong></td>
<td>0.29</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>C3</td>
<td>0.03</td>
<td>0.38</td>
<td>0.33</td>
<td>0.3</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>C4</td>
<td>0</td>
<td>0.97</td>
<td>0</td>
<td>0.05</td>
<td>0.33</td>
<td>0.08</td>
</tr>
<tr>
<td>C5</td>
<td>0</td>
<td>0.1</td>
<td>0.67</td>
<td>0.45</td>
<td>0.33</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 1: GMM clustering of fragments from “Respect.”

152 melodic fragments were found by the fragment finding procedure; all lead vocal and backing vocal parts were correctly discovered. All fragments were hand annotated into one of seven categories (lead vocal, backing vocals, bass, guitar, brass, keyboards, noise). Fragments were then clustered by the GMM algorithm into five clusters, which would ideally represent the melody (lead vocal), bass line, backing vocals, accompaniment and noise.

Results of the clustering procedure are given in Table 1. It shows percentages of fragments belonging to the seven annotated categories in the five clusters. Ideally, lead vocal fragments (melody) would all be grouped into one cluster with no additional fragments. Most (93%) were indeed grouped into cluster 2, but the cluster also contains some other fragments, belonging to backing vocals, brass and a small amount of noise. The majority of bass fragments were put into cluster 4, together with some low pitched keyboard parts, while other clusters contain a mixture of accompaniment and backing vocals. As our goal lies mainly in the discovery of the (main) melodic line, results are satisfying, especially if we take into consideration that practically no timbre based
features were taken into consideration when clustering. Most of the melody is represented by fragments in cluster 2, with some additional backing vocal fragments, which could actually also be perceived as part of the melody. The effect of negative and positive constraints on the clustering procedure was also assessed; somewhat surprisingly, constraints did not have a large impact on the clustering procedure. Small improvements were achieved mostly in separation of accompaniment from lead vocal and bass lines.

4. CONCLUSIONS

The presented approach to melody extraction is still in an initial phase, but we are satisfied with first obtained results. Currently, we are in the process of annotating a larger number of pieces, which will be used for improving the feature set used in GMM training, as so far, we settled for a very small number of parameters, mainly because of the small set of examples we worked with. We plan to concentrate on timbral features, which are expected to bring improvements, especially with mismatches in parts where accompaniment becomes dominant. The larger database will also enable us to test and compare several different clustering strategies.

5. REFERENCES

MUSICAL INSTRUMENT IDENTIFICATION IN CONTINUOUS RECORDINGS

Arie A. Livshin and Xavier Rodet

Analysis/Synthesis Team
Ircam, Paris, France
{livshin|rod}@ircam.fr

ABSTRACT

Recognition of musical instruments in multi-instrumental, polyphonic music is a difficult challenge which is yet far from being solved. Successful instrument recognition techniques in solos (monophonic or polyphonic recordings of single instruments) can help to deal with this task.

We introduce an instrument recognition process in solo recordings of a set of instruments (bassoon, clarinet, flute, guitar, piano, cello and violin), which yields a high recognition rate. A large and diverse solo database (108 different solos, all by different performers) is used in order to encompass the different sound possibilities of each instrument and evaluate the generalization abilities of the classification process.

First we bring classification results using a very extensive collection of features (62 different feature types), and then use our GDE feature selection algorithm to select a smaller feature set with a relatively short computation time, which allows us to perform instrument recognition in solos in real-time, with only a slight decrease in recognition rate.

We demonstrate that our real-time solo classifier can also be useful for instrument recognition in duet performances, and improved using simple “source reduction”.

1. INTRODUCTION

Most works on instrument recognition have dealt with classification of separate musical tones taken from professional sound databases, e.g. McGill, Studio Online, etc.

Instrument recognition in solo performances (monophonic or polyphonic musical phrases performed by a single instrument) is different and more complicated than dealing with separate note databases, as the time evolution of each sound (attack, decay, sustain, release) is not well defined, the notes are not separated, there are superpositions of concurrent sounds and room echo, different combinations of playing techniques, etc. Marques and Moreno [1] classified 8 fairly different instruments (bagpipes, clarinet, flute, harpsichord, organ, piano, trombone and violin) using one CD per instrument for learning and one for classification. They compared 3 feature types using 2 different classification algorithms, and achieved 70% recognition rate. Brown Houix and McAdams [2] classified 4 wind instruments (flute, sax, oboe and clarinet), compared 4 feature types and reached 82% recognition rate with the best combination of parameters and training material. Martin [3] has classified sets of 6, 7 and 8 instruments, reaching 82.3% (violin, viola, cello, trumpet, clarinet, and flute), 77.9% and 73% recognition rates respectively. He used up to 3 different recordings from each instrument; in each experiment one recording was classified while the rest were learned. The feature set was relatively large and consisted of 31 one-dimensional features. For a comprehensive review of instrument recognition, see [4].

The work on solo recognition is not yet exhausted. Although it seems that there are not many applications which actually require solo recognition, yet as we shall demonstrate at the end of this paper, knowledge of how to deal well with solos can also help in recognition of multi-instrumental music (where several instruments play concurrently). The subject of musical instrument recognition in multi-instrumental music is difficult and is just beginning to get explored (e.g. [5]).

We begin the paper by presenting a process for recognition of a set of instruments (bassoon, clarinet, flute, guitar, piano, cello and violin) which yields a high average recognition rate - 88.13% when classifying 1-second pieces of real recordings.

A large and very diverse solo database is used for learning and evaluating the recognition process. It contains 108 solo performances, all by different musicians, and apparently supplies a good generalization of the different sound possibilities of each instrument in various recording conditions, playing techniques, etc., thus providing a good generalization of the sounds each instrument is capable of producing in different recordings - what we call the “concept instrument”. In order to evaluate the generalization ability of the classifier, the same solos are never used both in the learning and test sets; we have proved that a classification evaluation process in which the training and learning sets both contain samples recorded in very similar conditions is likely to produce misleading results [6].

We use a very large collection of features for solo recognition – 62 different feature types [7] which were developed and used in the Cuidado project. Using our GDE feature selection algorithm, we select a smaller feature set best suited for solo recognition in real-time (of our 7 instruments), with only a small reduction in recognition rate (85.24%) compared to the complete feature set. We present the features of this real-time feature set, which was actually implemented in a real-time solo recognition program.

We end the paper by demonstrating that the same features and techniques we used for real-time solo recognition can also help to perform instrument recognition in duet performances. We use the same solo recognition program and the real-time feature set we used for solos, first to directly perform instrument recognition in duets, and then improve the results using a multiple-f0 detection program by C. Yeh [12] and simple “source reduction”.

Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx'04), Naples, Italy, October 5-8, 2004
2. SOLO DATABASE

Our sound database consists of 108 different ‘real-world’ solo performances (by “solo” we mean that a single instrument is playing, in monophony or polyphony) of 7 instruments: bassoon, clarinet, flute, classical guitar, piano, cello and violin. These performances, which include classical, modern and ethnic music, were gathered from commercial CD’s (containing new or old recordings) and MP3 files played and recorded by professionals and amateurs.

Each solo was performed by a different musician and there are no solos taken from the same concert. During the evaluation process we never use the same solo, neither fully or partly, in both the learning set and the test set. The reason for these limitations is that we need the evaluation process to reflect the system’s ability to generalize – i.e. classify new musical phrases which were not learned, and were recorded in different recording conditions, different instruments and played by different performers than the learning set. We have proved [6] that the evaluation results of a classification system which does learn and classify sounds performed on the same instrument and recorded in the same recording conditions, even if the actual notes are of a different pitch, are much higher than when classifying sounds recorded in different recording conditions. The reason is that such an evaluation process actually shows the system’s ability to learn and then recognize specific characteristics of specific recordings and not its ability to generalize and recognize the “concept instrument”.

2.1. Preprocessing

All solos were downsampling to 11Khz, 16bit. Only the left channel was taken out of stereo recordings1. A 2-minute piece was taken from each solo recording and cut into 1-second cuts with a 50% overlap – a total of 240 cuts out of each solo.

3. FEATURE DESCRIPTORS

The computation routines for the features we use in the classification process were written by Geoffroy Peeters as part of the Cuidado project. Full details on all the features, can be found in [7].

The features are computed on each 1-second solo-cut separately. Besides several features2 which were computed using the whole signal of the 1-second cut, most of the features were computed using a sliding frame of 60 ms with a 66% overlap. For each solo-cut of 1 second, the average and standard deviation of these frames were used by the classifier.

Initially, we used a very large feature collection – 62 different features of the following types [8]:

a) Temporal Features.

Features computed on the signal as a whole (without division into frames), e.g. log attack time, temporal decrease, effective duration.

b) Energy Features.

Features referring to various energy content of the signal, e.g. total energy, harmonic energy, noise part energy.

c) Spectral Features.

Features computed from the Short Time Fourier Transform (STFT) of the signal, e.g. spectral centroid, spectral spread, spectral skewness.

d) Harmonic Features.

Features computed from the Sinusoidal Harmonic modelling of the signal, e.g. fundamental frequency, inharmonicity, odd to even ratio.

e) Perceptual Features.

Features computed using a model of the human hearing process, e.g. mel frequency cepstral coefficients, loudness, sharpness.

Later in the paper we shall use our GDE feature selection algorithm to reduce the number of features in order to perform instrument recognition in real-time.

4. “MINUS-1 SOLO” EVALUATION METHOD

After the features are computed, they are normalized to the range of 0 – 1. For every solo in its turn, its 1-second solo-cuts are removed from the database and classified by the rest of the solos. This process is repeated for all solos, and the average recognition rate for each instrument is reported along with the average recognition rate among all instruments. These results are more informative than the average recognition rate per solo, as the number of solos performed by each instrument might be different.

The classification is done by first performing Linear Discriminant Analysis (LDA) [9],[10] on the learning set, multiplying the test set with the resulting coefficient matrix and then classifying using the K Nearest Neighbours (KNN) algorithm. For the KNN we use the “best” K from a range of 1 - 80 which is estimated using the leave-one-out method on the learning set3 [11].

5. FEATURE SELECTION

After computing the recognition rate using the full feature set, we use our Gradual Descriptor Elimination (GDE) feature selection method [11] in order to find the most important features. GDE uses LDA repeatedly to find the descriptor which is the least significant and remove it. This process is repeated until no descriptors are left. At each stage of the GDE the system recognition rate is estimated. In this Section we have set the goal to achieve a smaller feature set which will be quick to compute - allowing us to perform solo recognition in real-time, and will compromise the recognition rate as little as possible, compared with the results obtained by using

1 It could be argued that it is preferable to use a mix of both channels. Which method is actually better depends on the specific recording settings of the musical piece.

2 Some features contain more than a single value, e.g. the MFCC’s; we use the term “features” regardless of their number of values.

3 The “best K” for our database was estimated as 33 for the full feature set and 39 for the real-time set. Experiments with solo-cuts using an overlap of 75% instead of 50% (resulting in 480 solo-cuts per solo instead of 240), reported a “best K” of 78 for the full feature set and 79 for the “real-time” set.
the complete feature set. By “real-time” we mean here that while the solo is recorded or played the features of each 1-second fraction of the music are computed and classified immediately after it was performed, before the following 1-second has finished playing/recording.\footnote{Because the classified 1-second solo pieces can partially overlap, the theoretical upper limit for the recognition resolution is 1 sample.}

We removed the most time-consuming features and used GDE to reduce the feature-data until the number of features went down from 62 to 20. Using these features we have actually implemented a real-time solo phrase recognition program which works on a regular Intel Processor and is written in plain Matlab code (without compilation or integration with machine language boost routines).

We can see in Table 1 that the “Real-Time” average recognition rate is indeed rather close to the “Complete Set”. It is interesting to note that while reducing the feature set we have actually improved the recognition rate of the flute; LDA does not always eliminate confusion caused by interfering features.

![Figure 1: Real-time solo recognition process](image)

<table>
<thead>
<tr>
<th>“Real-Time”</th>
<th>“Complete Set”</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 features</td>
<td>62 features</td>
</tr>
<tr>
<td>Bassoon</td>
<td>86.25 %</td>
</tr>
<tr>
<td>Clarinet</td>
<td>79.29 %</td>
</tr>
<tr>
<td>Flute</td>
<td>83.33 %</td>
</tr>
<tr>
<td>Guitar</td>
<td>86.34 %</td>
</tr>
<tr>
<td>Piano</td>
<td>91.00 %</td>
</tr>
<tr>
<td>Cello</td>
<td>82.18 %</td>
</tr>
<tr>
<td>Violin</td>
<td>88.27 %</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>85.24 %</strong></td>
</tr>
</tbody>
</table>

Table 1: Minus-1 Solo recognition results

### 6.1. The Real-Time Feature Set

![Table 2: A sorted list of the most important features for real-time solo classification (of our 7 musical instruments) ![Table 2: A sorted list of the most important features for real-time solo classification (of our 7 musical instruments)](image)

Table 2: A sorted list of the most important features for real-time solo classification (of our 7 musical instruments)

We can see in Table 2 that the 10 most important features are the first 4 Moments and the Spectral Slope, computed in both the perceptual and spectral models. See [7] for a full explanation of each feature.

### 7. MULTI-INSTRUMENTAL EXAMPLES

In Table 3 we bring some examples for instrument recognition in real performance duets (where 2 instruments are playing concurrently) using our solo-recognition process with the real-time feature set. This Section is not pretending to be an extensive research of multi-instrumental classification, but rather comes to demonstrate that successful solo recognition might actually be useful for instrument recognition in multi-instrumental music.

From each real performance duet, a 1-minute section was selected in which both instruments are playing together.

In Table 3 we bring some examples for instrument recognition in multi-instrumental music. Each row of Table 3 shows two kinds of results; the upper results are of “Unmodified Recognition” and the bottom of recognition using simple “Source Reduction”.

We see in Table 3 that the 10 most important features are the first 4 Moments and the Spectral Slope, computed in both the perceptual and spectral models. See [7] for a full explanation of each feature.
225

Table 3: Duet Classification using our real-time solo recognition program.

<table>
<thead>
<tr>
<th>Source Reduction</th>
<th>Bassoon</th>
<th>Clarinet</th>
<th>Flute</th>
<th>Guitar</th>
<th>Piano</th>
<th>Cello</th>
<th>Violin</th>
<th>Total Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Castelnuovo:</td>
<td>16.2 %</td>
<td>10.4 %</td>
<td>16.6%</td>
<td>38.4 %</td>
<td>33.8%</td>
<td>1.8%</td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Sonatina</td>
<td>58.8 %</td>
<td>51.0 %</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.5 %</td>
</tr>
<tr>
<td>Stockhausen:</td>
<td>50.0%</td>
<td>50.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Tierkreis</td>
<td>50.0%</td>
<td>50.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Scleri:</td>
<td>28.9%</td>
<td>71.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Suite</td>
<td>33.0%</td>
<td>51.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Carter:</td>
<td>72.5%</td>
<td>45.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Esprit rude:</td>
<td>77.3%</td>
<td>41.8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Kirchner:</td>
<td>2.5%</td>
<td>15.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>100.0 %</td>
</tr>
<tr>
<td>Tripietech</td>
<td>1.7%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>97.5 %</td>
</tr>
<tr>
<td>Ravel:</td>
<td>1.7%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>97.5 %</td>
</tr>
<tr>
<td>Sonata</td>
<td>2.5%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>89.1 %</td>
</tr>
<tr>
<td>Martinu:</td>
<td>2.5%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>89.1 %</td>
</tr>
<tr>
<td>Duo</td>
<td>2.5%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>97.5 %</td>
</tr>
<tr>
<td>Pachelbel:</td>
<td>7.2%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>91.8 %</td>
</tr>
<tr>
<td>Canon in D</td>
<td>10.6%</td>
<td>41.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>91.8 %</td>
</tr>
<tr>
<td>Procaccini:</td>
<td>11.8%</td>
<td>23.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>91.8 %</td>
</tr>
<tr>
<td>Trois pieces</td>
<td>17.8%</td>
<td>35.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>91.8 %</td>
</tr>
<tr>
<td>Bach:</td>
<td>9.6%</td>
<td>45.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>90.4 %</td>
</tr>
<tr>
<td>Cantata BWV</td>
<td>18.2%</td>
<td>33.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>70.4 %</td>
</tr>
<tr>
<td>Sculptured:</td>
<td>11.1%</td>
<td>55.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>88.9 %</td>
</tr>
<tr>
<td>Psalmata</td>
<td>10.2%</td>
<td>25.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>80.3 %</td>
</tr>
<tr>
<td>Ohana:</td>
<td>5.0%</td>
<td>26.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>86.8 %</td>
</tr>
<tr>
<td>Flute duo</td>
<td>24.6%</td>
<td>72.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>72.1 %</td>
</tr>
<tr>
<td>Pachelbel:</td>
<td>13.6%</td>
<td>50.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>86.3 %</td>
</tr>
<tr>
<td>Canon in D</td>
<td>9.2%</td>
<td>51.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>95.0 %</td>
</tr>
<tr>
<td>Idrs:</td>
<td>44.2%</td>
<td>16.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>94.4 %</td>
</tr>
<tr>
<td>Aria:</td>
<td>2.7%</td>
<td>50.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>90.0 %</td>
</tr>
<tr>
<td>Feidman:</td>
<td>6.7%</td>
<td>37.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>59.4 %</td>
</tr>
<tr>
<td>Kleizmer:</td>
<td>0.0%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>77.5 %</td>
</tr>
<tr>
<td>Coplan:</td>
<td>0.0%</td>
<td>14.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>70.4 %</td>
</tr>
<tr>
<td>Sonata:</td>
<td>1.3%</td>
<td>32.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>89.1 %</td>
</tr>
<tr>
<td>Guigliani:</td>
<td>3.2%</td>
<td>35.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td>91.8 %</td>
</tr>
</tbody>
</table>

7.1. Unmodified Recognition

Each consecutive 1-second piece of the duet, without any modifications, is classified by our real-time solo recognition program.

7.2. Source Reduction

With this proposed technique, we reduce the volume of one of the playing instruments, and thus make it easier for our real-time solo-recognition program to recognize the other instrument.

First we use a multiple-f0 detection program by Chunghsin Yeh [12] (this article is also presented in DAFX’04), to get an estimation of two 5 f0s and their corresponding harmonics for every frame of the duet 6 . The frames are 180 ms each with a 75% overlap.

Next, consecutive frames with the same pitch (we use the estimated f0s, quantized to half-tones), are grouped together into “chunks” of at least ½ second in length 7 . Thus, each chunk of frames contains one note (at least) which is sustained throughout the chunk and presumably performed by a single instrument, and some other notes playing along with it.

Next, each chunk is used twice; in the first case we keep the harmonics but keep everything else – we call this the “anti-chunk”.

This filtering process is performed in the frequency domain using a phase vocoder and the same frame sizes which were used in the estimation of two f0s and their corresponding harmonics but keep everything else – we call this the “anti-chunk”.

First we use a multiple-f0 detection program by Chunghsin Yeh [12] (this article is also presented in DAFX’04), to get an estimation of two 5 f0s and their corresponding harmonics for every frame of the duet 6 . The frames are 180 ms each with a 75% overlap.

Next, consecutive frames with the same pitch (we use the estimated f0s, quantized to half-tones), are grouped together into “chunks” of at least ½ second in length 7 . Thus, each chunk of frames contains one note (at least) which is sustained throughout the chunk and presumably performed by a single instrument, and some other notes playing along with it.

Next, each chunk is used twice; in the first case we keep the harmonics of the sustained note and filter out every else out of the chunk. In the second case, we filter out the sustained note harmonics but keep everything else – we call this the “anti-chunk”.

This filtering process is performed in the frequency domain using a phase vocoder and the same frame sizes which were used in the estimation of two f0s and their corresponding harmonics but keep everything else – we call this the “anti-chunk”.

5 In cases of an instrument accompanied by a highly polyphonic instrument – guitar or piano, the “duets” might have many more concurrent notes than just 2. By using f0 detection of just 2 notes, we rely on the accompanied instrument to have a relatively high volume compared to the accompanying one [5], and thus be recognized as one of the 2 f0s.

6 The algorithms for the f0 estimation and their evaluation are out of the scope of this article.

7 Not all the musical piece is covered by chunks, as there are some sections where both instruments play notes that are shorter than the chosen ½ second.
multiple-f0 estimation. Overlapping harmonics of the two notes are not filtered out.

The filtered chunks and the anti-chunks are classified by our realtime solo-recognition program.

Note about the "scoring": the Source Reduction results in Table 3 (the bottom part of every row) are the percentage of time in which a specific instrument is recognized out of the total time of all the instruments. These are "net results" - in cases where some filtered chunks (or anti-chunks) overlap in time and are classified as the same instrument, we count the time of the overlapping part only once. For example, if two 1-second chunks overlap over ½ second, and are both classified as violin, it will count only as 1½ seconds for the violin and not 2 seconds. Each result in the 'Total Correct' column is the percentage of time in which the correct instruments were recognized out of the total time of all the instruments recognized in that duet.

7.3. Duet Recognition Results

The first column in Table 3 contains the partial name of the musical piece. Columns 2 to 8 contain the percentage of the total classification time which was classified as the corresponding instrument. The black cells indicate correct classifications - recognition of the instruments which actually played in the corresponding cuts, while the white cells indicate misclassified cuts. The last column is the total percentage of time in which correct instruments were recognized.

'V' beside the score in the 'Total Correct' column, indicates that the 2 most popular instrument recognitions are actually the 2 instruments playing in the duet. V means that only one of the 2 most popular recognitions is a correct instrument. This statistic is important; if the maximum number of instruments in a musical piece is known and we can rely on our classifier to correctly recognize them (e.g. the two correct instruments in a duet), then we can afterward limit the learning set to just these two instruments and perform the recognition process again, this time getting a much more precise segmentation of the musical piece into the correct playing instruments.

As already mentioned, each row of Table 3 shows at the top the results of Unmodified Recognition and at the bottom the results of recognition using Source Reduction. We immediately can see that there is a considerable number of duet examples where Unmodified Recognition produced correct classifications, although, as we know, this classifier is very naive and just attempts to classify unmodified duet pieces using a solo classifier.

Looking at the 'V' signs in the 'Total Correct' column, we see that the Source Reduction performed better than the Unmodified Recognition in finding which two instruments play in each duet, and except in one case, always correctly recognized them. On the other hand, just looking at the numbers in the 'Total Correct' column create the impression that in many cases the Unmodified Recognition outperforms the Source Reduction. Looking more carefully on these results along with the recognition rates for each instrument, we can conclude that in duets there is usually one instrument which is more dominant and relatively easier to recognize (e.g. with a higher volume). We see that the Source Reduction recognizes better the weaker instrument than the Unmodified Recognition (which has 'no choice' really, as it classifies each duet cut only once), and that the recognition results of the Source Reduction for both instruments are more even. For example, if we look at Pachelbel's Canon in D for Cello and Violin, we see that the total Unmodified Recognition score is 84.6% while the Source Reduction score is only 67.8%. However, with the Unmodified Recognition, the violin was not recognized at all. The Source Reduction, on the other hand, recognized the Cello with 37.2% and the Violin with 30.6%, and although the total score is lower than the Unmodified Recognition due to the fact that the source reduction process is not perfect and results in some misclassifications, still the correct instruments are the most popular ones – cello and violin.

We have demonstrated in this Section that a good solo recognizer can be useful for instrument recognition in multi-instrumental music.

8. SUMMARY

We presented a process for continuous recognition of musical instruments in solo recordings which yields a high recognition rate. Our results are based on evaluation with a large and very diverse solo database which allowed us a wide generalization of the classification and evaluation processes using diverse sound possibilities of each instrument, recording conditions and playing techniques.

We used our GDE feature selection algorithm with a big feature set and considerably reduced the number of features, down to a feature set which allowed us to perform real-time instrument recognition in solo performances. This smaller feature set delivers a recognition rate which is close to that of the complete feature set.

Lastly, we have shown that our recognition process and realtime feature set can also be useful for instrument recognition in duet music. This exemplifies our initial claim that learning to achieve high recognition rates in solos could also be useful for instrument recognition in multi-instrumental performances.

9. FUTURE WORK

We shall continue improving the solo recognition process in parallel to working on recognition in multi-instrumental music; we have shown that the first can help to achieve the second.

We will study the reasons why specific instrument combinations produce high recognition errors and how to better differentiate between these instruments.

A confidence level estimation could be added to the solo classifier. Instead of just reporting one instrument, it could give an estimated weight to each instrument, so each 'solo' classification will produce several recognition candidates.

New features will be developed and used in the feature selection process; some of them especially designed with multi-instrumental recognition in mind.

---

8 That is why we prefer the term "source reduction" instead of "source separation". We do not attempt to perform full separation of the sources, but just to "reduce" one of them as much as possible without altering the second one.
The multiple-f0 estimation and the source reduction algorithms still need to be improved, especially in order to provide an accurate segmentation into musical instruments of highly multi-instrumental music.

10. ACKNOWLEDGMENTS

Thanks to Geoffroy Peeters for using his feature computation routines and sharing his knowledge and experience.
Thanks to Chunghsin Yeh for using his multiple-f0 detection program.
Thanks to Emmanuel Vincent for sharing his solo database.

11. REFERENCES


Identifying chords and related musical attributes from digital audio has proven a long-standing problem spanning many decades of research. A robust identification may facilitate automatic transcription, semantic indexing, polyphonic source separation and other emerging applications. To this end, we develop a Bayesian inference engine operating on single-frame STFT peaks. Peak likelihoods conditional on pitch component information are evaluated by an MCMC approach accounting for overlapping harmonics as well as undetected/spurious peaks, thus facilitating operation in noisy environments at very low computational cost. Our inference engine evaluates posterior probabilities of musical attributes such as root, chroma (including inversion), octave and tuning, given STFT peak frequency and amplitude observations. The resultant posteriors become highly concentrated around the correct attributes, as demonstrated using 227 ms piano recordings with –10 dB additive white Gaussian noise.

1. INTRODUCTION

Chord identification has proven to be a long-standing problem in computer music research, despite the innate ability of musically trained humans to readily accomplish the task. A variety of historically successful approaches offer mostly rule-based or “blackboard” schema; of note are [1], [2], [3], among others.

When addressing the problem of automatic chord identification, it is important to consider the extent to which an approach attempts to model the human auditory system. The auditory modeling may be implicit, as in Klapuri’s use of the spectral smoothness principle [3], or more explicit, as in Martin’s use of a modified Meddis and Hewitt pitch perception model [1]. Rather than modeling the auditory mechanism, our goal is simply to identify chords and related musical attributes as well as possible.

Moreover, the problem becomes as much cognition as perception. Gang and Berger [4], for instance, emphasize the role of musical expectations in learning chord sequences in functional tonal music. In a Bayesian probabilistic framework, musical expectations may be easily encoded as prior and/or conditional distributions involving hidden musical attributes, e.g., note, chroma, root, octave, tuning, key, and mode. By so doing, we represent, in a purely statistical framework, pseudocognitive capacities such as temporal integration and the incorporation of knowledge from musical structure.

Several emerging approaches, for instance [5], [6], pursue probabilistic as opposed to rule-based schema. In [6], Sheh and Ellis facilitate temporal fusion via hidden Markov model (HMM) inference, using Fujishima’s pitch class profiles [7] as feature observations. Pitch class profiles discard octave information at the front end; however, for applications such as polyphonic transcription, it may be desired to retain the absolute pitches. We adopt short-time Fourier transform (STFT) peaks as features, enabling attributes such as octave to be retained or discarded on the back end, whichever the user may decide.

In [8], Goldstein introduces a probabilistic maximum likelihood pitch inference using STFT peak frequencies as feature observations. A related approach, developed by Thornburg and Leistikow [9], handles also spurious peaks from interference events, additionally incorporating timbral knowledge by statistically modeling joint frequency and amplitude peak observations. Furthermore, the Thornburg-Leistikow method may explicitly evaluate the likelihood of any candidate pitch component, rather than merely identifying the most likely component.

In this paper, we extend Thornburg and Leistikow’s single pitch likelihood evaluation to the multipitch case. Subsequently, the latter is embedded in a Bayesian chord identification schema inferring musical root, inversion, octave, and tuning as well as the individual pitch components comprising a chord.

2. PROBABILISTIC MODEL

Our probabilistic model is shown in Figure 1:

Figure 1: Probabilistic model.

Here, F consists of a list of observed STFT peak frequencies, and A the parallel list of amplitudes. Observed quantities are directly influenced by the following hidden attributes: tuning τ, octave O, root R, and chroma C. The latter are further influenced by higher-level contextual attributes: key K; mode M. Though key and mode inference proves difficult for single-frame data, the additional structure may facilitate multiframe extensions, as key and mode are likely to be constant across long segments of consecutive frames.
2.1. Attribute definitions

- **Tuning**: \( \tau \), in units of semitones, is discretized to a set of \( N_c \) equally spaced values \( \tau \in \{-0.5 + (l-1)/N_c\}_{l=1}^{N_c} \).
- **Key**: \( K \) takes the value of one of the twelve semitones in an octave \( K \in \{0, \ldots, 11\} \).
- **Mode**: The pair \( \{K, M\} \) designates what is usually called a “key” in music theory. For example: \( M \in \{\text{Major}, \text{minor}\} \).
- **Root**: \( R \) occupies one of the twelve semitones in an octave. The root is relative to a given key: \( R \in \{0, \ldots, 11\} \).
- **Octave**: \( O \) belongs to a consecutive integer range: \( O \in \{O_{\min}, \ldots, O_{\max}\} \).
- **Chroma**: \( C \) is represented by a set of intervals from the root. For instance, a minor triad is expressed: \( C = \{0, 3, 7\} \). Similarly, a Major-minor seventh chord admits the representation: \( C = \{0, 4, 7, 10\} \). \( C \) may belong to a standard codebook consisting of major, minor, augmented, and diminished triads, as well as the latter with major and minor sevenths added, with all standard inversions represented (three inversions in case of triads, four inversions in case of seventh chords). In total 44 chromas are represented of which 42 are unique thanks to the inherent ambiguity of augmented triad inversions.

2.2. Bayesian attribute inference

The factorization of the joint \( P(\tau, O, R, C, K, M, F, A) \), represented by the directed acyclic graph in Figure 1, is given as follows:

\[
P(\tau, O, R, C, K, M, F, A) = P(M)P(K)P(C|M) \times P(R|K, M)P(\tau|O, R, C) \times P(F, A|\tau, O, R, C) \tag{1}
\]

Priors, \( P(M), P(K), P(C|M), P(R|K, M), P(\tau|O, R, C) \) and \( P(F, A|\tau, O, R, C) \), encode domain-specific knowledge for a single STFT frame. The peak likelihood is given by \( P(F, A|\tau, O, R, C) \). Such constitutes the raw information needed to perform any attribute inference query.

Suppose we wish to identify some attribute, (say, a chroma \( C \) from the 44 possibilities), given only the peak observations \( F, A \). Our criterion is to construct a classifier \( T(F, A) = \hat{C} \) such that the probability of error (\( \hat{C} \neq C \)) is minimized. Formally, we desire:

\[
T^*(F, A) = \arg\min_{T(F, A)} P(T(F, A) \neq C) \tag{2}
\]

It is readily shown [10] that \( T^*(F, A) \) optimizing (2) maximizes the posterior \( P(C|F, A) \):

\[
T^*(F, A) = \arg\max_{C} P(C|F, A) \tag{3}
\]

The classifier (3) is called maximum a posteriori (MAP).

As optimal decisions involving any collection of musical attributes derive from analogous MAP rules, the key inference step involves the associated posterior. The latter may be derived by marginalizing irrelevant attributes from \( P(\tau, O, R, C, K, M|F, A) \). For instance, the associated posterior for chord recognition concerns chroma and root:

\[
P(C, R|F, A) = \sum_{\tau, O, K, M} P(\tau, O, R, C, K, M|F, A) \tag{4}
\]

2.3. Specification of priors

Priors \( P(M), P(K), P(C|M), P(R|K, M), P(\tau), P(\sigma) \), encoding domain-specific knowledge, become particularly concentrated when conditioning across frames, factoring across two levels: literal frame-to-frame continuities, and structural dependences across note transitions. Little can be said, however, when considering a single frame. Where we lack a priori knowledge altogether \( (K, M, \tau) \), maximum entropy arguments indicate the use of a uniform prior.

However, \( P(C|M) \) and \( P(R|K, M) \) may encode information specific to a given corpus. In functional tonal music, certain chroma are more common than others in a given mode. For instance, a major triad is far more likely than an augmented triad in Major mode. The latter influences \( P(C|M) \); for example: \( P(C = \{0, 4, 7\}|M = \text{Major}) > P(C = \{0, 4, 8\}|M = \text{Major}) \). Similarly, certain roots prove more common than others in a given key and mode. For instance, a chord rooted on the tonic of the key is more common than a chord rooted an augmented fourth higher. The latter influences \( P(R|K, M) \); e.g.: \( P(R = 0|K = 0, M = \text{Major}) > P(R = 6|K = 0, M = \text{Major}) \).

Specification of \( P(C|M) \) and \( P(R|K, M) \) by such common-sense reasoning may be suitable; however, we prefer to formally train these distributions from a corpus of musical data. We have not done so as of this writing; instead, we install uniform priors as placeholders, effectively removing \( K \) and \( M \) from the network.

3. MULTIPITCH LIKELIHOOD EVALUATION

The peak likelihood, \( P(F, A|\tau, O, R, C) \), is difficult to evaluate without knowing which observed peaks correspond to pitch components (and their associated harmonic numbers) and which peaks are altogether spurious. As such, we condition first upon a hidden layer, displayed in Figure 1. This layer consists of a set of pitch components, a template and a descriptor.

![Diagram](image-url)

Figure 2: Probabilistic model with exposed hidden layer.
3.1. Pitch components

For each set of attributes, there exist \( Q \) pitch components, where \( Q \) is the number of intervals in the chroma. Each \((k^{th})\) pitch component, \( k \in \{ 1, \ldots, Q \} \), is assigned a fundamental frequency \( f_0^{(k)} \) and a reference amplitude \( A_0^{(k)} \). Since the latter is unknown, \( A_0^{(k)} \) constitutes a nuisance parameter to be marginalized. Each fundamental frequency, \( f_0^{(k)} \), is computed accordingly:

\[
f_0^{(k)} = \frac{2\pi \cdot 2^{c_{440}}} {SR} \frac{[\pi^{12} + 12 + R]} {12 + c_{440}}
\]

where \( c_{440} = 4.0314 \) ensures \( A4 \leftrightarrow 440 \) Hz, and \( SR \) is the sampling rate in Hz.

3.2. Template

First consider the situation where every observed STFT peak results from, and hence may be linked to, exactly one harmonic of a single pitch component.

Define \( k^{(j)} \) to be the index of the pitch component to which the \( j^{th} \) STFT peak corresponds, and let \( h^{(j)} \) denote the harmonic number. In the absence of inharmonicity and noise, the ideal observed frequency would be \( h^{(j)} f_0^{(k^{(j)})} \).

Let \( F^{(j)} \) denote the peak frequency observation corresponding to the \( j^{th} \) observed STFT peak. We take the latter’s distribution to be Gaussian with mean \( F_{ib}^{(j)} \) and variance \( \lambda_{F,ib}^{(j)} \):

\[
F^{(j)} \sim \mathcal{N}(F_{ib}^{(j)}, \lambda_{F,ib}^{(j)})
\]

where

\[
F_{ib}^{(j)} = h^{(j)} f_0^{(k^{(j)})}
\]

\[
\lambda_{F,ib}^{(j)} = \Lambda_{F,ib}^{(j)} F_{ib}^{(j)} F^{(1)}
\]

where \( \Lambda_{F,ib}^{(j)} \) is a user-specified noise variance scaling.

Let \( A^{(j)} \) denote the peak amplitude observation corresponding to the \( j^{th} \) observed STFT peak. In the absence of noise, \( A^{(j)} = A_0^{(k^{(j)})} c_A^{(j)} \), where \( c_A \) is a user-specified spectral decay parameter. To allow for noise, \( A^{(j)} \) is most appropriately modeled as a scaled noncentral \( \chi^2 \) following [9]:

\[
P \left( \left[ A_{ib}^{(j)} \right]^2 / \Lambda_{A,ib}^{(j)} \right) \sim \chi^2 \left[ \Lambda_{A,ib}^{(j)} \right] / \Lambda_{A,ib}^{(j)} \] (8)

where

\[
A_{ib}^{(j)} = A_0^{(j)} c_A^{(j)}
\]

\[
\Lambda_{A,ib}^{(j)} = \Lambda_{A,ib}^{(j)} A_{ib}^{(j)} A_{ib}^{(1)}
\]

Of course, not all template peaks may appear in the STFT. Each template peak has a prior probability of being detected, denoted as \( P_{ib}^{(j)} \). The latter is modeled as decaying geometrically with the harmonic number:

\[
P_{ib}^{(j)} = \phi_b^{h^{(j)}} \] (10)

The aforementioned distributional parameters are organized into a template \( T \), containing all information necessary to evaluate \( P(F, A, \tau, O, R, C) \).

\[
T \triangleq \{ P_{ib}^{(j)}, \lambda_{F,ib}^{(j)}, A_{ib}^{(j)}, \lambda_{A,ib}^{(j)}, P_{ib}^{(1)} \} \}_{j=1}^{N_{ib}}
\]

3.3. Merging overlapped harmonics

A significant complication arises in the multipitch case where harmonics from different pitch components fail to be resolved by the STFT. The minimum frequency distance \( \Delta f \) between harmonics sufficient to resolve peaks depends on the analysis window’s length and shape, via the mainlobe width of the latter’s discrete-time Fourier transform (DTFT). For the length-\( M \) Hamming window used in our STFT front end, \( \Delta f = 8\pi / M \).

As each template peak is meant to describe the distribution of at most one observed STFT peak, we merge template peaks into clusters for which each peak frequency exists in a \( \Delta f \)-neighborhood of some other frequency within the cluster. As this clustering forms an equivalence relation, each peak in the original template is assigned to exactly one cluster.

Upon merging, each cluster, indexed by \( \{(j, l)\} \), is replaced by a single template peak. In the above, we let \( l^* \) refer to the index for which the amplitude noncentrality parameter \( \Lambda_{A,ib}^{(j,l)} \) is largest; we call the corresponding peak the primary peak.

Merged peak template parameters are obtained as follows.

- The merged frequency mean adopts the mean-amplitude-weighted average over the frequency means for each peak in the cluster:

\[
F_{ib}^{(j)} = \frac{\sum_{l=1}^{N_{ib}} A_{ib}^{(j,l)} F_{ib}^{(j,l)}} {\sum_{l=1}^{N_{ib}} A_{ib}^{(j,l)}} \] (12)

- The natural frequency variance is given via (7) as a continuous function of frequency. Hence, the frequency variances among all peaks in a given cluster should be roughly the same. We obtain the natural variance from the primary peak, and add to this the square of the spread of the frequency means within the cluster:

\[
\lambda_{F,ib}^{(j)} = \lambda_{F,ib}^{(j,l^*)} + \left( \max_{l} \left( F_{ib}^{(j,l)} - \min_{l} F_{ib}^{(j,l)} \right) \right)^2 \] (13)

- Peaks may overlap in any phase relationship. As such, the merged peak’s noncentrality parameter is taken to equal that of the primary peak, while the scale parameter adds to that of the primary peak’s squared amplitudes of the other peaks within the cluster. This scale parameter specification becomes exaggerated, accounting for only the worst cases: where all peaks interfere exactly in phase, or where all peaks but the primary peak interfere with the latter exactly \( 180^\circ \) out of phase:

\[
A_{ib}^{(j)} = A_{ib}^{(j,l^*)} \] (14)

\[
\lambda_{A,ib}^{(j)} = \lambda_{A,ib}^{(j,l^*)} + \sum_{l=1, l \neq l^*}^{N_{ib}} [A_{ib}]^2 \] (15)

- The merged peak’s survival probability is taken to equal that of the primary peak:

\[
P_{ib}^{(j)} = P_{ib}^{(j,l^*)} \] (16)
3.4. Descriptor and linkmap representation

Of course, the template only accounts for potential STFT peaks, as excessive noise and other types of interference may prevent the detection of some peaks associated with the template. Furthermore, interference events may generate spurious peaks in the observed peaklist which have no relation to those in the template. To evaluate the likelihood of the observed peaklist, then, we condition upon a descriptor $D$ encoding linkage between template and observed peaks. The desired unconditional likelihood $P(F, A|T)$ evaluates by summing over conditional likelihoods $P(F, A|D, T)$ weighted by an appropriate prior $P(D|T)$:

$$P(F, A|T) = \sum_D P(F, A|D, T)P(D|T)$$  \hspace{1cm} (17)

In [9], the authors propose a symbolic encoding of $D$ enabling distributions analogous to $P(F, A|D, T)$ for the single-pitch case to be written in closed algebraic form. In this paper, we adopt only the graphical linkmap representation, shown in [9] to be equivalent. Figure 3 illustrates an example linkmap. The scenario represents a perfect fifth interval where the third harmonic from the root and the second harmonic from the fifth are merged. Three pairs of linked peaks, two undetected peaks, and two spurious peaks are evinced by the Figure. Evaluating $P(F, A|D, T)$ and $P(D|T)$ requires a probabilistic model for the generation of spurious peaks. The spurious peak frequency distribution is modeled according to a Poisson process, while the squared-amplitude distribution is modeled according to a scaled $\chi^2$ distribution with scaling parameter $\sigma^2_{A,o}$ (see [9], Section 2).

Robustness in the presence of spurious peaks is primarily due to the difference in conditional likelihoods between spurious and linked peaks. If one of the peak amplitudes/frequencies is highly inconsistent with respect to all template peak distributions, the contribution $P(F, A|D, T)P(D|T)$ will be quite small for those $D$ for which the peak is linked, relative to those $D$ for which this peak is spurious. Hence, most of the unconditional likelihood will concentrate in $D$ for which this peak is spurious.

We now consider the benefits of merging. Figure 4 shows a hypothetical situation prior to merging template peaks. Since the amplitude of the overlap-interference peak is highly inconsistent with the template peak distributions (9), the present solution ends up discarding information from overlap-interference peaks altogether, effectively labeling them as spurious\(^1\). While overlap-interference peak amplitudes may be unreliable, the associated frequencies are especially likely to carry useful information as they correspond to

\(^1\)Technically speaking, the conditional likelihood concentrates in descriptors $D$ for which interference peaks are unlinked.

![Figure 3: Linkmap example; perfect fifth interval. All frequencies are expressed in ratios to the root’s fundamental frequency.](image)

![Figure 4: Illustration of STFT interference peak due to overlapping harmonics in the context of linked and spurious amplitude distributions.](image)

3.5. MCMC unconditional likelihood evaluation

The number of possible linkmaps and hence descriptors, with $N_{ib}$ template peaks and $N$ observed peaks, is as follows:

$$\#\{D\} = \sum_{n=0}^{\text{min}(N_{ib},N)} \binom{N_{ib}}{n} \binom{N}{n}$$  \hspace{1cm} (18)

If $N_{ib} = N$, (18) and Stirling’s approximation [11] yield:

$$\#\{D\} \approx 4^N \frac{N!}{\sqrt{\pi N} \left(1 - O\left(\frac{1}{N}\right)\right)}$$  \hspace{1cm} (19)

A further complication arises in that $P(F, A|D, T)$, via (9,11), requires knowledge of the reference amplitudes $A_{0,Q} \triangleq \{A_{0,k}^{(k)}\}_{k=1}^Q$, where $Q$ denotes the number of pitch components. Our solution is to marginalize the $A_{0,Q}$. Here, each $A_{0,k}^{(k)} \in A_{0,Q}$ is discretized to a grid $A$, the latter consisting of $N_A$ amplitudes uniformly positioned in dB space on a $[-9\,\text{dB}, +9\,\text{dB}]$ interval relative to the highest amplitude peak. Thus $A_{0,Q}$ belongs to the product space $A^Q$ consisting of $N_A^Q$ discrete possibilities. A uniform prior $P(A_{0,Q})$ is placed on these possibilities. Marginalization of $A_{0,Q}$ and (17) yield:

$$P(F, A|T) = \sum_{D \in \mathcal{D}, A_{0,Q} \in A^Q} P(F, A|D, T(A_{0,Q}))P(A_{0,Q})P(D|T(A_{0,Q}))$$  \hspace{1cm} (20)

The number of terms in the summation (20) grows exponentially with the common number of template and observed peaks, as well as the number of pitch components.
To facilitate the computation, we introduce a Markov chain Monte Carlo (MCMC) approximate enumeration, analogous to Section 4 of [9]. In practice, virtually all of the unconditional likelihood concentrates in a few \( \{D, A_{0,Q}\} \)-possibilities. Our goal then becomes to construct a Markov chain traversing the \( \{D, A_{0,Q}\} \) with the highest contributions to the sum (20). To this end, we specify the stationary distribution \( \pi(D, A_{0,Q}) \), as proportional to \[ P(F, A|D, T(A_{0,Q}))P(A_{0,Q})P(D|T(A_{0,Q})) \], where \( \gamma > 1 \).

The Metropolis-Hastings rule [12] is a general method for constructing a Markov chain admitting a desired \( \pi(D, A_{0,Q}) \), with a user-specified sampling distribution \( q(D', A_{0,Q}|D, A_{0,Q}) \). As long as \( q \) is irreducible, the algorithm will converge, though arbitrarily slowly unless this sampling distribution is well chosen. Our choice mixes alterations in \( A_{0,Q} \) (via \( q(A|A_{0,Q}|A_{0,Q}) \)) independently with alterations in \( D \) (via \( q(D|D'|D) \)), each chosen with probability 0.5.

In case of \( q_A \), we select one \( A_0^{(k)} \) uniformly among the \( Q \) possibilities and move it up or down one gridpoint (\( \Delta dB \) level), except for the boundary cases where only one adjacency exists. In case of \( q_D \), we exploit a similar notion of adjacency on the linkmaps. The following types of moves exist, as shown in Figure 5: 1) Remove a link; 2) Add a nonintersecting link; 3) Switch a link to the adjacent position on output; 4) Switch a link to the adjacent position on output.

Figure 5: Example linkmap moves.

A move type is selected with uniform probability over the types with at least one move possibility, then a move is selected uniformly among those possibilities.

The general Metropolis-Hastings rule proceeds over iterations \( i \) as follows. As a shorthand, the define \( S(i) = \{D(i), A_{0,Q}(i)\} \), and define \( S(i), S(i+1) \) analogously.

1. Sample \( S(i) \sim q(S(i)|S(i)) \)

2. Select

\[
S(i+1) = \begin{cases} 
S(i) & \text{w. prob } \alpha(S(i), S(i)) \\
S(i) & \text{w. prob } 1 - \alpha(S(i), S(i)) 
\end{cases}
\]

where

\[
\alpha(S(i), S(i)) = \min \left( 1, \frac{\pi(S(i))q(S(i)|S(i))}{\pi(S(i))q(S(i)|S(i))} \right)
\]

The irreducibility of \( q \) follows from the irreducibilities of \( q_D \) and \( q_A \) and the fact either distribution may be selected with probability 0.5, which is strictly positive. The irreducibility of \( q_A \) obtains since one may traverse any configuration of grid points in \( A \) for each \( A_{0,Q} \) by accumulating adjacent steps, each with probability at least \( 1/(2Q) \). Similarly, the irreducibility of \( q_D \) follows since one can reach any linkmap from any other linkmap by removing and adding links one-by-one. There are only finitely many such possibilities each of which occurs with strictly positive probability.

The initialization of \( A_{0,Q} \) proceeds by taking all elements \( A_{0,Q}^{(k)} \) equal to the maximum STFT peak amplitude. The initialization of \( D, A_{0,Q} \) follows by McAulay-Quatieri peak matching [9], [13]. In practice, we obtain rapid convergence using only \( \sim 600 \) MCMC iterations. Subsequent optimizations are as follows: first, we hash intermediate computations for previously visited \( D \) and/or \( A_{0,Q} \) values; second, we vary \( \gamma \) according to the annealing schedule \( \gamma(i) = 0.1, \gamma(i+1) = \min (1.005\gamma(i), 5) \), allowing a wider range of \( \{D, A_{0,Q}\} \) possibilities to be visited at the outset.

### Table 1: Prior and posterior chroma probabilities.

<table>
<thead>
<tr>
<th>True Chroma</th>
<th>Candidate Chords</th>
<th>Prior</th>
<th>Posterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maj</td>
<td>Maj</td>
<td>1.000007</td>
<td>1.000007</td>
</tr>
<tr>
<td>Maj</td>
<td>Maj</td>
<td>1.000007</td>
<td>2.31E-06</td>
</tr>
<tr>
<td>Maj</td>
<td>Maj</td>
<td>1.000007</td>
<td>0.000000</td>
</tr>
<tr>
<td>Maj</td>
<td>Maj</td>
<td>1.000007</td>
<td>0.000000</td>
</tr>
<tr>
<td>Maj</td>
<td>Maj</td>
<td>1.000007</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

The irreducibility of \( q \) follows from the irreducibilities of \( q_D \) and \( q_A \) and the fact either distribution may be selected with probability 0.5, which is strictly positive. The irreducibility of \( q_A \) obtains since one may traverse any configuration of grid points in \( A \) for each \( A_{0,Q} \) by accumulating adjacent steps, each with probability at least \( 1/(2Q) \). Similarly, the irreducibility of \( q_D \) follows since one can reach any linkmap from any other linkmap by removing and adding links one-by-one. There are only finitely many such possibilities each of which occurs with strictly positive probability.

The initialization of \( A_{0,Q} \) proceeds by taking all elements \( A_{0,Q}^{(k)} \) equal to the maximum STFT peak amplitude. The initialization of \( D, A_{0,Q} \) follows by McAulay-Quatieri peak matching [9], [13]. In practice, we obtain rapid convergence using only \( \sim 600 \) MCMC iterations. Subsequent optimizations are as follows: first, we hash intermediate computations for previously visited \( D \) and/or \( A_{0,Q} \) values; second, we vary \( \gamma \) according to the annealing schedule \( \gamma(i) = 0.1, \gamma(i+1) = \min (1.005\gamma(i), 5) \), allowing a wider range of \( \{D, A_{0,Q}\} \) possibilities to be visited at the outset.

The irreducibility of \( q \) follows from the irreducibilities of \( q_D \) and \( q_A \) and the fact either distribution may be selected with probability 0.5, which is strictly positive. The irreducibility of \( q_A \) obtains since one may traverse any configuration of grid points in \( A \) for each \( A_{0,Q} \) by accumulating adjacent steps, each with probability at least \( 1/(2Q) \). Similarly, the irreducibility of \( q_D \) follows since one can reach any linkmap from any other linkmap by removing and adding links one-by-one. There are only finitely many such possibilities each of which occurs with strictly positive probability.

The initialization of \( A_{0,Q} \) proceeds by taking all elements \( A_{0,Q}^{(k)} \) equal to the maximum STFT peak amplitude. The initialization of \( D, A_{0,Q} \) follows by McAulay-Quatieri peak matching [9], [13]. In practice, we obtain rapid convergence using only \( \sim 600 \) MCMC iterations. Subsequent optimizations are as follows: first, we hash intermediate computations for previously visited \( D \) and/or \( A_{0,Q} \) values; second, we vary \( \gamma \) according to the annealing schedule \( \gamma(i) = 0.1, \gamma(i+1) = \min (1.005\gamma(i), 5) \), allowing a wider range of \( \{D, A_{0,Q}\} \) possibilities to be visited at the outset.
Table 2: Prior and posterior octave probabilities.

Table 3: Prior and posterior tuning probabilities.

5. REFERENCES


A NEW SCORE FUNCTION FOR JOINT EVALUATION OF MULTIPLE F0 HYPOTHESES

Chunghsin Yeh, Axel Röbel
Analysis-Synthesis Team, IRCAM, Paris, France
cyeh@ircam.fr roebel@ircam.fr

ABSTRACT

This article is concerned with the estimation of the fundamental frequencies of the quasiharmonic sources in polyphonic signals for the case that the number of sources is known. We propose a new method for jointly evaluating multiple F0 hypotheses based on three physical principles: harmonicity, spectral smoothness and synchronous amplitude evolution within a single source. Given the observed spectrum a set of F0 candidates is listed and for any hypothetical combination among the candidates the corresponding hypothetical partial sequences are derived. Hypothetical partial sequences are then evaluated using a score function formulating the guiding principles in mathematical forms. The algorithm has been tested on a large collection of artificially mixed polyphonic samples and the encouraging results demonstrate the competitive performance of the proposed method.

1. INTRODUCTION

The estimation of the fundamental frequency, or F0, of a sound source from a given signal is an essential step for many signal processing applications. For the monophonic case there exist many approaches that achieve very high performance. Despite increasing research activities with respect to polyphonic signals the estimation of multiple F0s remains a challenging problem. Some of the generally admitted difficulties are: estimating the number of F0s, retrieving reliable time-frequency properties, treating mixtures of transient parts and stationary parts. In the following article, we propose a new method for multiple F0 estimation under the assumption that the number of F0s is known in advance.

There exist several approaches for multiple F0 estimation. A probabilistic signal modeling approach proposed in [1] applies specific prior distributions on the model parameters, such as the frequency and the amplitude of each partial, the number of partials, the detuning factor for each sinusoidal component, etc. This approach is computationally expensive and limited results are reported. In [2], a robust multipitch estimation is achieved by means of selecting reliable frequency channels as well as reliable peaks in the normalized correlograms. This technique has been reported to work for two-voice speech and the authors conclude that the proposed algorithm could be extended to more than two pitches. Klapuri’s iterative multiple F0 estimation algorithm handles most of the difficulties like estimating the number of F0s and treating the overlaps of coincident partials. Promising results are reported by evaluating a variety of polyphonic musical signals.

An iterative estimation and cancellation model has been proposed by de Cheveigné earlier in [3]. He compared an iterative approach and a full search approach which performs a joint evaluation. Based on this early study and later work in [3], he reported that a joint cancellation performs better than an iterative cancella-

tion in that a single F0 estimation failure may lead to successive errors in an iterative estimation cancellation manner. In fact, a joint evaluation strategy provides more flexibility in solving this problem. For each set of multiple F0 hypotheses, spectral components in the interleaved spectrum could be reasonably allocated to each F0 hypothesis and disturbed information provided by overlapped partials could be identified and taken care of in a more accurate way.

Therefore, we propose a new method for the joint evaluation of multiple F0 hypotheses. Based on a generative quasiharmonic spectral model, hypothetical partial sequences are constructed and evaluated using three physical principles: harmonicity, spectral smoothness and synchronous amplitude evolution within a single source. Harmonicity is the essential principle in nearly all F0 estimation techniques. It is known that using only harmonicity, however, often causes subharmonic/superharmonic ambiguity and thus more cues are necessary to improve the estimation performance. Both Kashino [4] and Goto [5] introduce tone models as a constraint on relative partial amplitudes. Klapuri has utilized the spectral smoothness principle [7] which assumes that the spectral envelopes of natural quasiharmonic sounds are in general rather smooth. Besides the two principles applied by the above authors, we include the synchronous evolution of sinusoidal amplitudes as another principle and finally formulate these principles into a new score function to rank all hypothetical combinations, which is one important contribution of this article. The second contribution is a new proposition to make use of the hypothetical F0s to determine reliable information in the observed spectrum.

This paper is organized as follows. In Section 2 the generative quasiharmonic model is described and the principles for F0 estimation are established. In Section 3 we introduce a frame-based F0 estimation method using the proposed score function. In Section 4 experimental results are shown, which proves the competitive performance of the proposed method. Finally, further improvements are discussed and conclusions are drawn.

2. GENERATIVE QUASIHARMONIC MODEL

The following algorithm is based on a polyphonic quasiharmonic signal model of the following form

\[ y[n] = \{ \sum_{m=1}^{M} \sum_{n=1}^{H_{m}} a_{m,n} \cos \left( (1 + \delta_{m,n}) \omega_{m} \right) + \phi_{m}[n] \} + v[n], \]

(1)

where \( n \) is the discrete time index, \( M \) is the number of sources, \( H_{m} \) is the number of partials for the \( m \)-th source, \( \omega_{m} \) represents the F0 of source \( m \), and \( \phi_{m}[n] \) denotes the phase. In the current context those parameters are either fixed or of minor interest. The
score function will make use of $a_{m,h_0}[n]$ and $\delta_{m,h_0}$, which are the time varying amplitude and the constant frequency detuning of the $h_0$-th partial and $v[n]$, which is the residual noise component. Generally it is supposed that the noise is sufficiently small such that a considerable part of the individual sinusoidal components can be identified.

Similar to [8] we understand the observed spectrum as generated by sinusoidal components and noise. Each spectral peak is characterized by its amplitude and frequency. A sinusoidal peak is assigned to one or more of the $M$ sources in equation [1], all unassigned peaks contribute to the noise component $v[n]$. The model supposes quasi-stationary frequency and, therefore, the sinusoidality of an observed peak is used to rate the requirement to include it into the quasiharmonic parts of the source model. Based on this model and given the observed spectrum and $M$, the most plausible $F_0$ hypotheses are going to be inferred. The procedure is close to the Bayesian model specified in [1], however, to prevent the huge computational requirements of numerically maximizing the likelihood a more pragmatic approach is proposed.

To construct and evaluate hypothetical sources, we use three physical principles for quasiharmonic sounds stated in the following.

**Principle 1**: Spectral match with low inharmonicity. For a $F_0$ hypothesis, a hypothetical partial sequence HPS$_{F_0}$ is constructed by selecting harmonically matched peaks from the observed spectrum in such a way that $\delta_{m,h}$ are minimized. The set HPS$_{F_0}$ should combinatorially “explain” the sinusoidal components in the observed spectrum. Under the assumption that the noise energy is small it is reasonable to favor $F_0$ hypotheses that explain more components of the observed spectrum as long as they are not contradicted by the following two principles.

**Principle 2**: Spectral smoothness. For natural quasiharmonic sounds, the spectral envelopes usually form smooth contours. While constructing HPS$_{F_0}$ of a source, the partials should be selected in a way that $\{a_{m,h_0}\}_{h_0=1}^{M}$ results in a smooth spectral envelope. For partial sequences fitting well to Principle 1, those with smoother spectral envelopes are more probable to be originated from natural sources such as musical instruments.

**Principle 3**: Synchronous amplitude evolution within a single source. Partial belonging to the same source should have similar time evolution of the amplitudes $\{a_{m,h_0}\}_{h_0=1}^{H}$ collected in a HPS. If the partials of a hypothetical source match mostly to noisy peaks, they evolve in a random manner and thus do not have a synchronous amplitude evolution.

3. MULTIPLE $F_0$ ESTIMATION

Based on the three principles described above, we design a frame-based multiple $F_0$ estimation system. The main task is to formulate these principles into four criteria serving as the core components in a score function for evaluating the plausibility of one set of multiple $F_0$ hypotheses.

3.1. Front end

3.1.1. Extracting hidden partials

When analyzing polyphonic signals with limited spectral resolutions, one often observes that the dense distribution of partials causes some peaks to be hidden by relatively larger coincident ones. Thus, extracting hidden partials is essential to increase spectral resolution, which leads to a more accurate harmonic matching in the later stage. As shown in the top of Figure 1 a peak of unsymmetric form might correspond to overlapped partials.

![Figure 1: Extracting the hidden partial](image)

To search for these hidden partials, we use a simple symmetry test for the shapes of the observed peaks. For each peak, we locate its neighboring valleys and choose the closer one to define a reference range (the bin number from one observed peak to its nearest valley). The degree of symmetry is defined as the summation of amplitude differences between the two sides of a spectral peak, considering the frequency bins within the reference range. Then a threshold is set for the degree of symmetry to select relatively unsymmetric peaks for further processing. After estimating the frequency and the frequency slope of each selected peak [9], we subtract it using the least square error criterion to extract the hidden peak as indicated in the bottom plot of Figure 1. To prevent the addition of simple residual energy as a new sinusoid, a resolved peak is kept as a successfully extracted partial only if it is not weaker than the original peak by $40$ dB and should be located further than half the mainlobe width away from the original peak.

3.1.2. Generating the candidate list

To generate a $F_0$ hypothesis list, we use an harmonic matching technique since harmonicity is the primary concern in $F_0$ estimation. The harmonic matching technique matches the regular spacing between adjacent partials to determine a coherent $F_0$ and has been widely used in $F_0$ estimation in the spectral domain [10].

Given a $F_0$, we construct a vector $d_{F_0}$ evaluating the degree of deviation from a harmonic model to the observed peaks. A tolerance interval around each harmonic is used to measure the goodness of the harmonic match. For the $i$-th observed peak matching the $h$-th harmonic, the degree of deviation is formulated as

$$d_{F_0}(i) = \frac{|f_{\text{peak}}(i) - F_{\text{model}}(h)|}{\alpha \cdot f_{\text{model}}(h)} \tag{2}$$

where $f_{\text{peak}}(i)$ is the frequency of the $i$-th observed peak, $F_{\text{model}}(h)$ is the frequency of the $h$-th harmonic of the model, and $\alpha$ determines the tolerance interval $2 \cdot \alpha \cdot f_{\text{model}}(h)$. If an observed peak situates outside the corresponding tolerance interval, it is regarded as unmatched and $d_{F_0}(i)$ is set to 1.
Since inharmonicity exists in most of the string instruments, it is necessary to dynamically adapt the frequencies of model harmonics according to the matched peaks. Thus, $f_{model}(h) = f_0 + (h-1) \cdot F0$ is calculated by means of adding $F0$ to the previously matched peak frequency. If not a single peak is matched for the previous partial, $f_{model}(h-1) + F0$ is used for the current match. The technique of selecting one single matched peak (among all the peaks situating in the tolerance interval) as a reference position makes use of Principle 2 and is described later.

Three vectors are chosen to weight $d_{F0}$: (i) the complex correlation between each observed peak and an ideal peak defined by the analysis window, (ii) the linear amplitudes of the observed peaks, and (iii) an attenuation vector favoring the first several partials\(^1\) as indicated in the top plot of Figure 2.

![Figure 2: Harmonic matching: a tenor trombone note at 137Hz](image)

The complex correlation favors peaks of better sinusoidality (shape and phase). The linear peak amplitude adjusts relative significance by considering peaks of larger energy more important. The third weighting vector attenuates less reliable matches for higher partials because they tend to be inharmonic and non-stationary. Besides, the gradual decay nature of higher partials reduces the reliability in the presence of stronger partials from other sources. Then the weighted deviation vector is summed and normalized between 0 and 1. The resulting indicator for harmonic matching is denoted as $D$. An example is shown in the bottom plot of Figure 2 the weighted sums of the deviation vectors for $F0$ hypotheses ranging from 50Hz to 2000Hz are plotted. A lower value means a better match and thus higher harmonicity. The harmonic matching indicator is applied to polyphonic spectra to select $F0$ candidates corresponding to local minima of $D$ for the joint evaluation.

Assume there are $P$ $F0$s in the candidate list and there are $M$ $F0$s to be estimated from the observed spectrum which results in the need to evaluate $C^P_M$ combinations of $F0$ hypotheses.

3.1.3. Generating Hypothetical Partial Sequences

Constructing $HPS$s of $F0$ hypotheses in the candidate list is realized by the partial selection technique. Both Parsons\(^{11}\) and Duijghuis\(^{12}\) have proposed selecting the nearest peak around a harmonic. However, this technique might fail if a partial is surrounded by spurious peaks and partials of other sources. Therefore, we try to increase the robustness by means of utilizing Principle 2 and the knowledge of spectral locations where partial overlaps may occur according to the current $F0$ hypotheses under investigation. The goal is to make the best of the available credible information.

The construction procedure has two steps: (i) Each $HPS$ is constructed by assigning the most plausible peaks, and (ii) the overlapped partials containing less credible amplitudes are removed from $HPS$ to ensure reliability for evaluating the spectral envelope in the score function.

To construct a $HPS$ we start with the first partial by simply assigning it to the closest peak observed. For the following partials we consider two candidate peaks: the closest one and the one of which the mainlobe contains the corresponding harmonic position. Compared to the formerly selected partials, the peak candidate forming a smoother envelope is sequentially allocated to the $HPS$. The case of overlapped partials requires special consideration. The treatment for this case is based on the idea that an overlapped partial still carries important information for at least the $HPS$ that locally has the strongest energy. Therefore, the algorithm aims to assign the overlapped partial to this $HPS$. The strategy for treating the overlapped partials is listed below:

(i) Partial having potential collision are determined from each hypothetical combination of $HPS$s.

(ii) The local energy strength of the envelope is obtained by means of interpolating the neighboring partial amplitudes that are not collided. By comparing the interpolated amplitudes estimated from all $HPS$s, the overlapped partials is exclusively assigned to the one having the most dominant interpolated amplitude among all and then labeled as “usable” which means that it could be used for interpolation for its neighboring partials. For the rest of the $HPS$s the overlapped partial is labeled as existing but without a specified partial amplitude.

(iii) If one neighboring partial happens to be overlapped, the non-overlapped partial at the other side is used instead. If the two neighboring partials are overlapped, the corresponding $HPS$ is not considered as having reliable information for interpolation and thus excluded.

(iv) If the amplitude of the overlapped partial is smaller than any interpolated amplitude, it is difficult to infer which $F0$ hypothesis contributes the most and thus partial assignment is not carried out but this overlapped peak in all $HPS$s are labeled as “usable” for further use of interpolation.

The score criteria explained in the following are designed to gracefully deal with this kind of incomplete $HPS$s. An example of treating the overlapped partials in $HPS$s of three notes is shown in Figure 3. The above plot shows the $HPS$s before the treatment and the bottom plot shows those after the treatment.

3.2. The score function

Having constructed the most reasonable peak sequences for each set of $F0$ hypotheses we design a score function to rank these hypothetical sets. The score function formulates the three principles into four criteria: harmonicity $HAR$, mean bandwidth $MBW$ and duration $DUR$ of the partial amplitude sequence, and the standard deviation of mean time $DEV$.

\(^{1}\)The third partial is tested to be a good starting point for attenuation.
**Figure 3:** Overlapped partial treatment

**Criterion 1**  
HAR is an indication of harmonicity and totally "explained" energy. It is formulated as

\[
HAR = \frac{\sum_{i=1}^{I} \text{Corr}(i) \cdot \text{Spec}(i) \cdot d_M(i)}{\sum_{i=1}^{I} \text{Corr}(i) \cdot \text{Spec}(i)}
\]  

(3)

where \( I \) is the number of peaks, \( i \) is the peak index, \( \text{Corr} \) is the complex correlation weighting vector, \( \text{Spec} \) is the linear peak amplitude and \( d_M(i) \) is obtained by combining \( \{d_{F0_m}(i)\}_{m=1}^{M} \) at the \( m \)-th peak in the following way:

\[
d_M(i) = \min(\{d_{F0_m}(i)\}_{m=1}^{M})
\]  

(4)

That is, each observed peak is matched with the closest partial among those of \( \{HPS_{F0_m}\}_{m=1}^{M} \) and thus each combination under evaluation could perform its optimal match.

**Criterion 2**  
To evaluate the smoothness of a HPS, we calculate the mean bandwidth \( MBW_{F0_m} \) of the partial amplitude sequence. Each HPS is assembled with its "mirror sequence" to construct a new sequence \( S_{F0_m} \) for further evaluation. It could also be interpreted as a hypothetical partial sequence constructed from a complex spectrum. An example of \( S_{F0_m} \) is shown in the middle plot of Figure 4.

Applying \( K \)-point Fast Fourier Transform on \( S_{F0_m} \) to obtain the linear spectral amplitude vector \( X_{F0_m} \), we can calculate the mean bandwidth \( MBW_{F0_m} \) as

\[
MBW_{F0_m} = \sqrt{\frac{\sum_{k=1}^{K/2} k^2 |X_{F0_m}(k)|^2}{\sum_{k=1}^{K/2} |X_{F0_m}(k)|^2}}
\]  

(5)

This indicates the degree of energy concentration in low frequency region and thus \( S_{F0_m} \) with less variation results in a smaller value of \( MBW_{F0_m} \).

The function of \( MBW_{F0_m} \) is to discriminate correct \( F0s \) from subharmonics. As the example shown in Figure 4, the spectral envelopes of a harpsichord note. Although the nature of the harpsichord does not form a smooth spectral envelope due to resonance, the HPS of its subharmonic \( F0/2 \) contains even more variations and thus larger \( MBW_{F0_m} \).

**Figure 4:** Spectral smoothness comparison between \( F0 \) and \( F0/2 \)

**Criterion 3** For a quasiharmonic sound, the spectral centroid usually lies around lower partials. Applying this general principle related to **Principle 2**, we could similarly evaluate the energy spread of the partial sequence, that is, the duration \( DUR_{F0_m} \) of \( HPS_{F0_m} \). Instead of removing the non-reliable components from \( HPS_{F0_m} \), we simply set them to zero to maintain correct positioning of all partials. Then the duration of \( HPS_{F0_m} \) could be calculated as

\[
DUR_{F0_m} = \sqrt{\frac{2}{L} \sum_{n=1}^{N_m} \frac{\|HPS_{F0_m}(n)\|^2}{\|HPS_{F0_m}(n)\|^2}}
\]  

(6)

where \( N_m \) is the length of \( HPS_{F0_m} \), \( L \) is a normalization factor determined by \( F_{90}/F_{0_{\min}} \), where \( F_{90} \) stands for the frequency limit containing 90% of spectral energy in the analyzing frequency range and \( F_{0_{\min}} \) is the minimal hypothetical \( F0 \) in search. Since spectral envelopes of natural sounds are not always smooth, this criterion functions as the further test of physical consistency of **Principle 2** and acts as a penalty function for subharmonics which "explain" more than one source in the observed spectrum.

**Criterion 4** To evaluate the synchronicity of the temporal evolution of the hypothetical sinusoidal components in a HPS, we rely on the estimation of the mean time for individual spectral peaks. Mean time is an indication of the center of gravity of signal energy[13] and the mean time of a spectral peak can be used to characterize the amplitude evolution of the related signal[14]. For a coherent HPS we expect synchronous evolution resulting in a small variance of the mean time for the HPS of a single source.

The mean time of a hypothetical source, denoted as \( T_{F0_m} \), is calculated as the power spectrum weighted sum of the mean time of the hypothetical partials. The variance of mean time of the partials in \( HPS_{F0_m} \) is then

\[
VAR_{F0_m} = \frac{1}{I} \sum_{i=1}^{I} \left( T_{F0_m} - T_{F0_m}(i) \right)^2 \cdot w_{F0_m}(i)
\]  

(7)

where \( T_{F0_m}(i) \) denotes the mean time of the \( i \)-th observed peak and the weighting vector \( \{w_{F0_m}(i)\}_{i=1}^{I} \) is constructed by the following steps:

1) Initially set \( \{w_{F0_m}(i)\}_{i=1}^{I} \) as the linear peak amplitude vector.
2) For the peaks situating too close in the observed spectrum, their spectral phases are probably disturbed. Therefore, we set the corresponding component in \( \{w_{F0m}(i)\}_{i=1}^M \) to 0.

3) According to the treatment of overlapped partials among \( \{HPS_{F0m}\}_{m=1}^M \), the components of \( \{w_{F0m}(i)\}_{i=1}^M \) corresponding to unusable partials are set to 0.

4) \( \{w_{F0m}(i)\}_{i=1}^M \) is then compressed by an exponential factor to reduce the dynamic range such that the sign change of noisy peaks is raised. This makes use of noisy peaks to penalize a hypothetical partial sequence containing more noisy peaks. Finally, \( \{w_{F0m}(i)\}_{i=1}^M \) is normalized to be a weighting vector.

\( DEV_{F0m} \) is then defined as the square root of \( VAR_{F0m} \) divided by half of the window size.

For each combination under investigation, \( MBW \) of a set of \( F0 \) hypotheses is defined as the weighted sum of \( \{MBW_{F0m}\}_{m=1}^M \):

\[
MBW = \sum_{m=1}^{M} \left( \frac{\sum_{n=1}^{N_{MBW}} HPS_{F0m}(n) \cdot MBW_{F0m}}{\sum_{m=1}^{M} \sum_{n=1}^{N_{MBW}} HPS_{F0m}(n)} \right)
\]  

(8)

This makes use of the credible components in each \( HPS_{F0m} \) as a weighting of relative importance. \( DUR \) and \( DEV \) are thus equivalently defined.

**Score function** We define the score function as

\[
D_{C_i}^{p_j} = \frac{1}{\sum_{j=1}^{4} p_j} \{p_1 \cdot HAR + p_2 \cdot MBW + p_3 \cdot DUR + p_4 \cdot DEV\}
\]  

(9)

where the weighting coef cients \( \{p_j\}_{j=1}^4 \) are to be trained by an evolutionary algorithm [15]. The score function is designed in a way that smaller values stand for higher scores. Notice that \( HAR \) generally favors lower hypothetical \( F0s \) while \( MBW, DUR \) and \( DEV \) favor higher ones. Therefore, the criteria perform in a complementary way and the weighting coef cients should be optimized to balance the relative contribution of each criterion such that the score function generally supports correct \( F0s \) the best.

### 4. EXPERIMENTAL RESULTS

To evaluate the proposed \( F0 \) estimation method, we perform a frame-based test using mixtures of musical samples. Since the criteria are designed for stationary quasiharmonic sounds, stationarily assigned and then samples ranging from 65Hz(C2) to 1980Hz(B6) are randomly selected to mix. Around 1500 samples for each database are generated in a way that each combination of note names are of equal proportion. Musical instruments not tting the quasiharmonic model are excluded. This database contains about 30 different musical instruments. To facilitate comparison, the database is published on the rst author’s web page.

The search range for \( F0 \) is set from 50Hz to 2000Hz and the maximal analyzing frequency limit is \( x \) ed at 5000Hz. A Blackman window is used for analysis and all parameters \( x \) ed for this evaluation.

Multiple \( F0 \) reference tables are built from single \( F0 \) estimation of monophonic samples before mixing. A correct estimate should not deviate from the corresponding reference value by \( 5\% \). The error rates are computed by the number of error estimates divided by the total number of target \( F0s \).

<table>
<thead>
<tr>
<th>polyphony</th>
<th>non-harmonical</th>
<th>harmonical</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWO</td>
<td>0.58%</td>
<td>7.28%</td>
<td>2.09%</td>
</tr>
<tr>
<td>THREE</td>
<td>1.48%</td>
<td>5.16%</td>
<td>2.68%</td>
</tr>
<tr>
<td>FOUR</td>
<td>2.46%</td>
<td>6.57%</td>
<td>4.50%</td>
</tr>
</tbody>
</table>

Table 1: \( F0 \) estimation results using a 186 ms window

### 4.2. Evaluation setups and results

Specifications for this evaluation are described below:

- Three databases: two-voice, three-voice and four-voice mixtures, labeled as TWO, THREE and FOUR respectively, are generated using McGill University Master Sample.[16] In combining \( M \)-voice polyphonic samples, \( M > 2 \) out of twelve (C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B) tones are preliminarily assigned and then samples ranging from 65Hz(C2) to 1980Hz(B6) are randomly selected to mix. Around 1500 samples for each database are generated in a way that each combination of note names are of equal proportion. Musical instruments not tting the quasiharmonic model are excluded. This database contains about 30 different musical instruments. To facilitate comparison, the database is published on the rst author’s web page.[16]

- The search range for \( F0 \) is set from 50Hz to 2000Hz and the maximal analyzing frequency limit is \( x \) ed at 5000Hz. A Blackman window is used for analysis and all parameters \( x \) ed for this evaluation.

- Multiple \( F0 \) reference tables are built from single \( F0 \) estimation of monophonic samples before mixing. A correct estimate should not deviate from the corresponding reference value by \( 5\% \). The error rates are computed by the number of error estimates divided by the total number of target \( F0s \).

Evaluation using two analysis window sizes, 186ms and 93ms, are performed and the results are shown in Table 1 and Table 2 respectively. Since musical samples mixed randomly surely contain notes with harmonically related \( F0s \), we present the error rates for two groups of samples: one group of mixtures containing harmonically related notes, labeled as “harmonical”, and the other group “non-harmonical”. The overall error rates are shown in the “total” column. The percentages of samples in the group “harmonical” are 22.43%, 32.78% and 49.46% for the three databases TWO, THREE and FOUR.

<table>
<thead>
<tr>
<th>polyphony</th>
<th>non-harmonical</th>
<th>harmonical</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWO</td>
<td>1.61%</td>
<td>7.59%</td>
<td>2.96%</td>
</tr>
<tr>
<td>THREE</td>
<td>3.27%</td>
<td>7.61%</td>
<td>4.69%</td>
</tr>
<tr>
<td>FOUR</td>
<td>5.68%</td>
<td>11.78%</td>
<td>8.70%</td>
</tr>
</tbody>
</table>

Table 2: \( F0 \) estimation results using a 93 ms window

The errors in the group non-harmonical are quite small which proves the satisfying performance of the proposed method. The overall errors are slightly better than the ones reported by Klapuri [16], however, this comparison is not conclusive due to the fact

---

http://theremin.music.uiowa.edu/MIS.html

---


http://www.ircam.fr/anasyn/cyeh/database.html

---
that the testing set comprises different samples and that in [16] a larger set of samples from four different databases has been used.

5. DISCUSSIONS

The score function sometimes fails to correctly resolve the ambiguity concerning target $F_0$s and their subharmonics or superharmonics especially $F_0/2$ and $2F_0$. This failure scenario accounts for a great proportion of the estimation errors. Polyphonic samples mixed with musical instrument samples of rich resonances often result in this kind of wrong estimate. Taking the string instruments for example, several predominant resonances occur with the excitation [17]. If strong resonances exist in the frequency range below the fundamental, the correct $F_0$s might lose too much score to subharmonics by the amount of explained energy (HAR). If strong resonances boost certain partials too much, correct $F_0$s might lose too much score to superharmonics by the spectral smoothness (MBW). Dealing with resonance peaks is a key to improving robustness.

The window size is still a concern. For those mixtures containing harmonically related $F_0$s, inharmonic partial structures might give a chance for correct estimation if a sufficient spectral resolution is provided. With the increase of polyphony, the performance suffers from the reduction of the window size. Therefore, investigating the techniques for treating overlapped partials is necessary.

The way of constructing polyphonic databases for evaluation should be carefully examined. With the increase of polyphony, the number of possible combinations among different notes and different instruments increases dramatically. A limited number of samples mixed in a random manner could not ensure a general representation of the large sample space. Besides, the number of harmonically related notes increases in higher polyphonic random mixtures and thus effective approaches to estimate $F_0$s of exact multiple relations become more important.

6. CONCLUSIONS

We have presented a new method for joint evaluating the plausibility of multiple $F_0$ hypotheses based on three physical principles. The three principles could be interpreted as reasonable prior distribution for all parameters in the generative spectral model. Instead of using an analytical approach, we optimize each hypothetical partial sequence based on these principles and then compare the credibility of possible combinations among $F_0$ hypotheses using a score function. Evaluation over a large polyphonic database has shown encouraging results. However, there are still issues to be addressed. We envisage that further improvements on the adequate treatment for overlapped partials will lead to higher robustness.

7. REFERENCES


SOUND SOURCE SEPARATION:
AZIMUTH DISCRIMINATION AND RESYNTHESIS

Dan Barry
Boh Lawlor

Dept. of Control Systems and Electrical Engineering,
Dublin Institute of Technology,
Kevin St, Dublin, Ireland.
barrydn@eircom.net

Eugene Coyle

Dept. of Control Systems and Electrical Engineering,
Dublin Institute of Technology,
Kevin St, Dublin, Ireland.
eugene.coyle@dit.ie

ABSTRACT
In this paper we present a novel sound source separation algorithm which requires no prior knowledge, no learning, assisted or otherwise, and performs the task of separation based purely on azimuth discrimination within the stereo field. The algorithm exploits the use of the pan pot as a means to achieve image localisation within stereophonic recordings. As such, only an interaural intensity difference exists between left and right channels for a single source. We use gain scaling and phase cancellation techniques to expose frequency dependent nulls across the azimuth domain, from which source separation and resynthesis is carried out. We present results obtained from real recordings, and show that for musical recordings, the algorithm improves upon the output quality of current source separation schemes.

1. INTRODUCTION
Our research is concerned with extracting sound sources from stereo music recordings for the purposes of audition and analysis. This is termed sound source separation and has been the topic of extensive research in recent years. In general, the task is to extract individual sound sources from some number of source mixtures. Currently, the most prevalent approaches to this problem fall into one of two categories, Independent Component Analysis, (ICA) [1],[2] and Computational Auditory Scene Analysis, (CASA) [3]. ICA is a statistical source separation method which operates under the assumption that the latent sources have the property of mutual statistical independence and are non-gaussian. In addition to this, ICA assumes that there are at least as many observation mixtures as there are independent sources. Since we are concerned with musical recordings, we will have at most only 2 observation mixtures, the left and right channels. This makes pure ICA unsuitable for the problem where more than two sources exist. One solution to the degenerate case where sources outnumber mixtures is the DUET algorithm [4], [5]. Unfortunately this approach has restrictions which make it unsuitable for use with music. CASA methods on the other hand, attempt to decompose a sound mixture into auditory events which are then grouped according to perceptually motivated heuristics [6], such as common onset and offset of harmonically related components, or frequency and amplitude co-modulation of components.

We present a novel approach which we term Azimuth Discrimination and Resynthesis, (ADRess). The approach we describe is a fast and efficient way to perform sound source separation on the majority of stereophonic recordings.

2. BACKGROUND
Since the advent of multi-channel recording systems in the early 1960’s, most musical recordings are made in such a fashion whereby N sources are recorded individually, then electrically summed and distributed across 2 channels using a mixing console. Image localisation, referring to the apparent position of a particular instrument in the stereo field, is achieved by using a panoramic potentiometer. This device allows a single sound source to be divided into two channels with continuously variable intensity ratios [7]. By virtue of this, a single source may be virtually positioned at any point between the speakers. So localisation is achieved by creating an interaural intensity difference, (IID). This is a well known phenomenon [8]. The pan pot was devised to simulate IID’s by attenuating the source signal fed to one reproduction channel, causing it to be localised more in the opposite channel. This means that for any single source in such a recording, the phase of a source is coherent between left and right, and only its intensity differs. It is precisely this that allows us to perform our separation. A similar mixing model is assumed in [9] and [10]. It must be noted then, that our method is only applicable to recordings such as described above. Binaural, Mid-Side, or Stereo Pair recordings will not respond as well to this method although we have had some success in these cases also.
3. METHOD

Gain-scaling is applied to one channel so that one source’s intensity becomes equal in both left and right channels. A simple subtraction of the channels will cause that source to cancel out due to phase cancellation. The cancelled source is recovered by first creating a “frequency-azimuth” plane, figure 1, and 2, which is then analyzed for local minima along the azimuth axis. These local minima represent points at which some gain scalar caused phase cancellation. It is observed that at some point where an instrument cancels, only the frequencies which it contained will show a local minima. The magnitude and phase of these minima are then estimated and an IFFT in conjunction with an overlap add scheme is used to resynthesise the cancelled instrument.

3.1. Azimuth Discrimination

The mixing process we have described can be expressed as,

\[ L(t) = \sum_{j=1}^{J} P_l j S_l(t) \]  
\[ R(t) = \sum_{j=1}^{J} P_r j S_l(t) \]

where \( S_l \) are the \( J \) independent sources, \( P_l j \) and \( P_r j \) are the left and right panning co-efficients for the \( j \)th source, and \( L \) and \( R \) are the resultant left and right channel mixtures. Our algorithm takes \( L(t) \) and \( R(t) \) as it’s inputs and attempts to recover \( S_l \) the sources. We can see from equation (1a) and (1b) that the intensity ratio of the \( j \)th source, \( g(j) \), between the left and right channels can be expressed as,

\[ g(j) = \frac{P_l j}{P_r j} \]  

This implies that \( P_l j = g(j) \cdot P_r j \). So, multiplying the right channel, \( R \), by \( g(j) \) will make the intensity of the \( j \)th source equal in left and right. And since \( L \) and \( R \) are simply the superposition of the scaled sources, then \( L - g(j) \cdot R \) will cause the \( j \)th source to cancel out. In practice we use \( L - g(j) \cdot R \), if the \( j \)th source is predominant in the right channel and \( R - g(j) \cdot L \) if the \( j \)th source is predominant in the left channel. This serves two purposes, firstly it gives us a range for \( g(j) \) such that: \( 0 \leq g(j) \leq 1 \). Secondly, it insures that we are always scaling one channel down in order to match the intensities of a particular source, thus avoiding distortion caused by large scaling factors.

So far we have only described how it is possible to cancel a source assuming the mixing model we have presented. Next we will deal with recovering the cancelled source. In order to this we must move into the frequency domain. We divide the stereo mixture into short time frames and carry out an FFT on each:

\[ Lf(k) = \sum_{n=0}^{N-1} L(n)W_N^{-k_n} \]  
\[ Rf(k) = \sum_{n=0}^{N-1} R(n)W_N^{-k_n} \]

where \( W_N = e^{-j2\pi/n} \) and \( Lf \) and \( Rf \) are short-time frequency domain representations of the left and right channels respectively. In practice we use a 4096 point FFT with a Hanning window and an analysis step size of 1024 points. We create a frequency-azimuth plane for left and right channels individually, see Figure 2. The azimuth resolution, \( \beta \), refers to how many equally spaced gain scaling values of \( g \) we will use to construct our frequency-azimuth plane. We relate \( g \) and \( \beta \) as follows,

\[ g(i) = i \times \frac{1}{\beta} \]  

for all \( i \) where, \( 0 \leq i \leq \beta \), and where \( i \) and \( \beta \) are integer values.

Large values of \( \beta \) will lead to more accurate azimuth discrimination but will increase the computational load. Assuming an \( N \) point FFT, our frequency-azimuth plane will be an \( N \times \beta \) array for each channel. The right and left frequency-azimuth plane are then constructed using,

\[ Az_l(k,i) = |Lf(k) - g(i) \cdot Rf(k)| \]  
\[ Az_r(k,i) = |Rf(k) - g(i) \cdot Lf(k)| \]

for all \( i \) and \( k \) where, \( 0 \leq i \leq \beta \), and \( 1 \leq k \leq N \).

It must be stated that we are using the term “azimuth” loosely. We are not dealing with angles of incidence. The azimuth we speak of is purely a function of the intensity ratio, created by the pan pot during mix down.

In order to illustrate how this process reveals frequency dependent nulls, we generated two test signals, each with 5 unique partials. A stereo mix was created such that both sources were panned to the right, but each with a different intensity ratio. Using this test signal, the frequency-azimuth plane in Figure 1 was created using equation (5a), with, \( \beta=100 \), and \( N=1024 \) point FFT. It can clearly be seen that partials from each source are at a minimum at the same point along the azimuth axis as in Figure 1 and Figure 2.

In order to estimate the magnitude of these nulls we redefine equation (5a) and (5b) as (6a) and (6b):

\[ Az_l(k,i) = \begin{cases} 
Az_l(k,i)_{\max} - Az_l(k,i)_{\min} & \text{if } Az_l(k,i) = Az_l(k,i)_{\max} \\
0 & \text{otherwise}
\end{cases} \]  
\[ Az_r(k,i) = \begin{cases} 
Az_r(k,i)_{\max} - Az_r(k,i)_{\min} & \text{if } Az_r(k,i) = Az_r(k,i)_{\max} \\
0 & \text{otherwise}
\end{cases} \]

Effectively, we are turning nulls into peaks as can be seen in Figure 2. However, the test signal described, represents the ideal case where there is no harmonic overlap between 2 sources. This is
almost never the case when it comes to tonal music. Harmony is one of the fundaments of music creation, and as such instruments will more often than not be playing harmonically related notes simultaneously which implies that there will be significant harmonic overlap with real musical signals. The result of this is that frequencies will not group themselves as neatly across the azimuth plane as in Figure 2. We have observed “frequency-azimuth smearing”. This is caused when two or more sources contain energy in a single frequency bin. The apparent frequency dependent null drifts away from a source position and may be at a minimum at a position where there is no source at all. For instance, if two sources in different positions, contained energy at a particular frequency, the apparent null will appear somewhere between the two sources. To overcome this problem, we define an “azimuth subspace width”, \( H \), such that \( 1 \leq H \leq \beta \). This allows us to recover peaks within a given neighborhood. These azimuth subspaces may overlap and often do. Nulls that drift away from their source positions can now be re-included for resynthesis. This allows the separation of common partials which are 2 sources, one at approximately 85 points along the azimuth axis, and the other at 33. The azimuth subspace width, \( H \), is then set such that the best perceived resynthesis quality is achieved. In practice, we centre the azimuth subspace over the discrimination index such that the subspace spans from \( d-H/2 \) to \( d+H/2 \). The peaks for resynthesis are then extracted using equations (7a) and (7b).

\[
Y_k^s(k) = \sum_{r=d-H/2}^{r=d+H/2} A_z(k,i) \quad 1 \leq k \leq N \tag{7a}
\]

\[
Y_k^f(k) = \sum_{r=d-H/2}^{r=d+H/2} A_z(k,i) \quad 1 \leq k \leq N \tag{7b}
\]

The resultant \( Y_s \) and \( Y_f \) are \( 1 \times N \) arrays containing only the bin magnitudes pertaining to a particular azimuth subspace as defined by \( d \) and \( H \). More specifically, \( Y_s \) and \( Y_f \) contain the short time power spectrum of the separated source. At this point it should be noted that, if two sources have the same intensity ratio, i.e. they share the same pan position, both will be present in the extracted subspace. This is particularly true of the “centre” position. It is common practice in audio mix down to place a number of instruments here, usually voice and very often bass guitar and elements of the drum kit too. In this instance, band limiting can be used to further isolate the source of interest. The bin phases could be estimated using a technique such as ‘magnitude only reconstruction’ but we have found that using the original bin phases is adequate, equation (8a) and (8b). Once we have bin phases and magnitudes we can convert from polar to complex form using equation (9). The azimuth subspace is then resynthesised using the IFFT, equation (10).

\[
\Phi_s(k) = \angle(Rf(k)) \tag{8a}
\]

\[
\Phi_f(k) = \angle(If(k)) \tag{8b}
\]

Polar to rectangular conversion is then carried out using equation (9).

\[
X(k) = \begin{cases} 
\text{Re}(X(k)) &= Y(k) \cdot \cos(\Phi(k)) \\
\text{Im}(X(k)) &= Y(k) \cdot \sin(\Phi(k)) 
\end{cases} \tag{9}
\]

Figure 2: The Frequency-Azimuth plane for the right channel. The magnitudes of the frequency dependent nulls are estimated. The harmonic structure of each source is now clearly visible as is their spatial distribution.

Figure 3: The Frequency-Azimuth Plane. The common partial is apparent between the 2 sources. The azimuth subspace width for source 1, \( H \), is set to include the common partial.

3.2. Resynthesis

In order to resynthesise only one source, we set the discrimination index, \( d \), to the apparent position of the source. In Figure 3, there are 2 sources, one at approximately 85 points along the azimuth axis, and the other at 33. The azimuth subspace width, \( H \), is then set such that the best perceived resynthesis quality is achieved. In
\[ x(n) = \frac{1}{N} \sum_{k=1}^{N} X(k)W_{N}^{-kn} \]  

(10)

where \( W_{N} = e^{\frac{j2\pi k}{N}} \).

The resynthesised time frames are then recombined using a standard overlap and add scheme. This algorithm has been implemented to run in real-time and it is the case that the control parameters \( d \) and \( H \) be set subjectively until the required separation is achieved. In effect, the user sweeps through the stereo space from left to right until the desired source is encountered. In much the same way as a pan pot places a source at some position between left and right, the ADRes algorithm will extract a source from some position between left and right.

4. RESULTS

We have applied the ADRes algorithm to a number of commercial recordings. The degree of separation achieved depends on, the amount of sources, the source proximity and the source level. If sources are proximate, it is likely that multiple sources may get extracted. If there is a large number of sources, partials may go missing. If the source level is too low, the resynthesis may have a bad signal to noise ratio. In general though, some degree of separation is possible. In order to illustrate this, we generated a synthetic stereo signal, using 5 general midi instruments: bass, piano, drums, vibraphone and French horn. They were panned to 5 unique positions as in Figure 4.

Figure 4: 5 sources panned to different positions. 1=bass, 2=vibraphone, 3=drums, 4=piano, 5=horn

The piece of music in Figure 5 was generated in a midi editor using these 5 instruments. The polyphony varies throughout the 2 bar segment with up to 9 notes sounding at once. In some cases 2 instruments are playing the same note at once. A stereo wav file, Figure 6, was then created using the score, instruments and panning parameters from above. This file was then processed by ADRes, with the relevant parameters set. The azimuth resolution, \( \beta \), was set to 10 points for each side. The azimuth subspace width, \( H \), was set to 2 in all cases. The discrimination index, \( d \), was set for each source position. A high quality of separation was achieved for all sources.

The resulting separations are of reasonably high quality. There are some obvious visual differences between the input and output time domain plots and there are some obvious audible artifacts but the quality is significantly high. Furthermore when the separations are ‘remixed’, the resultant mixture is almost free from artifacts. These examples and others can be downloaded at:

www.dmc.dit.ie/2002/research_ditme/dnbarry

Figure 5: The score which was generated for the 5 instruments.

Figure 6: The Stereo Mixture containing 5 panned sources.

Figure 7a: The 5 sources before mixing and processing.

Figure 7b: The 5 sources separated by the ADRes algorithm.
5. CONCLUSIONS

We have presented an algorithm which is able to perform sound source separation by decomposing stereo recordings into frequency-azimuth subspaces. These subspaces can then be resynthesised individually, resulting in source separation. The only constraints are that the recording is made in the fashion described in Section 2, and that the sources do not move position within the stereo field. We feel that ADRess is applicable to a large percentage of commercial recordings.

6. ACKNOWLEDGEMENTS

Many thanks to Derry Fitzgerald for knowledge imparted and also to Frank Duignan for work on the real-time java implementation of ADRess.

7. REFERENCES

ABSTRACT
This paper proposes a new scheme to reduce coding bit rate in array based multichannel audio applications like the acoustic opening, which can be used for modern teleconference systems. The combination of beamforming techniques for source separation and wave field synthesis allows a significant coding bit rate reduction. To evaluate the quality of this new scheme, both objective and subjective tests have been carried out. The objective measurement system is based on the Perceptual Audio Quality Measure of the binaural signal that the listener would perceive in a real environment.

1. INTRODUCTION

Over the last years there has been a significant development of multichannel audio. These technologies are evolving towards systems capable of recreating true 3D audio fields in a listening area as wide as possible. This evolution implies rising the number of loudspeakers used for sound reproduction. Today it is possible to find commercial products that use 5.1 channels but new systems will use many more channels to increase the perceived spatial sensations. If a high number of channels are involved in an audio system, this means that a large amount of audio material needs to be transmitted or recorded. This justifies recent attention on multichannel audio coding systems, that try to reduce the overall bit rate without penalizing quality.

One of the most promising multichannel audio systems is Wave Field Synthesis (WFS). Wave Field Synthesis technique reproduces an acoustic field inside a volume from the signals recorded or computed on a given surface. It is based on the Huygens principle. According to this principle, the propagation of a wave through a medium can be qualitatively described by adding the contributions of all secondary sources positioned along a wave front [1]. This means that if we know the wave field on the boundary surface S of a closed, source-free volume V it is possible to know the sound pressure in any point within that volume. From a practical point of view, this means that if we cover a plane with an array of omni directional loudspeakers, being driven with signals corresponding to the normal velocity distribution in that plane generated by virtual source, a spatially correct wave field of a point positioned behind the array is synthesized.

The application of microphone and loudspeaker array systems to enhance perceived sensations is under study. Using a high number of microphones (more than 20) in a linear array makes possible to sample the entire acoustic field. This field can be recreated in another location by means of Wavefield synthesis [2] using a loudspeaker array.

2. CODING APPROACHES

The two most used multi-channel codification systems are Dolby AC-3 and MPEG Advanced Audio Coding (AAC). AC-3 is the audio standard chosen for high-resolution television (HDTV), and it is able to compress 5.1 audio signals using 384 kbits/s. AAC is at the moment the most powerful multi-channel codification system within the family of MPEG coders. It is able to compress 5.1 audio signal using 320 kbits/sec without apparent loss of quality. Both schemes use perceptual models to hide coding distortions. Although they are very powerful systems that support a high
number of channels, they are optimized to encode 5.1 recordings. Thus, the multichannel strategies employed (Mid/Sum Coding and Intensity Coding) try to exploit the correlation between symmetric channel pairs (e.g. L-R and Ls-Rs), but are unable to eliminate the existing correlation among the rest of the channels.

Despite of this reduction, the number of channels to be transmitted still equals the number of microphones in the array. In order to reduce the number of channels to be transmitted, array processing methods are explored in the following.

3. SOURCE SEPARATION APPROACH

To develop a wave field synthesis approach we have two possibilities: in the first one, all microphone signals are transmitted to the same number of loudspeakers at the receiving end. This system receives the name of ‘hard-wired wave field transmission system’.

In this approach the compression gain comes from exploiting the correlation between channels (as seen in previous Sections). However there is another possibility where the signals that feed the secondary sources (the loudspeakers at the receiving room) are extrapolated of a enough dense set of measured impulse responses. This new approach has a tremendous impact from a codification point of view. Now, it is possible to send only the dry sources and the impulse responses of the room and recreate the wave field at reception. This leads us to the problem of obtaining the dry sources given that we only know the signals that the microphone array captured. Basically, this is a source separation problem. In Figure 4 we can see the full scheme where a WFS system synthesizes the wave field produced by primary sources in the simulated room.

From a mathematical point of view, the problem to solve can be resumed in expressions (1), (2) and (3). There are \( P \) statistically independent wideband sound sources \((S_1...S_P)\) in a \( M \)-microphone room \((P<M)\). Each microphone signal is produced as a sum of convolutions between sources and \( H_{ij} \), which represents a matrix of z-transfer functions between \( P \) sources and \( M \) microphones. This transfer function set contains information about the room impulse response and the microphone response. The number of sources \((M)\) is always lower than the number of microphones \((P)\). We have:

\[
X(z) = \begin{bmatrix} H_{11}(z) & ... & H_{1P}(z) \\ H_{21}(z) & ... & H_{2P}(z) \\ \vdots & \vdots & \vdots \\ H_{M1}(z) & ... & H_{MP}(z) \end{bmatrix} \begin{bmatrix} S_1(z) \\ S_2(z) \\ \vdots \\ S_P(z) \end{bmatrix} = HS
\]

(1)

\[
X = HS
\]

(2)

We make the assumption that source signals \( S \) are statistically independent processes, (which is a sufficient condition for source separation) so the minimum generating signals \( \Gamma \) will be the same as the number of sources \( P \). We need \( \Gamma \) to be as similar as possible to \( S \) (original dry signals). Ideally \( \Gamma \) would be the pseudoinverse of \( H \), however we may not know the exact parameterization of \( H \). In the real world spatial separation of sources from an output of a sensor array is achieved using beamforming techniques. Thus, we let

\[
\Gamma = JHS
\]

(3)
4. BEAMFORMING: GENERALIZED SIDELOBE CANCELLER

For acoustic openings, microphone arrays together with robust adaptive beamforming techniques allow the extraction of desired signals from many kind of interferers (background noise, reverberation, or competing talkers). One of the most used beamforming algorithms is the Generalized Sidelobe Canceller [6] that can obtain a high interference reduction performance with a small number of microphones arranged in a small space.

One of the biggest concerns in using GSC is that we need to know the direction of arrival (DOA) of the primary source. That means that we need to know quite accurately the position of the speakers in the room. This can be achieved using DOA-determination algorithms, like the MUSIC algorithm [7]. The MUSIC algorithm was developed by Schmidt to determine direction-of-arrival angles for multiple sources and although it offers good results if the primary sources are narrow band signals, with broadband signals (like voice) the results are not so good [8]. The resolution of this problem is beyond the objectives of this article, for our work we suppose the DOA is known.

We can see the general layout of the GSC in Figure 5. First of all, the microphone signals are time delayed steered (t₁, …, tₙ) to produce signals which ideally have the desired signal in phase with each other. If we add all these signals (d(ₙ)) we have a classical delay and sum beamformer. Usage of a simple delay-and-sum beam former leads to target signal cancellation at high frequencies so we need to complicate the system with an adaptive algorithm to improve the overall performance. A delayed version of d(ₙ): d' (ₙ) (to keep causality) is used as reference for the adaptive sidelobe canceling path. [8]. Depending on how precise is the DOA information this reference would be good enough. The delayed signals (before adding) are then sent to the blocking matrix. The purpose of the blocking matrix is to block out the desired signal from the lower part of the GSC. The idea is to adaptively cancel out noise and interference sources, therefore we only want noise to go into the adaptive filters F₁… Fₘ₋₁. At the present moment we have used a very simple blocking matrix, which means that the outputs of the matrix are the difference between successive signal samples:

\[
B = \begin{bmatrix}
1 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 \\
\end{bmatrix}
\] (4)

If the input signals of the adaptive filters (F₁… Fₘ₋₁) contain only interferences the multiple input canceller rejects the interferences and extracts the target signal (e(ₙ)). If the target signal leaks through the blocking matrix the adaptive algorithm (we use NLMS for simplicity) cancels both the interference and the desired signal (the original dry source we want to recover). This leakage can be caused by two different causes. First of all, bad DOA tracking; second, a highly reverberant room. If we are working with a highly reverberant room (high T₆₀) some of the reflections of the interference signals may leak into the main lobe. In this case even with highly complex blocking matrixes [9, 10] it may be very difficult to obtain target-free signals from the output of the BM. As we know the exact DOA, the current BM configuration is enough to extract a signal which is quite similar to the original dry source.

5. EXPERIMENTS AND CONCLUSIONS

To obtain the microphone signals we are employing the impulse-response recordings of the anechoic chamber in Bell Labs [11], corresponding to different audio source locations in the chamber and a 22-microphone linear array. This arrangement is perfect for studying the “virtual acoustic opening”. We have also considered free field propagation conditions (no chamber) to compare the effects of reverberation. Two different speakers (male and female), which act as primary sources, are placed on both sides of the room. These two signals are convolved with the impulse response of the chamber corresponding to those concrete locations to obtain the 22 microphone signals. In the case of employing free field conditions we only delay and scale properly the signals.

We have considered two different scenarios. In the first one we suppose that we are using a hard-wired WFS system. That means that we ‘send’ the full 22 channels and the sound field is reconstructed using directly the transmitted signals. In the second scenario we apply the beamforming algorithm to recover both original dry signals. In this occasion we suppose that we are only ‘sending’ these two signals. At reception we rebuild the acoustic field using WFS techniques. The results are based in the comparison of these two scenarios.

For this comparison we have used objective and subjective tests. One of the tools to be used is based on the ITU standard Perceptual Audio Quality: PEAQ [12] which is widely accepted for codec comparison. We have specifically used an implementation of the basic version of the recommendation ITU-R BS. 1387-1 [13]. This module measures the perceptual difference between the original and processed signal by means of the so called Objective Difference Grade (ODG). The output of the module is a figure between 0 and –4 where 0 means “no perceptible degradation” and –4 means “very annoying degradation”.

PEAQ was developed for evaluating the quality of mono of stereo signals, not for multichannel audio. The proposed solution to this problem is to synthesise the binaural signal which is computed with the following guidelines.

- In the first scenario, we obtain the loudspeaker driving signals supposing that we are using a hard-wired WFS systems as if we had sent the 22 channels.
- In the second one we obtain the pseudo-dry sources using the beamforming algorithm and reconstruct the loudspeaker driving signals using WFS.
- Loudspeakers are considered ideal; in a real non simulated environment. Equalization should be implemented.
- Free field propagation from the loudspeakers to the listener is assumed.
- To obtain a true binaural signal, the effect of external ear, head, shoulders, etc. is taken into account by using HRIRs (Head Related Impulse Responses). The signal coming from each loudspeaker is filtered with the HRIR that corresponds to that direction of arrival. The particular impulse response set is the one measured with KEMAR (Knowles Electronic Mannequin for Acoustic Research [14]) with diffuse equalization. The listener is positioned in the middle of the reception room. With a true mechanical opening, he would hear the male voice coming from the right and the female voice coming from the left.
Figure 4: Acoustical opening coding system.

Figure 5: Generalized sidelobe canceller.
If we use free field conditions the results are quite promising. The source separation is nearly perfect and the ODG value obtained is -0.7 which means that the distortions are nearly inaudible. This has a great impact from a coding point of view. In the first scenario we ‘sent’ 22 channels while in the second we only ‘sent’ two. The bit rate reduction is huge.

Problems arise when using the impulse responses of the varecoic chamber. Due to reverberation, source separation is not so good and you can hear an attenuated version of the female speaker signal in the separated male speaker signal and vice-versa. Also, the performance of PEAQ algorithm is not fully reliable in these conditions. However informal subjective tests with 6 listeners have showed that there is not a big difference between both scenarios in terms of quality at reception (after WFS and binauralization). The intelligibility is even better in the second scenario due to the reduction of the reverberation effect by the adaptive algorithm. The spatial sensations are also preserved (we still hear the male voice coming from the right and the female voice coming from the left) which is an important feature. We have noticed that the adaptive algorithm embedded in the sidelobe canceller performs much better when the interferer is white noise instead of a second speaker. The silences between words cause the nLMS algorithm to diverge. In the future it may be necessary to implement some kind of vocal activity detector to stop the nLMS adaptation algorithm in the silences. If we consider noise as the interferer, the source separation becomes a noise cancellation problem. For this case, we can see the behaviour of the GSC in Figure 6.

![GSC output](image)

Figure 6: Noise canceller behavior.

On the upper part of the picture we can see the original dry signal (male speech). In the middle, the signal recorded in the central microphone of the array (signal + noise). In the lower part we can see the GSC output (pseudo-dry signal). As we can see the noise reduction is quite effective. Taking a closer look at the first samples, you can notice a progressive noise reduction due to the nLMS convergence time. Playing with the adaptation step makes possible to decrease this convergence time. However, in this case, the final SNR would be higher.

The results presented in this paper are still preliminary but we think that are quite promising. Using beamforming together with Wavefield Synthesis to recreate a mechanical acoustic opening may provide us with a very useful tool to drastically reduce the number of channels to transmit (and consequently the bit rate). There still are problems to solve, like the DOA estimation, the effect of high reverberant rooms and the development of better quality measures, but it seems that the path is correct and that future teleconference systems may benefit from this approach.

6. REFERENCES

ANALYSIS OF CERTAIN CHALLENGES FOR THE USE OF WAVE FIELD SYNTHESIS IN CONCERT-BASED APPLICATIONS

Terence Caulkins, Etienne Corteel and Olivier Warusfel

IRCAM, 1 place Igor Stravinsky, 75004, France
caulkins@ircam.fr

ABSTRACT

Wave Field Synthesis (WFS) provides a means for reproducing 3D sound fields over an extended area. Beyond conventional audio reproduction applications, present research at IRCAM involves augmenting the realism of concert-based applications in which real musicians will be interacting on stage with virtual sources reproduced by WFS. The stake of such a situation is to create virtual sound sources which behave as closely as possible to real sound sources, in order to obtain a natural balance between real and virtual sources. The goal of this article is to point out physical differences between real sound sources and WFS reproduced sources situated at the same position, considering successively the sound field associated to the direct sound of the virtual source and its interaction with the room. Methods for taking into account and compensating these differences are proposed.

1. INTRODUCTION

Wave field synthesis, or holophony, is a sound reproduction technique based on Huygens’ principle that can be seen as the equivalent of holography for acoustic waves. For more complete theoretical background regarding this technique, the reader is invited to refer to [1], [2], [3]. In a previous article [4], the consecutive simplifications that must be operated upon the framework described by the Rayleigh integrals in order to achieve practical implementation of Wave Field Synthesis were listed. The stated simplifications include the reduction of the ideal planar distribution of secondary sources to a line, the truncation of the line, and the subsequent spatial sampling of the line.

Section 2 of the present paper aims to show a difference between the direct sound field radiated by a real source and that of a virtual source reproduced by WFS. This difference is seen to be a consequence of the reduction of the plane of secondary sources to a line, and manifests itself as a dispersion of the virtual source’s direct sound field over the listening area. The discussion will be initiated by considering the radiative properties of an ideal line source. The impulse response of this type of source is shown to be initiated by considering the radiative properties of an ideal line source. The impulse response of this type of source is shown to be instantaneous and its consequences upon the reproduced sound field are examined. This approximation is shown to lead to similar conclusions concerning the dispersive quality of the reproduced wave field. These theoretical considerations are illustrated by simulations for three distinct reproduction situations.

Section 3 of this paper goes on to study the difference between a real source and a WFS reproduced source in terms of interaction with the listening room. An analysis of the power effectively emitted by an ideal monopole array reproducing a virtual dipole source shows a difference with what is expected of true dipole sources. This entails a modification of the ratio of direct/reverberated sound level throughout the listening area for WFS reproduced sources as compared to real sources.

2. SCATTERING OF THE SOUND FIELD EMITTED BY THE SECONDARY SOURCE ARRAY

2.1. Radiation of an ideal line source

A review of the properties inherent to line sources may help to improve the overall comprehension of the physical properties of an ideal WFS line array before truncation and sampling. Let \( \Omega_n \) represent an n-dimensional infinite, homogeneous, and isotropic space. It is well documented fact that a line source in \( \Omega_n \) is the physical equivalent of a point source in \( \Omega_3 \) [5]. The propagative behavior of acoustical waves emitted by a line source in \( \Omega_2 \) can thus be derived from the Green function associated to a point source in \( \Omega_2 \).

Furthermore, the 2D Green function \( g_2 \) can be deduced from the 3D Green function using Hadamard’s “method of descent” [6] and yields the following expression:

\[
g_2(|\vec{r} - \vec{r}_0|, t - t_0) = \frac{c}{2\pi \sqrt{c^2(t - t_0)^2 - |\vec{r} - \vec{r}_0|^2}} \times \left[ U(c(t - t_0) - |\vec{r} - \vec{r}_0|) \right],
\]

where \( U \) is the step function and \( c \) the speed of sound. This expression signifies that if a point source of \( \Omega_2 \) situated in \( \vec{r}_0 \) emits an impulse at \( t_0 \), the pressure field received in \( \vec{r} \) will consist of an immediate radiation \( g_2 \) followed by a residual field of which the magnitude depends on the distance from the source. In other words, the wavefront emitted by a point source in \( \Omega_2 \) is followed by a tail or wake [7]. This is known as diffusive, as opposed to sharp, propagation: the point (resp. line) source in 2D (resp. 3D) behaves as if it were emitting a wave that propagates simultaneously at all velocities between 0 and \( c \) [6]. Moreover, the shape of the wake varies as a function of the reception point \( \vec{r} \). The Fourier transform of \( g_2 \) yields the following expression:

\[
G_2 = -\frac{i}{4} H_0^1(|k|\vec{r} - \vec{r}_0)|
\]

where \( H_0^1 \) represents the cylindrical Hankel function of the first kind of order 0 and \( k = \frac{2\pi f}{c} \) represents the wave number.

---

Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx'04), Naples, Italy, October 5-8, 2004

250 — DAFx’04 Proceedings — 250
Figure 1 displays the magnitude of $G_2$ expressed in dB as a function of the distance $r$ to the considered source. This value is normalized by a factor $\sqrt{r}$ to compensate for the $3dB$ attenuation per distance doubling, $r$ being the radius (i.e. distance to the line source). A global $\sqrt{r}$ factor of frequency correction is also included to account for the $-3dB$ per octave magnitude attenuation that occurs in the far field of line sources. One can observe that the corrected frequency response displays very little energy for low frequencies in the near field (where $kr \ll 1$), meaning that the “true” (uncorrected) frequency response tends to be flat in the near field. In the far field, the corrected pressure field exhibits a flat frequency response, meaning that the magnitude of the “true” pressure field is attenuated by $-3dB$ per octave. This shows that the global level of bass frequencies increases with the distance to the line source, i.e. “focused” radiation (see Figure 2). An important specificity of WFS systems is the capacity to reproduce sound sources within the listening area, i.e. “focused” sources. These sources are generated simply by applying a time reversal on the delays suggested by the stationary phase equations.

$$P(r, k) = \sqrt{r} \exp^{-j(kr - \frac{\pi}{4})} \left( \frac{|y_R - y_L|}{|y_R - y_0|} \right) \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cos \phi_{inc} d\omega$$

where $S(\omega)$ represents the signal fed to the notional source.

This approximation is based on the oscillatory nature of the exponential function. For the needs of this article, let it suffice to say that the accuracy of the approximation increases for large values of $k$, $r_0$ and $\Delta r_0$.

Noting that $\sqrt{r} = \exp j \frac{\pi}{4}$, a set of driving functions $Q_\psi(x_L, k)$ for the secondary monopole sources situated along line $L$ (represented by the term $\exp^{-j(kr - \frac{\pi}{4})}$ in equation (3)) can now be extracted. This is done for an average listening depth $y_{Rav}$, introducing only an amplitude error in the synthesized wave field for receivers on lines $y_R \neq y_{Rav}$:

$$Q_\psi(x_L, k) = \sqrt{r} \exp^{-j(kr - \frac{\pi}{4})} \left( \frac{|y_R - y_L|}{|y_R - y_0|} \right) \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cos \phi_{inc} d\omega$$

The term $\sqrt{r} \exp^{-j(kr - \frac{\pi}{4})}$ situated outside of the brackets in equation (4) makes it clear that the driving functions applied to the secondary sources are frequency dependent in regard to both phase and magnitude. The term $\exp j \frac{\pi}{4}$ (constant in the frequency domain) can be translated as a “negative delay” of an eighth of a period to be applied to all the frequency components of the source signal in the time domain. Moreover, the $\sqrt{r}$ term implies a magnitude dependence of $+3dB$ per octave. In other words, the stationary phase suggests that low frequencies be emitted with lower levels and in advance in comparison to higher frequencies.

An important specificity of WFS systems is the capacity to reproduce sound sources within the listening area, i.e. “focused” sources. These sources are generated simply by applying a time reversal on the delays suggested by the stationary phase equations. The driving function for such sources is thus equal to:

$$Q_{inc}^{\psi}(x_L, k) = \sqrt{r} \exp^{-j(kr + \frac{\pi}{4})} \left( \frac{|y_R - y_L|}{|y_R - y_0|} \right) \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cos \phi_{inc} d\omega$$

Note that for focused source reproduction low frequencies must be emitted after high frequencies because of the phase inversion appearing in equation (5).

Aside from virtual point sources, WFS systems allow the reproduction of “plane waves”. These correspond to point sources situated at very large distances as compared to the size of the listening area. It is to be remarked that the situation described in Section 2.1, i.e. a line source emitting an impulse, is identical to the emission by an ideal linear WFS array of a “plane wave” propagating perpendicularly to the array. It is therefore clear that dispersive effects will also appear in the sound field of WFS reproduced plane waves.
In any case, the stationary phase approximation proposes a correction of dispersive effects (frequency dependent delays and $\sqrt{k}$ filtering) independently of virtual source and listening area positioning. Although it is clear that dispersive qualities appear in sound fields emitted by infinite continuous line arrays, nothing guarantees that this is also the case for the finite and discrete linear monopole arrays used in Wave Field Synthesis. Simulations in the following Section will allow to decide whether dispersive effects exist in ideal WFS setups.

### 2.3. Simulations

![Figure 3: Typical concert situation in which a WFS system is being used to render virtual sources alongside real instruments.](image)

The configuration for the following simulations consists of a linear array of 32 ideal monopole sources with regular 16.5cm spacing reproducing sound sources situated on stage as well as inside of the listening area. This is the simulation of a real situation in which the loudspeaker array would be placed between the stage and the audience so as to generate virtual sources to accompany real instruments (Figure 3).

Three WFS sound reproduction situations are chosen to illustrate dispersive effects in the reproduced soundfield.

- **Situation 1**: Reproduction of a virtual source at various distances behind the monopole array (i.e. on stage).
- **Situation 2**: Reproduction of a virtual source 1 meter in front of the monopole array (i.e. in the listening area).
- **Situation 3**: Reproduction of a plane wave travelling perpendicularly away from the array into the listening area.

**Situation 1** exhibits a source/array positioning that enters into the theoretical framework set by Huyghens’ principle (i.e. primary source situated outside of the listening area). Virtual sources are placed on the perpendicular line running through the center of the monopole array so as to limit windowing effects as much as possible. The resulting soundfield is recorded for different source positions (1m, 5m, 10m and 50m behind the array) on a virtual omnidirectional microphone situated in the listening area. This recording simulation is carried out at 1m and 10m in front of the monopole array. The results are represented in Figure 5. It can be seen that between 100 and 1000 Hz the frequency responses simulated 1m in front of the monopole array tend to be flat (+1dB). In the same frequency band but 10m away, all sources (except for the one 1m behind the array), exhibit frequency responses that become more and more disturbed as the virtual source moves away from the monopole array. Below 100Hz none of the virtual sources exhibit flat frequency responses and the magnitude differences between the two recording positions are maximal.

![Figure 4: Situations chosen to illustrate dispersive qualities in WFS sound fields. Situation 1 involves reproducing virtual point sources behind the monopole array; Situation 2 involves reproducing a point source in front of the array; Situation 3 involves reproducing a plane wave.](image)

**Situation 2** displays the reproduction of a focused source situated 1m inside of the listening area. In order to center the description on the virtual source, the sound field is recorded upon concentric microphone arrays of increasing radii ($r = 0.1m$, 0.5m and 50m). These arrays are reduced to arcs situated in the listening area as shown in Figure 6 so as to be contained in the visibility window of the focused source [4]. Microphone positions are numbered from 1 to 28 starting from the left extremity of the array. Figure 7 shows the results for this simulation. The first observation that can be made is that in the immediate vicinity of the source...
and underneath the aliasing frequency (≈ 1200 Hz for this setup),
the wave field exhibits a +3dB/octave frequency response as well
as phase delay for lower frequencies. At this point, the contribu-
tions of the entire array arrive simultaneously (by design) and are
summed independently of frequency, which means that the $\sqrt{-jk}$
filtering suggested by the stationary phase approximation locally
distorts the sound field. As the distance to the source increases,
the frequency response flattens out, as does the phase diagram. These
observations are well in agreement with the theoretical considera-
tions of Section 2.2. Indeed, the $\sqrt{-jk}$ filtering suggested by the
stationary phase approximation ensures a realistic sound field in
the far field of the monopole array ($r = 10m$, $f \geq 100$).

**Situation 3** describes the reproduction of a plane wave propa-
gating perpendicularly to the monopole array into the listening
area. The wave field is recorded on linear microphone arrays run-
ning parallel to the monopole array as shown in Figure 6. Fig-

![Figure 6: Left: Mic arrays (radii resp. 0.1m, 0.5m, and 10m) recording a focused source situated 1m in front. Right: Mic ar-
arrays recording a plane wave (0.1m, 10m and 50m from array)](image)

Figure 6: Left: Mic arrays (radii resp. 0.1m, 0.5m, and 10m) recording a focused source situated 1m in front. Right: Mic ar-
arrays recording a plane wave (0.1m, 10m and 50m from array)

ure 8 shows the results for this simulation. Phase and frequency
responses in the nearfield of the loudspeaker array and below the
aliasing frequency are seen to be flat. They tend towards a +3dB/oct-
ave frequency response in the far field as well a phase advance for
low frequency components. To explain this, one may turn towards
linear array radiation prediction techniques. It is a classical ap-
proximation to consider that the contributions of all the sources
composing a linear array become coherent in the farfield around
the perpendicular bisector of the array. The same situation arises
at the focal point of virtual sources located within the listening
area, causing an inaccuracy in the corrections suggested by the
stationary phase approximation (cf **Situation 2**).

### 2.4. Consequences on practical implementation

Practical implementation of WFS involves the application of a
multiequalization scheme to compensate for the complex direc-
tivity patterns exhibited by real loudspeakers. For detailed knowl-
gedge on this subject the reader is invited to refer to [9].

Application of the multiequalization technique ensures that the
reproduced sound field is correct along a certain microphone con-
trol line. The multiequalization scheme “automatically” takes into
account the scattering effects described and simulated in Sections
2.2 and 2.3 and compensates them. However, for positions situated
before and after the control line (in regard to the natural progres-
sion of the wavefront) nothing is known a priori about the validity
of the wave field. The simulations carried out in the previous part
point to the fact that the wave fields for the three types of WFS
sources display frequency and phase characteristics that vary dur-
ing propagation.

This knowledge may prove to be useful when installing WFS
setups in large concert halls (where the reproduction zone must be
as large as possible but far away from the loudspeaker system) or, oppositely, when dealing with WFS setups in small rooms (where the reproduction zone is situated near the reproduction system and is reduced in size). For small reproduction rooms, the microphone array upon which the desired wave field is specified may be placed close to the loudspeaker array since the reproduced wavefield will have very little space to disperse over. For larger rooms, the control array of microphones must be placed further away to account for dispersion effects. This suggests the use of different filter banks according to the size of the reproduction room to ensure a realistic sound field over the targeted listening area.

3. DIRECT/REVERBERATED SOUND RATIO FOR DIRECTIVE SOURCES

Rendering realistic spatial impression involves reproducing, beside the direct sound of the virtual source, a coherent room effect, especially when real and virtual sources are mixed together on stage. Contrarily to a classical audio situation where synthetic room effect is rendered in addition to direct sound, the aim of this article is to explore the ‘natural’ room effect emanating from the interaction between the WFS virtual source itself and the listening room. A priori if the direct sound field reconstructed by WFS were entirely accurate (which is not the case, as was shown in Section 2), the resulting room effect would automatically be entirely accurate. This Section aims to give a measure of the accuracy of the reproduced room effect for WFS. This can be done simply by characterizing the ratio of direct/reverberated sound in the listening room. Section 2 dealt with describing the direct sound field, which can also be piloted using directive sources [4]. The energy density of the reverberated sound field is for its part linked to volume and absorption of the listening room, as well as the power emitted by the source itself. The power effectively emitted by the WFS array when reproducing the ideal properties of sources such as monopoles and dipoles will therefore be calculated so as to compare its behavior in terms of direct/reverberated ratio with that of real sources.

The proposed configuration is a WFS reproduction system made of a linear array of 32 ideal monopole transducers with 16.5cm spacing surrounded by a circular array composed of 64 evenly distributed omnidirectional microphones. The array is used to reproduce a virtual source situated at different positions (in front or behind the WFS array), and associated to various directivity patterns and/or orientations (cf Figure 9).

The first analysis deals with the synthesis of a source situated 3m in back of the array, which represents the case of a source situated on stage. As described in [4], WFS allows for the synthesis of a directivity pattern associated to the virtual sound source. We consider here the case of a dipole pattern simulated with different orientations. Results are shown in Figure 9. It appears that the power emitted by the array for different dipole orientations varies. This obviously would not be the case for a real source of which the power level is independent of its orientation in space. Moroso, the right side of Figure 9 shows that the expected ratio of -4.7dB for dipole/monopole power is attained only for certain dipole orientations (∼40° for the source synthesized behind the array and ∼45° for the source generated 1m in front of the array).

One reason for which the direct/reverberated sound level in the listening room varies for different dipole orientations that the window through which the source feeds the listening room is limited by the size of the loudspeaker array. A proposition for compensating this windowing effect was proposed in [4]. It involves injecting artificial image sources using extra loudspeaker arrays along the lateral walls of the listening room. This would then naturally modify the ratio of direct/reverberated sound for the different possible

![Figure 8: Phase (top figures) and frequency (bottom figures) evolution of a WFS synthesized plane wave recorded on linear microphone arrays situated at different distances (left: 0.1m, middle: 10m, right: 50m) from the monopole array](image)
dipole orientations. A less costly solution in terms of calculation power would be to add extra room effect using dedicated room effect channels [10], provided that the power error is negative.

The other reason for which the direct/reverberated level varies lies in the linear nature of the WFS array. The emitted soundfield manifests a symmetry around the axis of the array ("cylindrical propagation"). Directivity patterns or orientations that do not exhibit such cylindrical symmetry will not be accurately rendered in terms of associated power.

4. CONCLUSION

After examining the consequences of the reduction of the plane of secondary sources to a line, simulations were carried out showing dispersive qualities in sound fields emitted by WFS monopole arrays when synthesizing sources situated in front and behind the array, as well as plane waves. This observation led to the conclusion that synthesizing realistic sources in Wave Field Synthesis entails adapting the multiequalization scheme to the reproduction room size, depending on whether we are considering a large hall for concert applications or a small room for virtual reality applications. The next part of the article was committed to studying the ratio of direct/reverberated sound generated by an ideal loudspeaker array when synthesizing basic directivity figures (monopole, dipole). It was shown that this ratio cannot be fulfilled by WFS and depends on the orientation of the virtual sources. Solutions for restoring the perceptual effect linked to room interaction were proposed involving the injection of artificial image sources or reverberation in order to compensate for the observed differences.

5. REFERENCES

A MAXIMUM LIKELIHOOD APPROACH TO BLIND AUDIO DE-REVERBERATION

Massimiliano Tonelli
ANWIDA Soft
Strada Montefeltro 81, 61100 Pesaro, Italy
tonelli@anwida.com

Nikolaos Mitianoudis
Electronic Engineering Dept., Imperial College
Exhibition Road, London SW7 2AZ UK
n.mitianoudis@imperial.ac.uk

Mike Davies
DSP Group, QMUL
Mile End Road, London E1 4NS UK
mike.davies@elec.qmul.ac.uk

ABSTRACT

Blind audio de-reverberation is the problem of removing reverb from an audio signal without having explicit data regarding the system and/or the input signal. Blind audio de-reverberation is a more difficult signal-processing task than ordinary de-reverberation based on deconvolution. In this paper different blind de-reverberation algorithms derived from kurtosis maximization and a maximum likelihood approach are analyzed and implemented.

1. INTRODUCTION

In non ideal acoustic spaces, reverb may deteriorate the listening experience. This is common for rooms that have not been explicitly designed for such functionalities. The recommended approach in this case is to improve the acoustic of the space by modifying its physical properties. However, this might not be possible for constraints of several natures (e.g. historical buildings, churches, cockpit of a car).

In similar conditions, speech intelligibility, or more generally the quality of the listening experience, is badly affected and often the “duty” of enhancing it is left entirely to the public addressing (PA) system. Therefore, it is a common trend for today’s music equipment industry to build “intelligent” PA systems able to address such problems.

In other cases, it might be of interest to enhance audio streams obtained in environments with bad acoustic characteristics (e.g. recording restoration, pre-processing for speech recognition software).

If we assume the acoustic paths as linear, the acoustic system can be modelled using a linear transfer function. Under this hypothesis, two families of identification methods exist:

- “input-output” system identification methods (supervised learning) If both the input and the output of the system are known, a supervised learning strategy is feasible. If there is a necessity to characterize the system by the measurement of a “steady” transfer function, swept sinus or maximum length sequences (MLS), identification techniques offer excellent results. Otherwise, to estimate a time varying transfer function, adaptive filtering strategies or input-output block based cross-correlation methods can be used.

- “blind” identification methods (unsupervised learning) If only the output of the system is available (e.g. a speaker talking in a room with no microphone, or acoustic instruments playing in a theatre), an input-output identification strategy is not possible.

This paper will discuss this second approach.

2. IMPULSE RESPONSE AND ROOM TRANSFER FUNCTION

The impulse response (IR) of a generic system is defined as its output, when a Dirac δ function is applied to its input. If the system is linear time invariant (LTI) the IR specifies the system completely.

In fact, if:

1) \( x(n) \) is the signal obtained sampling the speaker
2) \( y(n) \) is the reverberated signal that is perceived by the listener
3) \( h(n) \) is the IR of the source-listener acoustic path

then \( y(n) \) can be expressed as

\[
y(n) = h(n) \otimes x(n) = \sum_{k=-\infty}^{\infty} x(k) \cdot h(n-k)
\]

that is: the convolution between \( x \) and \( h \).

An IR of finite length can be represented as a finite impulse response filter (FIR) having as coefficients the samples of \( h(n) \).

The transfer function that characterizes the source-listener path in an acoustic space can be approximated by a linear, time-invariant system. Such transfer function is called room transfer function (RTF).
2.1. Impulse response, echogram and early reflections

The number of coefficients necessary to model an RTF using an FIR model is proportional to the sampling frequency (Fs) and to the time necessary for the energy present in the room to vanish, once the excitation source has been interrupted.

As an example, using a sampling frequency of 22.05kHz, the FIR model of an RTF of a lecture room with a duration of 0.73 seconds, that requires about 16000 taps, is shown in Figure 1.

An echogram, as shown below, is a useful tool to analyse the IR of an acoustic space. The echogram is calculated by the logarithm of the absolute value of the IR.

In the echogram it is easier to detect the most energetic, discrete reflections that characterize the IR. As reported in [1] the number of these reflections is small in comparison to the length of the whole IR. For example, in the reported IR there are only 61 coefficients that have intensity above –30 db (Figure 3). Usually these reflections are contained in the first 100ms of the IR (early reflections).

3. INVERSE FILTERING

From “Iterative cepstrum-based approach for speech de-reverberation” by Radlovic and Kennedy, 1999 [2]:

"Let \( h(n) \) be a causal and stable, non-minimum phase impulse response between an acoustic source and a microphone placed some distance apart in a reverberant room.

If the requirement for signal processing is that the waveform of the input (source) signal remains unchanged after passing through the equalized room transmission system, we may think of the equalization problem as that of finding an inverse impulse response function \( p(k) \) such that

\[
\begin{align*}
  h(n) \ast p(n) &= \delta(n - N_d) \\
  \text{where } n \text{ is a nonnegative time index, } \ast \text{ signifies the discrete linear convolution, } \delta(k) &\text{ is the unit sample sequence } (\delta(k)=1 \text{ for } k=0 \text{ and } \delta(k)=0 \text{ for any other } k) \text{ and } N_d \text{ a delay.}
\end{align*}
\]

3.1. Inverting and equalizing a single echo

Let us consider a single echo RTF of the kind:

\[
h(n) = \delta(0) + \alpha \cdot \delta(k)
\]

where \( k \) is the delay expressed in samples and \( \alpha \) the reflection gain.

Its Z transform is

\[
H(z) = 1 + \alpha \cdot z^{-k};
\]

this filter is also known as an FIR comb. Its inverse transfer function is

\[
G(z) = \frac{1}{1 + \alpha \cdot z^{-k}};
\]

this filter is also known as IIR comb. The inverse Z transform of the IIR comb is

\[
\alpha^{-n} \cdot \delta(n - m) \text{ with } m = 0, k, 2k, 3k, \ldots
\]

In the following example the IR of a single echo (\( k=50 \) and \( \alpha=-0.95 \)) and its truncated inverse filter are shown. Their convolution gives as a result a almost perfect approximation of a Dirac \( \delta \).

This simple example clarifies two facts:

1) By finding the inverse filter \( P(n) \), perfect de-reverberation can be achieved.
2) The problem of equalizing a complex RTF can be seen, by linearity, as equalizing multiple single reflections. But even a single strong reflection may require a very long (theoretically infinitely long) and very resonant inverse filter. However, since reverberation is essentially due to energy that is decaying in a finite amount of time, a FIR structure is still a reasonable approximation.

4. BLIND RTF EQUALIZATION

Many techniques based on supervised identification approaches to measure an IR exist [3][4]. Once the IR has been measured, the inverse filter can be estimated and an equalized version of the signal can be obtained. Even if robust inverse filter design approaches have been proposed [5], several problems are still open in the application to practical audio systems. If the system input is unknown, the previous techniques cannot be applied and the equalization problem becomes even more complex.

A large class of blind identification methods are based on higher order statistics [6]. A non-Gaussian statistic is needed since second order statistic is "phase blind". This means that second order statistic cannot distinguish between different minimum and non-minimum phase system representations.

Furthermore, for signals having complex spectral structure (e.g., speech, music) the problem of blind RTF identification is complex since the observed power spectrum is a combination of the signal and reverb characteristics. Therefore de-reverberation cannot be obtained by direct spectral "whitening". A method of separating the two models must be provided.

Blind de-reverberation is still largely an unsolved problem.

4.1. Kurtosis and LP residual

LPC analysis has been proposed, as a way to decouple the spectral structure of the input and the reverb [7][8].

Kurtosis has been proposed as a suitable higher order statistic for blind system identification [8].

\[
\text{kurtosis} = E \left[ \frac{(x - \mu_x)^4}{\sigma_x^4} \right] - 3
\]  

(7)

Kurtosis is a measure of the "peakedness" of the probability density function of a real-valued random variable [6].

A signal with sparse peaks and wide low level areas is characterized by a high positive kurtosis values.

In particular it has been observed [8] that, for clean speech signals, the kurtosis of LP residual can serve as a reverberation metric.

This observation can be explained roughly by the Central Limit Theorem (CLT): the sum of N arbitrarily (but identically) distributed RVs, converges to a Gaussian distribution.

By its nature, reverberation is the process of summing a large number of filtered and delayed copies of the same signal. Thus, very crudely, by the CLT, the reverberated signal has a more Gaussian distribution with respect to the original one.

The LPC residual of speech is mainly constituted by the glottal pulses, so it is sparse and characterized by high, positive kurtosis. Reverberation causes this signal to assume a more Gaussian distribution with decreased kurtosis.

As a consequence, by building a filter that maximizes the kurtosis of the residual it may be possible to identify the inverse function of the RTF and therefore to equalize the system. This approach is blind since it requires only the evaluation of the kurtosis of the residual of the system output.

In [8] the following time domain single channel structure was proposed.

![Figure 5: A single channel time-domain LMS algorithm for maximizing kurtosis of the LP residual.](image)

During tests performed with this structure using batch processing, we have observed stability problems and unexpected results. In some apparently unpredictable conditions, the algorithm captures a harmonic component and creates a resonant spike in the residual. Therefore even if kurtosis is smoothly maximized, the inverse filter might converge to a highly resonant state.

By creating isolated strong peaks in the residual (thus making it sparser) kurtosis increases, but the result is irrelevant for de-reverberation purposes. This issue can be associated to the extreme sensitivity of kurtosis to "outlying" values.

Examining the simplified expression of kurtosis for a zero mean, unitary variance RV \( y \)

\[
\text{kurtosis} = E \left[ y^4 \right] - 3
\]  

(9)

it can be noticed how its value is greatly affected by the values of \( y \) greater than 1. Similar criticisms of kurtosis have been raised in the context of Independent Component Analysis [9].

Let the RV \( y \) be generated by filtering a RV \( x \):

\[
y(n) = h^H(n) \cdot x(n)
\]  

(10)

the derivative of kurtosis expression is

\[
\frac{\partial \text{kurtosis}}{\partial h} = E \left[ 4y^3 \frac{\partial y}{\partial h} \right] = 4E \left[ y^3 \frac{\partial h^H \cdot x}{\partial h} \right] = 4E \left[ y^3 \cdot x \right];
\]  

(11)

due to it is not bounded and can theoretically diverge.

4.2. Maximum likelihood approach

In order to minimize the sensitivity of the algorithm to "outlying" values we propose a maximum likelihood (ML) approach.

Assuming an FIR filter so that

\[
y(n) = h^H(n) \cdot x(n)
\]  

(12)

the idea is to build \( h \) in order to achieve any desired probability density function of the output \( y \):

\[
\max_h E \left[ \log(P(y)) \right] = \max_h E \left[ \log(P(h^H x)) \right]
\]  

(13)
Defining the cost function as
\[ J = E\left\{ \log(P(y)) \right\} \]
its gradient is
\[ \frac{\partial J}{\partial h} = E\left\{ \phi(y) \frac{\partial P(y)}{\partial h} \right\} = E\left\{ \phi(y) \cdot x \right\} \]
where
\[ \phi(y) = \frac{1}{P(y)} \frac{\partial P(y)}{\partial y} \]
therefore the update equation to maximize \( J \) is
\[ h(n+1) = h(n) + \mu \cdot \nabla J(h(n)) = h(n) + \mu \cdot E\left\{ \phi(y) \cdot x \right\} \]

The probability density function of \( y \) is chosen in order to have high kurtosis and bounded derivative.
A probability density function with these properties is
\[ P(y) = \frac{1}{\cosh(y)} \]
giving
\[ \phi(y) = \frac{1}{P(y)} \frac{\partial P(y)}{\partial y} = \cosh(y) \frac{\partial}{\partial y} \frac{1}{\cosh(y)} = -\tanh(y) \]
The update equation now becomes
\[ h(n+1) = h(n) - \mu \cdot E\left\{ \tanh(y) \cdot x \right\} \]
Thus the hyperbolic tangent function replaces the kurtosis term. While the latter is unbounded and dominated by a cubic term, the former is bounded and insensitive to outliers.

5. RESULTS

In order to evaluate the proposed algorithm:
- an anechoic speech signal has been reverberated convolving it with a 16000 sample IR measured from a real room
- a 2000 tap FIR filter (corresponding to a time window of 90.7ms) has been used to equalize the real room IR.

To achieve a perfect equalization, a filter with a number of taps much greater than the length of the IR should be used. However, a shorter filter is expected to be less sensitive to noise during the adaptation and to be able to reduce the intensity of the early reflections in the considered time window (e.g. 2000 taps at a sampling frequency of 22050≈90.7ms). It is important to remember that the most energetic portion of an RTF is usually concentrated in the first 100ms.

From the echograms it can be noted that both the algorithms attenuate the most prominent early reflection (at sample position 1138). However, the kurtosis approach provides worst result and it introduces noise in the considered time window, making the processed IR noisier than the original one.

The batch ML algorithm delivers improved de-reverberation. Within the IR equalized using the ML algorithm, the strongest early reflection is attenuated by 13.5dB:

Original IR, strongest ER intensity = -14.7dB
Equalized IR, strongest ER intensity = -28.2dB

The average attenuation of all the IR intensity is of 4.93dB.
The average attenuation of the first 2000 taps is of 10.25dB.
5.1. Multi-channel algorithm

Equalization with multiple microphones is potentially easier than using a single source [10]. As suggested by Gillespie et al [8], a multi-channel implementation can be directly extended from a single-channel system (see Figure 8). There a four channel system based on the improved-kurtosis, online algorithm was developed.

As stated before, the objective is to maximise the kurtosis of the LP residual of \( y(n) \), where \( y(n) \) is given by

\[
y(n) = \sum_{c=1}^{C} h_c^T(n) \cdot x_c(n),
\]

where \( C \) is the number of channels.

The multi-channel update equation thus becomes

\[
h_i(n+1) = h_i(n) + \mu \cdot f(n) \cdot \tilde{x}_i(n)
\]

where the feedback function \( f(n) \) can take various forms (e.g. kurtosis or sigmoid). To jointly optimise the filters, each channel is independently adapted using the same feedback function.

While the extension to a multi-channel system is simple, the differences between a single channel approach and a multi-channel one are quite dramatic (indeed [8] provided no results for the single channel case). The reason is that it is often possible to represent the inverse filter using a set of FIR filters [10]. Recall that in the single channel case even a single echo can only theoretically be inverted using an IIR filter. Also, even when the full inverse is not attained the multi-channel setup is more similar to a beam-former and, since it mixes together filtered versions of the inputs, it can exploit constructive and destructive phase interference to de-reverberate. This greatly simplifies the filter complexity (see results).

This implies:

1) greater insensitivity to noise and better gradient estimation
2) less computational demand
3) less memory requirements

However the down side is it cannot offer “one channel to one channel” signal enhancement.

5.2. Synthetic example - equalization using a 4 channel system

Four long echoes (with delays of 2000, 2100, 2200 and 2300 samples and gains respectively of –0.5, 0.5, –0.5, 0.5) have been used to add reverb to the speech file. Figure 9 shows the impulse response of 1 of the 4 filters of the multi-channel system.

Here, it can be seen how the filter models the time delay between the taps (100, 200, 300 samples), rather than the actual delays of 2100, 2200 and 2300.

The spectrograms of the original, reverberated and equalized files are shown in Figure 10. The algorithm manages to suppress almost completely the echoes present in the reverberated file.

6. CONCLUSION

It has been demonstrated how the ML approach is effective in reducing the strength of the early reflections of an acoustic IR. However, the performances are poor in removing the late reverberant tail.

In this sense, two approaches should be investigated:

1) verify if a longer filter used with a larger quantity of input data can effectively remove the long reverberant tail (this approach is considered to be unlikely effective due to RTF’s coherence instability).

![Figure 8: Two channel system.](image)

![Figure 9: Impulse response of 1 of the 4 filters of the multi-channel system.](image)

![Figure 10: (a) original speech signal, (b) reverberated, (c) equalized by the multi-channel system.](image)
2) use a two step process cascading the proposed algorithm with a denoiser specifically designed to attenuate the long term reverb

The LP residual approach has been proposed as a method of decoupling the harmonic structure of speech and reverb. However, it can be criticized since both the LPC and the de-reverb FIR filter are convolutional operators. Thus there is a worrying ambiguity in the identifiability of the de-reverb filter. Essentially such algorithms are making an implicit assumption that these filters are in some sense orthogonal. Alternative methods based on the transients present within the input signal may provide a better criterion. This new approach could be extended to a more general class of musical signals.

Since the generation of sound by musical instruments is often associated with impulsive like phenomena, it should be investigated if and how the metric based on the kurtosis of LP residual can be generalized.

The ML algorithm has not been optimised for real-time purposes, even if its modification in this sense seems to be straightforward. It would be interesting to develop a multi-channel versions of the algorithm with an approach similar to the one proposed in [8].

Finally other techniques of blind identification are reported in the literature. It would be of interest to apply them to the de-reverberation problem and compare their results.

7. REFERENCES


PRACTICAL IMPLEMENTATION OF THE 3D TETRAHEDRAL TLM METHOD AND VISUALIZATION OF ROOM ACOUSTICS

Stanley J. Miklavcic and Johan Ericsson

Department of Science and Technology
University of Linköping,
S-601 74, Norrköping Sweden
stami@itn.liu.se

ABSTRACT

This paper concerns the implementation of a 3D transmission line matrix (TLM) algorithm based on a tetrahedral mesh structure and visualization of room acoustics simulation. Although a well known method, TLM algorithms implemented in 3D are less commonly found in the literature. We have implemented the TLM method using a tetrahedral mesh of pressure nodes with transmission lines lying superimposed on nearest neighbour bonds of a tetrahedral atomic lattice. Results of simulations are compared with those of a standard 3D cartesian mesh and a 2D mesh implementation of TLM. An important feature is a useful graphics interface designed for user-friendly control of room acoustics simulation and visualization in arbitrary shaped rooms containing objects of arbitrary size and number. The paper includes brief discussions of results of using different techniques for modeling totally absorptive or partially absorptive boundaries.

1. INTRODUCTION

Significant effort has been directed to applications of the transmission line matrix (TLM) method in the field of room acoustics simulation. Discussions have focused on topics ranging from the implementation of different boundary conditions [1] to the means of minimizing numerical problems inherent in the discrete TLM approach, such as dispersion effects [2], to means of incorporating boundaries that are not coincident with discrete mesh nodes [3]. The advances made in these directions increase the potential of the TLM method as a method for quantitative study of room acoustics. This is becoming all the more important as computer resources improve and as new areas of application appear. One of the modern areas of application is auralization (the audio counterpart of visualization), in which the artificial reproduction of sound in a simulated environment is accomplished by a convolution of a pure sound signal (representing a given event) and the impulse response of a simulated room. This area of application has a considerable commercial market value for the computer games industry and revolves around coupling realistic audio effects to virtual graphics effects. Naturally, this must be achieved in real-time and must take into account as large a proportion of the frequency spectrum as possible and must also be able to accommodate mobile sources and receivers. These three requirements limit the usefulness of geometric acoustics techniques (such as ray-tracing and beam-tracing [4, 5, 6, 7]). Even though geometric techniques may be quicker to model early reflections, the TLM method, with its ability to incorporate full room reverberation effects including interference, diffraction and frequency effects as well as mobile sources, can represent a potentially viable alternative.

Theoretical studies of the TLM method in terms of defining the quality of acoustic wave reproduction, quantitative accuracy, physical limitations, error analysis, etc, have predominantly been based on the 2D TLM case, either explicitly or with that case implicitly in mind [1, 2, 3, 8, 9, 10]. Furthermore, applications to real systems based on implementation of the 2D TLM method still appear in the literature [11, 12]. Practical examples of 3D implementation are not as prevalent, though some examples can be found [13, 14]. The two strongest reasons for the relatively little effort devoted to 3D TLM modeling are, firstly, the increased simulation time involved with updating a 3D mesh of points and secondly the difficulty of eliciting or extracting information from the 3D results. In this paper we argue that the latter need no longer be a problem with the present availability of computer graphics software, while the former will become less significant with the continuing advance in computer power. Indeed it is now apt to question whether the greater emphasis on 2D simulations should continue. In this paper we discuss two 3D implementations of the TLM method and reflect on their relation to 2D versions.

The obvious generalization to 3D of the most common 2D TLM implementation involves an extension of the square lattice of nodes and equal length transmission lines to a regular cartesian cubic lattice of nodes, each connected to its nearest neighbours by 6 transmission lines. Naturally, this can and does lead to an undesirable increase in computation time. To minimize this increase we have implemented the TLM on an alternate, tetrahedral lattice structure. Here, the 3D array of nodes are once again connected by 4 transmission lines. To justify the numerical effort we rely on the analysis of Van Duyne and Smith III [15, 16] who have, unknown to us at the time of implementation, also considered the notion of a tetrahedral lattice as an alternative and shown that the TLM method on a tetrahedral lattice is locally equivalent to the 3D acoustic wave equation. We compare results of our 3D tetrahedral TLM implementation with those of a 3D cartesian implemented scheme as well as with results of 2D cartesian implementation. We apply our numerical simulation(s) to a realistic model of a concert hall as presented in [12].

2. 3-D TRANSMISSION LINE MATRIX

2.1. Basic Principles

We imagine space to be divided up into a discrete mesh of nodes, each connected with its nearest neighbours by means of digital, bi-
directional waveguides. The waveguides allow the transmission of acoustic signals in two directions. At a node several waveguides meet which gives rise to an impedance mismatch between any one of the waveguides and the remainder. This mismatch is a source of reflection at the junction end of the waveguide.

The transport behavior of signals along waveguides and at waveguide junctions can be determined assuming that the cross-sectional dimensions of the waveguides are such that only plane wave signals propagate. The plane wave approximation leads to a simple proportionality relation between acoustic pressure, $p$, and volume velocity, $v$, in terms of acoustic impedance, $p/v = \pm Z$, where the positive/negative signs depend on direction.

Suppose we have the general situation depicted in Figure 1.

![Figure 1: Schematic of a waveguide junction.](image)

An input signal along waveguide, 0, meets a junction and experiences a partial reflection. The total pressure along this waveguide is the sum of these two pressure signals, $p_0^{(i)} + p_0^{(r)}$, while the volume velocity is $v_0^{(i)} + v_0^{(r)}$. The partially transmitted energy is divided up into signals which carry along the remaining waveguides, $p_k$.

The total transported material must be conserved. For a constant density fluid and constant cross-sectional waveguides this gives the condition that the volume velocity must be conserved

$$v_0^{(i)} + v_0^{(r)} = \sum_{j=1}^{N} v_j. \tag{1}$$

Furthermore, the pressures at the junction must be equal

$$p_0^{(i)} + p_0^{(r)} = p_1 = p_2 = \cdots = p_N. \tag{2}$$

This gives rise to the junction pressure relation

$$p_0^{(i)} + p_0^{(r)} = \frac{\sum_{j=1}^{N} p_j}{Z_0} = \frac{1}{Z_0} \sum_{j=1}^{N} \frac{1}{Z_j} Z_0. \tag{3}$$

The reflection and transmission coefficients for the signal in waveguide 0, are

$$R = \frac{p_0^{(r)}}{p_0^{(i)}} = \frac{1}{Z_0} \sum_{j=1}^{N} \frac{1}{Z_j}, \quad T = 1 - R. \tag{4}$$

For waveguides of identical characteristic impedance these reduce to

$$R = \frac{1 - N}{1 + N}, \quad T = \frac{2N}{1 + N}. \tag{5}$$

For a 2D cartesian mesh with each node being the junction for $N + 1 = 4$ equal-impedance waveguides, the pressure transmission coefficient per waveguide, $t = T/N = 1/2$ and the pressure reflection coefficient, $R = -1/2$. For incident waves along all 4 waveguides at junction, $K$, this result generalizes to a signal transmission described by the scattering matrix,

$$S_{2D} = \frac{1}{T} \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}, \tag{6}$$

while the total pressure at junction, $K$, can be expressed as

$$p_K = 2 \sum_{j=1}^{N+1} \frac{p_j}{Z_j} \sum_{j=1}^{N+1} \frac{1}{Z_j}. \tag{7}$$

### 2.2. Boundary Effects

Scattering matrix (6) cannot be used at boundaries. Not only is signal scattering along certain directions prevented, there are different acoustic properties at a boundary. Boundary properties considered here are perfectly reflecting, perfectly absorbing, and partially absorbing-partially reflecting boundaries. In the reflecting cases, only specular reflection is considered for which incoming signals to boundary nodes are returned along the same waveguide with opposite sign (for diffuse reflection one can invoke the ideas of Laird, et al. [3]). With partial absorption we have only used the simple method of multiplying the incident waveguide signal with a reflection coefficient, $\alpha \in [0, 1]$ to obtain the reflected signal traveling back along the same waveguide. This implies use of the matrix

$$S_{\alpha} = \alpha \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{8}$$

In the case of total absorption we have implemented and tested several methods ranging in complexity. The simplest approach is to set the above reflection coefficient, $\alpha$, to zero. This is well known to give a good but not ideal representation of absorbing boundary conditions, with ghost reflections prevailing. A second option, Berenger’s perfectly matched layer [17, 18], is to surround the physical boundaries with additional layers of nodes each of which behave as a normal interior node except with a scattering matrix $S_{\beta} = \beta S_{2D}$ with $\beta \in (0, 1)$. This allows signals to extend outside the simulation space, tapering off gradually. We have found this method works well in 2D but becomes prohibitive in 3D due to the significant number of nodes that must be introduced.
Finally, we have implemented the Taylor series approximation approach [1], whereby a Taylor series expansion is used to estimate the signal strength to be input into a boundary node, based on signals at neighbouring nodes at previous iterations. We remark that the algorithm as presented in [1] seems best suited to a 2D cartesian mesh, while questions arise as to how to implement the algorithm in 3D, especially for the tetrahedral geometry.

Variations on these themes must be considered for objects existing inside the room itself (e.g., tables, chairs, people, etc.). For example, an absorptive 3D object or even a 2D object (such as a thick heavy curtain) would be difficult to model using the Berenger method which is designed for exterior boundaries (additional sites are introduced outside the simulation space). Here the simplest approach has been taken for all interior cases: use of the scattering matrix, $S_\alpha$ with $\alpha \in [0, 1]$.

### 2.3. 3D Cartesian Lattice

In 3D the space filling generalization of the 2D square matrix usually employed, involves a 3D array of nodes each connected to nearest neighbors via 6 bi-directional, equal-length waveguides. Apart from the increase in number of nodes, the added effort of shunting signals along the extra waveguides increases the computational demand, thereby decreasing the effectiveness of the TLM algorithm. Here, using (5) with $N = 5$, we have $R = -2/3$ and $t = T/N = 1/3$. The scattering matrix for this case of six transmission lines is then a $6 \times 6$ matrix, $S_{6,6}^{DC}$, with diagonal elements $R = -2/3$ and off-diagonal elements $t = 1/3$.

### 2.4. 3D Tetrahedral Lattice

Past experience with 3D space-filling lattice structures [19, 20], has led us to consider the tetrahedral lattice structure of diamond, which has a coordination number of 4. Thus, if the TLM were based on a tetrahedral mesh with waveguide junctions placed on the atomic lattice sites, then each junction or node could be connected to its nearest neighbours via 4 waveguides of equal length. Scattering then again involves (6). This should in principle introduce a computational saving compared to the 3D cartesian lattice, back to the state enjoyed by the 2D simulation. The qualifier that these waveguides are of equal length implies that the signals will be transmitted in an arbitrary way with the same speed. Artificial anisotropy in wavespeed is thus reduced [2, 15, 16, 22].

In addition, comparing the elements of the scattering matrices, $S_{6,6}^{DC}$ and $S_{6,6}^{T}$, we see that the number of operations involved in the tetrahedral case is also reduced since only division (by 2) is involved as opposed to division (by 3) and multiplication (by $m$).

Furthermore, division by 2 can be implemented as a right-shift in binary arithmetic [21, 23] with some minor additional computational saving. Finally, to model a given volume requires fewer tetrahedral nodes than cartesian nodes with the same $\Delta l$. For example, to model a $1m^3$ box with $\Delta l = 0.01m$, a cartesian mesh requires $1/(0.01)^3 = 1,000,000$ nodes; a tetrahedral mesh requires only $1/((2 \cdot 0.01 \sin(54.45^\circ))^2 (2 \cdot 0.01 \cos(54.45^\circ))) \approx 649,519$ nodes, a saving of roughly a factor of 1/3.

Taking the waveguide length to be $\Delta l$, the positions of the nearest neighbour junctions to a given junction can be determined knowing that the angle between waveguides is $100.5^\circ$. One fact not previously advertised in the literature is the single greatest disadvantage of the tetrahedral mesh. This is that the mesh naturally distinguishes between two different types of nodes, depending on the orientation of their neighbours. In Figure 2 we see that the nearest neighbour directions from node $A$ are in counterclockwise direction about $z$-axis $(1, -1, -1), (1, 1, 1), (1, -1, -1)$ and $(-1, 1, 1), 1$, while the nearest neighbour directions from node $B$ are $(1, -1, -1), (1, 1, -1), (1, 1, 1)$ and $(-1, -1, -1)$. The neighbours of node $B$ are rotated $90^\circ$ about the $z$-axis with respect to neighbours of node $A$. These two node types lie on different planes. Nodes lying on the same plane as node $A$ will bear the same relation to their neighbors as $A$ does to its. This feature repeats every second plane: the plane containing node $C$ is structurally identical to the plane containing node $A$. Likewise, the plane with node $D$ is identical to the plane with node $B$. This periodicity can be utilized in the programming of the TLM scattering process by defining two different scattering processes, one for each type of node [23].

As mentioned three possible boundary types: perfectly reflecting, perfectly absorbing (transparent) and partially absorbing were considered. Reflecting and even partially reflecting boundary conditions are sufficiently easy to implement and are quite accurate. For partially reflecting walls, the error associated with ghost reflections are relatively minor compared to the dominating reflected signal strength. For perfectly absorbing walls we have experienced significant complications with our interpretation of Murphy and Howard’s scheme [1]. The Taylor series ABC on the tetrahedral lattice did not produce the desired result. We expect that in the original formulation the method relied to some extent on orientation of the boundary: lying parallel to one of the coordinate axes and perpendicular to the other. This is not the situation in our 3D implementation. Work aiming to adapt the idea to the present case is continuing.

### 2.5. Graphics Interface

One of the difficulties incurred with implementing a 3D simulation is in assessing acoustic information. Simple pressure -v- time plots associated with specific points in the available 3D space, as we show below, are necessary for a true quantitative analysis. However, these are limited in scope. In particular, it is difficult to know apriori what space points need to be considered for sam-

---

[Figure 2: Tetrahedral lattice]
pling of information. This is especially true if the virtual room is complex in shape and totally lacking in symmetry. This problem can be remedied if one is able to visualize the entire acoustic field. Then one can quickly ascertain in which regions interesting phenomena occur and where quantitative focus should be placed. Figure 3 shows a graphic interface we have created in tandem with our TLM simulation [24]. With this interface one can obtain both an overall view of the acoustic field in three dimensions (the large window in the screen dump) and quantitative pressure - vs- time plots (lower window) as recorded by any number of virtual microphones positioned in $xyz -$space according to user specifications (data shown either as acoustic pressure in Pa or pressure level in $\text{dB}$). The interface also allows the latter information to be saved as raw data for further audio or analytical processing. In the main window, the room, whose shape and size is specified by the user, can be rotated about any axis through the central point in order to be able to see the acoustic field everywhere - there are no blind spots. Finally, apart from room shape and size, the interface allows the user to introduce arbitrary objects to the room. The surfaces of these objects can be modeled as having any degree of absorption capability from perfectly absorbing to perfectly reflecting. Since the visualization takes a significant fraction of the total simulation time, it is also possible to turn off this feature to speed up the simulation while allowing continued flow of data from the virtual microphones. The latter data can be plotted, saved as raw data or used in audio playback.

This user-friendly interface has the potential to be used for commercial purposes, e.g., by designers and architects in the construction industry, for research by acousticians, as well as for teaching purposes to students of physics and mechanics (studying wave phenomena), engineering (in particular, for acoustic engineering) as well as students of music.\footnote{This is currently being used at this university for students of media technology and for students of music production.}

3. RESULTS

3.1. Comparison of 3D Cartesian and 3D Tetrahedral Meshes

In Section 2.2 we gave reasons for why a saving in computational time would be expected in comparisons between the 3D cartesian and tetrahedral mesh-based TLM modeling. This expectation was realised in all our simulations. As an example, Table 1 shows results of a simple simulation where sound in a rectangular prism shaped room of dimensions $2.0 \times 2.0 \times 1.0$ was simulated using TLM on both 3D cartesian and 3D tetrahedral meshes. In both cases, the nearest neighbour internode distance was kept constant at 0.02 m. With or without the added computational cost of visualizing the simulation, the tetrahedral TLM performs considerably faster that the 3D cartesian algorithm, by approximately a factor of 2.\footnote{All calculations were run on a Pentium III 1000 Mhz PC with 512 Mb RAM and 30 Gb ROM. The significance with the information given lies, however, more with the difference in simulation times rather than absolute times.} This factor also applies to the visualization component, which in relation to total time taken is 25%. The differences in simulation time are mostly due to the fewer number of nodes to be updated, but also the fewer waveguide/node scattering operations to be performed.

![Screen dump of the final programme's graphical interface. Screen shows the structure inherent in the concert hall model studied by Morton (2001).](image)

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>3D Tetrahedral</th>
<th>3D Cartesian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internode distance (m)</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>TLM iteration time (s)</td>
<td>0.490</td>
<td>0.843</td>
</tr>
<tr>
<td>Visualization time (s)</td>
<td>0.121</td>
<td>0.264</td>
</tr>
<tr>
<td>Total iteration time (s)</td>
<td>0.611</td>
<td>1.107</td>
</tr>
</tbody>
</table>

Table 1. Comparison between TLM simulation using cartesian and tetrahedral meshes for a room of size $2.0 \times 2.0 \times 1.0 = 4.0 \text{ m}^3$.

Other comparisons can be found in [23]. More detailed results addressing a quantitative comparison between the tetrahedral 3D model and analytic data for a rectangular geometry will be published elsewhere, as will discussions of spatial dispersion [2, 22].

3.2. 2D -vs- 3D Room Acoustics Simulation - A Practical Example

Clearly, a 2D simulation will not produce the same result as a 3D simulation. A 2D model can be thought of as a 3D model, being infinite in the third dimension, in which case information about the effect of boundaries in this third dimension is lost. A pressure point source in 2D is then re-interpreted as a line source in 3D and pressure values are quoted in units of pressure per unit length. Alternatively, a 2D model can be considered when confinement in a given direction is extreme compared with the other two. With either interpretation it is inappropriate to compare directly quantitative 2D and 3D pressures in extended 3D situations. Consequently, quantitative information provided by a 2D simulation is also inappropriate for assessing the acoustic properties of a real world environment. Since 2D simulations are used to study acoustic behavior in 3D systems [11, 12], it is legitimate to even question the relevance of qualitative information produced. We address these issues using a particular case study. Here, we demonstrate the disparity between 2D and 3D TLM models using the example of a concert hall studied by Morton [12].
The hall, whose original purpose was as a sports venue, doubles as a concert hall for theatres and musical performances. Morton’s simulations aimed at modeling solutions to the inherent acoustic problems of the $34 \times 18 \times 16 \text{m}^3$ hall. The 2D model of the hall employed by Morton and the true 3D shape of the hall are shown in Figures 4 and 5, respectively. Materials such as wood, fibreboard and concrete are used in its construction and furnishings; the concrete walls are draped by heavy curtains during performances. Relevant acoustic data are provided by Morton [12].

To perform our comparison, Morton’s 2D simulations were reproduced by us using information on acoustic and structural features of the concert hall as provided by Morton. Unfortunately, some details (in terms of dimensions, absorption properties, especially of objects in the room) were not provided, which made quantitative comparison between our respective 2D simulations somewhat difficult. Nevertheless, a reasonably accurate reproduction was achieved (see [23] for details). To allow for greater flexibility in our comparison between 2D and 3D simulations, we used our own implemented 2D model, rather than rely on data provided by Morton. Times quoted in the figures are based on iteration time (different for 2D and 3D) multiplied by number of iterations.

In both models a Gaussian pulse is sent out from a virtual source placed on a stage fixture. Recordings were made by 8 virtual microphones placed at increasing distances from the stage as shown in Figures 4 and 5. Typical quantitative comparisons of acoustic pressures and pressure levels are shown in Figures 6 and 7. Again, the 2D and 3D magnitudes are not comparable. Overall, the 3D decays much more rapidly than in 2D, as expected. Clearly, 2D models would give a false prognosis of reverberation times for a real structure. With regard to detail, there are few similarities between the results shown in Figures 6 and 7. This is in contrast to a comparison between 2D and 3D simulations of a straight rectangular prism [23]. The simpler geometry there led to more similarly structured curves making for easier identification of contributing reflections (both included and absent in 2D). Here, the more com-
plex geometry with the sloping roof gives rise to a complicated reflection scenario. Only the strong reflections from the back wall of the hall (second strong peaks in Figure 7) and similar ones from the front are discernible.

4. SUMMARY AND CONCLUSIONS

With the continued advance in computer performance we are likely to see an increased usage of room acoustics simulation methods for architectural purposes, music instrument design, as well as the expanding new front of virtual environments. The 2D TLM method has always intimated the potential of TLM as a contender for three dimensional applications. However, in comparison with 2D, 3D acoustic implementations have not been so forthcoming, due partly to a more involved programming task, partly to the increased simulation time and partly the problem of extracting information from a simulation. We have here demonstrated that this potential can indeed be realised in practice by using a judicious choice of discrete mesh and newly available computer graphics methods. Our results confirm the fact that the tetrahedral mesh, despite the inherent programming complexity, is superior to the cubic mesh in terms of computational time as well as total mesh size. 3D visualisation adds the dimension of allowing one to see the acoustic field during the simulation and decide where special attention need be directed. 2D TLM models fail to represent real room acoustics for rooms of complex form. They can, however, give some qualitative information for simple box-shaped rooms (data not shown).

Further steps can be taken to improve the performance of the 3D tetrahedral TLM algorithm, primarily by exploiting the natural existence of two interconnecting classes of nodes. It is possible to utilize the two sets of matrices in a parallelized code to speed up computation time. A second important issue is that of the boundary conditions. More efficient approaches, better suited to the current tetrahedral system, must be applied.

5. REFERENCES

ROOMWEAVER: A DIGITAL WAVEGUIDE MESH BASED ROOM ACOUSTICS RESEARCH TOOL

Mark J. Beeson and Damian T. Murphy

Media Engineering Group, Dept. of Electronics
University of York, Heslington, York, UK, YO10 5DD
mjb128@ohm.york.ac.uk, dtm3@ohm.york.ac.uk

ABSTRACT

RoomWeaver is a Digital Waveguide Mesh (DWM) based Integrated Development Environment (IDE) style research tool, similar in appearance and functionality to other current acoustics software. The premise of RoomWeaver is to ease the development and application of DWM models for virtual acoustic spaces. This paper demonstrates the basic functionality of RoomWeaver’s 3D modelling and Room Impulse Response (RIR) generation capabilities. A case study is presented to show how new DWM types can be quickly developed and easily tested using RoomWeaver’s built in plug-in architecture through the implementation of a hybrid-type mesh. This hybrid mesh is comprised of efficient, yet geometrically inflexible, finite difference DWM elements and the geometrically versatile, but slow, wave-based DWM elements. The two types of DWM are interfaced using a KW-pipe and this hybrid model exhibits a significant increase in execution speed and a smaller memory footprint than standard wave-based DWM models and allows non-trivial geometries to be successfully modelled.

1. INTRODUCTION

Architects and composers have long known the dramatic gravitas that can be imposed upon music or spoken word through the creative use of acoustic spaces. This has been exploited for hundreds, if not thousands of years in places of worship and great concert halls by composers such as Mozart, Berlioz and Mahler. In more modern times the psychoacoustic cues that impart this sense of space are usually generated by sound engineers to give an acoustic image that differs from the listening environment into which the sound is actually delivered. Such techniques are often used in theatre, cinema and more recently, interactive multimedia applications and give the composer/audio engineer/sound designer the artistic freedom to place the sound source and listener where required with absolute and complete control.

Acousticians, engineers and practitioners have sought to better understand the underlying physical principles that govern the behavior of sound in an enclosed space. This knowledge, when combined with modern computing techniques has yielded numerous models that aim to synthesize a RIR suitable to impart the notion of a particular acoustic environment upon an anechoic (or dry) sound. These techniques can be traced back to simple electro-mechanical devices, the first digital feedback networks, right through to the modern techniques described below.

One option for acquiring RIRs is to make a direct measurement in the desired space although this can be both time consuming and logistically difficult. The other option (which is the only possibility if the RIR of a virtual space is required) is to use a computational model to produce a virtual RIR that is an accurate analogy of the equivalent real-world space.

Geometrical acoustic techniques are the most popular solution for virtual space modelling with current architectural acoustics programs [1], [2] making use of hybrid image-source [3], ray-tracing [4] and beam tracing [5] techniques to derive RIRs as well as other more general acoustic properties. This is achieved by calculating a sufficient proportion of all the possible propagation paths that exist between a sound source and receiver through the geometric interaction of a sound-ray and the surfaces present in the model. Although these techniques produce RIRs appropriate for a digital implementation using delay lines, ray-tracing techniques have a number of limitations. The discrete, clearly defined reflection patterns that result from these geometrical methods have to be convolved with HRTF data in order to make them suitable for auralization purposes and they do not take into account wave interference effects. As such, these geometric methods are valid for high frequencies only and are less appropriate for low frequencies where the wave based properties of sound propagation and the presence of sparsely distributed modal frequencies tend to dominate. They are further limited in their ability to successfully model diffraction effects and hence by extension, sound occlusion due to objects being present in the propagation path, resulting in potential spatialisation errors. The rays used in ray-tracing have no cross-sectional area, whereas the rays used in beam-tracing are often conical or tetrahedral in shape and expand in area as they travel away from the source sound. This allows greater geometrical coverage than ray-tracing for the same number of rays or beams, resulting in quicker detection of valid source-receiver paths.

Finite Element Models and Boundary Element Models offer iterative methods for calculating the resonant frequencies present within an enclosed space. Although accurate these methods are computationally intensive, depending on dense mesh structures to produce results across the audible spectrum. These techniques have been used to create RIRs of virtual spaces [6], but alternative modelling techniques that produce equally valid results with less computational overhead and greater flexibility in terms of implementation and realization are currently more common. One such method uses the multi-dimensional digital waveguide mesh (DWM) and this paper introduces a new research tool based on this algorithm and is laid out as follows. Section 2 introduces the basic concept of the DWM and summarizes some of its implementation and realization are currently more common. One such method uses the multi-dimensional digital waveguide mesh (DWM) and this paper introduces a new research tool based on this algorithm and is laid out as follows. Section 2 introduces the basic concept of the DWM and summarizes some of its limitations. Section 3 presents the RoomWeaver DWM modelling tool and describes how it defines a bounded geometry and...
populates it with a mesh structure. The extensible and open-ended design of RoomWeaver is demonstrated in Section 4 through the introduction of a new efficient 2D mesh-type based on the KW-pipe. Results based on the use of this new mesh-type are then presented demonstrating its validity for RIR generation and its computational efficiency over other mesh structures. Section 5 summarizes the work presented in this paper.

2. THE DIGITAL WAVEGUIDE MESH

Digital Waveguide Mesh models techniques have been shown to be appropriate for simulating the acoustic properties of an enclosed space [7], [8], [9]. Although computationally intensive for large spaces or long reverberation times, wave propagation effects are an inherent part of the implementation [10], requiring no additional computational load, and hence this method seems appropriate for modelling the spatial scattering effects due to architectural features that may be present in an enclosed space.

The digital waveguide mesh is derived from the 1-D digital waveguide used extensively for physical modelling synthesis [11]. Higher dimension mesh structures are constructed using bi-directional delay lines and scattering junctions which act as spatial and temporal sampling points within the modelled space. It is also possible to use different mesh topologies to model the same physical space. Figure 1 shows two such topologies – the tetrahedral mesh and the 3-D rectilinear mesh – both of which can be used to model wave propagation through a 3-D space.

Figure 1: The 3D tetrahedral waveguide mesh (a) and the 3D rectilinear waveguide mesh (b), both of which could be used to model the same 3-D space.

Waveguide mesh models are limited by dispersion error, where the velocity of the propagating wave is dependent upon both its frequency and direction of travel, leading to wave propagation errors and a mistuning of the expected resonant modes. The degree of dispersion error is highly dependent upon mesh topology and has been investigated in [12] and can be compensated for to some extent using frequency warping techniques [13]. Current work involves the development of DWM models to deal with frequency and direction dependent reflection at a boundary. Diffusion has yet to be modelled successfully and research into a perfect anechoic boundary shows some promise although they have yet to be implemented with complete success [14].

Perhaps one reason for the lack of perceived awareness of DWM methods in more general room acoustics modelling is that no intuitive software exists for the non-specialist and specialist alike to experiment with. At present the generation of RIRs using DWM models requires meticulous and time consuming editing of computer code to set up all but the simplest room geometries and mesh topologies. This means that any research scientist wishing to do work in the field has a steep learning curve to overcome as they write and maintain their own code base, making it difficult to quickly try out new ideas and share knowledge. It is for these reasons that the IDE style RoomWeaver research tool was commissioned allowing the user to intuitively set up the geometrical and source/receiver parameters required to generate a RIR by means of a simple Graphical User Interface (GUI). The user may then pick from a variety of mesh topologies and types to generate an appropriate RIR.

Additionally RoomWeaver allows the DWM researcher to develop and test new mesh topologies and types using an extensible plug-in architecture. This system shields the mesh developer from the complexities of the RoomWeaver code base and greatly reduces the time required to develop and test new mesh structures.

3. AN OVERVIEW OF ROOMWEAVER

The current implementation of RoomWeaver (Figure 2) runs on a PC using the Windows operating system and is driven by a platform independent scripting language. This scripting language is used to define all relevant parameters, from the definition of workspaces, projects, room geometry, surface properties, material properties and mesh properties, down to the position and Input/Output attributes of sources and receivers. All of these files observe a similar syntax and standard constructs such as If-Else statements and For-Loops, and variable manipulation is supported in all script files. This architecture means that a simple command line port of RoomWeaver should be trivial to implement on any platform. Unfortunately the DLL based plug-in system is specific to the Windows platform, so if this aspect is to be transferred across platforms some redesign must be undertaken to achieve platform independency.

3.1. Modelling Rooms using RoomWeaver

RoomWeaver uses a hierarchal approach when organizing the data required to successfully model a room. This data is browsed using the workspace navigator pane in the GUI as shown in Figure 2.

Figure 2: A Screenshot of RoomWeaver showing the 3D View and Workspace navigator
3.1.1. Defining the Geometry

Once a project has been set up it is necessary to define a room geometry. Within RoomWeaver a geometry consists of several surfaces that in turn consist of a series of connected points. These surfaces may be organised into surface groups, allowing geometry to be defined in a logical, hierarchical manner. The rooms must be defined in a text file using RoomWeaver’s object-orientated scripting language. The scripting language allows each surface and group to be added individually, but it also caters for Objects and Models. For instance it is possible to define a model of an auditorium chair at only one point in a file and then create several instances or Objects of the model within the geometry with a single line of text (See Figure 3). The geometry file can also include Model Libraries to make the best use of any pre-written models. Models can take input arguments that change their behaviour, and surfaces and groups can be moved, rotated and scaled using the script.

3.1.2. Manipulating Materials

Materials are assigned to surfaces in the geometry file, but once loaded the materials on the walls can be easily changed and saved as surface sets, effectively allowing the same geometry to have several ‘outfits’. RoomWeaver includes a library of materials that can be applied to a surface (adapted from the Odeon Material Library, downloadable as part of the Odeon package from www.dat.dtu.dk/~odeon/), the properties of which may be edited and new materials can also be created from within the program. Materials have absorption coefficients for any number of angles in eight octave bands, plus diffusion coefficients for any number of surface sizes, again in eight octave bands.

3.1.3. Sources and Receivers

When satisfied with the geometry settings, sources and receivers are defined within the space. Again, the location and input/output properties of sources and receivers are defined in script files and are organized in a hierarchical manner. Sources and receivers may have geometry associated with them - for instance if a binaural response is required a model of a human can be inserted into the geometry set with virtual microphones placed at the ears [10]. RoomWeaver also offers input/output features for transducers, including auto-normalized output, resampling of input files to the mesh sampling rate, time delayed & signal-triggered sources plus individual channel gains for input and output signal files.

3.2. Running the Simulation

Once a virtual space has been defined the structure will need populating with a DWM, either 2D or 3D, together with the associated topologies and mesh types.

3.2.1. Creating the Mesh

RoomWeaver maps all mesh topologies to a rectangular array in the 2D case, or cubic array in the 3D case. The mesh array takes its size from the bounding box of the geometry. It is possible to rotate the geometry to create the smallest possible meshing volume. 2D meshes can exist along any plane within the geometry and can even be locked to key sources and receivers ensuring that the mesh intersects transducers of interest.

The choice of mesh topology available will depend on whether the mesh is 2D or 3D. Although certain topologies exhibit better dispersion characteristics than others, there will also be an associated implication on the efficiency of the RIR...
generation algorithm. Once the topology is decided upon, a compatible mesh-type plug-in must be selected (ie. **Finite Difference or Wave-based**), and again mesh-types will vary in efficiency, quality and the types of geometry they can successfully deal with. RoomWeaver will only allow mesh-types that are appropriate for the chosen topology.

Meshes are grown to fit the room at simulation run-time using the selected topology plug-in. The mesh will grow from a user defined point in the room, with the meshing algorithm filling the modelled space with free-air type nodes until it encounters a surface. At these boundaries special boundary nodes are created that can take on the properties of the encountered surface - Figure 4(a). RoomWeaver also provides a tool to examine the meshing process. This is extremely useful for testing new topology plug-ins and checking the geometry for leaks if warning messages are reported.

### 3.2.2. Generation of Room Impulse Responses

The main variable set at simulation run-time is the mesh sample rate given by:

$$f_{\text{update}} = \frac{c \sqrt{N}}{d}$$  \hspace{1cm} (1)

where $c$ is the speed of sound, $N$ is the dimension of the mesh and $d$ is the inter-nodal distance. For example, for a 2D mesh with a target sample rate of 44.1 kHz, and $c = 343$ m/s, an inter-nodal distance of 0.011 m is required. Ultimately this value of $f_{\text{update}}$ will dictate the quality of the eventual output, with previous studies showing that a typical mesh gives a valid bandwidth only as far as $f_{\text{update}}/4$ [15]. However, large sample rates require exponentially denser meshes and hence use more computer memory and take longer to run. Other options at run time include setting a time or dB limit that, once reached, will terminate the modelling process. When the mesh is actually run the mesh-type plug-in is loaded and executed. The mesh nodes are created according to the plug-in specification and the algorithm will begin to generate RIR data for all of the active receivers present in the model. The impulse responses are stored as wave files in a unique file folder for each simulation run, preventing the user from accidentally overwriting previous RIR data.

Visualization options allow the user to choose to watch an animation of 2D or 3D meshes as they run as shown in Figure 4(b). This is useful for checking that the model is running as intended, observing wave behaviour and for testing mesh-type plug-ins, although visualization is processor intensive and will significantly slow down the simulation process. Due to the lengthy running time of these mesh models a function that enables room settings to be saved as **snapshots** is included, which stores all the present settings that affect the modelling process. These snapshots can be then be used to queue a sequence of simulations at run-time, allowing several simulations to be run overnight or at a weekend.

## 4. PLUG-IN CASE STUDY

To demonstrate the flexibility of RoomWeaver a short case study is presented that implements a new, efficient mesh type as a RoomWeaver plug-in.

### 4.1. The Hybrid Mesh & The KW-Pipe

Traditionally there have been two mesh types used in DWM simulations; the wave based scattering mesh and the finite difference (FD) mesh - a simplified formulation of the scattering equations. The scattering equations are derived from the 1D digital waveguide, where the sound pressure at a propagating wave signal at any given junction is defined as the sum of the input and the output to the node:

$$p_i = p_i^{+} + p_i^{-}$$  \hspace{1cm} (2)

Further, the sound pressure at a lossless scattering junction with $N$ connected waveguides can be expressed as:

$$p_j = \sum_{i=1}^{N} p_i^{+} Z_i^{-1}$$  \hspace{1cm} (3)

where $p_i$ is the pressure in waveguide element $i$, $Z_i$ is its associated impedance and $p_j$ is the total pressure value at the scattering junction itself. As the waveguides are equivalent to bi-directional unit-delay lines, the output to a scattering junction is equal to the output from a neighbouring junction into the connecting waveguide at the previous time step:

$$p_{i,j}^{+} = z Z_i p_{i,j}^{-}$$  \hspace{1cm} (4)

Using (2), (3), and (4) it is possible to derive an equivalent finite difference formulation for these scattering equations in terms of junction pressure values only:

$$p_j(n) = \frac{2}{N} \sum_{i=1}^{N} p_i(n-1) p_i(n-2)$$  \hspace{1cm} (5)

The FD mesh is the most computationally efficient, but boundary conditions are currently only simply implemented [16], limiting their application to basic rectangular geometries. The scattering mesh is the more flexible at a boundary, particularly when a triangular topology is used [16], and as such it is possible to design a mesh that will provide a better fit to a more irregular geometric structure. However meshes based on the scattering equations are more computationally inefficient and so it would clearly be useful if the speed and efficiency of the FD mesh could be combined with the enhanced flexibility of the scattering mesh. It is possible to interface these two mesh types using the KW-pipe adapter [17]. The KW pipe is an all-pass network that exhibits a time delay in one direction only, which allows finite difference elements to be seamlessly connected to scattering elements. It is termed ‘KW’ as scattering based nodes are often termed ‘W’ nodes and FD nodes are often termed ‘K’ nodes.

By incorporating the KW-pipe into the scattering junction implementation a new “Black Box” scattering/KW-pipe junction is produced that looks identical to the finite difference version. These KW-Scattering junctions can be placed at the mesh boundaries where their geometrical flexibility is required, while the free-air nodes remain as regular FD nodes as shown in Figure 5. The free air nodes make up the overwhelming majority of the mesh, resulting in the significant speed and memory benefits of a purely FD based mesh together with the geometric and topological benefits of the scattering mesh.
4.2. Testing & Evaluating the Hybrid Mesh

To test these new mesh-types three simulations were conducted. A fan shaped room was selected due to its relatively simple acoustics and the fact that it is a non-rectangular geometry. Each simulation was conducted in 2D and used the same room with the following parameters:

- Dimensions – 10 m tall, 8 m at the bottom, 4 m at the top.
- 100% reflecting walls
- Mesh sample rate of 44.1 kHz.
- Sample rate sets waveguide size to 0.011 m
- Dimensions and waveguide size yield 260,330 nodes
- Sound source located towards the top center of the room
- Receiver located towards the bottom right of the room.

Each simulation evaluated the behaviour of a certain mesh type:

- Simulation 1 - standard Rectilinear Scattering Mesh,
- Simulation 2 - Rectilinear Hybrid Mesh
- Simulation 3 - Triangular Hybrid Mesh.

Visualizations of the hybrid models are shown in Figure 6. The ripples behind the wavefronts that have bounced off the side walls are due to discontinuities in the mesh as it tries to fit itself to the room geometry. This effect is less pronounced in the hybrid triangular mesh as its additional connections allow it to fit more accurately to the actual room geometry.

Figure 7 shows a spectral analysis (up to 500 Hz) of the output signals from the rectilinear scattering and rectilinear hybrid meshes and highlights the accurate low frequency response of both the scattering and hybrid DWM models. The axial mode at 16 Hz between the parallel top and bottom walls is clearly visible in both responses. It is interesting to note that the modes from the hybrid mesh are more pronounced in amplitude than those of the scattering mesh. One explanation for this is that it is easier to excite the hybrid mesh with a broadband signal as an unfortunate property of the scattering equations is that input signals get distorted due to an imbalance of incoming and outgoing energy at the excited node (See equation (3)).

An experiment was conducted to evaluate the improvements of the hybrid mesh in terms of processing time and memory requirements. The results were achieved on a PC with a 2.88 GHz P4 processor and 512 MB of RAM. A speed increase of 200% and memory usage decrease of 50% was observed when using the Hybrid rectilinear over the scattering rectilinear mesh. RoomWeaver makes obtaining such data simple as it reports the time taken to run a mesh and how much memory it is allocating to the mesh.
5. CONCLUSIONS

This paper helps to establish DWM models as an alternative and viable means to generating virtual RIRs for an enclosed acoustic space over more traditional methods. The RoomWeaver system has been presented and been demonstrated as a useful acoustical research tool, allowing DWM models to be applied to more complex room geometries than those previously investigated, resulting in the acquisition of valid virtual RIRs suitable for a variety of auralization applications. The introduction and implementation of the hybrid mesh based on the KW-pipe demonstrates that RoomWeaver has potential as a valuable and extensible research tool, making the development of new DWM techniques quick and easy. Future work will focus on expanding the current portfolio of mesh-types and implementing improved boundaries that are currently being developed in a parallel study.

6. ACKNOWLEDGEMENTS

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant Number GR/S01481/01 and in part by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), contract IG2002-2049.

7. REFERENCES


REAL TIME MODELING OF ACOUSTIC PROPAGATION IN COMPLEX ENVIRONMENTS

Fabio Antonacci, Marco Foco, Augusto Sarti, Stefano Tubaro
Dipartimento di Elettronica e Informazione - Politecnico di Milano
Piazza L. Da Vinci 32 - 20133 Milano, Italy
{antonacci|foco|sarti|tubaro}@elet.polimi.it

ABSTRACT
In order to achieve high-quality audio-realistic rendering in complex environments, we need to determine all the acoustic paths that go from sources to receivers, due to specular reflections as well as diffraction phenomena. In this paper we propose a novel method for computing and auralizing the reflected as well as the diffracted field in 2.5D environments. The method is based on a preliminary geometric analysis of the mutual visibility of the environment reflectors. This allows us to compute on the fly all possible acoustic paths, as the information on sources and receivers becomes available. The construction of a beam tree, in fact, is here performed through a look-up of visibility information and the determination of acoustic paths is based on a lookup on the computed beam tree. We also show how to model diffraction using the same beam tree structure used for modeling reflection and transmission.

In order to validate the method we conducted an acquisition campaign over a real environment and compared the results obtained with our real-time simulation system.

1. INTRODUCTION
Finding all the paths that link a source and a receiver in a complex environment using the laws of geometric reflection is a problem of crucial importance in a variety of applications ranging from realistic sound rendering to the modeling of indoor multipath fading in electromagnetic propagation. Several methods have been proposed for a fast determination of paths in both 3D and 2D space [1]. In particular, the beam tracing method [2] proved to be one of the most efficient solutions for determining such paths. One interesting aspect of this approach is that, given the location of the source, we can pre-compute the branching topology of all beams as paths passing through this point. For this reason, we can pre-compute the branching topology of all beams that propagate from that source through the construction of a data structure called beam tree. If we consider a receiver placed in any point in space, we can then quickly determine (through beam-tree lookup) which beams pass through that point and retrieve all the information that is needed to rapidly construct the paths between source and receiver. In conclusion, given a source location, this approach allows us to determine very quickly how the paths change as receiver moves. On the other hand, the beam tree depends on the reflectors' configuration as well as the source location, therefore every time we move the source, we need to recompute it. This operation can be rather costly, as it needs us to re-evaluate the visibility from the new source location.

A solution to this problem was recently proposed in [4]. The idea behind that method was to first compute the visibility information on the environment (reflectors) from an arbitrary point in space, which is equivalent to the visibility of a generic reflector from a point on a generic reflector. This information is computed and stored in a specific data structure in a preliminary analysis phase. As soon as we specify the source location, we can then iteratively construct the beam tree through lookup of the visibility information. As soon as we specify the receiver's location, we can iteratively determine the paths between source and receiver through beam tree lookup. A clever arrangement of the visibility information based on visibility diagrams (defined in the dual of the geometric space) enables a fast update of the beam-tree, which means that both sources and receivers can move in the environment during the auralization process.

In this paper we propose a method that extends this approach in such a way to model diffraction as well as geometric reflections. In particular, we extend the concept of visibility diagram in order to account for the diffracted field. We assume that the acoustic environment is 2.5D (vertical walls perpendicular to floor and ceiling). This allows us to simplify the analysis of 3D acoustic propagation and visualize it on a 2D floor plan. The approach that we propose, however, can be generalized to the case of a full-3D environment.

Diffraction is a fundamental mode of propagation in densely occluded environments. As a matter of fact, if source and receiver are not in direct visibility (and the transmission of sound through walls is negligible), then the first significant acoustic arrival will follow the shortest diffracted path.

In general, the diffracted field tends to enhance the spatial impression of the environment in which the receiver is immersed. In order to account for it, we consider a geometric approximation of the diffracted field based on the addition of a virtual source on each diffractive wedge. The diffractive paths can thus be thought of as "geometric Fermat paths" passing from a point on the diffractive edge. We will show that these edges, in our 2.5D approximations, are points on the floor plan, therefore diffractive paths are modeled as paths passing through this point. For this reason, we can pre-compute a beam tree for every potentially diffractive point in the map and use this propagation tree in runtime to evaluate diffracted propagation paths. Given these diffractive paths, in this paper we propose three different methods for simulating the diffractive effect with different computational loads.

We also show the results of some validation tests conducted in a real office environment. The tests consist of comparing some descriptors of the estimated impulsive response with the measured one.

This paper is organised as follows: in Section 2 we illustrate
the key concept of visibility diagram. In Section 3 we describe the extension of the visibility diagram in order to account for diffraction; Section 4 describes the methods used for measuring the impulse response of a room; finally Section 5 shows the results of the validation tests.

2. VISIBILITY DIAGRAMS

One key concept behind our work is the visibility diagram, which is a parameter-space representation of the visibility between reflectors. The visibility function of a reflector from an arbitrary viewpoint is here defined as a boolean function of the plenoptic space (the parametric space that describes a ray that departs from a generic point in space in an arbitrary direction). This function tells us whether or not the reflector will be visible from that viewpoint while looking in the considered direction. A two-dimensional plenoptic space is thus described by three parameters: two for the viewpoint location and one for the viewing angle. Notice, however, that all points on a visual ray share the same value of the visibility function. This tells us that a plenoptic parametrization is, in fact, redundant. This fact is well-known in applications of image-based rendering, where the plenoptic space is often replaced by a reduced-dimension space (see, for example, the Lumigraph [3]).

In our case this dimensionality reduction can be easily achieved by considering only the viewpoints that lie on a reference section of the geometric space (a reference line in 2D environments and a reference plane in the 3D case). This section, in principle, can be chosen arbitrarily, as long as it does not lie on the reflector whose visibility we are evaluating. It is important to remember that the visibility function will be iteratively looked up for tracing beams in the geometric space, therefore it is important to choose the reference section in such a way to simplify this process. We will see that this can be achieved by making the reference section coincide with another reflector. This corresponds to defining the visibility of a reflector from another reflector. A complete evaluation of the environment visibility is thus given by the whole collection of visibility functions of all reflectors from all reflectors.

With reference to Figs. 1 and 2, the visibility of reflector 2 from reflector 1 can be expressed as a boolean function of two parameters \( q \) and \( m = \tan \phi \). This function indicates whether a visual ray in position \( q \) on reflector 1 pointing in the direction \( \phi \) passes through any point of reflector 2. Notice that the visibility region on the plane \((m, q)\) corresponds to the dual of the reflector 2 with respect to a reference frame attached to reflector 1.

Let us consider the visibility diagram of reflector \( r_1 \) in Figure 1, which describes how the other reflectors are seen from viewpoints on \( r_2 \). The first step consists of choosing a reference frame attached to \( r_1 \), which is normalized in such a way that \( r_1 \) will correspond to the segment \((x_1, y_1)\), with \( x_1 = 0 \) and \(-1 \leq y_1 \leq 1\). This choice allows us to delimit the parameter space to the reference strip corresponding to \(-\infty \leq m \leq \infty \) and \(-1 \leq q \leq 1\) (dual space of the reference reflector).

The rays departing from the reference segment and hitting the other segments correspond in the \((m, q)\) space with visibility regions, for example the visibility region of \( s_2 \) is showed in Figure 2.

Considering the dual space interpretation, the visibility regions of the various reflectors with respect to the reference one can be computed in closed form [4]. Notice, however, that the visibility regions of the various reflectors overlap in regions corresponding to visual rays that intersect more than one reflector. Figuring out which reflector occludes which corresponds to sorting out which regions overlaps which. This ordering operation can be performed very quickly by back-tracing one ray for each connected overlapping area. Once overlaps are all sorted out, the visibility of environment of Figure 1 from reflector 1 is shown in Figure 3.

Visibility diagrams can all be computed in a pre-analysis phase, and this information can be used for a fast construction of a beam tree and a fast determination of all geometric paths between source and receiver. As soon as the source location is specified, the initial beam departing from it will split into a number of sub-beams, each inciding on a different reflector. The reflected beams will then branch out again as they reach other reflectors. In order to trace all such reflections and branchings, we can implement an iterative process that involves looking up visibility information.

At the generic step of the branching process, a beam is characterized by a (real or virtual) source and that portion of a reflector
that is “illuminated” by the beam (active portion of the reflector). The visibility from the active portion of the reflector can be readily obtained from the visibility of the whole reflector by narrowing the reference strip in the parameter space. Similarly, all the rays that depart from a source location correspond to a line on the visibility diagram (we recall that the parameter space of the visibility diagram is, in fact, the dual of the geometric space). The beam will thus be the intersection between the narrowed reference strip (illuminated portion of the reflector) and the dual line corresponding to the source (set of all visual rays that depart from the source’s origin). In conclusion, in order to determine which reflectors the beam will encounter in its path after being reflected by \( r_i \), we just need to determine the intersection between the dual of the source (a line) and the visibility regions of all the reflectors as seen from \( r_i \).

Once the beam tree is constructed, all paths corresponding to a given receiver location can be readily found through a simple beam tree lookup as described in [4] and [2].

3. ALGORITHM

As already mentioned above, in a 2.5D environment all diffractive edges are can be visualized as points on the floor plan as we assume that all diffractive edges are vertical. In order to render the diffracted field, we use the Uniform Theory of Diffraction (UTD), which models diffraction by placing a virtual source on each diffractive wedge. In practical cases, the UTD is not used when the diffractive edge is not long enough, as the diffractive field would in this case be negligible.

In order to render the diffracted field the very first phase of the proposed algorithm consists of the diffractive wedge selection. In facts not all the wedges in the environment can obscure the line of visibility path between source and receiver.

![Figure 4: Two examples of wedges, the hand-left side one will not be considered a diffractive wedge, while the right-hand side one will be considered a diffractive wedge.](image)

After selecting the diffractive wedge we trace two beam trees from a virtual source placed on the diffractive edges (see Figure 5): the first-level beams will be traced in the two regions marked as I and II.

At this point, the first level beams follow a reflection procedure that is completely similar to a regular beam tracing approach based on visibility diagram [4]. The depth of the diffractive beam tree must be determined in a different fashion compared to reflective beam trees for reasons of relevance and computational load. This operation can be done in a pre-computation phase, i.e. without knowing source and receiver positions. Once the diffracted beam trees are traced, we only need to test source and receiver positions in the two previously computed beam trees in order to obtain all paths between source and diffractive wedge and from the diffractive wedge to the receiver.

![Figure 5: An example of a diffractive wedge and the angular regions in which the first-level beams will span in the beam tracing execution.](image)

3.1. Rendering the diffracted field

In order to render the diffracted field we propose three approaches. The first rendering technique is based on the computation of a filter for every diffractive path using the Uniform Theory of Diffraction (UTD). The second approach sacrifices some of the accuracy in exchange of a reduction of computational complexity. In fact, in most of the applications of virtual acoustics we are more interested in producing a convincing sound rather than a physically-accurate one. A simplified method is based on a double interpolation. In particular, during a precalculation phase we record the complex value of the diffracted field at the origin of the beam (“penumbra” area) and in proximity of the wall in the shadow region. This operation is done by placing the source at eight angles in the angular range where the beam tree spans and for eight frequencies between 0 and 3 kHz. The source-wedge and receiver-wedge distances are assumed to be the half the distance between unoccluded walls. We use this information in the calculation phase as follows: we first check that source and receiver fall in the beam trees departing from the considered wedge; we calculate then the complex-value of the diffracted field of penumbra and shadow zone as linear interpolation of nearest values in the precalculated structure. The interpolation is angle-based. Let \( D_{f,h}(\beta_i) \) and \( D_{f,h}(\beta_{i+1}) \) be the magnitude of the diffracted penumbra field at two of eight angles that lie the closest to the source (whose angle with the closest side of the wedge is \( \beta_i \)) and at frequency \( f \). The first interpolation is

\[
D_{f,h}(\beta_i) \approx D_{f,h}(\beta_i) \frac{\beta_{i+1} - \beta_i}{\beta_{i+1} - \beta_i} + D_{f,h}(\beta_{i+1}) \frac{\beta_i - \beta_i}{\beta_{i+1} - \beta_i}.
\]
The same interpolation is used for phase and magnitude of $D_{f,h}(\beta_s)$.
A similar relationship can be written for phase and magnitude of the diffracted field at the shadow zone, $D_{f,s}(\beta_s)$:

$$D_{f,s}(\beta_s) \approx D_{f,s}(\beta_i) \frac{\beta_{i+1} - \beta_i}{\beta_{i+1} - \beta_i} + D_{f,s}(\beta_{i+1}) \frac{\beta_i - \beta_{i+1}}{\beta_i - \beta_{i+1}}.$$  

(2)

Given the position of the receiver, we use a linear angle-based interpolation between the penumbra and shadow values of the diffracted filter at the frequencies we are considering.

Let $\alpha_r$ be the angle between the wedge side that lies closest to the source and receiver. If $\alpha_1$ and $\alpha_2$ are the angles between the same segment and the beginning and the end of the lobe in which the receiver falls, then using the eqs. (1) and (2), the magnitude and the phase of the diffracted field at the receiver can be written as follows:

$$D_f(\alpha_r) \approx D_{f,s}(\beta_i) \frac{\alpha_r - \alpha_1}{\alpha_2 - \alpha_1} + D_{f,h}(\beta_i) \frac{\alpha_2 - \alpha_r}{\alpha_2 - \alpha_1}.$$  

An inverse Fourier transform is then computed in order to obtain the required filter.

![Figure 6: RMS error of the diffracted field for a $2\pi - \pi/4$ wedge as function of the receiver angle (left) and maximum RMS error as function of the wedge opening.](image)

In order to validate the double interpolation method and compare his computational complexity, we executed a comparison of the filter calculated using UTD and double interpolation techniques. In Figure 6 is shown the RMS error of the interpolated filter and the non interpolated filter, the RMS error is computed as follows:

$$E_{RMS}(\sigma_n) = \sqrt{\frac{\sum_n [h_d(n, \alpha_d) - h(n, \alpha_d)]^2}{\sum_n [h_d(n, \alpha_d) + h(n, \alpha_d)]^2}},$$

where $h_d(n, \alpha_d)$ is the $n$-th sample of the interpolated filter when the receiver is placed at angle $\alpha_d$, while $h(n, \alpha_d)$ is the $n$-th sample of the UTD filter at the same receiver position.

In the left-hand side of Figure 6 the wedge opening is $2\pi - \pi/4$ and, as expected, the maximum error takes place when the receiver is in the middle of the diffracted beam. In the implementation a wedge of such angular opening would not be auralized by UTD. In the right side of Figure 6 we plot the maximum RMS error moving the receiver in the middle of the diffracted beam for various angular openings of the wedge itself. The larger the angular opening, the larger the interpolation error.

The third proposed approach renders the diffracted field using a sample for every diffracted path, with a RMS value that equals the RMS value of the diffractive filter.

### 4. MEASURING THE ACOUSTICAL IMPULSE RESPONSE

The aim of this Section is to validate the algorithm illustrated above. We compared synthetic parameters obtained from the computed impulse responses, with those computed from the measured impulse responses. The impulse responses were measured using a Maximum Length Sequence (MLS) as source and recording the corresponding signal with a condenser microphone. Let $h(n)$ be the impulse response (of length $N$) that we need to measure, $\psi(n)$ the MLS signal (of length $M$), and $y(n)$ the recorded signal. We can estimate the impulse response as

$$\hat{h}(n) = \sum_{m=0}^{M} y(n + m)\psi(m).$$  

(3)

If the acquisitions are corrupted by an environmental noise (of RMS value $\sigma_n$) that is incorrelated with the MLS signal, then the Signal to Noise Ratio obtained with this method is given by

$$SNR = \frac{M}{N} \frac{\sigma_n^2}{\sigma_n^2}.$$  

(4)

The impulse response acquired with eq. (3) keeps trace of every transfer function encountered by the signal: D/A converter, speaker, environment, microphone and A/D converter. In order to reduce the impact of the transfer functions related to rendering and acquisition devices we implemented a deconvolution algorithm (details can be found in [5]). The algorithm uses the first echo as deconvolution element. Assuming that the air has no filtering effect on acoustic signal, an echo corresponding to the line of visibility path between source and receiver will keep trace only of the transfer functions introduced by acquisition and output devices. A different situation can be devised when the direct signal between source and receiver is absent: in fact the first echo will correspond to the diffracted signal and will be filtered by the typical low-pass effect of diffractive propagation. The last task will be a high-pass filtering in order to lower the effect of electric interferences.

### 5. VALIDATION OF THE ALGORITHM

The floor plan of the testing environment is shown in Figure 7. The first test we conducted aims at comparing the simulated acoustical response with the acquired one. In this work we neglected propagation modes such as diffusion and late reverberations. In order to reduce the effect of such propagation modes we considered only the most energetic samples of the measured impulse response. The microphone position used in this first test is marked with $R$, while speaker position is marked with $S_1$.

Our algorithm focuses on early reflections and low-order diffraction, therefore we will compare only the first part of the impulse responses.

As predicted, the first arrival of measured and simulated impulse responses (neglecting transmission by walls) comes from diffractive paths (marked in Figure 8 with the numerals 1 and 2). Although the first arrival comes from diffractive path, the most energetic samples of the impulse responses correspond to reflective...
paths (marked in Figure 8 with the numerals 3, 4 and 5). Finally, we notice a densification of the echoes in the last portion (marked with 6) of the impulse response: this is the effect of high-order reverberation and in beam tracing algorithm these echoes are modelled by high-level beams in the beam tree data structure.

In order to compare simulation and acquisition results on a larger scale we had to properly choose the model parameters using the measured impulse responses. Several parameters can be computed from the envelope of the impulse response and a number of solutions are available in the literature. We adopted a method based on the Schröder integration, whose details can be found in \[6\].

Figure 9 shows a measured acoustic impulse response and the corresponding Schröder envelope. A typical Schröder envelope exhibits a first portion where the curve decays linearly, while the following portion dims down very rapidly. The temporal location of the Schröder curve’s slope change can be taken as a measurement of the impulse length.

Once we compute the Schröder envelope we can define several parameters:

- Early Decay Time (EDT): the time where the Schröder envelope decays 10 dB from original level. In literature the EDT is thought as the time where the early reflections are terminated.
- T15: the distance between the time where the value of the Schröder envelope is \(-5\) dB and \(-20\) dB. Similarly we can define the T20 (decay from \(-5\) dB to \(-25\) dB).

Other parameters can be defined directly from the impulse response:

- Centre time: it is the first order momentum of the squared pressure impulse response, starting from the arrival of the direct wave.
- Energy of the impulse response.

In order to validate the algorithm, in this context we compared the Early Decay Time, and the centre time of measured, simulated with and without diffraction impulse responses.

In Figure 10 is shown the Early Decay Time of the measured (top) simulated with diffraction (centre) and without diffraction (bottom) impulse response in the test environment. From a visual comparison is clear that the simulated response using diffractive paths better approximates the measured response. In particular, we notice that the region near the door is not well represented in the absence of diffractive paths.

Figure 11 shows a comparison between the energy of the measured impulse response (top), that of the simulated response in presence of diffractive paths (centre) and in absence of diffractive paths (bottom). We notice that the presence of diffractive paths provides a better approximation of the measured response.
Figure 10: Comparison of energy of measured (top), simulated with diffraction (centre) and without diffraction (bottom) responses. We can observe that the contrast between illuminated region and obscure region is smaller in presence of diffractive paths, approximating better the real situation.

6. CONCLUSIONS

In this paper we proposed a novel method for high-quality rendering of audio in complex 2.5D virtual environments. The method generalizes to the case of diffracted sound propagation a beam tracing method driven by visibility functions. The method proved to be able to produce realistic rendering of both the reflected and the diffracted acoustic field with a modest computational complexity. We also proposed alternative solutions that sacrifice accuracy in exchange of computational efficiency, which produce very convincing results.

7. REFERENCES

DECORRELATION TECHNIQUES FOR THE RENDERING OF APPARENT SOUND SOURCE WIDTH IN 3D AUDIO DISPLAYS

Guillaume Potard, Ian Burnett

School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, Wollongong, Australia
gp03@uow.edu.au

ABSTRACT

The aim of this paper is to give an overview of the techniques and principles for rendering the apparent source extent of sound sources in 3D audio displays. We mainly focus on techniques that use decorrelation as a mean to decrease the Interaural Cross-Correlation Coefficient (IACC) which has a direct impact on the perceived source extent. We then present techniques where decorrelation is varied in time and frequency, allowing to create temporal and spectral variations in the spatial extent of sound sources.

2. PSYCHOACOUSTICS OF SOUND SOURCE EXTENT

Source extent has been studied in a large amount of literature (see [1], [10] and [11] for a review) under the names of apparent sound source width, tonal volume and others. It has been shown that the perceived source extent depends on the value of the inter-aural cross correlation coefficient (IACC) [12], sound loudness [13], pitch and signal duration [14]. The IACC coefficient is a widely used parameter in acoustics [1], [15] to determine the spaciousness and envelopment of concert halls. An IACC value close to zero will introduce a sense of diffuseness and of spatially large sound source; in contrast, an IACC absolute value close to 1 will produce a narrow sound image.

The IACC coefficient is defined as the maximum absolute value of the normalised interaural cross correlation function in turn defined as:

\[ IACC(\tau) = \frac{\int_{-\infty}^{+\infty} s_L(t - \tau) s_R(\tau) dt}{\sqrt{\int_{-\infty}^{+\infty} s_L^2(t) dt \int_{-\infty}^{+\infty} s_R^2(\tau) dt}} \]

where \( s_L(t) \) and \( s_R(\tau) \) are the ear canal signals at the left and right ears. The normalised cross-correlation function is bounded between -1 and 1.

3. METHODS FOR RENDERING APPARENT SOURCE WIDTH

This Section describes the techniques employed to create and control the spatial extent of sound sources.
3.1. Uncorrelated point sources

A commonly used technique to render the extent of sound sources in virtual auditory displays relies on the observation that a physically broad sound source can be decomposed into several, spatially distinct, point sound sources (Figure 1a). However, for this effect to take place, the signals emitted by the point sources must be statistically uncorrelated from one another. This is due to the fact that if correlation is high between the point sources, the binaural system perceives them as a single auditory event [1]. This results in a summation phenomenon and consequently only a narrow sound source is perceived at the center of gravity (Figure 1b). The position of the center of gravity depends on the positions and intensity gains of the point sources. This equates to amplitude panning performed between several speakers. In contrast, if the signals generated by the point sources are weakly correlated, the binaural system perceives the point sources as distinct auditory streams. This results in the perception of a spatially wide sound source (Figure 1c). In reality however, if the point sources are densely distributed, it might not be possible to distinguish every single point source as a different stream because the binaural system produces a final impression of a single, spatially large, sound source.

Figure 1: a) Decomposition of a broad sound source into point sources, b) High correlation between point sources creating a narrow sound image, c) Low correlation creating a wide sound image.

We now review different techniques to obtain decorrelated signals from a monaural sound signal.

3.1.1. Full band Decorrelation

The simplest way to obtain decorrelated signals is to introduce a small time delay between them. Although simple, this method can only produce a limited number of decorrelated signals as the upper permissible delay is restricted by the perception of an echo; this is typically around 40 ms. On speakers, this technique should however be avoided due to the possible comb-filtering effects caused by delays.

Decorrelation is most commonly achieved by filtering the input signal with all-pass filters having random, noise-like, phase responses [7]. Due to the ear instability to phase variations and the preservation of the signal amplitude spectrum (i.e., all-pass response), the obtained output signals are perceptually equal but statistically orthogonal.

Decorrelating all-pass filters can be implemented in FIR, IIR [7] or Feedback Delay Network (FDN) architectures. This technique can be used to create only a finite and relatively small number of uncorrelated signals, as a high correlation value will eventually occur between a pair of signals, due to the finite length of the filters. Thus the filter phase responses also need to be maximally orthogonal and need to be obtained by a best performance selection process. With this technique, we were able to obtain only five to six totally decorrelated signals. The filter length used was typically 100 poles and zero.

In order to obtain further signals, time-varying or dynamic decorrelation [7] is introduced.

3.1.2. Dynamic Decorrelation

Time-varying or dynamic decorrelation can be defined by the use of time-varying all-pass filters. The advantage of dynamic decorrelation over fixed decorrelation is that a higher number of uncorrelated signals can be obtained. This is due to the fact that time-varying decorrelation will introduce time-varying levels of decorrelation, depending on the orthogonality of the filter phase responses, but if these variations are fast enough and cannot be tracked by the ear, the perceived mean correlation value is low.

With all-pass filters, dynamic decorrelation is obtained by calculating a new random phase response for every new time frame. FIR or IIR lattice filter structures are best suited for this task due to their resistance to the instabilities that can occur during frequent filter coefficient updates.

Dynamic decorrelation also generates special audible effects not obtained with fixed decorrelation: it has been said [7] that dynamic decorrelation creates micro-variations simulating the time-varying fluctuations caused by moving air.

However we found that dynamic decorrelation can also have a distracting effect and even creates fatigue due to noticeable changing positions of objects in a recorded scene. This is likely due to phase differences between point sources that produce an Interaural Time Difference (ITD). Therefore, it is left to the discretion of the sonification designer whether fixed or dynamic decorrelation should be used.

3.1.3. Sub-band Decorrelation

So far we have only looked at decorrelation that is applied to the full signal spectrum. We now introduce a novel technique that allows us to alter decorrelation differently in each frequency band. Using this technique, a set of signals can be obtained where, for instance, their low-frequency components are uncorrelated while their high frequency components are left correlated. Using the point source method described above, this can lead to interesting effects where the spatial extent of a sound source varies in frequency. Therefore a sound source can be split into frequency bands having different spatial extents and positions. We call this effect the spatial Fourier decomposition effect. This effect can easily be noticed after some training.

The sub-band decorrelation technique is depicted in Figure 2. The input is first split into different frequency bands by a decomposition filterbank made of high order low-pass, band-pass and high-pass filters. Each sub-band signal is then decorrelated using any of the decorrelation technique described above. Crossfader modules are then used to control the amount of correlation
in each frequency band by a decorrelation factor $k$. This works by re-injecting some common sub-band signal into each decorrelated signal. For example, if total decorrelation is wanted, $k$ equal zero, then no common signal is injected. If $k$ equals one, the cross-fader outputs only the common signal and no decorrelated signal, therefore the correlation coefficient is one. It is also possible to set $k$ to any intermediate correlation value. A constant power cross-fading technique is preferable so that no change in signal level can be observed when $k$ is changed. Finally the different sub-bands of the respective decorrelated signals are added together to form the final set of partially decorrelated signals.

We have implemented such a decorrelator on the MAX/MSP platform [18] with low (0–1 kHz), medium (1–4 kHz) and high (4 kHz–20kHz) sub-bands. A higher number of sub-bands could be employed in order to obtain a finer grain on the correlation spectrum.

Finally, we note that it is also possible to combine dynamic decorrelation and sub-band decorrelation to obtain time and frequency varying levels of signal correlation.

\[ \text{Input signal} \rightarrow \text{Decomposition filterbank} \]
\[ \text{low} \rightarrow \text{Decorrelation filter} \]
\[ \text{medium} \rightarrow \text{Decorrelation filter} \]
\[ \text{high} \rightarrow \text{Decorrelation filter} \]
\[ \text{Cross-fade} \\
\]
\[ k_1 \rightarrow \text{Correlation coefficient} \]
\[ k_2 \rightarrow \text{Correlation coefficient} \]
\[ + \rightarrow \text{Partially Uncorrelated signals} \]

3.1.4. Time-varying Decorrelation

Time-varying decorrelation is obtained by periodically re-injecting the original signal into the decorrelated signals. When decorrelation is changed at up to 10 Hz with correlation coefficients between sound sources varying between 0 and 1, this creates a sound source with a constantly varying spatial extent. Above 10Hz this effect is destroyed by the lag of the binaural system to derive an IACC coefficient.

3.1.5. Other decorrelation techniques

Other decorrelation techniques besides delay and all-pass filtering exist (see [19] for an overview), these are often used in echo-cancellation systems. However, we discarded these techniques for virtual auditory display applications because they either degrade the signal (artificial introduction of noise and distortion), create large source localisation shifts and a disturbing phasing effect (Hilbert transform based techniques), they destroy the signal (KLT transform) or do not generate a high enough number of decorrelated signals.

3.2. O-format

A different approach for reproducing the spatial extent of sound sources relies on the encoding of the sound source spatial dimensions and directivity into spherical harmonics impulse responses, these techniques known as O-format and W-panning [20], [21] are offspring of Ambisonics theory [22]. We have not yet experimented with these techniques but it seems that low IACC at the listeners ears could also be achieved if the convolution of the monaural source signal with the spherical harmonics impulse responses creates enough decorrelation between parts of the broad sound source.

4. EVALUATION

The aim of this experiment was to study the shift between the intended source extent (ie the extent wished by the sound engineer or artist) and the actually perceived source extent by subjects.

The experiment was performed on the Configurable Hemispheric Environment for Spatialised sound (CHESS) [23] which uses fourth order Ambisonics spatialisation on a 16 speaker dome array. The space is not anechoic but has some acoustic proofing.

4.1. Stimuli

To create sound sources with various spatial extents, we employed six point sources and the technique described in Section 3. The point sources were spatialised using Ambisonics spatialisation and fed with independent white noise sequences having inter-correlation coefficients of 0.

We constructed 49 sound sources having various spatial extents, locations and geometry (Figure 3). Firstly, horizontal lines were made with a spatial extent of 60 and 180 degrees (sequences 1-4 and 11-14 respectively). We then constructed vertical lines with 40 and 90 degree extents (sequences 5-8). We also created small and big square sound sources having spatial extents of 60 degrees horizontally and 30 degrees vertically (sequence 10) and 180 degrees horizontally and 40 degrees vertically (sequence 9).

Finally we investigated the perceived spatial extent of a single speaker (sequence 15-16).

4.2. Procedure

Subjects were asked to draw the spatial extent of the noise sequences they were listening to on an answer sheet that represented a top-down view of the dome speaker array. On the answer sheet, the center therefore represents the zenith of the dome. Subjects were placed at the center of the dome and facing the zero degree orientation. Head rotations were allowed.

Although not perfect and subject to transcription errors, this elicitation method seemed appropriate for the transcription of the sound source extents perceived by subjects.

Fifteen subjects with no particular experience or knowledge in the audio field participated in the experiment.
4.3. Results and discussion

Areas where subjects had drawn were counted, and from this, density graphs generated. Due to limited space, we only show sixteen sequences out of the obtained 49 (Figure 3).

The graphs show that, in general, the mean perceived source extent follow the intended sound source extent (thick line).

For sources with an horizontal extent of 60 degrees (sequences 1-4), the perceived source extent was narrower than intended. This is probably due to the source density being too high; this creates a narrower source extent. This effect has been observed in previous experiments that we have carried out [9][4]. For sources with an horizontal extent of 180 degrees (11-13), the perceived source extent matched the intended extent, however subjects perceived some elevation in the sound which was not actually present. Sequence 14, which is an horizontal sound source placed at 40 degrees elevation was perceived as being higher, but not with a great precision however. Sources with a vertical extent (sequences 5-8) can be seen as having been discriminated from the horizontal sources. The sources with a square extent (9-10) where perceived roughly like the horizontal sources, but with slightly more vertical extent. In general, we can also notice that the ability to assess source extent is diminished for sounds coming from behind. Finally we can see in sequences 15 and 16 that even a single speaker is not perceived as a point source and has some spatial extent.

In general we can conclude that localisation of the the wide sound sources were correct and that the mean perceived spatial source extent matches coarsely the intended extent. It can be seen however that subjects can discriminate horizontal from vertical sources with different extent.

The mean perceived source extent matches coarsely the intended extent but there can be a lot of variation on the perception of extent (or the elicitation error) between subjects.

Also, using spatially small speakers would also help in the sharpening of source extent.

5. CONCLUSIONS

We gave an overview of the psychoacoustics concepts that control the perception of sound source extent. We then reviewed existing techniques to render the spatial extent of sound sources. We have then introduced a technique to alter the level of correlation of signals in different frequency bands. The resulting effect is that of a spatial Fourier decomposition where the different frequency bands of the signals are perceived in different positions with different spatial extents. This effect was clearly perceivable by the author but requires a substantial amount of training. We are planning to carry out experiments in order to assess further this effect on subjects.

In conclusion, we have seen that the apparent extent of sound sources can be controlled on several dimensions: spatial, temporal and spectral. We have, so far, only studied the spatial case.

Artificially produced spatial source extent based on temporal and spectral dependant decorrelation can be used to create special 3D audio effects that need to be further investigated and tested on subjects.

6. REFERENCES


[18] www.cycling74.com


DIGITAL EMULATION OF ANALOG COMPANDING ALGORITHMS FOR FM RADIO TRANSMISSION

Jürgen Peissig
Sennheiser Electronic, Wedemark, Germany
peissigj@sennheiser.com

Jan Remmer ter Haseborg
Technical University Hamburg-Harburg, Germany
jr@terhaseborg.de

Florian Keiler, Udo Zölzer
Helmut-Schmidt-University, Hamburg, Germany
udo.zoelzer@hsu-hamburg.de

ABSTRACT

Analog compander systems have been used to suppress the perception of noise in low dynamic range analog signal storage (tape recording) and signal transmission (FM radio). Commercial compander systems have been analyzed with respect to their signal processing requirements. The general structures of single- and multi-band compander systems have been implemented on a high performance audio PC workstation. Audio tests and measurements with the optimized compander algorithms and parameters show very good performance. Even for transmission channels with very low signal-to-noise ratio (SNR of only 40 dB) an optimized digital multi-band compander emulation removes the channel noise perceptively from the output signal of the transmission system.

1. INTRODUCTION

Analog tape recorders as well as analog FM radio transmission systems show an audio dynamic range of only 50 to 70 dB depending on tape material or RF reception. This reduced dynamic range results in a clearly audible noise floor which is very distracting. In order to reduce the noise perception audio compressors have been used prior to the recording or transmitting process. The compressor reduces the dynamic range of the input signal (e.g. 100 dB to 50 dB at a compression ratio of 2) and as a result all signal amplitudes are above the noise threshold of the tape or the FM transmission. During play-back or in the FM receiver an appropriate expander restores the original dynamic range of the signal by applying an attenuation depending on the signal amplitude. This results in an expansion (e.g. from 50 dB to the original 100 dB) of the dynamic range which on the other hand reduces the noise level by 50 dB. With a compression ratio of 2 for example the perceived signal-to-noise ratio (SNR) can be increased by a factor of 2 (e.g. from 50 dB to 100 dB).

This processing is called companding (compression and expansion). Companding is a time-variable processing and thus can cause audible alterations of the processed signal. By using complementary compressor and expander circuitry the original signal indeed can be restored at the output without any alteration if no noise is added in the compressed path. In real companded signals with noise added we find signal alterations and disturbance like noise pumping and breathing as well as distortion caused by dynamic mistracking of the level of the respective compressor and expander control path inputs. Most of these artifacts cannot be perceived because of being psychoacoustically masked by the preceding sounds (below masking threshold).

Several commercial analog compander systems were on the market. Each system was optimized for its main application. Optimization could be done by choosing appropriate frequency processing structures (broadband, sliding band or multi-band) as well as optimizing the time dependent parameters like attack and decay times of the envelope (level) estimator and the compression factors. Well known compander systems for professional applications like studio tape recorders and movie sound recording are the Dolby A and SR type and the TELEFUNKEN Telecom C4 compander as well as the dbx compander. Principles of those companders have been simplified for usage in consumer variants like Dolby B and C, the TELEFUNKEN HighCOM system and the Burwen Noise Eliminator for the use in cassette tape decks.

Studio tape recorders and tape decks are not produced any more and their usage is very limited nowadays to those locations where a considerable amount of tape material is available (e.g. the archives of radio stations). Nevertheless analog compander systems can still be found in wireless microphone systems. All large brands use analog broadband or multi-band compressors in their wireless transmitters and complementary expanders in the receivers (e.g. the Sennheiser HiDyn Plus compander or the dbx-like compander circuitry in the Shure wireless FM systems).

As digital signal processing circuitry becomes cheaper but more powerful, smaller, and less power consumptive, many analog circuits are emulated on digital systems. This holds also for compander systems. Most of the above mentioned compander systems are emulated digitally and are used where the original equipment is not available any more (e.g. in the radio-station to restore old analog tape material onto digital media.) The combination of analog FM transmission and digital companding algorithms is also used in the field of wireless microphones. A digital emulation of the analog compander principles yields several advantages:

a) more complex compander algorithms can be realized on a smaller circuit board size
b) compander parameters can be programmed via presets
c) more complex / adaptive signal processing can be used.

All signal processing techniques aim at the reduction of companding artifacts, at the optimal adaptation of the compander settings...
to the FM transmission characteristics and at increased reliability and convenience of the wireless FM link.

![Compander system diagram](image)

**Figure 1: Compander system.**

### 2. THEORY OF OPERATION

Compander systems are based on a compressor system before the transmission and expander system at the receiver (see Figure 1). Existing compander systems are built complementary and have time-variant transfer functions $H_C(z, t)$ and $H_E(z, t)$, where the expander transfer function $H_E(z, t)$ is the inverse of the compressor transfer function $H_C(z, t)$ according to

$$H_E(z, t) = \frac{1}{H_C(z, t)}. \quad (1)$$

The notation $H(z, t)$ denotes a time-variant Z-transfer function, because we are aiming at discrete-time realizations for the compressor and the expander. If the compander processing does not meet equation (1), the original signal cannot be restored and the compander processing becomes audible. For compander systems the compression factor $k$ is calculated from the signal input level $P_s(t)$ in dB and the signal output level $P_y(t)$ in dB according to

$$k = \frac{P_s(t)}{P_y(t)}. \quad (2)$$

For the noiseless transmission case the compander relation can be written as

$$y(t) = F_E(x(t)) \quad (3)$$

$$F_E(y(t)) = x(t), \quad (4)$$

where $F_E(x(t))$ and $F_C(x(t))$ denote the expander and compressor law (gain functions) that depend on the input signal $-1 \leq x(t) \leq 1$. For the compressor law the following relations hold

$$F_C(-x(t)) = -F_C(x(t))$$

$$F_C(\pm 1) = \pm 1$$

$$|F_C(x(t))| \geq |x(t)|.$$

Broadband (syllable) compensators as well as multi-band (formant) compensators use the signal envelope $Env_s(t)$ instead of the actual signal $x(t)$ to calculate the gain factors $F_C$. The compressor law of a syllable or formant compander working with the envelope or an appropriate estimate is then defined by

$$Env_s(t) = F_C(Env_s(t)) \quad (5)$$

and for the expander law respectively

$$Env_s(t) = F_E(Env_s(t)). \quad (6)$$

The compressed signal $y(t)$ can thus be calculated by the envelope $Env_s(t)$ of the uncompressed signal or the envelope $Env_s(t)$ of the compressed signal accordingly to

$$y(t) = \frac{F_C(Env_s(t))}{Env_s(t)} \cdot x(t) \quad (7)$$

$$y(t) = \frac{Env_s(t)}{F_E(Env_s(t))} \cdot x(t). \quad (8)$$

During reconstruction the expanded signal $x'(t)$ is calculated accordingly from the received compressed signal $y'(t)$ with the envelope of the input or the output of the expander given by

$$x'(t) = \frac{F_E(Env_s(t))}{Env_s(t)} \cdot y'(t) \quad (9)$$

$$x'(t) = \frac{Env_s(t)}{F_C(Env_s(t))} \cdot y'(t). \quad (10)$$

These different approaches result in different possible processing structures for syllable and formant companders which are shown in Figure 2. The control signals $s_C(t)$ and $s_E(t)$ are the so-called gain factors derived from the signal envelopes by nonlinear mappings based on the compression factor $k$.

Those structures that derive the control signal from the compressed signal are favoured in real applications because the envelope detector operates on a signal with reduced dynamic range, especially when the level detector contains true RMS (root mean square) processing. Working on signals with reduced dynamic range imposes less demands on the dynamic range of the analog circuitry or the digital word length $\frac{2}{4}$. According to $\frac{2}{4}$ both structures yield equal results. The input/output characteristics of a compressor or expander are shown in Figure 3. The upper curve pair...
describes the compressor law and the lower two curves describe the expander law. The compressor input and the expander output are drawn in dB on the abscissa. The compressor output and the expander input are drawn in dB on the ordinate.

2.1. Envelope and level estimators

In real compander systems the envelope is approximated by functions like

1. the RMS of the signal amplitude with integration time constant
2. a peak amplitude detection with attack and decay time constants
3. a modified peak detection with multiple decay time constants

All approximations comprise a temporal behaviour caused by the integration (by low-pass filtering) of the squared or absolute value input signal. The deviations also differ according to the frequency content of the input signal. Figure 4 gives an impression of the envelope estimates of the various kinds for two sine burst signals with different frequencies.

2.2. Problems with existing companders

There are several unwanted side effects which are more dominant in single-band companders compared to multi-band companders. The time variable processing leads to distortions in the compressed signal that sometimes cannot be recovered by the expander. This behavior is called dynamic mistracking and leads to more perceived roughness of the low frequency content. This is caused by modulation with the envelope of the high frequency content and vice versa. The temporal integration of the control factor on the other hand leads to pumping and breathing sounds perceived after a sudden mute of a loud signal. The integration/decay constant has to be large to not follow a low frequency amplitude. On the other hand it has to be as fast as the decay of the post masking curve so that a sudden silence causes the expander to attenuate the output signal resulting in the system noise floor always being below threshold.

3. EXAMPLES OF ANALOG COMPANDERS

Only the most important existing compander systems from Dolby, TELEFUNKEN and dbx are explained in detail here. Less known companders are the Toshiba Address-System and the JVC SuperANRS as well as the TELEFUNKEN derivate HighCOM II.

3.1. Dolby A and Dolby SR

The block diagram of the Dolby A compander is shown in Figure 5. It is a four-band system with cut-off frequencies at 80 Hz and 3000 Hz and two high-pass filters working in parallel with cut-off frequencies of 3 kHz and 9 kHz. The compression curves are shown in Figure 6.

This compressor was used for tape recording and could not be used for FM radio because of its non-linear characteristics that resulted in frequency response and signal distortion if the expander received a compressed signal level differing from the compressor output level. Compressor and expander input levels had to be adjusted carefully with an adjustment signal before operation. Usually compressor and expander worked with identical circuitry that could be switched to operate on input (compressor) or on output (expander).

A more sophisticated compander is the Dolby SR (Spectral Recording) system that was used for high end studio tape recording. A block diagram is shown in Figure 7. It is working with fixed and variable filter cut-off frequencies at three different activating
levels (High: -30 dB, Medium: -48 dB and Low: -62 dB). Compression takes place only below these input levels. In the High and Medium setting the frequency band is divided at 800 Hz by a sliding band filter. In the Low setting only frequencies above 800 Hz are affected. The time constants of the analysis modules (Modulation Control Circuits A,B,C) are adapted to the psychoacoustics of hearing. The system is capable to modify the signal spectrum in five bands depending on the frequency content of the input signal.

3.2. TELEFUNKEN Telcom C4

The high end studio compander system from TELEFUNKEN was called Telcom C4. A block diagram is shown in Figure 8. This compander uses a compression ratio of 2/3 working in four different bands separated at 215 Hz, 1450 Hz and 4800 Hz. The compression characteristic is shown in Figure 9.

It shows different compression onsets and thresholds for each frequency band and more compression at higher frequency bands. In the low frequency bands higher levels are expected and thus the compression onset is set to 0 dB whereas at the 10 kHz frequency band a much lower threshold yields a higher gain of up to 25 dB at low levels. The envelope detection method of the Telcom C4 compander uses three time constants: attack, slow release and fast release (1/10\textsuperscript{th} of the slow release time interval. This behavior is illustrated in Figure 4 where the temporal behavior of different envelope estimates according to attack and release time constants is shown.

3.3. Dolby B and C

Dolby B and C (see Figure 10) are the single sliding band consumer derivatives of the professional systems. The B type acts on frequencies only above 500 Hz. Depending on the signal energy this cut-off frequency moves higher. A maximum noise reduction of 20 dB between 2 kHz and 10 kHz can be achieved with this version. Dolby C is a two step version derived from Dolby SR. It also has a constant spectral skewing and improves the high frequency dynamic recording level (reduction of overload inclination). The maximum noise reduction here is approximately 15 dB. The compression characteristics are shown in Figure 11.

![Figure 6: Compression characteristic of Dolby A. Curve 2: characteristic of the control signal added to the direct path. This results in the compression (3) and expansion (4) characteristics.](image)

![Figure 7: Block diagram of the Dolby SR compander system.](image)

![Figure 8: Block diagram of the TELEFUNKEN Telcom C4 compander system.](image)

![Figure 9: Compression characteristic of the TELEFUNKEN Telcom C4 compander system.](image)

![Figure 10: Block diagram of the Dolby B (top) and Dolby C (bottom) compander systems.](image)
3.4. TELEFUNKEN HighCOM

Figure 12 shows the block-diagram of the semi-professional consumer version of the Telekom C4 compander. It is a single-band feed-back version with pre- and de-emphasis from 1.2 kHz to 8.6 kHz, a compression factor of 2 and different compression thresholds (see Figure 13).

Figure 13: Compression characteristic of HighCOM [3].

3.5. dbx and Burwen Noise Eliminator

The dbx compander (Figure 14) was used for professional and consumer tape recording. It is a single band multiplicative feedback compander with a compression ratio of 2 and an envelope extraction according to the true-RMS principle. It also incorporates pre- and de-emphasis from 370 Hz to 1.59 kHz to avoid high frequency overload, the level detector loop contains a low-pass filter with 10 kHz corner frequency and an additional pre-emphasis from 440 Hz to 4.8 kHz. A similar system is the Burwen Noise Eliminator which uses the same characteristics but with a compres-

Figure 14: Block diagram of the dbx compander system [3].

4. DIGITAL EMULATION OF ANALOG COMPANDERS

Most of the above mentioned analog systems have been implemented digitally in C-code. This comprises single-band and multi-band structures with peak, modified-peak, RMS and Hilbert level detectors. Adjustable compression factors and adjustable time constants as well as adjustable compression thresholds and compression onsets in each frequency band have been implemented. The block-diagram of the single-band compander emulation is shown in Figure 16.

With this structure several existing types (e.g. Sennheiser HiDynPlus, dbx and the HighCOM) can be re-evaluated. In the single-band version pre- and de-emphasis filters account for the
Figure 17: Block diagram of the digitally implemented feedback multi-band additive compander emulation. HP/LP/BP: high/low/band-pass, ED: envelope detection, AWGN: additive white Gaussian noise.

Figure 18: GUI of the compander emulation program.

perceptive properties of the ear. All filters have been realized as shelving filters according to [4]. Pre-emphasis in the control path reduces the high frequency dynamic range. To overcome this problem an additional LP filter can be added to the control path. In order to get even better performance for professional usage we also implemented a multi-band system. Dolby A or SR structures have not been implemented because of their well known disadvantages in FM applications.

The multi-band system is shown in Figure 17 and is based on shelving and peak filter designs introduced in [4]. By using this multi-band structure, equation (11) holds and the transfer function of the compressor \( H_C(z, t) \) and expander \( H_E(z, t) \) can be expressed as

\[
H_C(z, t) = 1 + H_{FB}(z, t) \quad (11)
\]

\[
H_E(z, t) = \frac{1}{1 + H_{FB}(z, t)}. \quad (12)
\]

\( H_{FB}(z, t) \) denotes the time-varying transfer function of the filter bank. It depends on the signal spectrum of the compressed signal. Compression factors, time constants and envelope estimators define the time dependency of the transfer function. These parameters control the envelope detection and the compression factor calculation in block ED in Figure 17. The control path is separated from the signal path. Both paths work with the same filter bank transfer function. The recursive structure in the expander has delay free loops and is calculated by the technique introduced in [5]. A limiter and a AWGN channel is used between compressor and expander to simulate an analog FM transmission link.

All compander algorithms are written in float or 24 bit integer resolution and are controlled by a GUI offering different IO paths (.wav-file and sound card) and analyzer options. The C++ program can be linked via soundcard to a real-time FM modulator and demodulator. The aim of this program is to have a basis to do high quality audio test sessions to investigate acoustically optimal compander structures and compander settings for wireless FM microphone systems. Figure 18 shows the GUI of the compander evaluation program [6]. It allows to change all compander settings during playback of the test materials. Audio test material can be processed in real-time. Measurements of total harmonic distortion, intermodulation distortion, on/off behavior and dynamic tracking can be performed for different compander structures and settings. All results are comparable to the analog circuitry. The multi-band compander outperforms the single-band compander. With a high compression ratio in the upper frequency bands and different compression onsets and thresholds a channel noise with a SNR of 40 dB can perceptively be suppressed by the multi-band compander.

5. CONCLUSIONS

The digital implementation of a multi-band compander is suitable to work as a high-end compander system for professional wireless FM microphone systems. The real-time PC implementation of the novel multi-band compander structure allows the possibility to optimize the compander presets by artists and sound engineers. The possibility to integrate an analog FM link into the signal chain offers the evaluation of the compander performance, increases the reliability of the compander settings also under real operating conditions, and also reduces the development time for the audio signal processing significantly.

6. REFERENCES

HIGH QUALITY VOICE TRANSFORMATIONS BASED ON MODELING RADIATED VOICE PULSES IN FREQUENCY DOMAIN

Jordi Bonada

MTG, Institut Universitari de l’Audiovisual
Universitat Pompeu Fabra, Barcelona, Spain
jordi.bonada@iua.upf.es

ABSTRACT

This paper introduces a method to transform voice based on modeling the radiated voice pulses in frequency domain. This approach tries to combine the strengths of classical time and frequency domain techniques into a single framework, providing both an independent control of each voice pulse and flexible timbre and phase modification capabilities.

1. INTRODUCTION

Our goal is to achieve high quality natural transformations of singing and speech voiced utterances. The type of targeted transformations is quite wide and covers both high (pitch-shifting, time-scaling, timbre modification) and low level ones (vocal disorders).

During voiced utterances the airflow going through the vocal folds is chopped into a set of glottal pulses which are then filtered by the vocal tract, tongue and lips. The resulting waveform can be understood as the result of overlapping all the filtered glottal pulses, therefore showing an impulsiveness feature. In terms of frequency domain representation, this impulsiveness comes out from the phase synchronization between harmonics existing at the pulse onsets.

Classical time domain techniques [1][2] can transform the voice and preserve this phase alignment with a relatively low computational cost. The basic procedure consist on cutting short fragments from the waveform (typically one period) and overlap and add them at synthesis. However most difficulties appear when we want to achieve complex timbre modifications. Maybe the main drawback of these techniques is that they can’t isolate one single impulse response but they cut the result of several overlapped ones, thus degrading the quality.

FOF (Formant Wave Function Synthesis, [3]) is another time-domain technique which features a flexible and fast additive synthesis method. It models the human voice as the sound from an impulse generator, equivalent to the vocal chords, passing through a set of band-pass filters which represent the characteristics of the vocal tract (each filter corresponds to a vocal formant). This technique can successfully control independently each pulse. However, it roughly represents the analysis amplitude spectrum and doesn’t reproduce the voice phase alignment at glottal pulses.

On the other hand, classical frequency domain techniques [4][5][6] can accomplish a wider range of transformations with a high computational cost, including complex timbre modifications and exotic effects (inharmonizer, morphing). However, they require continuing the phase of the harmonics and preserving their phase relation at pulse onsets (which is rather tricky). Besides, if the signal is modeled with pure sinusoids the transformations sound often artificial and synthetic, even adding some residual. Advanced phase vocoder techniques [7] have shown encouraging results preserving the local spectrum (amplitude and phase) behavior around the harmonics. This local spectrum describes somehow the context of the partial (amplitude, frequency evolution), but it is not yet clear how to properly modify this context (for example adding a vibrato, so changing the frequency evolution). In addition complex peak picking strategies must be considered to avoid problems with noisy or masked peaks.

In this paper we present an algorithm which combines the waveform preservation ability of time-domain techniques with the flexible and wide transformations of frequency-domain techniques, while at the same time avoiding the complexity and the contextual problems of them. This algorithm consists on modeling the radiated voice pulses in frequency domain, and is able to transform independently each one of them.

2. MODELING RADIATED VOICE PULSES IN FREQUENCY DOMAIN

The STFT of a windowed frame \( x(n) \) can be expressed as

\[
X(e^{j\Omega}) = \sum_{n=0}^{N-1} x(n)e^{-j\Omega n}
\]

where \( s(n) \) is the input signal and \( w(n) \) is the window function.

Let’s assume a rectangular window of \( N \) samples and a finite input signal shorter than \( N \) and covered by the window. If the input signal is delayed by \( \Delta n \) samples, then its STFT will be

\[
S_{\text{delayed,}\Delta n}(e^{j\Omega}) = \sum_{n=0}^{N-1} s(n-\Delta n)e^{-j\Omega n} = \\
= \sum_{n=\Delta n}^{N-1} s(n-\Delta n)e^{-j\Omega n} \approx X(e^{j\Omega})e^{-j\Omega \Delta n}
\]

where the last approximation would become an identity if the delayed signal was also fully covered by the window.

Let’s now consider \( y(n) \) as the sum of \( R \) identical signals \( s(n) \) delayed by \( \Delta n \) samples, where some overlap is possible. We can calculate its STFT as follows:

\[
y(n) = s(n) + s(n-\Delta n) + s(n-2\Delta n) + \ldots + s(n-(R-1)\Delta n)
\]

therefore
The effect of the term $\text{sinc}_a(\Omega \Delta n)$ is somehow sampling the spectrum of $X(e^{i\omega})$, since this term can be seen as a train of equidistant pulses located each $2\pi/\Delta n$ radians (see Figure 1). All the pulses have a constant value of $R$, as can be seen below:

$$\text{sinc}_a(\Omega \Delta n) \approx e^{-j\pi a(\Omega \Delta n)} \sin(\pi R \Delta n)$$

$$\lim_{\Delta n \to 0} \text{sinc}_a(\Omega \Delta n) = R$$

Consequently, if we assume $|X(e^{i\omega})|$ and $\Delta X(e^{i\omega})$ vary slowly along frequency, we can estimate $X(e^{i\omega})$ from $Y(e^{i\omega})$ by interpolating the values at frequencies $\Omega$. The interpolation algorithm could be just a linear interpolation but a spline method is preferred. The resolution depends on $\Delta n$, where a bigger value means better resolution. The reconstructed signal $x'(n)$ can be computed by means of the inverse STFT of the estimated $X'(e^{i\omega})$

$$x'(n) = \frac{1}{N} \sum_{n=0}^{N-1} X'(e^{i\omega}) e^{i\omega n}$$

and finally we could approximate the input signal as

$$s'(n) = x'(n)$$

In the case of voiced utterances in a speech or singing voice recording, $s(n)$ corresponds to a radiated glottal pulse filtered by the vocal tract. However, what we usually deal with is the result of several consecutive voice pulses, as we see in Figure 2 which are overlapped along time and not fully covered by the window. In this case $\Delta t$ is straightforwardly related to the voice pitch by $\Delta t = f_s/\text{pitch}$, where $f_s$ is the sampling rate. Besides, the radiated voice pulses are not identical because the voice production system is changing its characteristics continuously along time. However, if the window is short enough so that the signal is stationary, then the above method would allow estimating a single radiated voice pulse.

$$Y(e^{j\varnothing}) = \sum_{n=0}^{N-1} y(n)e^{-j\varnothing n} = X(e^{j\varnothing})[1 + e^{-j\Delta \varnothing} + e^{-j2\Delta \varnothing} + \ldots + e^{-j(N-1)\Delta \varnothing}] =$$

$$= X(e^{j\varnothing}) \sum_{n=0}^{N-1} e^{-j(n-1)\Delta \varnothing} = X(e^{j\varnothing}) \frac{1 - e^{-j(N-1)\Delta \varnothing}}{1 - e^{-j\Delta \varnothing}} = X(e^{j\varnothing}) e^{-j(N/2)\Delta \varnothing}$$

$$\approx X(e^{j\varnothing}) \text{sinc}_a(\Omega \Delta n)$$

Figure 2: Spectrum estimation of a single radiated voice pulse

$$R = \frac{N \cdot \text{pitch}}{f_s}$$

3. VOICE ANALYSIS

The analysis is done as a constant frame-rate process where we get an estimation of the STFT of a centered radiated pulse for each step (see Figure 3). In our experiments we have used about 172 frames per second. The analysis starts windowing the input voice signal and computing its STFT. Then the spectral peaks are detected out of the amplitude spectrum using a parabolic interpolation and inputted into the pitch detection algorithm, actually an extension of the TWM algorithm [3]. The pitch is then used by the peak selection module to choose the harmonic peaks considering as a guide both the perfect harmonic distribution and the amplitude and frequency of spectral peaks. Next, the phase of the harmonic peaks is modified by the maximally flat phase alignment (MFPA, see Section 3.1) algorithm which centers the voice pulse in the window. The final step is to interpolate the harmonic peaks and estimate the spectrum of the radiated voice pulse $X'(e^{j\omega})$. The amplitude spectrum is interpolated using a 3rd order spline method and afterwards scaled by $1/R$, where $R$ is the number of pulses contained in the analysis window. $R$ can be computed from the pitch by

$$R = \frac{N \cdot \text{pitch}}{f_s}$$

Figure 1: $\text{sinc}_a(\Omega \Delta n)$, $R=32$, $\Delta=8$, $N=256$. 

Figure 2: Spectrum estimation of a single radiated voice pulse

Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx’04), Naples, Italy, October 5-8, 2004
Related to the window size $N$, we need a good frequency resolution in order to precisely detect the spectral peaks, therefore a long window. However, in such case, the transitions, attacks and releases would become considerably smoothed by the windowing since radiated pulses with different characteristics (energy, timbre) would be processed together and pitch would not be approximately constant along the window. The compromise adopted has been to adapt the size of the window to cover around three periods, which assures enough resolution to discern the peaks while at the same time increases the temporal resolution and improves the handling of non stationary parts. Besides, zero padding is applied in order to improve the spectrum resolution.

3.1. Maximally flat phase alignment (MFPA)

As will be seen later in the synthesis section, there is a need to control the position in time of each voice pulse. Therefore we need a way to estimate the position of each pulse or at least of one of them. One way to deal with this issue is to shift in time the input signal so that one of the voice pulses becomes centered in the window.

When the voice pulse is almost centered it happens that harmonics are usually synchronized in a way that the phase spectrum is nearly flat with phase shifts under each resonance (i.e. formant) area. This can be seen in Figure 2. Whenever we move the voice pulse, the corresponding time shift adds a phase shift which varies linearly along frequency ($e^{j\Delta \omega t}$). Thus, one way to estimate the pulse location is to estimate the slope of this linear phase shift. However, phase wrapping complicates the problem because all phase values are contained in the range $[-\pi, \pi]$.

We have come up with an easier method to approximate the center location. With this procedure we pretend to find the time shift $\Delta t$ which minimizes the phase differences between harmonics, therefore obtaining a maximally flat phase alignment (MFPA), as explained in the following:

a) Define several fundamental phase candidates $\phi'_{i0}$ in the interval $[-\pi, \pi]$

b) For each candidate, apply the corresponding time shift $\Delta t_i$ to each harmonic peak. The phase of each harmonic $\phi_i$ will be rotated by $\phi_i = 2\pi f_i \Delta t_i$, where $f_i$ is the frequency of the harmonic.

c) Compute the sum of rotated phase differences as $\phi_{eff} = \sum |\text{princarg}(\phi_{i0} - \phi_i)|$, where $\text{princarg}$ is a function which puts an arbitrary radian phase value into the interval $[-\pi, \pi]$ by adding an integer number of periods ($2\pi$).

After several candidates are estimated we obtain an error function which is similar to a sinusoid and whose minimum sets the desired fundamental phase $\phi_{min}$ which approximately centers the voice pulse (see Figure 4).

Figure 3: Block diagram of the analysis process.

Figure 4: Maximally flat phase alignment

Actually only low frequency harmonics (up to around 3 KHz) are considered since usually most energy is located at low frequencies and also because higher frequency harmonics are often unstable or noisy. Besides a continuation algorithm for $\phi_{min}$ is used for better handling noisy parts.

3.2. Phase Unwrapping and interpolation

Whenever we want to interpolate the peak’s phases we have to consider the wrapping issue and choose the shortest way between two consecutives phases. The unwrapping procedure sets the initial phase to be $\phi_0 = \phi_{min}$. The next unwrapped phases $\phi'_i$ can be obtained iteratively by

$$
\phi'_i = \phi_{min} + \text{princarg}(\phi_i - \phi_{min})
$$

The estimated phase spectrum $\hat{X}(e^{j\omega})$ is then computed interpolating the unwrapped phases $\phi'$. It may happen for successive harmonics that during phase unwrapping different number of periods are added for consecutive frames. In such case we will get discontinuities between the harmonics’ frequencies after interpolation (see Figure 5).

In order to avoid this artifact, for each harmonic a phase correction ($\Delta \phi'_{\text{corr}}$) is added which compensates the period differences and slowly decays to zero, getting 0.1 radians closer each frame. This way, if in frame $n-1$ two consecutive harmonics ($i-1$ and $i$) have a phase difference of $-0.9\pi$ and in the next frame $n$ the difference becomes $-1.1\pi$, the phase correction would be $\Delta \phi'_{\text{corr}} = -2\pi$. In the next frame $n+1$, the phase correction would decay a little, thus $\Delta \phi'_{\text{corr}} = -2\pi + 0.1$, and if the phase difference becomes again $-0.9\pi$ then $2\pi$ would be added to the phase correction, so it would be $\Delta \phi'_{\text{corr}} = 0.1$. The final sequence would be process together and pitch would not be approximately constant along the window. The compromise adopted has been to adapt the size of the window to cover around three periods, which assures enough resolution to discern the peaks while at the same time increases the temporal resolution and improves the handling of non stationary parts. Besides, zero padding is applied in order to improve the spectrum resolution.

3.1. Maximally flat phase alignment (MFPA)

As will be seen later in the synthesis section, there is a need to control the position in time of each voice pulse. Therefore we need a way to estimate the position of each pulse or at least of one of them. One way to deal with this issue is to shift in time the input signal so that one of the voice pulses becomes centered in the window.

When the voice pulse is almost centered it happens that harmonics are usually synchronized in a way that the phase spectrum is nearly flat with phase shifts under each resonance (i.e. formant) area. This can be seen in Figure 2. Whenever we move the voice pulse, the corresponding time shift adds a phase shift which varies linearly along frequency ($e^{j\Delta \omega t}$). Thus, one way to estimate the pulse location is to estimate the slope of this linear phase shift. However, phase wrapping complicates the problem because all phase values are contained in the range $[-\pi, \pi]$.

We have come up with an easier method to approximate the center location. With this procedure we pretend to find the time shift $\Delta t$ which minimizes the phase differences between harmonics, therefore obtaining a maximally flat phase alignment (MFPA), as explained in the following:

a) Define several fundamental phase candidates $\phi'_{i0}$ in the interval $[-\pi, \pi]$

b) For each candidate, apply the corresponding time shift $\Delta t_i$ to each harmonic peak. The phase of each harmonic $\phi_i$ will be rotated by $\phi_i = 2\pi f_i \Delta t_i$, where $f_i$ is the frequency of the harmonic.

c) Compute the sum of rotated phase differences as $\phi_{eff} = \sum |\text{princarg}(\phi'_{i0} - \phi_i)|$, where $\text{princarg}$ is a function which puts an arbitrary radian phase value into the interval $[-\pi, \pi]$ by adding an integer number of periods ($2\pi$).

After several candidates are estimated we obtain an error function which is similar to a sinusoid and whose minimum sets the desired fundamental phase $\phi_{min}$ which approximately centers the voice pulse (see Figure 4).

Figure 3: Block diagram of the analysis process.

Figure 4: Maximally flat phase alignment

Actually only low frequency harmonics (up to around 3 KHz) are considered since usually most energy is located at low frequencies and also because higher frequency harmonics are often unstable or noisy. Besides a continuation algorithm for $\phi_{min}$ is used for better handling noisy parts.

3.2. Phase Unwrapping and interpolation

Whenever we want to interpolate the peak’s phases we have to consider the wrapping issue and choose the shortest way between two consecutives phases. The unwrapping procedure sets the initial phase to be $\phi_0 = \phi_{min}$. The next unwrapped phases $\phi'_i$ can be obtained iteratively by

$$
\phi'_i = \phi_{min} + \text{princarg}(\phi_i - \phi_{min})
$$

The estimated phase spectrum $\hat{X}(e^{j\omega})$ is then computed interpolating the unwrapped phases $\phi'$. It may happen for successive harmonics that during phase unwrapping different number of periods are added for consecutive frames. In such case we will get discontinuities between the harmonics’ frequencies after interpolation (see Figure 5).

In order to avoid this artifact, for each harmonic a phase correction ($\Delta \phi'_{\text{corr}}$) is added which compensates the period differences and slowly decays to zero, getting 0.1 radians closer each frame. This way, if in frame $n-1$ two consecutive harmonics ($i-1$ and $i$) have a phase difference of $-0.9\pi$ and in the next frame $n$ the difference becomes $-1.1\pi$, the phase correction would be $\Delta \phi'_{\text{corr}} = -2\pi$. In the next frame $n+1$, the phase correction would decay a little, thus $\Delta \phi'_{\text{corr}} = -2\pi + 0.1$, and if the phase difference becomes again $-0.9\pi$ then $2\pi$ would be added to the phase correction, so it would be $\Delta \phi'_{\text{corr}} = 0.1$. The final sequence
of phases for the harmonic \(i\)-th would be \(\phi_{i-1} - 0.9\pi\), \(\phi_{i} - 1.1\pi\), \(\phi_{i+1} - 0.8\pi\), with no unwrapping artifact.

4. SYNTHESIS

The analysis outputs the estimated pitch and spectrum of radiated voice pulses centered in the window. In order to synthesize a voiced utterance, the first thing we have to do is to generate a sequence of pulse locations. Such sequence can be derived from the pitch envelope by separating consecutive pulses by one period \(T = 1/pitch\). If we want to better match the input signal pulse sequence we could also use the unwrapped fundamental phase envelope and the outputs of the MFP4 to determine the position of each pulse in the original sequence. We don’t need to care about each pulse’s amplitude since the original amplitude is preserved through the estimated spectral amplitude itself.

Once we have filled the sequence we have to map each pulse to one analysis frame and get the radiated voice pulse spectrum. We can choose the nearest frame or interpolate the spectrum of the two closest frames to get smoother timbre transitions, which can be useful specially when applying a time-stretch transformation. At this point we can synthesize each pulse independently by means of an IFFT and then position it in the desired location. This would require an interpolation of the samples in time domain because pulse time onsets are not quantized to the sampling rate.

However the positioning can also be done directly in frequency domain by adding a linear phase slope to the phase spectrum proportional to the required time-shift, which can be written as

\[
\Delta\phi_r(\Omega) = (t - t_{\text{frame}})\Omega
\]

where \(\Delta\phi_r(\Omega)\) is the phase increment for the \(r\)-th pulse and \(t_{\text{frame}}\) is the center time of the current frame. Here we have to be careful about the hop and window size values because there is a limit of how much the signal can be shifted by this method since the window is finite and short. If the window size is \(N\) we can not time-shift the pulse more than \(N/2\) samples, otherwise the pulse will appear in the opposite part of the window as an aliasing effect. This time-shift operation can be computed efficiently on a complex spectrum since the phase shift increases by a constant amount for consecutive bins, so we can compute only two cosinus for the whole spectrum and then two complex multiplications per bin.

Depending on the synthesis configuration it can be good to combine into a single IFFT several pulses so to speed up the process. This way we would generate a complex spectrum for each pulse and add it to the IFFT buffer. The computational cost for each synthesis frame would include a polar to complex conversion for each analysis frame used plus a time shift of the whole spectrum (see equation (9)) for each synthesized pulse. The computational cost would then be given by

\[
C = n_p \cdot 2 \cdot c_m + 2 \cdot n_s (\text{cmul} \cdot N + \text{cos}) + \text{IFFT}_N
\]

\(p_c\) = Polar to complex conversion
\(c_m\) = complex multiplication
\(\text{cos}\) = cosine calculation
\(N\) = synthesis window size
\(n_p\) = analysis frames used
\(n_s\) = synthesis frames used
\(\text{pitch hopsize}\)

However, in certain contexts the polar to complex conversion can be done in the analysis stage, this way reducing significantly the synthesis computational cost. This would be the case, for example, of a sample based singing voice synthesizer which reads a preanalyzed database.

The resulting time domain signal after the IFFT must be multiplied by an overlapping window and added to the output sound buffer. A triangular window would be fine but it must have a size shorter than \(N\) to avoid artifacts due to time aliasing (the edges of the synthesized buffer will not be used). Here it is interesting to point out that the estimated spectrum of the radiated voice pulse \(X'(e^{j\Omega})\) is not convolved anymore with the analysis window, thus the reconstructed signal doesn’t need to be divided by it before overlapping.

In Figure 6 we can observe several examples where a single pulse has been synthesized out of the analysis of the voiced utterance. Besides, in Figure 7 we can see how it looks an isolated radiated glottal pulse is visible.

4.1. Voice residual

In the estimation of \(X'(e^{j\Omega})\) we have obtained a spectrum from the interpolation of the harmonic peaks. It is clear that we have disregarded all the data contained in the spectrum bins between the harmonic frequencies. This data often explain irregularities of the pulse sequence (time and amplitude) plus noisy or breathy characteristics of the voice. These irregularities could be somehow reproduced by analyzing the unwrapped fundamental phase envelope and the outputs of the MFP4. On the other hand, the breathy part could be synthesized using some of the original spectrum data.

Initially we tried to use the residual obtained by subtracting in frequency domain perfect sinusoids with the amplitude, frequency and phase values of the detected harmonics to the original spectrum. However, sometimes we found that some harmonic information was kept, especially in transitions, producing artifacts. Better results can be obtained by applying comb-like filters to this residual which attenuate the frequency bands around the harmonic
4.2. Unvoiced segments

The proposed algorithm is thought to be used only in voiced sections, since it assumes a periodic signal as input. Actually, in our experiments we have combined it with a phase-vocoder based technique for unvoiced segments, in which a white noise source is filtered with the spectral amplitude of the input signal. This method allows applying transformations with high quality results. However, it fails to properly handle transients, so plosives consonants are smeared and intelligibility is degraded. In order to improve the results we have adapted a transient processing algorithm which is able to detect them and discriminate which spectral peaks contribute to them.

5. VOICE TRANSFORMATIONS

Several voice transformations can be achieved using this technique, from high level (pitch-shifting, time-scaling, timbre modifications) to low level ones (independent pulse control). In this section we will briefly present how some of them can be applied.

Pitch and Time-Scaling

Pitch and time-scaling transformations can be implemented in a similar way to TD-PsOLA, controlling the speed at which the sequence of analysis frames is read and distance between pulses. The timbre will be preserved as long as the estimated amplitude spectrum is not modified.

Timbre Modification

Timbre can be modified by scaling, warping or equalizing the estimated spectral amplitude of each pulse. In the real case, if a formant is shifted in frequency the related phase shift (see 3.1) is shifted as well. This means it would be desirable to scale and warp the spectral phase envelope as well. This could be done by applying the same transformation to both amplitude and phase envelopes of the polar spectrum or just applying it once to the complex spectrum.

Voice Disorders

Several voice disorders, intentional or not, can be characterized by irregularities in the excitation glottal pulse sequence, both in time (jitter) and amplitude (shimmer). These irregularities can be described mostly by the appearance of subharmonics in the spectrum, which are hard to manipulate in frequency domain. However, in our case, we have an independent control of the location and amplitude of each pulse, thus easily different patterns of irregularities can be synthesized and even vary along time. We have observed that sometimes in a growl pattern (as the one shown in Figure 8) voice pulses have different timbres which differ by some equalization. In this example the two pulses with more amplitude experience a significant boost around 4 KHz. This behavior can also be reproduced with the presented technique by equalizing differently each pulse’s amplitude spectrum.

6. RESULTS AND CONCLUSIONS

The presented method is able to isolate radiated voice pulses and model them in frequency domain. Several high and low level transformations can be applied to voiced utterances with high quality results. It provides an independent control of each synthesized pulse allowing to easily apply voice disorders patterns. The proposed algorithm is not suitable for unvoiced sections and therefore we have combined it with a phase-vocoder based technique to deal with them.

Compared to classical time-domain methods, the presented algorithm is able to obtain the whole voice pulse response and to apply complex timbre transformations. Compared to frequency-domain techniques it provides a way to control each pulse independently and minimizes artifacts due to noisy peaks and local spectrum contextualization.

This method has some difficulties to process voice utterances with vocal disorders. In such case, pulse onsets need to be detected, as well as some timbre differences at pulse level. More research should be done in this area.

We have tested the algorithm only with vocal sounds. It would be nice to test it with other instruments as well. This would lead us to think that maybe the presented algorithm is not so much related to voice modeling but to periodic signal modeling. However, it has been developed thinking on dealing with voice most relevant characteristics (periodicity, impulsive characterization as result of filtering a sequence of pulses, formants and their effect on the harmonic phase synchronization...) and also trying to achieve the most typical and interesting voice transformations.

7. REFERENCES

ADAPTIVE EFFECTS BASED ON STFT, USING A SOURCE-FILTER MODEL

Vincent Verfaille, Philippe Depalle
Sound Processing and Control Laboratory
Faculty of Music – McGill University
555, Sherbrooke Street West
Montréal, Québec H3A 1E3, Canada
{vincent|depalle}@music.mcgill.ca

ABSTRACT
This paper takes the opportunity of presenting a set of new adaptive effects to propose a generic scheme for adaptive effects built upon a spectral source-filter decomposition and a Short-Time Fourier analysis-resynthesis. This allows for a better formalization of the involved signal processing algorithms and leads to a simple classification of adaptive effects already presented in the literature, that falls into this category.

We discuss the motivation and the advantages of combining source-filter modeling and phase vocoder representation for the design of adaptive digital audio effects. Then we detail the general structure that includes STFT analysis and re-synthesis scheme, the source filter decomposition, and an adaptive control unit composed of a feature extraction system and a sound mapping unit that might be driven by a gestural control section.

1. INTRODUCTION
Adaptive effects are audio processing systems which controls are derived from sound features, in order to take into account the evolution of their structural properties (often called “musical gestures”). As a canonical example, cross-synthesis (as defined in [2]) uses the information from one signal to modify the spectral envelope of a second one. In this paper, we present a generic scheme for adaptive effects built upon a Short-Time Fourier analysis-resynthesis and a spectral source-filter decomposition.

During the 1930ies, H. Dudley invented the vocoder (Voice Operated reCORDEr) at the Bell labs [3]. The vocoder consisted of analyzing a voice signal with a filter-bank and was designed for voice coding in telecommunication applications. Its musical use has been popularized by The Beatles’ song “Tomorrow Never Knows” in 1966 and by W. Carlos’ song “TimeSteps” in 1971 (soundtrack of “A Clockwork Orange”).

An improved version of the vocoder was introduced in the mid 1960ies (namely the phase vocoder) [4], then explicitly linked to the Short-Time Fourier Transform (STFT) [5] and has benefited for many optimization, such as in [6]. At the same time, analysis and synthesis techniques were developed in the same context of voice coding and synthesis, namely source-filter model and linear prediction coding (LPC) [2]. Both techniques allows for many transformations of sounds, among which interpolation, pitch-shifting with formant preservation and robotization.

Recently, more and more effects that takes into account sound features of the sound under process have been developed, such as morphing [2] [8] [9], voice morphing [10] [11], etc. The generic structure we propose allows for a better formalization of the involved signal processing algorithms and leads to a simple classification of adaptive effects already presented in the literature, that falls into this category. After introducing the techniques and the general structure (Section 2), we present a classified review of usual and new adaptive source-filter effects (Section 3) that processes the spectral envelope, the source or both of them.

2. TECHNIQUES
2.1. Motivations
Based on STFT, the phase vocoder represents exhaustively any sound without any error, allows for filtering with very accurate frequency response, for time-scaling without pitch-shifting and reciprocally, as well as many other exotic transformations [12] [13]. This time-frequency representation can be viewed as a block-by-block processing system [4] [5].

The source-filter model can be viewed in two ways: a signal production model that directly acts in the time domain, and a spectral classification of sound properties. It sorts the slow frequency-varying part of the spectrum which is interpreted as the frequency response of a filter, while the fast frequency-varying features are considered to be the spectrum of the source. Contrarily to the STFT, the source-filter model is not a representation, i.e. most of the signals cannot be exhaustively represented without an error.

However, a combination of the strengths of both methods is possible by separating the source and the filter in the spectral domain (e.g. using the STFT with the cepstrum technique [14]). Then we can benefit from the abstract and efficient coding of the source-filter model while preserving the processed signal from any modeling error.

2.2. Adaptive Digital Audio Effects (A-DAFx)
We call “adaptive digital audio effects” the generalization of effects and their control [15] [16] in the context of “intelligent effects” [17] and “content-based transformations” [18]. There are two types of control (see Fig. 1):
- adaptive control which is a time-varying control derived from sound features (i.e. an analysis step) [17] and modified by appropriate mapping functions;
- gestural control for realtime access through input devices.

Several forms of adaptive effects exist, depending on the sound signal from which features are extracted [15] [16]. We name
- “adaptive” effects when features are extracted from \( q[n] \);
- “auto-adaptive” effects when extracted from \( x[n] \);
- “feedback”[1] “adaptive” effects when extracted from \( y[n] \);
- “cross-adaptive” effects when two input signals are used.

The mapping between sound features and effect control values includes non-linearities as well as feature combinations [20]. The effect controls and their mapping can be modified by gestural control [21].

2.3. STFT

In order to describe the complete processing system, we briefly present the block processing structure (see Fig. 2) based on the STFT [4, 5]. The STFT \( F_x[m, k] \) of a signal \( x[n] \) is defined as

\[
F_x[m, k] = \sum_{n=-(N-1)}^{\infty} x[n] \cdot w[mR_A - n] \cdot e^{-j2\pi mk/N},
\]

where \( w[n] \) is a window which length defines the block size, \( R_A \) is the step increment, \( m \) is the current time frame index, \( N \) is the number of spectral bins, indexed by \( k = 0, \ldots , N - 1 \). The synthesis stage uses the classical overlap-add technique [22] with the same time increment, chosen to guaranty a perfect reconstruction of the signal [5].

2.4. Source-Filter Processing

In order to get a description of a sound signal in terms of a source-filter model, one need to deconvolve the signal by first estimating its spectral envelope \( H_x[m, k] \). Depending on the chosen technique or the appropriate coding, \( H_x \) is going to be represented by a set of parameters \( \{P_{H_x}[m] = E_{\alpha}[H_x]m, \bullet \} \), that denotes either reflection coefficients, autoregressive coefficients, cepstral coefficients, correlation coefficients or formant coefficients. The filter estimation is achieved by various and well-known techniques such as LPC [7], cepstrum [14] or spectral breakpoint functions [23]. We then deduce the STFT of the source \( S_x[m, k] \) (see Fig. 2)

\[
F_x[m, k] = H_x[m, k] \cdot S_x[m, k].
\]

[1]The word “adaptive” is commonly used in this context, where the output signal is analyzed and used to minimize an error function, such as in adaptive filtering (telecommunications) [15].

[2]The notation \( G[m, \bullet] \) stands for the frequency vector at time index \( m \).

Figure 1: Diagram of the gesturally controlled adaptive effect.

Figure 2: Diagram of adaptive source-filter block-by-block processing.

In case the adaptive control and the source-filter processing part of the A-DAFx use an identical analysis step and the same input signal, they can be factorized.

2.5. Useful Notations

The STFT of the output signal \( F_y \) is derived from input sources’ and filters’ STFTs by a given transformation noted \( T_F \)

\[
F_y = T_F(F_x, F_q).
\]

When \( F_y \) is separable we consider the transform to be a source-filter one

\[
F_y = T_S(S_x, S_q) \cdot T_H(H_x, H_q).
\]

Notice that in general, \( T_F, T_S \) or \( T_H \) might depend on features extracted from the signals \( x \) or \( q \).

Usual effects often involve frequency shifting, scaling or warping of a time-frequency function \( G_a \), i.e. \( F_a, S_a, \) or \( H_a \), as well as multiplication by another time-frequency function. Frequency shifting of \( G_a[m, k] \) by \( \beta[m] \) frequency bins is denoted

\[
Shift[G_a, \beta][m, k] = G_a[m, \beta[m] + k].
\]

As soon as we shift \( G_a \), we have to prevent aliasing due to spectral content moved out of the bounds, and fill in emptied regions at the other bound [24]. \( Shift \) includes this process.

Scaling (or dilation) can be applied to the spectral envelope and to the source (it is then called pitch-shifting), with the same boundary management than the one used for shifting. Scaling by a ratio \( \lambda[m] \) is denoted

\[
Scale[G_a, \lambda][m, k] = G_a[m, \lambda[m]k].
\]

Warping (or arbitrary modifying) consists of applying an arbitrary function \( W[m, k] \) to \( G_a[m, k] \), and is denoted

\[
Warp[G_a, W][m, k] = G_a[m, W[m, k]].
\]

Shifting and scaling are simple cases of warping function. We note \( Mul \) the multiplication transform \( (Mul[G_a, E] = G_a \cdot E) \) and \( Id \) the identity transform \( (Id[G_a] = G_a) \).

We define the time-varying interpolation \( G_{a\gamma}[m, \bullet] \) of spectral functions \( G_a[m, \bullet] \) and \( G_q[m, \bullet] \) by using a time-varying interpolation coefficient \( \gamma[m] \) that evolves in the range \( [0; 1] \)

\[
G_y = \text{Interp}[G_x, G_q, \gamma].
\]
Notice that $\gamma[m]$ is not necessarily a monotonic function of time. Warping does not necessarily guaranty a valid STFT. Therefore, special frequency warping functions need to be used [23].

3. EFFECTS BASED ON A SOURCE-FILTER PARADIGM

We will present effects that are based on a source-filter modeling of spectral information [4]. We will successively consider effects that modify the spectral envelope only, the source only or both.

3.1. Effects on the Spectral Envelope

When applying the identity function to the source, only the spectral envelope is transformed

$$F_y = S_x \cdot T_H[H_x, H_q]. \quad (10)$$

In this class of effect, we can regroup not only spectral envelope interpolation such as cross-synthesis and hybridization, but also adaptive spectral envelope modifications. When the processing is linear, $T_H$ simply becomes a multiplication of the spectral envelope by a time-frequency function, and then the explicit source-filter separation is no more needed.

3.1.1. Adaptive Spectral Envelope Modifications

Modifying a spectral envelope is useful not only for transforming voiced sounds as it distorts and moves the formants, but also for creating new sounds for electro-acoustic music as it strongly modifies the spectral envelope over time. Based on the functions defined in sec. [25], we have built several adaptive spectral envelope modifiers to generate timbre modulation according to the content of the signal [10]. Adaptive shifting of the spectral envelope is then given by

$$H_y = \text{Shift}[H_x, \beta]. \quad (11)$$

Adaptive spectral envelope scaling depends on a scaling ratio $\lambda[m]$ greater or lower than 1 (expanding/compressing) that is given by

$$H_y = \text{Scale}[H_x, \lambda]. \quad (12)$$

Adaptive spectral envelope warping uses a non linear curve $W$ which varies in time. We have found useful to express $W$ as a linear interpolation between the identity warping ($W[m, k] = k$) and a vector $c_2[m, k] \in [0; 1], k = 1, ..., N$. This allows for an easy balance between a bypass effect and a warping effect. In practice, the interpolation ratio $c_1[m] \in [0; 1]$ is for example derived from the RMS, or the voiciness, whereas $c_2$ is derived from the spectral envelope or its integral

$$W[m, k] = c_1[m] \cdot c_2[m, k] \cdot N + (1 - c_1[m]) \cdot k. \quad (13)$$

The spectral envelope is given by

$$H_y = \text{Warp}[H_x, W]. \quad (14)$$

3.1.2. Vocoding Effect [3][2]

The classical vocoding effect consists of applying the spectral envelope $H_q$ to $F_x$. This adaptive effect is historically implemented using the channel vocoder [3] but can also be implemented through the use of the phase vocoder [4].

$$T_H[H_x, H_q] = H_x \cdot H_q, \quad (15)$$

$$F_y = F_x \cdot H_q. \quad (16)$$

In practice, the input signal $x[n]$ that provides the source must be a rich and complex one (e.g. a monophonic or polyphonic harmonic sound, or an inharmonic sound such as a bell). The “filtering sound” $q[n]$ must exhibit strong formants, such as the human voice. Indeed, most of vocoding effect examples consist of making instruments talk.

Cross-synthesis such as defined in [2] is another occurrence of the vocoding effect. The only difference when compared to the classical one is that the spectral envelope $H_q$ is explicitly extracted by using an LPC analysis and then applied by using an IIR filter or a phase vocoder [26]. The output STFT is

$$F_y = S_x \cdot H_q. \quad (17)$$

Even though cross-synthesis is used to describe the process involved in [2], we prefer to define cross-synthesis as a more general processing, that blends parameters coming from two different sounds to generate a third one.

3.1.3. Interpolation Between Two Spectral Envelopes

We will consider the “L-interpolation” $H_y$ between two spectral envelopes $H_x$ and $H_q$ as the interpolation between time-varying parameter sets $P_{H_x}[m, \bullet]$ and $P_{H_q}[m, \bullet]$ that represent the spectral envelopes ($H_x$ itself, cepstral coefficients, auto-regressive coefficients, formant coefficients, autocorrelation coefficients, reflection coefficients). The interpolated value

$$P_{H_y}[m, \bullet] = L^{-1}(H_y)[m, \bullet] \quad (18)$$

given by

$$L^{-1}(H_y) = \text{Interp}[H^{-1}_x, \gamma[m], \gamma], \quad (19)$$

with the time-varying interpolation ratio $\gamma[m] \in [0; 1]$. We denote

$$H_y = L^{-}\text{Interp}[H_x, H_q, \gamma]. \quad (20)$$

this interpolation between parameters of $H_x$ and $H_q$. The cross adaptive effect may imply complex strategies when the sets to interpolate do not have the same number of parameters (e.g. from 5 to 4 formants). We now focus on the particular case of hybridization.

3.1.4. Hybridization (Voice Morphing) [10][11]

We call hybridization or voice morphing the cross-synthesis of one voice by another voice previously analyzed. Its purpose is to make a singing voice fit to specific properties: e.g. recreating the castrato’s voice of Farinelli [10] (see Fig. 3), or singing with someone else’s voice such as in the Voice Impersonator in a karaoke

\footnote{A famous example is the talk box, popularized by Pete Frampton in the 1970ies. The guitar signal goes through a tube into the mouth, and the spectral envelope of the mouth is superimposed to the guitar harmonics.}
context [11]. The spectral envelope $H_y$ is computed by interpolating between the envelope of the signal itself and the envelope of the signal stored in the database, according to specific mapping rules. The output STFT is given by

$$F_y = S_x \cdot H_y,$$

(21)

### 3.1.5. Adaptive Equalizer

Starting from the structure of an equalizer, which is essentially to filter a signal by a given spectral frequency response, we have extended it to an adaptive effect, where the frequency response $H_y$ is provided through the adaptive control section of the processing system displayed on Fig. 2. In this adaptive effect, $H_y$ can be generated from any vector feature of the input signal $q$ after some specific mapping (see [16] for more details). The only constraint imposed to $H_y$ is that for any time $m$, $H_y[m, \bullet]$ evolves sufficiently slowly as a function of frequency to be considered as a spectral envelope. The output STFT is given by

$$F_y = H_y \cdot F_x,$$

(22)

Moreover, we have to ensure that the time evolution of $H_y[m,k]$ is sufficiently slow so that, given the time increment, its spectrum fits in the bandwidth mentioned in sec. 2.2 in order to prevent aliasing of parameters.

One of the typical examples we have developed is the adaptive spectral panoramization [16]. It consists of generating a stereophonic signal by an adaptive splitting of the spectrum of the input signal $x$. We evaluate a panaramization angle vector $\theta[m, \bullet]$, which is derived from input sound features (e.g. the waveforms). STFTs of the two output signals $y_1$ and $y_2$ are computed from the Blumlein law

$$\left\{ \begin{array}{l} Y_1 = \frac{2}{\pi} \sin \theta \cdot F_x, \\ Y_2 = \frac{2}{\pi} \cos \theta \cdot F_x. \end{array} \right.$$  

(23)

In practice, each frequency bin of the STFT $F_x$ is moved to a specific location in space, and the signal is split with more or less independent motions and speeds.

### 3.2. Effect on the Source

Effects on the source preserve the spectral envelope so $H_y = H_x$. The possible effect that falls into this category are pitch-shifting, ring modulation (particular case of sec. 3.3.1) and frequency warping.

#### 3.2.1. Pitch-Shifting with Formant Preservation

Pitch-shifting consists of resampling the source and preserving the sound duration, e.g. using the overlap-add of the STFT technique [24]. However, the spectral envelope is also scaled (leading to the “Donald Duck” effect). In order to preserve formants, a pitch-shifting algorithm must be applied to the source only [27][28]

$$S_y = Scale[S_x, \beta],$$

(24)

with $\beta$ being the pitch-shifting ratio. When $\beta[m]$ varies in time, the pitch-shifting becomes adaptive (allowing intonation changes and adaptive resampling [21]).

#### 3.2.2. Adaptive Spectral Warping

Time-varying spectral warping allows to change a musical sound from an harmonic one to an inharmonic one, while preserving the spectral envelope [10]. The source is computed as

$$S_y = Warp[S_x, W],$$

(25)

where the warping function $W$ has the same structure than the one specified in Eq. (17).

### 3.3. Effect on the Source and the Spectral Envelope

#### 3.3.1. Adaptive Ring-Modulation

The famous ring modulation is a simple amplitude modulation with no DC component [6]. It consists of multiplying a signal $x[n]$ by a modulator signal. We only consider the case where the modulator is a simple sinusoidal wave $x_{mod}[n] = \sin(2\pi f_{mod} n)$, with $f_{mod} = \frac{f_s}{N}$, where $f_s$ is the audio signal sampling rate. The whole spectrum is then duplicated and shifted

$$F_y = \text{Shift}\{F_x, k_{mod}\} + \text{Shift}\{F_x, -k_{mod}\},$$

(26)

The modulation frequency is over 20 Hz, so that it is timbre and not amplitude (and so rhythm) that is modified.
which implies that both source and spectral envelope are modified. The adaptive ring modulation uses a time-varying modulation
frequency \( f_{\text{mod}} \). We designed an adaptive ring modulation by controlling the frequency of the modulator from mapped input
signal features [16].

By a careful selection of a sound feature, a ring modulator that
preserve the fundamental frequency can be designed [29]. If
\( f_{\text{mod}} = M f_0 \) with \( M \) an integer, pitch is unchanged. When \( f_{\text{mod}} = M f_0 / P \) with \( P \) an integer, pitch is transposed
down. These particular cases only affect the spectral envelope (sec. 3.2).

For any other values \( f_{\text{mod}} = M f_0 / P \), the sound
remains inharmonic. In this case, the adaptive ring-modulation can
be applied to the source only [16]

\[
S_y = \text{Shift}\{S_x, k_{\text{mod}}\} + \text{Shift}\{S_x, -k_{\text{mod}}\}. \tag{27}
\]

As it only modifies the source, it can be considered as an effect of
sec. 3.2. This effect provides an inharmonization of a voice while
preserving its intelligibility; this is also useful for transposing
inharmonic sound, such as bell sounds.

### 3.3.2. Adaptive Spectral Envelope and Source Modification

Adaptive spectral envelope and source modifications (shifting, scal-
ing or warping) can be combined [16] to provide interesting effects
for processing electroacoustic sounds. The output STFT is

\[
F_y = T_S\{S_x\} \cdot T_H\{H_x\}. \tag{28}
\]

A particular case is the gender change.

#### 3.3.3. Gender Change [30]

To transform a female voice into a male voice, and vice-versa [31][18], pitch has to be shifted (up for male-to-female, down for
down for male-to-male) and the formants have to be warped, more
precisely this is a combination of scaling the spectral envelope in or-
der to take into account changes in the length of the vocal tract
and a shift of the spectral envelope in order to take into account
synchronization between the fundamental frequency and the first
formant’s frequency for high pitched sounds. It is an auto-adaptive
effect, controlled by \( f_0 \), with the output STFT

\[
F_y = \text{Scale}\{S_x, \alpha\} \cdot \text{Warp}\{H_x, \beta\}. \tag{29}
\]

#### 3.3.4. Robotization with Spectral Envelope Modifications

Robotization consists of nulling the phase for every block of a
voice signal \( x[n] \cdot w[mR_A - n] \) [31], such as defined in Eq. (1).
The hop size between blocks imposes the robot’s pitch, which can
also adaptively evolve in time according to sound features [15].
This effect only modifies the pitch. Formants can be independently
modified, for example to obtain a Donald Duck robot effect.

\[
F_y = |S_x| \cdot T_H\{H_x\}. \tag{30}
\]

### 3.3.5. Effects Based on Interpolation [17][24][35][36][46]

Effects based on spectra L-interpolation such as defined in Eq. (9)
appear in the literature under many different names, not only de-
pending on the interpolated parameter sets, but also depending on
the authors. Here is a list of existing effects that falls into this
category, with the names proposed by the authors.

- **timbral morphing** [32][33] when \( \gamma[m] = \gamma_0 \) and \( P_s, P_q \) are
  additive parameters (partials’ amplitude and frequency);
- **mutation** [34][35][31] when

\[
F_y[m, \bullet] = \text{Interp}\{p_x, p_y, \gamma_0\} \cdot e^{j\text{sharp}(f_x - f_y, \gamma_0)};
\]

with the specific case found in [17] when \( \gamma_0 = 0 \) or 1, and
\( \gamma_0 = 1 - \gamma_\rho \), this is called hybridization by the author.
Notice that it differs from the definition of hybridization we
use in sec. 3.3.4.

- **spectral interpolation** [46] when

\[
F_y[m, \bullet] = \text{Interp}\{S_x, S_q, \gamma_s\} \cdot \text{Interp}\{H_x, H_q, \gamma_h\};
\]

with the particular case of voice morphing (hybridization in
our definition) [10][11] when \( \gamma_s = 0 \), and voicing
effects/cross-synthesis [9][4][1][24][20] when \( \gamma_s = 0 \) and
\( \gamma_h = 1 \).

The so called **timbral metamorphosis** [35] is the general case when
\( \gamma[m] \) monotonically evolves from 0 to 1 and takes continuous or
discrete values. Timbral morphing and mutations cannot be con-
sidered as based on the source-filter model.

### 3.4. About Resampling Control Parameters

According to the structure of the adaptive audio effects consid-
ered in this study, the result of the mapping of extracted features
\( G_e[\bullet] \) has to be a vector of size \( N \), whatever the features are.
Another constraint is that \( G_e \) is sampled at frequency \( f_s/R_A \).
Therefore, it has to be low-pass filtered to prevent aliasing and un-
der sampled when the feature is generated at a higher rate, and re-
sampled when provided at a lower rate. Nevertheless, in a musical
situation, one may want to use and control the degree of aliasing
to add modulations in the resulting sound.

### 4. CONCLUSIONS

We presented a general structure for adaptive effects built upon
a Short-Time Fourier analysis-re-synthesis and a spectral source-
filter decomposition. This structure demonstrated to be very flex-
ible and allows to reformulate in an unified framework many al-
ready existing adaptive audio effects. As a part of the review of
these existing effects, we have also presented several new adaptive
effects that we developed: adaptive shifting, scaling and warping
of the spectral envelope, adaptive equalizer and spectral panora-
mization, adaptive ring modulation of the source and robotization
with spectral envelope modifications. We hope that this formula-
tion which combines STFT, source-filter decomposition and adap-
tive control will give rise to the development of new adaptive audio
effects.

### 5. REFERENCES

[1] E. Mètois, “Musical Gestures and Audio Effects Process-
ing,” in Proc. COST-G6 Workshop on Digital Audio Effects

Computer Music Applications,” Journal of the Audio Engi-


GABOR’S LEGACY: NEW DEVELOPMENTS IN GRANULAR ANALYSIS AND SYNTHESIS

Curtis Roads

Media Arts and Technology
3431 South Hall
University of California
Santa Barbara CA 93106-6065 USA
clang@create.ucsb.edu

ABSTRACT
Granular synthesis has evolved from the theories of Gabor and Xenakis into a broad range of techniques that treat sound as a stream of acoustic particles.

This lecture traces the particle paradigm from its origins to the present day. Through myriad sound examples, I will demonstrate the power of the particle paradigm as a tool for synthesizing transforming sound. Some of these examples will be drawn from my book Microsound (2002, The MIT Press).

I will demonstrate PulsarGenerator, a program developed by Alberto de Campo and me for particle sound synthesis. PulsarGenerator was inspired by the sounds of the early analog electronic music. After a brief presentation of the matching pursuit wavelet transform, which Garry Kling and I will present at another point in this conference, I will unveil a new program for generalized control of particle synthesis, EmissionControl, developed by David Thall and me. Finally, I will present examples of music realized with particle techniques with accompanying videos by Woon Seung Yeo and Brian O’Reilly.

Curtis Roads teaches in CREATE, Center for Research in Electronic Art Technology, in the Department of Music and also in Media Arts and Technology (MAT), University of California, Santa Barbara.

He studied music composition at California Institute of the Arts, the University of California, San Diego (B. A. Summa Cum Laude), and the University of Paris VIII (PhD).

From 1980 to 1986 he was a researcher in computer music at the Massachusetts Institute of Technology and the MIT Media Laboratory. He then taught at the University of Naples “Federico II,” Harvard University, Oberlin Conservatory, CCMIX (Paris), and the University of Paris VIII. He has led master classes at the Australian National Conservatory (Melbourne) and the Prometeo Laboratorio (Parma), among others.


He has served on the composition juries of the Ars Electronica (Linz) and the International Electroacoustic Music Competition (Bourges, France).

Certain of his compositions feature granular and pulsar synthesis, methods he developed for generating sound from acoustical particles.

Upon arrival at UCSB in 1996, he developed the Creatophone, a system for spatial projection of sound in concert. Another invention is the Creatovox, an expressive new instrument for virtuoso performance that is based on the synthesis of sound particles. The Creatovox, developed in collaboration with Alberto de Campo, was first demonstrated to the public in March 2000.

His composition Clang-Tint (1994) was commissioned by the Japan Ministry of Culture (Bunka-cho) and the Kunitachi College of Music, Tokyo. His music is available on compact discs produced by Asphodel, MODE, OR, the MIT Media Laboratory, and Wergo.


His electronic music collection POINT LINE CLOUD won the Award of Distinction at the 2002 Ars Electronica and is being released as a CD + DVD on the Asphodel label (San Francisco) in 2004.
SOFTWARE FOR MEASURING AND IMPROVING ESOPHAGEAL VOICES

B. García, J. Vicente

“ESOIMPROVE” Group: Dpt. of Telecommunications
University of Deusto, Bilbao. Spain
mbgarcia@eside.deusto.es

ABSTRACT

The main aim of this paper is to describe a new software program for esophageal speech treatment developed at the University of Deusto. The software tool, named “ESOIMPROVE”, allows both to characterize and to modify this speech, and provides the necessary framework to achieve a high quality and intelligible transformed esophageal speech by applying a complete range of sound effects and algorithms. In this field, this tool represents a considerable advance in the study of these voices. The final objective of the project is to obtain an esophageal speech with acceptable levels of quality and intelligibility, and some more works in this direction are being actually developed.

1. INTRODUCTION

A complete set of software tools have been developed for the here presented software package in order to allow the characterization of laryngectomees’ esophageal voice through the measurement of its feature parameters: pitch, jitter, shimmer and harmonics-to-noise ratio (HNR), and the application of filters and specific algorithms for the transformation and enhancement of low-quality speech signals.

The main problem of using commercial software for the measuring and characterization of esophageal voices is that most of it is only designed to work with normal voices. Therefore, when used to assess the quality of esophageal speech, the values of the calculated feature parameters are often wrong and inconsistent. The main reason for their incorrect esophageal speech feature measurements is their inability to detect each cycle peak, which is a fundamental aspect of the method used in the application here presented.

2. METHODS

Matlab 6.5 [1] has been the chosen framework for the development and implementation of this software program. One of the latest characteristics of the application is that it can calculate esophageal speech feature parameters such as jitter, shimmer, pitch and HNR, i.e. [2]. None of these parameters can be calculated using commercial applications, since the resulting values exceed the normal ranges.

In order to determine the characteristics, that the program should have, a complete study about esophageal speech was done, especially in two aspects: First of all, which features of this speech should be modified to improve quality. And secondly, why commercial programs were unable to measure and modify these parameters, with the obtained results “ESOIMPROVE” was designed, an example of the program’s main screen can be appreciated in Figure 1.

2.1. Period Marking

The first step in the development of this software was to design a special algorithm to determine the marks [3] corresponding to each cycle peak, using the Discrete Time Fourier Transform in order to measure the harmonics and correctly identify the periodic cycles of the signal, which constitutes the base for all the algorithm included in the software.

The problem of detecting correctly the fundamental period’s maximums is critical to the application. In order to achieve a correct detection, the sonority of small frames is calculated, and a threshold for it established, if the sonority is above this threshold, the considered frame contains a maximum. Once a small range for the position of the peak has been determined, it is possible to apply a conventional maximum detection algorithm in order to detect the exact location of it.

Figure 1: Main screen
2.2. Speech Processing

Apart from featuring the speech signal, the software includes some special algorithms designed for its treatment, concretely in two aspects: First, an algorithm that modifies the poles of the system that models the vocal tract making them more stable, an example of how this is done can be seen in Figure 2, and second an algorithm that implements a pitch scaling effect over the speech signal.

The first part called ‘Formant evolution reconstruction’ modifies the behavior of the main formants along the time, so that they don’t suffer big power losses, due to the air pressure reduction in laryngectomized people’s voice.

This algorithm analyses the energy of the formants in each instant and detects its sudden decrease, applying then the correction formula:

$$MIN_{mod} = \frac{2}{2} \left( \frac{MAX_1 + MAX_2}{2} + MIN \right)$$

After calculating the modified minimum, the points between the two maximums are adjusted to this new value with the help of two correcting straight lines: one for the left, and the other one for the right part of the minimum. With this modification we get the energy loss not to be so heavy.

Also some additional functions for speech transformation are included, such as applying various filters, taking signal pieces, and the possibility of watching the broad and narrowband spectrograms.

3. RESULTS

The feature parameters obtained after the application of the mark calculating algorithm match the results obtained with the manual analysis of the spectrograms, validating therefore the mark based measuring method.

The program allows the application of algorithms to transform the low quality esophageal speech signal [4] and improve its intelligibility. These algorithms can be executed step by step, in order to study the partial results at each stage and adjust the parameters that optimize the quality of the resulting speech signal. Figure 2 shows how the user can adjust the parameters the algorithm will apply, by typing the optimal values in the text-boxes.

It is important to remark the utility of the software here presented in the field of medicine, specialists could use it to measure the improvements of the patients after surgery (laryngectomy) and during esophageal speech learning process. It is also very useful for the digital speech processing due to the possibility of applying different effects to an esophageal speech.

4. DISCUSSION

Laryngectomees can’t objectively evaluate their voices either after surgery or after the esophageal speech learning period, since there is no valid software application available. Algorithms applied by commercial programs mistake speech signal cycles due to esophageal speech signal’s high noise level and to the fact that correct feature parameter values happen to be out of the normal ranges these programs work with.

The described mark calculation method solves these problems, in Figure 3 the parameter calculation screen can be seen, allowing the correct calculation of the esophageal speech feature parameters, in Figure 4 we can see how this application detect correctly the pitch [6] while the commercial programs are unable to measure esophageal voices.

5. CONCLUSIONS

The software application here presented allows the measurement and transformation of esophageal speech with the same feature parameters which define normal speech. Therefore, the objective evaluation of the improvements achieved either with new esophagus based speaking techniques or with new digital esophageal speech signal processing techniques is possible. In such a sense, any digital esophageal speech signal processing algorithm developed in Matlab can be integrated and evaluated against an esophageal speech data base, partial results can be studied through spectrograms and poles-zeros diagrams and the algorithm configuration parameters can be optimized in order to obtain a better speech quality.
Table 1: Pitch evolution measured with a commercial program and with “ESOIMPROVE” software

<table>
<thead>
<tr>
<th>Time (ms)</th>
<th>Original pitch</th>
<th>Pitch measured with a commercial software</th>
<th>Pitch measured with “ESOIMPROVE”</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-20</td>
<td>61Hz</td>
<td>101Hz</td>
<td>59Hz</td>
</tr>
<tr>
<td>20-40</td>
<td>70Hz</td>
<td>111Hz</td>
<td>73Hz</td>
</tr>
<tr>
<td>40-60</td>
<td>68Hz</td>
<td>125Hz</td>
<td>69Hz</td>
</tr>
<tr>
<td>60-80</td>
<td>74Hz</td>
<td>94Hz</td>
<td>74Hz</td>
</tr>
<tr>
<td>80-100</td>
<td>55Hz</td>
<td>107Hz</td>
<td>59Hz</td>
</tr>
<tr>
<td>100-120</td>
<td>63Hz</td>
<td>98Hz</td>
<td>62Hz</td>
</tr>
</tbody>
</table>

Table 2: Percentage error in average pitch measurement for five different voices

<table>
<thead>
<tr>
<th>Voices</th>
<th>Original pitch</th>
<th>Pitch measurement error with a commercial software</th>
<th>Pitch measurement error with “ESOIMPROVE”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice 1</td>
<td>61.475Hz</td>
<td>65.34%</td>
<td>3.33%</td>
</tr>
<tr>
<td>Voice 2</td>
<td>69.921Hz</td>
<td>68.23%</td>
<td>3.45%</td>
</tr>
<tr>
<td>Voice 3</td>
<td>84.409Hz</td>
<td>55.76%</td>
<td>2.76%</td>
</tr>
<tr>
<td>Voice 4</td>
<td>58.085Hz</td>
<td>73.83%</td>
<td>1.35%</td>
</tr>
<tr>
<td>Voice 5</td>
<td>70.494Hz</td>
<td>64.32%</td>
<td>2.27%</td>
</tr>
<tr>
<td>Average error</td>
<td>-</td>
<td>65.498%</td>
<td>2.62%</td>
</tr>
</tbody>
</table>
Tables 1 and 2 represent the comparison of final results of pitch calculation between a commercial program and “ESOIMPROVE”, as it can easily be seen a commercial program is completely unable to estimate signal’s pitch and clearly measures a much higher pitch than real. On the other hand, “ESOIMPROVE” measures correctly pitch’s value, with a very small error (2.62% in average). This proofs that specific algorithms work correctly to measure esophageal speech’s features. These excellent results can also be extrapolated to other feature parameters such as jitter, shimmer or HNR.

Such methods of measuring constitute the base for effective transformation algorithms, like the ones which have been here explained, because if it is possible to detect speech’s irregularities, it won’t be difficult to find a way of correcting them. In this sense, as it has been said, two algorithms have been developed, one correcting the characteristic instability of the poles and another one that is able to modify esophageal speech’s low pitch. With these two algorithms we are able to enhance substantially alaryngeal speech.

These algorithms can then be used in the fabrication of devices such as telephone adapters, in order to enhance the esophageal speech signal before sending it through the telephone line, so that the receiver has a better intelligibility during the conversation. But also it is possible to think in other type of traditional applications for example: special sound effects designed to be applied in different fields, could also be modified to work with esophageal voices, in this case this effects would not be oriented to regeneration but to modification of speech.

6. ACKNOWLEDGEMENT

The authors wish to thank the support from the University of Deusto during the development of this work and also the valuable contributions of all the scholarships who have taken part in the project. Equally, they sincerely appreciate the help of the “Asociación de Laringectomizados de Bizkaia”.

7. REFERENCES


IMPROVEMENT OF ESOPHAGEAL VOICES’ PITCH

B. García, J. Vicente, I. Ruiz, A. Alonso, E. Loyo

Esoimprove Group: Dpt. of Telecommunications
University of Deusto, Bilbao. Spain
mbgarcia@eside.deusto.es

ABSTRACT

In this paper it is described a new algorithm for esophageal speech regeneration, based on pitch and jitter modification. Traditional phase vocoder and resampling pitch scaling techniques have been used to develop a new adaptive method which scales the low esophageal speech pitch and applies a variable scaling factor significantly reducing its jitter. This method has shown to considerably improve esophageal speech quality, reducing its hoarseness and increasing its intelligibility. The presented algorithm pretends to be an important step forward in the regeneration of esophageal speech.

1. INTRODUCTION

Laryngectomees have to generate their speech by insufflation of the esophagus and vibration of the pharyngoesophageal segment in replacement of the vocal folds. These special conditions make esophageal speech’s feature parameter differ from normal speech. This algorithm is part of a set of algorithms intended to correct these parameters.

Among esophageal speech irregularities there are many related to the pitch. The fundamental frequency of the resulting speech signal is lower than the pitch of normal voices, due to the different shape and size of the vibration organs. However, this is not the only problem. The unsustained air flow responsible for the vibration of the pharyngoesophageal segment causes the decrease of the fundamental frequency which does not remain constant over time. Therefore, esophageal speech has a very low pitch and a very high jitter [1].

With the aim of correcting these irregularities, the designed algorithm raises the fundamental frequency and lowers the jitter of esophageal speech. In this aspect the algorithm constitutes a total revolution for the esophageal speech treatment, and sets up many new lines of investigation.

2. METHODS

In order to raise the fundamental frequency of esophageal speech, standard pitch scaling techniques [2] have been modified to achieve an adaptive algorithm that responds to instantaneous pitch variations, modifying its pitch scaling factor and reducing the deviation from the medium pitch. A simplified example of how this is done is shown in Table 1. As it can be seen, if the original pitch is higher than the average one then the applied scaling factor is lower and vice-versa.

<table>
<thead>
<tr>
<th></th>
<th>Original pitch</th>
<th>Applied PSF</th>
<th>Modified pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average</strong></td>
<td>65</td>
<td>1.2</td>
<td>78</td>
</tr>
<tr>
<td><strong>1st Period</strong></td>
<td>55</td>
<td>1.418</td>
<td>78</td>
</tr>
<tr>
<td><strong>2nd Period</strong></td>
<td>75</td>
<td>1.04</td>
<td>78</td>
</tr>
<tr>
<td><strong>3rd Period</strong></td>
<td>65</td>
<td>1.2</td>
<td>78</td>
</tr>
</tbody>
</table>

Table 1: Examples of the application of the variable PSF.

In Figure 1 the whole process of the pitch scaling is shown: the first step consists of detecting the signal’s peaks, with a special algorithm adapted for esophageal speech. Once the periods have been correctly determined, it is possible to measure the average speech’s pitch. In the third and fourth steps the algorithm works period by period, first calculating the pitch scaling factor (PSF) appropriate for the period (the process is explained in Figure 2) and then applying the pitch scaling to it (pitch scaling can be applied in time or in frequency). Finally, the modified signal is reconstructed from the modified periods.

Calculating the variable PSF is not an easy task because if the duration of each period is set to be the same, the naturalness of the resulting speech is significantly reduced. Therefore, the recalculation of the scaling factor involves establishing a maximum deviation from its original value, in order to preserve speech naturalness. The developed algorithm involves a frame by frame calculation [3] of the average and instantaneous pitch, and the application of a different scaling factor according to the deviation from the average pitch in order to significantly reduce these pitch irregularities and, as a consequence, the speech signal’s jitter.

![Figure 1: Block diagram of the pitch scaling algorithm.](image-url)
As previously mentioned, the jitter cannot be reduced to zero because the speech would lose naturalness. In order to obtain a clear and natural signal, the variable scaling factor is calculated in the following way (Figure 2): first of all, the pitch of the period is calculated and compared with the average pitch. Secondly, as a result of this comparison, an initial variable factor is calculated: this factor would reduce jitter to zero. In the third step, the difference between the fixed and the variable factor is calculated. If this difference is larger than 0.5 and smaller than 1 the variable scaling factor is limited. This jitter reduction produces a significant enhancement of voice’s quality, since speech’s characteristics such as hoarseness and reverberance nearly disappear, making the speech much more intelligible. At the same time, the limiting of the scaling factor preserves the naturalness of the speech.

3. RESULTS

Table 2 and 3 show the results obtained after applying the algorithm to esophageal speech samples from a database of voiced patrons recorded from ten different laryngectomee patients. Scaling factors of 1.2 and 1.3 [4], have been used in order to increase the average esophageal speech pitch value from around 60 Hz – 80 Hz to 90 Hz. As it is shown in Table 2, pitch scaling has been achieved in all the samples and jitter has been significantly reduced in both cases.

As shown in the mentioned tables, the pitch of the voices rises according to the expected values. For example, in the first case, if the voice’s original pitch was 61.475 Hz, with a PSF of 1.2 it should have risen to 73.77, and in fact the real value (74.310) is very close to this value.

<table>
<thead>
<tr>
<th>Voices</th>
<th>Originals</th>
<th>PSF = 1.2</th>
<th>PSF = 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice 1</td>
<td>61.475</td>
<td>74.310</td>
<td>80.850</td>
</tr>
<tr>
<td>Voice 2</td>
<td>104.112</td>
<td>122.137</td>
<td>134.595</td>
</tr>
<tr>
<td>Voice 3</td>
<td>69.921</td>
<td>86.184</td>
<td>94.598</td>
</tr>
<tr>
<td>Voice 4</td>
<td>58.085</td>
<td>71.577</td>
<td>78.730</td>
</tr>
<tr>
<td>Voice 5</td>
<td>60.556</td>
<td>80.301</td>
<td>81.901</td>
</tr>
<tr>
<td>Voice 6</td>
<td>84.409</td>
<td>101.124</td>
<td>109.871</td>
</tr>
<tr>
<td>Voice 7</td>
<td>70.494</td>
<td>85.383</td>
<td>92.510</td>
</tr>
<tr>
<td>Voice 8</td>
<td>62.626</td>
<td>75.959</td>
<td>83.627</td>
</tr>
<tr>
<td>Voice 9</td>
<td>61.485</td>
<td>75.553</td>
<td>82.748</td>
</tr>
<tr>
<td>Voice 10</td>
<td>58.754</td>
<td>67.408</td>
<td>74.240</td>
</tr>
<tr>
<td>Voice 11</td>
<td>58.284</td>
<td>69.534</td>
<td>75.128</td>
</tr>
</tbody>
</table>

Table 2: Pitch (Hz) resulting of the application of the algorithm to ten samples of a data base
Table 3: Jitter (%) resulting of the application of the algorithm to ten samples of a data base

<table>
<thead>
<tr>
<th>Voices</th>
<th>Originals</th>
<th>PSF = 1.2</th>
<th>PSF = 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice 1</td>
<td>10.333</td>
<td>2.505</td>
<td>2.690</td>
</tr>
<tr>
<td>Voice 2</td>
<td>16.349</td>
<td>9.051</td>
<td>6.864</td>
</tr>
<tr>
<td>Voice 3</td>
<td>18.339</td>
<td>4.519</td>
<td>6.169</td>
</tr>
<tr>
<td>Voice 4</td>
<td>17.960</td>
<td>8.012</td>
<td>9.977</td>
</tr>
<tr>
<td>Voice 5</td>
<td>20.302</td>
<td>12.348</td>
<td>10.782</td>
</tr>
<tr>
<td>Voice 6</td>
<td>15.635</td>
<td>5.620</td>
<td>5.126</td>
</tr>
<tr>
<td>Voice 7</td>
<td>9.037</td>
<td>2.810</td>
<td>4.547</td>
</tr>
<tr>
<td>Voice 8</td>
<td>12.312</td>
<td>4.338</td>
<td>3.462</td>
</tr>
<tr>
<td>Voice 9</td>
<td>10.747</td>
<td>4.717</td>
<td>3.625</td>
</tr>
<tr>
<td>Voice 10</td>
<td>7.020</td>
<td>3.957</td>
<td>4.442</td>
</tr>
<tr>
<td>Voice 11</td>
<td>3.536</td>
<td>4.373</td>
<td>2.697</td>
</tr>
</tbody>
</table>

With a scaling factor of 1.2, in 90% of the samples jitter is reduced to a half or even a quarter of its original value. For a scaling factor of 1.3, results are still better since all samples improve their jitter after the application of the algorithm. These results show that the proposed algorithm solves the esophageal speech characteristic low pitch and high jitter problem.

4. DISCUSSION

Improving esophageal speech is a difficult task, especially hard when such an enhancement involves the detection of the speech signal’s fundamental periods. Therefore, the design of the adaptive pitch scaling algorithm has involved the development of a method to detect each speech cycle’s peak first. The developed technique allows us to obtain the speech signal’s average pitch and the pitch of each speech frame. Once these values have been calculated, the algorithm can be applied.

This algorithm is an important step forward in esophageal speech regeneration, since two characteristic problems of esophageal speech, low pitch and high jitter, are solved at the same time.

Future work should be focused on the development of a shimmer correction algorithm, attempting to improve esophageal speech’s particularly high shimmer value. The effect of applying this set of algorithms to the excitation signal of voiced phonemes rather than to the speech signal should also be investigated.

![Original and modified signal](image-url)
5. CONCLUSIONS

The presented algorithm has achieved the goals of raising the fundamental frequency of esophageal speech to normal speech's average values and of reducing its jitter level. These two improvements make esophageal speech sound more natural and intelligible. The obtained results show, as it can be seen in figures 3 and 4, that esophageal speech can be enhanced and regenerated to normal values. These results support our research and encourage us to develop new algorithms that correct other esophageal speech irregularities, in order to help laryngectomees recover the voice they lost after their illness.

It has been demonstrated once more, that engineering can make easier the life of those who suffer some kind of “disability,” helping them to integrate and build a fairest society.

6. ACKNOWLEDGEMENT

The authors wish to acknowledge the University of Deusto which kindly lent infrastructures and material for this investigation. They would also like to thank all the scholarships that so enthusiastically have collaborated with this project. And especially it cannot be forgotten the help of the members of the “Asociación Vizcaína de Laringectomizados” whose voices constituted the data base for the investigation; without their help it would not be possible to carry out this project.

7. REFERENCES


ACOUSTICAL SIMULATIONS OF THE HUMAN VOCAL TRACT USING THE 1D AND 2D DIGITAL WAVEGUIDE SOFTWARE MODEL

Jack Mullen, David Howard and Damian Murphy

Department of Electronics
University of York, Heslington, York, YO10 5DD, UK
{jm220|dh|dtm3}@ohm.york.ac.uk

ABSTRACT

This paper details software under development that uses the digital waveguide physical model to represent the sound creation mechanism and environment associated with the production of speech, specifically the human vocal tract. Focus is directed towards a comparison between the existing 1D waveguide method, on which several studies have already been conducted, and the developing 2D waveguide mesh method. The construction of the two models and the application of the tract geometry is examined, in addition, the inclusion of dynamic articulatory variations to increase the ability of such systems to create natural sounding speech is discussed. Results obtained from each suggest that the 2D model is capable of producing similarly accurate vowel spectra to that already accomplished with the 1D version, although speech-like sounds created with the 2D mesh appear to exhibit greater realism.

1. INTRODUCTION

The artificial reproduction of human speech sounds is accomplished to within acceptable levels of naturalness using established and widely used techniques such as linear prediction [1] or formant based synthesis [2]. Although of appropriate quality, these methods do not exploit the benefits inherent in the use of a physics based model in the quest for synthesised sounds of an organic nature.

Physical modelling synthesis is focused on the discretisation of continuous real world mechanics into manageable lumped element systems exhibiting behaviour that approximates the target structure. It has greatly improved the naturalness of sound created in the simulation of musical instruments over spectral reconstruction methods. The underlying mathematical laws governing sound and vibration are used to accommodate a virtual source-environment coupling in such models to allow for a greater scope of representation, and hence produce synthesised sounds of greater realism. Digital waveguide physical modelling can be used to produce accurate representations of vibrating structures, such as the column of air within a clarinet, or a plucked guitar string. Digital waveguides constructed in multidimensional mesh formation can be used to model resonating bodies of air and have been applied in room acoustics simulations [3].

Research focused on the development of a physical model of the vocal tract employing the digital waveguide method in one dimension [4] has produced synthesised speech sounds of an organic nature. Currently little is known about the possible improvements in naturalness of synthesis that may be achievable in a similar model extended towards a multidimensional waveguide system.

2. THE DIGITAL WAVEGUIDE MESH

Based on the discretisation of the acoustic wave equation, the digital waveguide comprises a bi-directional unit delay which forms the left going and right going components of a simulated pressure wave. The sum of such components represents the pressure value at each element. Multiple waveguides connected in a line as in Figure 1, with some applied termination constitutes the basic digital waveguide method of, for example, a resonating tube or string.

Figure 1: A 1D Chain of Waveguides

The length of the modelled structure is apparent in the length represented by each waveguide multiplied by the number of waveguides within the model. Scattering equations are performed, resulting in lossless propagation of an applied excitation along the chain of waveguides with boundary reflections observed at either end. Non-uniform scattering within the model can be set as an impedance discontinuity between waveguides.

This modelling method can be extended to higher dimensions with the construction of a digital waveguide mesh (DWM). The width of the target structure can be set within the model in the same manner as the length, resulting in the 2D DWM, modelling, for example, vibrations on the surface of a drum skin or the propagation of sound in a 2D plane through a room. A 3D DWM can be constructed in a similar manner.

Figure 2: (a) A Unit Junction, and (b) A Rectilinear Mesh
Figures 2(a) and (b) detail the scattering junction with \( i \) arbitrary connections and the formation of the rectilinear (junctions with 4 neighbours at 90° from each other in a cartesian coordinate system) mesh, respectively. In Figure 2(b), air pressure values labelled \( p_{J,i} \) indicate an incoming pressure at node \( J \) from node \( i \) (at a unit time step before), and those labelled \( p_{i,J} \) show the outgoing pressure at node \( J \), to node \( i \) (reaching node \( i \) a time step later). The application of the following three equations to each node in the mesh gives rise to accurate lossless scattering of pressure:

- The pressure \( p \) at a lossless junction with \( N \) equal impedance waveguide connections is:
  \[
  p_J = \frac{2}{N} \sum_{i=1}^{N} p_{J,i}
  \]  
  \( (1) \)

- The pressure output \( p_{i,J} \), on each waveguide connected to a junction is directly related to its input:
  \[
  p_{i,J} = p_J - p_{J,i}
  \]  
  \( (2) \)

- The time step is then incremented to distribute all junction output pressures along waveguides to become neighbouring junction input pressures:
  \[
  p_{J,i} = z^{-1}p_{i,J}
  \]  
  \( (3) \)

Mesh boundaries are simulated using scattering equations derived from impedance matching techniques, allowing for a proportional amount of incident energy to be reflected back into the mesh, as defined by the reflection coefficient \( r \), such that the pressure on a single connection boundary node is:

\[
 p_J = (1 + r)p_{J,1}
 \]  
(4)

Equations (1)-(3) can also be derived as an equivalent finite difference scattering algorithm:

\[
 p_J(n) = \frac{2}{N} \sum_{i=1}^{N} p_i(n-1) - p_J(n-2)
 \]  
(5)

This mathematical simplification results in a mesh scattering methodology which is easier to implement and more efficient in terms of memory requirements and speed of computation.

3. THE WAVEGUIDE VOCAL TRACT MODELS

Both 1D and 2D waveguide models representing the vibrating air cavity in the vocal tract between the glottis (vocal folds) and the lips have been constructed. Windows dialog-based software (Figure 3) has been developed to allow the comparison of 1D and 2D performance at equal levels of sophistication so as to highlight the potential benefits available from the increased dimensionality.

Quantitative comparison between the two models are made on the proximity of simulated formant frequencies and their bandwidths to that predicted by natural recorded speech. Although not an exact measure of accuracy owing to the varied nature of speech, this method allows for model parameters to be initialised.

The speech sound producing capability of each of the models depends greatly on accurate data describing the individual geometrical structure of the tract in the creation of each of the vowels. Such information in the form of functions describing the cross sectional area along the tract from x-rays taken of Russian speakers have been used in these simulations [5].
the open lip end $r_{lips}$, the closed glottis end $r_{glottis}$, and the internal fleshy wall reflection $r_{wall}$. Excitation is applied in the form of either noise or the LF glottal waveform, injected on to the mesh along the line of nodes closest to the glottis. Output is then measured as an average of all nodes along the lip boundary.

Figure 5: (a) QWR and (b) The Vocal Tract 2D DWM Models

Visual output from the two models is presented in the form of an OpenGL 3D graphics window on the main dialog panel (Figure 3) illustrating a planar view of the air pressure within the tract as it varies over time. Data output is saved as a .wav file. This allows for continual monitoring of the chain/mesh behaviour at any spatial or temporal point in a simulation.

3.2. Features

Both models can be constructed using the scattering or finite difference methods to allow the study of the various benefits offered by each approach. Mixed modelling, as supported by the use of the KW interface [7], is facilitated in the 1D model; the potential mobility of the 2D model boundaries leads to the application to the higher dimensional system.

In order to increase the quality of the model such that it is closer to natural speech or singing, many additional factors are included. Vibrato, tremolo and noise can be introduced into the input waveform to include a varied, more human element to the excitation. The simulation of constrictions to the air flow and plosive speech like sounds is achieved with both the modulation of the reflection coefficient at the lips end of the model, and a decrease in tract width modelling the tongue.

3.3. Diphthong Simulation

An important factor in the synthetic production of speech is the ability to create diphthongs and triphthongs, sounds formed from a slide between two or three vowels. In a physics based model this can be accomplished by linear interpolation between area functions set within the model. The 1D model allows for a simple update of the impedance values in an allotted time frame (typically about 250ms for a triphthong), whereas implementing this for the 2D system proves less trivial.

The application of the area function to the 2D model as the amount of waveguides across each section of the tract adds much complexity to the idea of a dynamic model. A mesh with moving boundaries requires dynamic node reallocation to accommodate the addition and removal of extra nodes into the mesh and the adjustment of those surrounding each change. Balancing of waveguide pressures around each of the changing nodes will also be necessary to accommodate this alteration to the model. This feature of the 2D model is currently under construction, but it is believed that once complete it will enhance the potential of the multi-dimensional model to create speech sounds of an organic nature.

4. RESULTS

The output from the simulations produced by the software can be used to gain an understanding of the importance of the use of physical modelling in speech synthesis, specifically any benefits that might be gained by moving to a 2D or 3D formulation.

4.1. Vowel Simulation

The simulation of any speech-like sounds relies heavily on the creation of many of the different vowel types obtainable in the real world, each characterised by its unique formant frequency pattern. The 1D vocal tract model has been seen to exhibit accurate formant simulation and hence vowels that approximate the target with some realism [4]. Similarly, more recent research into the 2D mesh model has shown that it produces speech-like output with accurate formant values for certain vowels [8]. Figure 6 shows the spectrum synthesised with the 1D model when creating the vowel in the word ‘boot’. The frequencies of the simulated formants give a good agreement with those predicted by recordings of natural speech [9]. Although there is up to 30% variability in some of the formant frequencies, the results are considered accurate enough due to the variability of the formant frequencies in measured natural speech.

Figure 6: Frequency Response of the ‘boot’ Vowel 1D Model

Figure 7 illustrates the formant pattern produced by the 2D mesh model for the shape of the tract held when creating the vowel in the word ‘boot’. The frequencies of the simulated formants give a good agreement with those of natural speech, with only a 6.4% maximum variability between them.

Figure 7: Frequency Response of the ‘boot’ Vowel 2D Model

Initial results show the frequency peaks for both models to be of a reasonable accuracy and as such both methods produce spectra of an expected shape. With application of the LF glottal waveform to both models vowel sounds are achieved that bear a close resemblance to their targets. The 2D model presents a slight increase in
quality of naturalness over the 1D model, which exhibits a small amount of metallic ringing.

4.2. Formant Bandwidths

The main difference in the two spectra in Figures 6 and 7 is observed as a reduction in bandwidth produced by the 1D model. The ability to control the bandwidths of the formants created will have an important implication on the system’s overall potential to create realistic sounding speech. The bandwidths of the formants produced by both waveguide models are directly influenced by the reflection values set at the boundaries. The extended control offered by the additional boundaries in the 2D model may therefore present greater flexibility. Current research is involved in the optimisation of reflection coefficient values for both the two (1D) and four (2D) boundary systems. Figure 8 shows the variations in bandwidth of the first formant in the spectrum of the vowel in ‘beet’, achieved by changing \( r_{\text{glottis}} \), keeping \( r_{\text{tips}} = 0.6 \). For comparison, 2D formant bandwidth variations achieved by varying \( r_{\text{wall}} \), with higher, more realistic reflection coefficient values \( r_{\text{tips}} = 0.9 \) and \( r_{\text{glottis}} = 0.98 \), are also shown in Figure 9. Preliminary results highlight more clearly defined peaks generated by the 2D model when compared to the 1D equivalent, approaching target bandwidths of between 80 – 100 Hz. Similarly, greater sensitivity towards smaller changes in \( r_{\text{wall}} \) are apparent in the larger range of 2D bandwidths achieved.

![Figure 8: 1D 'beet' Vowel Formant Bandwidth Variations](image)

![Figure 9: 2D 'beet' Vowel Formant Bandwidth Variations](image)

5. CONCLUSIONS

The software presented in this paper is designed to allow a thorough comparison of the 1D and 2D waveguide vocal tract models. Current results suggest that the 2D model can accurately simulate vowel formant frequencies, verifying the methods ability to generate the building blocks of speech. The 2D model also allows for greater control of formant bandwidths, giving enhanced sensitivity for smaller changes in mesh reflection coefficients. The ease with which the 1D model employs a vowel-to-vowel slide, when compared with the complex junction reassignment necessary in the 2D case, may present difficulties in justifying the eventual argument of higher dimensional superiority, but it is hoped that diphthongs of greater realism may be achieved with 2D modelling given its superiority for static vowel sounds.

Further features such as mesh boundaries simulating a more realistic frequency dependent reflection, a lip radiation filter accounting for the open end, and a nasal tract, will increase the potential of the 2D model further. With more comprehensive tract geometry data, an additional consideration might be the inclusion of a third dimension to the model, resulting in a full physically modelled vocal tract. Future testing will involve the use of area functions generated from natural recorded vowels, which can then be re-synthesised with the waveguide models for a direct comparison. The eventual goal is to extend both systems to the sophistication and realism achieved in more developed models, such as the SPASM system, with a view to examining the possible increase in naturalness that may be achieved with the higher dimension model.

6. REFERENCES

ROOM IMPULSE RESPONSE SHAPING FOR ENHANCEMENT OF PERCEIVED SPACIOUSNESS AND AUDITORY DISTANCE

Dae-young Jang
Broadcasting Media Research Group
ETRI, Daejeon, Korea
dyjang@etri.re.kr

Kyeongok Kang
Broadcasting Media Research Group
ETRI, Daejeon, Korea
kokang@etri.re.kr

Han-gil Moon
Applied Acoustic Lab.
Seoul National Univ., Seoul, Korea
fullmoon@acoustics.snu.ac.kr

Hoe-Kyung Jung
Dept. of Computer Engineering
Paichai Univ., Daejeon, Korea
hkjung@pcu.ac.kr

ABSTRACT

Room impulse response is one of most important information with localization of sound source in 3D audio. Actually we can adjust the distance and spaciousness of a sound source with impulse response of the room. Through consequent experiments, we found that C80 and EDT are varying systematically with sound source distances, and these variations are due to early reflection decay curves. This paper contains brief explanation of the two parameters as auditory distance cues, shaping of early reflection decay curves for control of auditory distance, and psychological test results of auditory distance control with early refection decay curve shaping. With these validations, we can confirm early reflection decay curve shape is effective factor for control of perceptual auditory distance and spaciousness in the room.

1. INTRODUCTION

People who research on 3-dimensional sound have mainly performed the 3-dimensional sound localization (elevation and azimuth) of the sound sources for the past tens years. But there have been also little researches for the distance cue of the sound source. Many previous researches have revealed that the conventional distance cues are composed of loudness, spectral information, reverberation and binaural information [1], [2].

Distance cues are classified into relative cues and absolute cues. Relative cues include loudness and spectral information of the source. They have an advantage of easy implementation, but also have a limit of their effect, because they cannot determine the absolute distance without priori information about loudness or spectrum of the sound source. Basically loudness cue is according to the inverse square law. And this cue has very powerful ability to determine changes in the distance of a constant amplitude sound source. Molecular absorption of the air or air absorption filtering on the sound propagation path is the major origin of spectral cue. In fact, over very long distance this cue has a considerable low-pass filtering effect.

On the other hand, absolute cues are consisted of familiarity, binaural information and reverberation. If the listener is efficiently familiar with the sound sources, the relative cues can be used to judge absolute distance. Listener’s familiarity with both the source signals and the acoustic environment is clearly a key factor in any model for auditory distance perception. For near field listening, we can consider IID & ITD as other cues providing not only directional information but also distance information.

Another important distance cue is the relative loudness of reverberation. When sound is produced in a reverberant space, the associated reverberation may often be perceived as a background ambience, separate from the foreground sound. The foreground sound consists largely of the sound that propagates directly from the sound source to the listener, this so-called direct sound decreases in amplitude as the distance to the listener increases. The amplitude of the reverberation, on the other hand, does not decrease considerably with increasing distance. The ratio of the direct to reverberant amplitude is greater with nearby objects than it is with distant objects. Thus, distant objects sound more reverberant than close objects. An example of this relationship used by Wave Arts Acoustic Environment Modeling is diagrammed in Figure 1. The direct sound amplitude drops 6 dB for each doubling of distance. However, the reverberation amplitude drops 3 dB per doubling of distance [3].

Figure 1: An example of direct to reverb energy ratio [3].
Many other researches also reported that reverberation is very important cue for distance, but the actual parameters for controlling perceived distance with reverberation cues are not well defined. But according to our previous research, we found that EDT (Early Decay Time) and C80 are varying systematically with distance [4], and these variations are due to early reflection decay curves.

In this paper, we will show a method of handling and shaping of early reflection decay curves for control of auditory distance, and psychological test results of auditory distance control with early reflection decay curve shaping. With these validations, we can confirm early reflection decay curve shaping is effective for control of perceptual auditory distance and spaciousness of the sound sources.

2. BACKGROUND

Referring to previous research [4], we found that EDT (Early Decay Time) and C80 are varying systematically with distance. Namely, as the distance between source and listener goes farther, C80 more decreased and EDT more increased. Figure 2 and Figure 3 show the experiment results on this phenomenon. On the other hand, reverberation time of the specific room doesn’t vary so much even if the measuring distances from sound source are increased.

Figure 2: C80 versus auditory distance.

Figure 3: EDT versus auditory distance.

To investigate the reason that EDT and C80 are varying systematically with distance, we divided measured impulse responses into early reflection part and late reverberation part. And we normalized early reflection parts in order to compare their decay tendencies of each impulse response.

C80 is the energy ratio of early part and late part and EDT is the reverberation time for the first 10 dB drop. So these two parameters are very sensitive to the shape of the early decay curve. As the distance varies, the early reflection part varies significantly and this change of early reflection makes C80 decrease as the distance increases and vice versa in EDT. So if we can control these two parameters by designing the early parts of impulse responses, we can control the perceived distance with the reverberation cue. So C80 and EDT can be used as the parameters for controlling perceived distance with the reverberation cue.

Figure 4 shows the decay curves of early reflections and rate reverberations for 1m and 16m distance. We can identify the decay curve slopes of early reflections are varying significantly, while the decay curves of rate reverberations have almost the same shape. Figure 5 shows the systematic changes of early reflection decay curve with distances.
3. EARLY DECAY CURVE SHAPING

With previous results (systematic changes of EDT and C80 with distance) and specific measured impulse response, we tried to make simple function to provide distance information about a sound source for listening test. In analyzing measured impulse responses of a specific room, we found that late reverberation parts are maintained constantly whereas early reverberation parts vary systematically with distance. The slope of the early reflection curves of impulse responses measured at the near place of the source is more abrupt than the one at the distant place of the source.

From this investigation, we devised a method to implement situations above mentioned. First of all, we selected a measured 2m-impulse response as a reference impulse response. Then we made 1m-impulse response and 4m-impulse response using this reference impulse response. And we chose first 80ms as the threshold of early reflection part of the impulse response. This method can be divided into two parts. In the first step, we make late reverberation part according to various distance of the source. Namely, we chose late reverberation part of the measured impulse response at 2m in a specific room and used it as the late reverberation part of any other point of the room impulse response. Because late reverberation part of the impulse response has regular pattern with distances as we could see in measured data. Then it makes no difference to insert 2m-late reverberation part to 1m or 4m-late reverberation.

The second step is to make the early reflection part according to the distance change. We adjusted the EDT values with distance by varying the slope of early decay curve. When we adjust EDT, reverberation time (RT) must be fixed. If we consider a straight line with linear slope to measured 2m-impulse response appropriately, we can get some impulse responses that we want. But when we apply a linear straight line to early parts of the impulse response, there can be overemphasized reverberation and reverberation time of the room will not be fixed. Therefore we applied several linear lines of different slope for approximately shaping the curves of different distances. The curve can be represented by a pair of time and power value for the calculation of curves for the early reflection parts:

\[ \{(t_0, e_0), (t_1, e_1), (t_2, e_2), \ldots, (t_n, e_n)\} \]

Theoretically element \( t_0 \) is the propagation time for the path from sound source to receiver. And elements \( e_0, e_n \) can have the values of 2, 0 respectively. Then actual modification value for \( t_0 \) (direct sound) at half distance will be 4 (square of 2) and that for \( t_n \) will be 1 (0 power of 2).

When the position of sound source is changed by user interaction, the propagation time also varied. Thus we have to compensate this changed propagation delay, as insert or delete first samples of impulse response. The propagation time difference \( \Delta t \) for changing the position of sound source can be calculated with following numerical formula:

\[ \Delta t = \frac{(d - \text{ref}_d)}{S}, \]

where \( d \) is changed distance, \( \text{ref}_d \) is reference distance, and \( S \) is the velocity of sound propagation.

Figure 6 shows the way to control the early decay curve. These modified impulse responses are convolved with dry sound sources. The outcomes after convolving with distance are normalized to investigate the only effect of EDT excluding the influence of loudness. We have focused on how important the EDT value is in affecting the auditory distance with other acoustic parameters fixed like reverberation time. By means of outcomes above, we performed subjective listening tests with well-trained trainers to verify our hypothesis, that is, not the late parts but the change of early decay curve affect perceiving the distance of the source.

4. VALIDATION TEST

We chose dry sound sources of tenor solo, soprano solo and orchestra as test materials. Two kinds of subjective listening tests were performed. First, randomized the order of the different impulse responses and paired with two samples. Then let the subjects listen to the pair of samples, which are the same sounds with different distances, and asked them to choose which sample is perceived closer. Secondly, we arranged impulse responses in a distance order, e.g. 2m-4m-8m or 8m-4m-2m pair. Then we also asked the subjects whether they feel the sample becomes closer or distance order, e.g. 2m-4m-8m or 8m-4m-2m pair. Then we also asked the subjects whether they feel the sample becomes close or distant. In the first tests, when a subject finds a correct answer he gets +1 point, otherwise he gets –1 point. And in the second method, if subject finds a correct answer he gets +1 point and when he can’t recognize any change, he gets 0 point. And if he answers incorrectly, he gets –1 point. So a perfect score is +3, and the lowest point is –3. Total 9 subjects participated in the subjective listening tests.

Figure 7 and Figure 8 show the results of subjective listening tests for two test methods respectively.

These graphs also show the total averaged scores of subjective responsiveness regardless of any kind of recording dry sound source. We tried to look into listeners’ ability to perceive auditory depth on the average. So method 1 has a perfect score as +9 and the worst mark as –9, method 2 has a perfect score as +6 and the worst score as –6. Generally listeners had a higher mark in method 2. It shows that people perceive well in case of listening samples along to the distance in serial. Overall we obtained the satisfactory result through subjective listening tests in validating previous results, which explains how EDT affects the distance of the source.
Many previous researches show that reverberation can provide absolute information. But the difficulty of implementation and no effectively defined parameters have prevented the reverberation from being used efficiently. We found EDT and C80 are varying systematically with the distance from sound source to receiver. This systematic change can be explained by examining the divided impulse responses. Because the early reflection part of impulse response varies significantly with distance whereas the late reflection part does not vary much, C80 that is the energy ratio of the early part over the late part of the impulse response decreases as the distance gets farther and vice versa in case of EDT. Both parameters are related to the reverberation of the room. So EDT and C80 can be used as the parameters for controlling perceived distance with the reverberation cue.

With this result and relatively simple method, we functionalized reverberation cue with distance. We made artificial reverberation curve composed of early parts with various slopes and late parts with the one specific slope to imitate impulse responses of different distances. The significance of this paper is to extract effective distance cues and to define EDT or C80 as distance cue maintaining reverberation time constant. To provide distance information of sound source in virtual audio environment, we need some practical algorithm. This paper also shows a good possibility to implement the reverberation parameter with proposed reverberation decay curve, which is composed of the slope of the early part changed and late part unchanged as distance variation.

For future work, we are going to investigate multi-channel impulse responses for multi-channel reproduction environment. And also we are going to make clear the time domain variation of early reflections of impulse response according to various positions of sound source in the room.

6. ACKNOWLEDGEMENTS

This work was supported by the Ministry of Information and Communication (MIC) of Korean government under the title of “Development of Super-intelligent Multimedia Anytime-anywhere Realistic TV (SmarTV) Technology”. We, authors, faithfully thank to the project funding managers.

7. REFERENCES


5. CONCLUSIONS
SEMIAUTOMATIC AMBIANCE GENERATION

P. Cano, L. Fabig, G. Gouyon, M. Koppenberger, A. Loscos†
† Institut Universitari de l’Audiovisual
Universitat Pompeu Fabra, Barcelona, Spain
pcano@iua.upf.es

A. Barbosa‡ ‡ Dept. Som e Imagem
Universidade Católica, Porto, Portugal
abarbosa@porto.ucp.pt

ABSTRACT

Ambiances are background recordings used in audiovisual productions to make listeners feel they are in places like a pub or a farm. Accessing to commercially available atmosphere libraries is a convenient alternative to sending teams to record ambiances yet they limit the creation in different ways. First, they are already mixed, which reduces the flexibility to add, remove individual sounds or change its panning. Secondly, the number of ambient libraries is limited. We propose a semi-automatic system for ambiances generation. The system creates ambiances on demand given text queries by fetching relevant sounds from a large sound effect database and importing them into a sequencer multitrack project. Ambiances of diverse nature can be created easily. Several controls are provided to the users to refine the type of samples and the sound arrangement.

1. INTRODUCTION

The audio component of audiovisual productions has long been regarded as of minor importance. Nevertheless, in the last years and especially after productions such as Apocalypse Now (1979), its importance has been acknowledged. Audio is gaining interest for its evocative and overall immersive experience for the audiences. Audio has an immense power, even when accompanying coarsely-drawn cartoons, for creating the illusion of reality.

Traditionally, from the film production process point of view, sound is broken into a series of layers: dialog, music and sound effects—from now SFX [1]. SFX can be broken further into hard SFX (car doors opening and closing, and other foreground sound material) and foley (sound made by humans, e.g: footsteps) on the one hand, and ambiances on the other hand. Ambiances—also known as atmospheres—are the background recordings which identify scenes aurally. They make the listener really feel like they are in places like an airport, a church, a subway station, or the jungle. Ambiances have two components: The ambient loop, which is a long, streaming, stereo recording, and specific or stingers, which are separate, short elements (e.g: dog barks, car horns, etc) that trigger randomly to break up repetition [2].

Sound engineers need to access sound libraries for their video and film productions, multimedia and audio-visual presentations, web sites, computer games and music. Access to libraries is a convenient alternative to sending a team to record a particular ambiances (consider for instance “a Rain forest” or “a Vesuvian eruption”). However, the approach has some drawbacks:

1. Accessing the right ambiances is not easy due to the information retrieval models, currently based mainly on keyword search [3].

2. The number of libraries is large but limited. Everybody has access to the same content although sound designers can use them as starting point and make them unrecognizable and unique.

3. The ambiances offered by SFX library providers are already mixed. There may be SFX in the mix that the sound engineer does not want in that position of may be does not want at all. It is a hassle to fix it.

In this context, we present a system for the automatic generation of ambiances. In short, the system works as follows: the user specifies his need with a standard textual query, e.g: “farm ambiances”. The ambiance is created on-the-fly combining SFX related to the query. For example, the query “farm ambiance” may return “chicken”, “tractors”, “footsteps on mud” or “cowbells” sound. A subset of retrieved sounds is randomly chosen. After listening to the ambiance, the user may decide to refine the query—e.g: to remove the “cowbells” and add more “chickens”—, ask another random ambiance—with a “shuffle-type” option—or decide that the ambiance is good enough to start working with. The system outputs the individual SFX samples in a multitrack project.

The intended goals of the approach can be summarized as follows:

Enhance creativity: Sound engineers have access to a huge ever-changing variety of ambiances instead of a fix set of ambiances. The combination of individual SFX provides a substantially larger number of ambiances.

Enhance productivity: Engineers can have several possible sonications in a short time.

Enhance flexibility: Having different SFX of the ambiance separately in a MultiTrack gives more flexibility to the ambiance specification process, some sounds—a bird singing in a forest ambiance can be removed or their location in the timeline changed. It also allows for spatialization using 5.1.

Enhance quality: With a very low overhead—basically clicking on a “shuffle” button and adjusting some sliders, sound engineers can obtain several ambiance templates. Hence, the production cycle reduces. The producers can give their feedback faster and their opinions be incorporated earlier in the production improving the overall quality.

2. SYSTEM DESCRIPTION

The system is based on a concept-based SFX search engine developed within the AudioClas project (www.audioclas.org). The objectives of the project were to go beyond current professional SFX provider information retrieval model, based on keyword-matching, mainly through two approaches [4]:

— DAFx’04 Proceedings —
Semantically-enhanced management of SFX using a general ontology, WordNet [5].

Content-based audio technologies which allow automatic generation of perceptual meta data (such as prominent pitch, dynamics, beat, noisiness).

These two approaches are the building blocks of the semi-automotive ambiance generation. Current prototype uses 80,000 sounds from a major on-line SFX provider. Sounds come with textual descriptions which have been disambiguated with the augmented WordNet ontology [3]. WordNet is a lexical database that, unlike standard dictionaries which index terms alphabetically, indexes concepts with relations among them.

\begin{itemize}
  \item \text{thrush} (songbirds characteristically having ...) => oscine, oscine bird
  \item \text{thrush} (brownish upper plumage with a spotted breast)
  \item \text{nightingale} (European songbird noted for its melodious nocturnal song)
  \item \text{nightingale} (Luscinia megarhynchos)
  \item \text{birdcall} (the characteristic sound produced by a bird)
  \item \text{brown} (brownish upper plumage)
  \item \text{call} (song, songbird)
  \item \text{birdsong} (song)
  \item \text{phonetic} (object, physical object)
  \item \text{chordate} (organism, being)
  \item \text{animal} (animate thing)
  \item \text{animate} (object, physical thing)
\end{itemize}

Accordingly, the sound “Thrush And Nightingale Various Calls” becomes labeled with the following set of concepts:

01234719n thrush -- (songbirds having brownish upper plumage with a spotted breast)
01237641n nightingale, Luscinia megarhynchos -- (European songbird noted for its melodious nocturnal song)
05680983n birdcall, call, birdsong, song -- (the characteristic sound produced by a bird)

The numbers before the definitions correspond to the unique identifiers, offsets, of the concepts, or synonym sets, synsets as referred in the WordNet literature [5].

There are two main functional blocks in the system. The first one retrieves the relevant sounds of the SFX Database and a second one organizes the sounds in a multitrack according to some heuristic rules (see Figure 1).

3. SOUND SELECTION AND RETRIEVAL

The first step has been mining ambiance sounds to learn the type of sources used. We use a database of SFX that has been labeled with concepts rather than with words (see [3] for details). We are therefore able to study the co-occurrence of concepts in sounds. For example, the ambiance “Farm Ambiance Of Rooster And Hen With Wagtail In Background” has been converted to:

01466271n hen, biddy -- (adult female chicken)
01260115n wagtail -- (Old World bird having a very long tail that jerks up and down as it walks)
02893350n farm -- (workplace consisting of farm buildings and cultivated land as a unit)

By mining this information we learn that farm is related to the concept hen and the concept wagtail. Moreover, there are relations encoded in WordNet, which knows that hen and chicken are related. Whenever a user asks for farm sounds we can retrieve a set of sounds where farm appears. Besides we can also search for the sounds of the related concepts, such as chicken. A random subset of the relevant sounds is forwarded to the subsequent block, the sound sequencing.

4. SOUND SEQUENCING

Besides the definition and selection of the suitable SFX, a significant part of the work of the sound designer is setting up parameters and time lines in a multitrack project, such as volumes or panoramic envelopes. This Section details some of the rules used to mix all fetched tracks and compose the synthetic atmosphere. The SFX retrieval module returns mono and dry (no effect has been applied) tracks. Whenever available, the module differentiates between two types of tracks: long ambient tracks and several short isolated effects. One long track is selected to serve as a ambient loop on which the short sounds, or specifics, are added. With such picture of the workspace we hint some rules on how to place the tracks in the mix, how to adjust channel controls (gain, panning and equalization), and which effects (echo, reverb) can be applied to each track.

The systems automatically distributes the tracks along the mix, placing first the ambient loop and inserting sequentially the specifics, with a probabilistic criterion. This probabilistic criterion is based on the inverse of a frame-based energy computation. This means that the more energetic regions of the mix will have less probability to receive the following effect track. This process is depicted in Figure 2.

It is a cunning feature to keep a certain degree of randomness. Again, a shuffle button can remix the atmosphere as many times as desired. Also, further models can take into account other parameters such as energy variation (in order to avoid two transients happening at the same time), such as spectrum centroid (in order to avoid as much as possible the frequency content overlap), or others.

Another important feature is the automatic adjustment of channel controls: gain, panning and equalization. Regarding the levels, these are set so that the track maximum levels are 3 dB above the long ambient mean level and that no saturation / clipping problems appear. Regarding the stereo panning the ambient sound is centered and the isolated tracks are panned one left one right along time in order to minimize time overlap. The amount of panning depends on how close are two consecutive tracks, the closer, the more panned. Equalizing is only applied to those tracks that overlap significantly in frequency domain with the adjacent tracks or with the ambient loop sound. In these cases the effect track is 6-band equalized to flatten down to -12 dB the overlapping frequency region.

Finally, the strategy for the automation of the effects we propose is based on rules. These rules are mainly related with the context of the ambiance. Say we are reconstructing an office atmosphere, we will apply a medium room reverb to whatever effect track we drop to the mix; if we are reconstructing a mountain atmosphere, we can apply some echo to the tracks.
4.1. Integration in professional environments

The advent of high quality audio and spatialization surround setups (e.g., 5.1), first in the film industry, and more recently in home entertainment with DVD, offers the possibilities to create more engaging and immersive ambient sound. It is now possible to have ambient loops that take advantage of very low pitch sound (using subwoofers). It is possible to simulate movement in a three-dimensional space or specific sound elements that pan in every direction we wish. On the other hand the complexity of setting up a multitrack project for a surround scenario increased a lot. It would be extremely useful for a sound designer to specify at a higher level which surround characteristics are desirable for the ambiance, so that the system can provide him a multitrack project file, and respective sound files, already configured to be integrated in his main project.

5. EXAMPLES

Let us now give critical comments on some typical examples on ambiance generation:

Some of the ambiances created had too many events in it. The “jungle” ambiance had plenty of tropical birds, elephants and monkeys and sounded more like a zoo than a jungle.

Some of the ambiances need greater detail in the specification. A “war” ambiance query returned war sounds of different epochs, e.g: bombs, machine guns, swords and laser guns.

The sex ambiance retrieved sounds produced by too many people to be realistic.

These experiences lead us to the conception of a refinement control to add/remove specific sound classes or another control for the density of specifics.
As a multitrack application, we have used the free editor Audacity\(^3\). In addition to common sound editing functionalities, Audacity allows to mix several tracks together and apply effects to tracks. Audacity allows to save multitrack sessions yet it does not read sessions created by external programs. We have therefore tweaked the application in order to load our automatically generated ambiance multitrack sessions.

6. CONCLUSIONS AND FUTURE WORK

We have presented a system for semi-automatic ambiance generation. The ambiances generated by textual query can be further refined by the user. The user controls the number of sounds that should be returned and can add and remove types of sounds, e.g. “more penguin sounds”. Furthermore the ambiance is delivered to the user as a multitrack project, providing thus flexibility to fine tune the results. We plan to extend this work to semi-automatic sonifications of audiovisual productions given scripts (or briefings) and some information of the timing.

7. ACKNOWLEDGMENTS

We thank the staff from the Tape Gallery for all the support, discussion and feedback.

This work is partially funded by the AUDIOCLAS Project E! 2668 Eureka (http://www.audioclas.org). We thank the collaboration from Sylvain Le Groux, Julien Ricard and Nicolas Wack. We thank the review and feedback from Perfecto Herrera and José Lozano.

8. REFERENCES


\(^3\)http://audacity.sourceforge.net/
MATCONCAT: AN APPLICATION FOR EXPLORING CONCATENATIVE SOUND SYNTHESIS USING MATLAB

Bob L. Sturm
Electrical and Computer Engineering
Graduate Media Arts & Technology Program (MAT)
University of California, Santa Barbara, USA
b.sturm@mat.ucsb.edu

ABSTRACT
The author has developed an application in MATLAB implementing concatenative sound synthesis (CSS) using feature matching. CSS is a process of combining short pieces of recorded sound to construct new sonic forms. Historically, CSS was developed for text-to-speech synthesis, but recently it has been explored as a musical sound synthesis method. The results have been called ‘musics,’ the sonic analogue to mosaics made from small pieces of colored tile. Though this MATLAB application is less sophisticated than other audio mosaic algorithms, it is meant to be a free and open application for demonstrating and experimenting with the process. The author has used this application to create many interesting and entertaining sound examples. It has also been used to create several electroacoustic compositions. The application, and all of the sound examples presented here, can be downloaded for free from http://www.mat.ucsb.edu/~b.sturm.

1. INTRODUCTION
A mosaic is a picture assembled by smaller pieces that contribute to the overall perception of an image. Close up the picture is not clear, but further away an image emerges. Figure 1(a) shows a mosaic assembled by hundreds of photographs, Figure 1(b), instead of colored tiles [1]. This process, called ‘photo-mosaicing,’ selects picture-tiles that are most similar to portions of the original image.

![Figure 1: A Photomosaic.](image)

A method similar to photo-mosaicing exists in the synthesis of speech, called ‘concatenative speech synthesis’ [2]. This technique, developed in the early sixties, is used mostly for text-to-speech synthesis. A computer segments written text into elementary spoken units that are synthesized using a large database of sampled speech sounds, like “ae”, “oo”, “sh”. These components are pieced together to obtain a synthesis of the text.

These methods have recently been applied to creating “audio mosaics,” or “musics” ([3], [4], [5], [6], [7]). As in photo-mosaicing, a ‘target’ sound is approximated by samples from a ‘corpus.’ Schwarz [6] uses intelligent segmentation of the sounds by demarcating notes, or analyzing with a MIDI score. A deeper analysis is made by subdividing the segments into attack, sustain, and release portions. For each analyzed ‘unit’ Schwarz calculates a feature vector using several parameters, including mean values, normalized spectra, and unit duration. These units are then used to synthesize a target that is specified by either a symbolic score (MIDI) or audio score (sound-file). The units are selected based on their ‘cost,’ or perceptual similarity, to the original unit. Minimizing this cost results in the best synthesis possible using the database.

So far creative application of concatenative sound synthesis (CSS) is minimal, and software for exploring it is not available. The author thus decided to create an application to explore this technique. MATConcat is an implementation of CSS using feature matching in MATLAB. With this program a sound or composition can be concatenatively synthesized from audio segments in a database of any size. CSS provides many interesting and unique possibilities for sound design and electroacoustic composition. MATConcat has been used to create several intriguing sound examples, as well as some electroacoustic compositions. These demonstrate the potential of this technique for sound synthesis.

2. MATCONCAT
The algorithm used in MATConcat, Figure 2, is much more simple than in [4], [5] or [7]. Instead of segmenting the audio using an auxiliary score, or attempting to determine the content of a unit, the analysis produces feature vectors for ‘frames’ taken by sliding a user-specified window across the audio by a constant hop-size. A six-element feature vector is created for each frame of the sound. Table 1 shows the current dimensions of the feature vector and interpretations of each component.

The analysis database of sound used for the synthesis is called the corpus, which can be several seconds to hours long. The analyzed sound being approximated is called the target. Iterating through the frames of the target analysis, optimal matches are found in the corpus database using specified matching parameters and thresholds. For instance, in the screenshot of MATConcat, Figure 3, the user has specified in the bottom-middle pane to first find all corpus frames that have a spectral centroid within ±10% of each target analysis frame; and from these matches pick the corpus frame that is within ±5% of the target analysis frame RMS.
There are currently six synthesis options. Specifying the features when short frames are suddenly expanded to reveal longer phrases. One can also specify to reverse the corpus samples, or convolve the target and corpus frames.

\[ \text{Target} \]  
\[ \text{Corpus} \]  
\[ \text{Extract Features} \]  
\[ \text{Target Feature Vectors} \]  
\[ \text{Matching Criteria} \]  
\[ \text{MATCH} \]  
\[ \text{Corpus Feature Vectors} \]  
\[ \text{Synthesis} \]  

![Figure 2: Algorithm of MATConcat.](image)

### Table 1: Current Feature Vector Dimensions.

<table>
<thead>
<tr>
<th>Feature Measure</th>
<th>Meaning of Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Crossings</td>
<td>General pitchness</td>
</tr>
<tr>
<td>RMS</td>
<td>Mean acoustic energy</td>
</tr>
<tr>
<td>Spectral Centroid</td>
<td>Mean frequency of spectral energy distribution</td>
</tr>
<tr>
<td>Spectral Drop-off</td>
<td>Frequency below which 85% of energy exists</td>
</tr>
<tr>
<td>Harmonicity</td>
<td>Deviation from harmonic spectrum</td>
</tr>
<tr>
<td>Pitch</td>
<td>Estimate of fundamental frequency</td>
</tr>
</tbody>
</table>

The user can specify any number of features to match in any order; but as the number of features increases, the probability of finding matching frames decreases unless the corpus grows in size. Once the best matches are found, a frame is either selected at random from these or the most optimal frame is chosen (an option specified by the user). The matching audio frame is then accessed from the corpus sound-file and written into the target synthesis according to the settings given in ‘synthesis parameters,’ e.g. window shape, size, and skip.

It is not necessary to keep the window or hop-sizes the same for the analysis and synthesis. One can specify a short hop-size for the target analysis and synthesize it with a larger hop-size. This will obviously make the synthesis longer than the original. For instance, in Figure 5, the panels at the top-left show information about the analysis databases. Note that the target was analyzed using a window size of 512 and window skip of 256 samples (512, 256). The corpus was analyzed with resolution (16384, 1024). If the synthesis uses a hop-size of 1024, its total duration will be four times that of the target.

Once the synthesis process has finished, MATConcat displays the synthesized sound in the upper-right corner and the matching process output in the lower-right corner. As can be seen, in frame 10 the number of corpus frames matching the spectral centroid criteria is 39; and from this the number of frames satisfying the RMS threshold is only 1. If no match is found then the frame is either left blank, a best match is forced, or the previous match is extended to fill the gap—depending on specified synthesis options.

There are currently six synthesis options. Specifying the ‘Force Match’ option finds the next best match if none is found within the given thresholds. If many matches are found, the default action is to select one closest to the original; this can be overridden by selecting ‘Random Match.’ ‘Force RMS’ will normalize the match to the RMS of the target frame. In this way one can preserve the amplitude envelope of the target while satisfying other matching criteria. If no match is found, this can either be left blank, or when the ‘Extend Matches’ is selected, the last successful match will be extended to fill the gap. This creates interesting moments when short frames are suddenly expanded to reveal longer phrases. One can also specify to reverse the corpus samples, or convolve the target and corpus frames.

### 3. EXAMPLES

Several intriguing sound examples have been created using MATConcat. The dramatic percussion crescendos from Gustav Mahler’s second symphony have been synthesized using corpora of monkey and animal sound effects, a Muslim Imam chanting the Koran, an hour of vocal music by John Cage, three hours of nostalgic Lawrence Welk, and all four string quartets of Arnold Schoenberg.

The example using the monkey vocalizations, shown in Figure 3, is particularly amazing. In this example the RMS and spectral rolloff components are matched to within ±5% and ±10%, respectively. The slowly building crescendo is ‘aped’ by the monkeys, creating a sense of increasing hysteria. At the climax the dominant gorillas grunt as lesser monkeys cower in submission. Synthesizing the same target using the same matching criteria but from a corpus of John Cage’s vocal music, creates an entirely different experience. The impressions of Mahler’s crescendi remain however.

A recording of American President George W. Bush has been synthesized by corpora of monkeys, alto saxophone, and Lawrence Welk, and Bach’s Partita for flute. By choosing the right window parameters the speech can still be understood—perhaps though only after hearing the original. When specifying a suitably small spectral centroid and rolloff, much of the sibilance and breathiness remains, especially when using the saxophone and flute corpora.

Specifying a target that is polyphonic understandably leads to trouble. The beginning of Schoenberg’s fourth string quartet pro-
vides an interesting example. A solo viola plays the main theme, punctuated by the other players. The first eight seconds of this piece were concatenatively synthesized using alto saxophone within a pitch threshold of $\pm 1\%$. The original time series and sonogram, along with the sonogram of the synthesis, are shown in Figure 5. Only at the times 0 – 1, 3, and 4.2 – 5 seconds does there appear to be any success. Auditioning the result confirms this observation; the melody is very discontinuous, but can be heard with effort.

It is quite easy for the human listener to hear the melody as continuous in the original passage; but for the machine this task becomes impossible without expert knowledge, i.e. gestalt principles [9], timbre recognition, score following, etc. In the synthesis, the moments at which the theme becomes clear are those in which the viola is the only instrument heard. All other mo-

![Figure 3: Screenshot of MATConcat.](image)

![Figure 4: Mahler’s crescendi performed by London Symphony Orchestra (Gilbert Kaplan, cond.) (top), and performed by ensemble of Monkeys (bottom).](image)

![Figure 5: Beginning of Schoenberg’s fourth string quartet performed by the Arditti String Quartet (top and middle), approximated by Anthony Braxton on alto saxophone, matching pitch $\pm 1\%$ (bottom).](image)
ments are squeaks and squawks attempting to accommodate the transients created by the accompanying strings playing staccato. A score following technique, such as that implemented by Schwarz [5], would probably work best for polyphonic targets.

Using CSS and MATConcat, two multi-movement electroacoustic compositions have been written by the author. The incredible amount of work done by composer John Oswald in his “Plunderphonics” pieces [9], where he combines by hand short samples of sound [10], cannot be reproduced so easily. CSS however leads to other interesting compositional possibilities, which can be explored quite rapidly with MATConcat.

3.1. Dedication to George Crumb, American Composer

At a composition master class given by the composer George Crumb a student asked about the influence of world music on his composition. Crumb related a story about how he collected recordings of musical traditions all around the world. Someone specifically asked about American Indian music and he stated he had never heard it.

For this stereo composition, a recording of a short movement of Crumb’s was used as the target. It is recomposed into three movements, each using a different corpus of recorded American Indian music: a Navajo man and woman singing (45 minutes), three pieces for end-blown flute (5 minutes), and group dances of different tribes (53 minutes). The target and corpora were analyzed at several different resolutions to produce multiple sound files, which were then arranged to form each movement.

3.2. Gates of Heaven and Hell: Concatenative Variations of a Passage by Mahler

The dramatic percussion crescendi in the final movement of Gustav Mahler’s second symphony, ([11], measures 191–193, Figure 9) is said to signify the gates of hell opening. These short variations (1–3 minutes in duration) explore this brief passage, using five versions by different conductors. Each variation uses a target or corpus created from one or several of these renditions. The targets are sometimes the unmodified or even reversed originals. But to create more complex forms than the crescendo and decrescendo, the renditions themselves are chopped up and rearranged.

All movements explore the possibilities of CSS, and its application to composing variations of a theme. What is very unique about most of the movements is that they do not sound electronic or tampered with, and none are really recognizable. The Bach Partita for solo flute used in one variation is completely dissolved, but its acoustic essence remains. This poses interesting questions for the legality of such timbral appropriation.

4. CONCLUSION

Through the sound examples and compositions created, MATConcat demonstrates that this relatively simple implementation of CSS, compared with machine listening and score following, creates effective and intriguing sound and music material. MATConcat serves well as a massive sample-mill, grinding sound into minuscule pieces for reconstitution into familiar forms. Surely with machine listening and score analysis, other interesting possibilities will emerge; but currently this implementation of CSS is far from being exhausted.

In a sense, the version of CSS implemented by MATConcat is just granular synthesis [12] with grains selected from sample data by matching features. Thinking of the algorithm in this way leads to interesting ideas for extensions: parameter envelopes, variable window-sizes and grain delay, pitch-synchronous and asynchronous synthesis, grain spatialization, etc. For instance, one could specify strict thresholds and gradually relax them, or suddenly change them. One could also specify fades between any number of corpora. These will be explored in future work.

Many improvements will be made to MATConcat, especially increasing the dimensions of the feature vector, and expanding the list of synthesis options. There are many more feature measures than the six currently implemented; and their use will serve to further characterize the frames. Future work will implement the features of the MPEG-7 audio framework standard [13]. These extensions will further open up the interesting avenues for creative concatenative composition. MATConcat, and the sounds and compositions mentioned above, are available for free at

http://www.mat.ucsb.edu/b.sturm

5. REFERENCES

SONIFICATION OF THE FISSION MODEL AS AN EVENT GENERATION SYSTEM

Sinan Bokesoy
CICM, University of Paris VIII, Paris, France
bokesoy.sinan@wanadoo.fr

ABSTRACT
I am proposing an event generation system for sonification purposes, where a simplified chain reaction model known as nuclear fission in physics is used. The basic background of the fission model, mapping of parameters as sonic entities and technical aspects of the realization procedure are presented.

1. INTRODUCTION
1.1. Sonification of event generation systems
Algorithmic composition and sound synthesis in computer music has intensively employed mathematical models. These models have been introduced in order to generate several types of data streams for compositional and synthesis control sources. In particular, micro-sound generation methods in computer music need to deliver immense data flow in order to build macro-structures.

The control sources of data flow can be stochastic laws [1], chaotic functions [2], Cellular Automata models [3], genetic algorithms [4][5], terrain data, physical models and nature events [6] etc., which could create dynamic event generation systems from deterministic to non-deterministic and linear to non-linear results. The mapping could be applied to both the time and frequency representation of sound. The granular distribution in time domain gives us the micro-sound composition tool, termed granular synthesis, which has been inspired by the quantum representation of sound in physics devised by D. Gabor in 1947 [7]. The additive synthesis, which operates in the frequency domain, can benefit from different mapping of sources according to the needs of intensive data flow for modification of each partial in sonic spectrum [8].

Looking to the past, we see that the idea of distributing sonic events in sound space was first realized by Iannis Xenakis, starting with his work “Achorripsis” (1957) and then followed by his “ST” series of compositions [1]. He proposed the utilization of probability functions and invented the stochastic composition. “Stochos” is an example application, which permits us to attain the micro-sound time level event generation and in addition provides multiple control sources working in parallel in order to manipulate the sonic parameters on any event time level, by using the stochastic synthesis mapping [9].

The granular synthesis pioneers C. Roads [10][11] and B. Truax [12] have created event generation systems where the application assigns specific features for each sonic event. C. Roads “Cloud Generator” (CG) software is an example of how the user can control the event distribution process which creates clouds of grains, filling up a sound space changing in time.

We consider each event in an event generation system as mapped as a sonic entity or a sonic vector dimension in sound space. The sonic vector should at least have time information about onset and duration of the following event for an accurate definition in time-space. Information like pitch, intensity, spectral or spatial data could be assigned in order to define further dimensions of this vector. With the sonification process we can further modify the data flow or mapping structure for compositional purposes.

1.2. About the fission model
Fission was discovered in 1934 when Enrico Fermi irradiated uranium with neutrons and believed he had produced the first transuranic element [13][14]. In a nuclear reaction laboratory experiment, a beam of particles of type x is incident on a target containing nuclei of type X. After the reaction, an outgoing particle y is observed in the laboratory, leaving a residual nucleus Y. We write the reaction as

\[ x + X \rightarrow y + Y. \] (1)

Mostly, the heavy residual nucleus Y loses all its kinetic energy (by collisions with other atoms) and therefore stops within the target. Since a nuclear reaction takes place under the influence only of internal forces between the projectile and target, we expect the reaction to conserve energy, linear momentum, and angular momentum. In nuclear reaction experiments, usually two basic properties of the particle y; its energy, and its probability to emerge at a certain angle with certain energy is measured. The latter implies also a reaction probability, which leads to different energy states having different probabilities. Furthermore, the compound nucleus after the impact decays to \( y + Y \) in different ways, which is also based on purely statistical considerations. Within this stochastic nature, we can interpret this process as a dynamic system.

In the process of fission, a heavy nucleus, such as that of uranium, splits into two lighter nuclei. It is expected that most of the fission energy is transferred to the fragments. In fact, 80 percent of the energy released in fission does appear as the kinetic energy of the fragments and the remaining 20 percent appears as decay products and kinetic energy of neutrons emitted in the fission process. Each neutron can initiate another fission process, resulting in the emission of still more neutrons, followed by more fissions. A typical fission nuclear reaction is:

\[ ^{235}\text{U}_{143} + n \longrightarrow ^{93}\text{Rb}_{56} + ^{141}\text{Ce}_{86} + 2^n. \] (2)
We can assume each neutron life as an event and, therefore, we can extract additional parameters for the mapping system. It is assumed that the velocities of the nuclear particles are sufficiently so that one can use the non-relativistic kinematics approximation.

We consider a projectile \( x \) moving with momentum \( P_x \) and kinetic energy \( K_x \). The target is at rest and the reaction products have momentum \( P_y \) and \( P_z \). The particles \( y \) and \( Y \) are emitted at angles \( \theta_y \) and \( \theta_Y \) with respect to the direction of the incident beam. Figure 1 illustrates this reaction. We assume that the resultant nucleus \( Y \) is not observed. Assuming that \( X \) is initially at rest, we have:

\[
\text{initial energy} = \text{final energy}
\]

In a real fission reaction, the average number of neutrons produced is greater than one, making the chain reaction possible. The two neutrons emitted in the fission process are prompt neutrons. They are emitted at the instant of fission. About 1 percent of the neutrons in the fission process are delayed neutrons emitted following the decays of the heavy fragments.

The reaction speed is about \( 10^{20} \) s. In reality, the space between the uranium atoms is huge compared to the size of the colliding particles, so that there must be sufficient conditions for the reaction process not to die out quickly. It is very likely that the neutrons do not hit any target.

For our model we make the following considerations. At first we restrict the conditions of the system for the purpose of having a basic prototype. We assume that every reaction creates two new particles, which is the case for a chain reaction. Then we watch for their collision time, location and speed vectors. For the time being, we neglect the effect of the delayed neutrons, which occur after a very long time through the decay products compared to the fission reaction time.

### 1.3. The rules of the model

For our sonification model we created a 3D space filled with the target atoms. In this experiment, we choose a size in three dimensions of \( 400 \times 400 \times 400 \). For making the calculation process much easier, we locate all the target particles on the integer coordinates, so that we have 64 million atoms resting in our space.

We define the collision rules such as there will be two new particles created with every collision. Each of them will have a speed vector \( v'_p \). Using this information, we can calculate when and where their collision with the next target will occur. If they do not hit any target, they might go out of our space and feedback from the opposite site of their point of leaving. We are free to set these rules, since this is a virtual model. Every atom after the reaction becomes a by-product, which cannot be part of another reaction.

In summary, we need to know the following parameters:

- The collision coordinates \( C_x, C_y, C_z \)
- The 3D speed vectors of the 2 new particles after the collision \( V_{p1}, V_{p2} \)
- The event life \( t_1 \) and \( t_2 \) as the two neutrons life durations.

Having set these parameters, we can apply a mapping for the sonification of this system. We could simply assign an oscillator loaded with the mapping parameters for each event for this task.

### 2. SONIFICATION PROCESS

The realization of this model and the sonification process has been implemented in Csound. The source file can be found at

http://perso.wanadoo.fr/sinan.bokesoy/fission.csd

The following considerations apply:

**a)** The initial parameters are assigned to their initial values. We inject the system with two initial neutrons at a specific location \( (px, py, pz) \) and also two speed vectors \( (vx, vy, vz) \) and \( (vx2, vy2, vz2) \). Each time the program calculates the collision parameters; the new particles call up again the same collision calculation block, while updating the distribution of particles in space.

**b)** Using integer numbers for the coordinates in this model brings enormous ease and speed for the calculations. There we compromise some features like preservation of momentum in the reaction kinematics. For each new particle, a randomly calculated velocity value is assigned in the range \( (0-5) \). The resolution of the numbers is one digit after the decimal point: 1.4, -0.2 or 0.6, for example. Therefore, if we know the departure point and the velocity of the new particle, we can immediately calculate its collision point with atoms. The program routine looks after whether there exists a target or not at the calculated point.

**c)** After each collision, the target at the collision point will be cleared in the 3D space and, namely, its corresponding table in the program. This happens by calculating the index value of the target coordinate to reach the correct array address in the table, which should address 64 million points. The table size limit in Csound is 16777217, therefore we use multiple tables to link them as one large table.
d) Also the coordinates of each collision $xc$, $yc$, $zc$ and $xc2$, $yc2$, $zc2$, their velocities $vx$, $vy$, $vz$ and $vx2$, $vy2$, $vz2$, the particle life durations $time1$ and $time2$, and the collision onset times are written in their corresponding tables.

e) A recursive routine updates the tables and looks for new collisions and generates the information. At the end we obtain 7 parameters ready to map for each event.

f) Each event is calling the instrument definition in Csound with using the ‘event’ opcode. Besides the onset time, which is the collision time, the neutron life duration $time1$ and $time2$ accurately define these events in time space.

<table>
<thead>
<tr>
<th>$xc$</th>
<th>$yc$</th>
<th>$zc$</th>
<th>$vx$</th>
<th>$vy$</th>
<th>$vz$</th>
<th>$time1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f1$</td>
<td>$f2$</td>
<td>$f3$</td>
<td>$g1$</td>
<td>$g2$</td>
<td>$g3$</td>
<td>$dur$</td>
</tr>
</tbody>
</table>

Table 3: Parameter mapping

Figure 4: Visualization of collisions data in two different experiments

Figure 5: Histograms for the onset time distributions of the experiment

Table 3 shows how to map the collision coordinates and the neutron speed components to frequency and intensity parameters of 3 oscillators. Their duration is determined by neutron life ($time1$). One could use another set of oscillators and assign $time2$ as their durations and distribute them with the collision coordinate mapping in the stereo field.

The scaling of source and destination parameters in the mapping can be adjusted at will. Certain scale factors help us understand the evolution of the process more clearly.

Figure 4 shows the collision data of two experiments. The rules for the speed vector calculations are different. The first experiment, plotted on the left, shows a distribution of collisions where the speed vectors are assigned randomly for the particles. The second experiment, plotted on the right, shows a distribution with the velocity vector of the first particle fixed to 1, 1, 1 and of the second particle to -1, -1, -1 always. These experiments show the extreme conditions of between disorder and order of this dynamic system. Figure 5 shows us the histogram data of the onset time distributions for the experiments above.
The parameters supplied as tables could also be fed to a 3D modeling software in order to visualize the dynamic system in 3D environment and gain better understanding of the motion of the particle system. This basic model was a simplified version of a fission model and did not use the following known features of the fission model:

- Considerations on preserving energy, angular and linear momentum in the reaction
- Delayed neutrons produced by the by-products
- The prompt neutron number was fixed as two, but could be changed
- The distribution of resting atoms were fairly regular and just on the integer coordinates

The last two conditions were introduced in order to increase the speed of calculation, which is not in real-time.

### 3. CONCLUSIONS

We tried to bring up a simplified model of nuclear fission for sonification purposes and explained an example mapping process. The chain reaction creates event data and has been mapped with Csound as instrument parameters.

The sound examples that can be found at [http://perso.wanadoo.fr/sinan.bokesoy/ex1.mp3](http://perso.wanadoo.fr/sinan.bokesoy/ex1.mp3) and at [http://perso.wanadoo.fr/sinan.bokesoy/ex2.mp3](http://perso.wanadoo.fr/sinan.bokesoy/ex2.mp3) give a quick idea about the onset time distribution in the chain reaction. Since the collision coordinate of one axis was mapped to the pitch, the change of the granular pitch gives an idea about the distribution of the collisions.

Many different mapping trials can be performed with the rest of the parameters. I found that the model produces strong realistic images of explosions or thunder events. It would be useful for future expansions to use the data output of the model for convolution implementations [15]. In this way, the model can be employed to process other sounds, which gives a digital audio effect quality. What we have done here, as first step, is to interpret the particles of this fundamental reaction as acoustical quanta. Controlling acoustical quanta makes it necessary the use of automated systems with high-level control features such as the presented sonification model [1]. We extracted the macro parameters of control from the fission model here, which are derived from elementary laws of nature.

A system capable of responding to the input parameters in real-time is the goal; triggering multiple collisions at the same time on different coordinate would also be very interesting. Only then we could consider the model as a sonic instrument and as a compositional tool or a real-time effect system, which has the ability to interact with the real-time input parameters and feedback processes. In order to achieve this, other programming environments than Csound, like the new Java implementation of Max/MSP, will be considered as the next step.

### 4. ACKNOWLEDGEMENTS

I would like to thank to Ibrahim Semiz at the Department of Physics of the Bosphorus University Istanbul for the information provided about nuclear reactions.

### 5. REFERENCES

MEMORY REDUCTION TECHNIQUE OF SPREADING FUNCTION IN MPEG AAC ENCODER

Shih-Way Huang, Liang-Gee Chen

DSP/IC Design Lab, Grad. Inst. of Elect. Eng.,
Dept. of Electrical Engineering
National Taiwan University, Taipei, Taiwan, ROC
shihway@video.ee.ntu.edu.tw

Tsung-Han Tsai

Dept. of Electrical Engineering
National Central University,
Chung-Li, Taiwan, ROC
han@ee.ncu.edu.tw

ABSTRACT

In order to reduce the computational complexity of an MPEG AAC encoder, calculation of spreading function, which is the critical algorithm in the psychoacoustic model, can be replaced with look-up tables. However, the memory size of the table is considerably large in DSP or hardware implementation for portable devices. The paper deals with the methods to reduce the memory size. We modified a method originally used in an MP3 software encoder and adopted it in an MPEG AAC encoder. The result shows that only about one-third of the original size is required in programmable processor implementation such as DSP. Moreover, for hardware implementation, we analyzed the wordlength and reduced the size to only 3.6% compared to that for a 16-bit DSP while maintaining audio quality.

1. INTRODUCTION

Portable electronic devices such as smart mobile phones, digital cameras, PDAs, USB drives, and digital audio devices with audio playback and audio recording have been attractive because of both the prevalence of MP3 audio files and the development of flash memory cards. The MPEG AAC encoder, so-called MP3’s successor, is to be embedded into these portable devices because of its higher coding efficiency. Therefore, low power and low cost (area) are important considerations in DSP and hardware implementations. In the previous work, Tsai et al. proposed that the calculation of spreading function can be replaced with a look-up table to reduce the computational complexity dramatically. However, the size of the table is considerably large in the implementation. For example, Gayer et al. showed that about 10000 words of data RAM and 7000 words of data ROM are required for a typical AAC LC encoder on DSP. On the other hand, the memory size of spreading function requires 6614 words (at sampling rate 44100 Hz) without optimization. Therefore, it is important to reduce the memory size of spreading function.

The paper proposes methods to reduce the memory size of spreading function. According to the characteristics of the table, we utilized two methods to reduce the size of the table. Referring to an MP3 software encoder, we employed two arrays to store the values. Only about one-third of the original size is required. Besides, for dedicated hardware implementation, where area can be further reduced with word length analyses of the table values, the memory size was reduced to only 3.6% compared to that for a 16-bit DSP.

2. SPREADING FUNCTION IN PSYCHO-ACOUSTIC MODEL

In psychoacoustics, masking is an effect that one sound is made inaudible because of the presence of another sound. Since the simultaneous masking effects are not band-limited within the boundaries of a single band, inter-band masking also occurs. A masker centered within one critical band has effects across the other bands. The effect, predictable on detection thresholds in other critical bands, is known as the spread of masking and often modelled in coding applications by an approximately triangular spreading function. A spreading function is adopted in ISO/IEC MPEG Psychoacoustic Model 2, which is employed in MP3 and AAC.

Figure illustrates the spreading function of psychoacoustic model (PAM) in an MPEG AAC encoder. The above of the figure is a block diagram of MPEG AAC encoder, and the below are the 13 steps of PAM according to MPEG AAC. In step 5, both partitioned energy and unpredictability are convolved with the spreading function in order to estimate the effects across the partitioned bands.

Spreading function, shown in Figure, is composed of a series of functions, including some complex operations such as square roots, divisions, exponential operations, etc, which increase the complexity. Tsai et al. proposed that the repeated calculation of spreading function is dominant in PAM, and it can be replaced with a look-up-table memory because it is only dependent on the sampling rates and the block type used. Table shows the reduction rate of computational complexity when the calculation of spreading function is replaced with a look-up table. The reduction rate is up to 36% for an MPEG AAC encoder. However, a problem is left—the size of the look-up table is considerably large. For example, if the memory for sampling rate 44100 Hz (CD-quality) is required, a two-dimension table due to convolution in step 5 is calculated as follows:
reduced with word length analyses of the table. Therefore, it is important to reduce the table size and its according access bandwidth in an efficient way.

### 3. ANALYSES OF SPREADING FUNCTION TABLE

In order to minimize the size of the two-dimension table, we pre-calculated all the values (outputs) of the spreading function and then analyzed them. We found that the table was a sparse table with the following characteristics. Figure 3 illustrates a table at sampling rates 44100 Hz and LONG block type. The size is 70 x 70 as the square number of partitioned bands. The table is characterized as follows:

- Most of the values are exactly zero. This is because the function (Figure 2) returns zero when the condition tmpy < -100 is true, which is often satisfied.
- The distribution of the zero values is regular in both the upper triangular parts and lower ones.
- The number of the non-zero values is very small (in the shape of necktie in Figure 3). Table II shows that it occupies only 27.7% of the table.
- The non-zero values are positive and smaller or equal to one. Most of them, however, are very small up to approximately zero.

In next Section, we will discuss how to reduce the size of the table according to these characteristics.

---

**Table I: The reduction rate by replacing the calculation of spreading function with a look-up table.**

<table>
<thead>
<tr>
<th>Reduction method</th>
<th>Array of value</th>
<th>Array of index</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>6664</td>
<td>0</td>
<td>6664</td>
</tr>
<tr>
<td>Reduction</td>
<td>100.0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table II: The number of zero and non-zero values in the table of spreading function (sampling rate 44100 Hz).**

<table>
<thead>
<tr>
<th>Reduction method</th>
<th>Array of value</th>
<th>Array of index</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>4900</td>
<td>1764</td>
<td>6664</td>
</tr>
<tr>
<td>Zeros</td>
<td>3623</td>
<td>1198</td>
<td>4821</td>
</tr>
<tr>
<td>Non-zeroes</td>
<td>1277</td>
<td>566</td>
<td>1843</td>
</tr>
</tbody>
</table>

**Table III: The number of values for storage (sampling rate 44100 Hz).**

<table>
<thead>
<tr>
<th>Reduction method</th>
<th>Array of value</th>
<th>Array of index</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>1843</td>
<td>224</td>
<td>2067</td>
</tr>
<tr>
<td>Reduction</td>
<td>31.0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. MEMORY REDUCTION METHODS

According to the characteristics in the last section, we employed two methods to minimize the size of the table. First, we referred to an MP3 software encoder, which utilizes a linear array to store values and an array to store indices. Secondly, we analyzed the wordlength of the values and further reduced the size in hardware implementation.

4.1. Storage in two arrays

Since ISO/IEC MPEG Psychoacoustic Model 2 is employed in both MP3 and AAC, there is no essential difference between them in the utilization of spreading function. However, the psycho-acoustic parameters such as bark values and partitioned bands are different. Because the partitioned bands in AAC are larger than those in MP3 are, the size of the spreading function table in AAC is larger than that in MP3. Besides, in the MP3 reference software [7], the spreading function is by use of look-up tables, whereas in the AAC reference software [7], the spreading function is by use of repeated calculation. Therefore, referring to the MP3 software encoder, LAME [8], where a linear-based array is utilized to store the values of the spreading function and an array is used for indices, we modified the table of the spreading function in AAC. It is illustrated in Figure 4. The non-zero values are stored into the linear-based array segment-by-segment and row-by-row. The start indices and the end indices of each row are also stored in the array of indices, which is an overhead. Compared to the table, the indices array (overhead) is very small. The number of values for storage is shown in Table III and only 31% of the original number is required. Table IV shows the reduction of tables at different sampling rates. Most of the table’s size could be reduced to about one-third of the original.

4.2. Reduction in wordlength

We know that if the wordlength of the values of the table decreases, the area of the table would decrease while the sound quality would degrade. This is a trade-off. However, because of the last characteristic that most of the non-zero values are approximately zero, they have inferior impacts on the convolution in step 5 and thus they can be replaced with zeroes in smaller wordlength.

Table IV: The reduction of table’s size at different sampling rates.

<table>
<thead>
<tr>
<th>Sampling Rate (Hz)</th>
<th>Original</th>
<th>After Reduction (with overhead)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>4304</td>
<td>1732</td>
<td>40.2%</td>
</tr>
<tr>
<td>11025</td>
<td>5345</td>
<td>1963</td>
<td>36.7%</td>
</tr>
<tr>
<td>12000</td>
<td>5553</td>
<td>2002</td>
<td>36.1%</td>
</tr>
<tr>
<td>16000</td>
<td>5809</td>
<td>1993</td>
<td>34.3%</td>
</tr>
<tr>
<td>22050</td>
<td>6085</td>
<td>2007</td>
<td>33.0%</td>
</tr>
<tr>
<td>24000</td>
<td>6472</td>
<td>2071</td>
<td>32.0%</td>
</tr>
<tr>
<td>32000</td>
<td>6292</td>
<td>1980</td>
<td>31.5%</td>
</tr>
<tr>
<td>44100</td>
<td>6664</td>
<td>2067</td>
<td>31.0%</td>
</tr>
<tr>
<td>48000</td>
<td>6525</td>
<td>2053</td>
<td>31.5%</td>
</tr>
<tr>
<td>64000</td>
<td>6010</td>
<td>1958</td>
<td>32.6%</td>
</tr>
<tr>
<td>88200</td>
<td>6553</td>
<td>2125</td>
<td>32.4%</td>
</tr>
<tr>
<td>96000</td>
<td>6337</td>
<td>2082</td>
<td>32.9%</td>
</tr>
</tbody>
</table>

5. SIMULATION AND RESULTS

In order to verify our methods to reduce the size of the table, we simulated our design by an MPEG AAC encoder with MDCT-based psychoacoustic model [4], which was derived from the reference software. In addition to subjective listening, NMR (Noise-to-Mask-Ratio) and ODG (Objective Difference Grade) [9]...
were employed as objective audio quality measurement. EQUAL was used to calculate NMR and ODG. The sound quality would not be affected when only the first method “Storage in two arrays” was employed. Table II shows the wordlength reduction versus the degradation of sound quality. The wordlength of indices (overhead) is calculated as 7 bits, and the number of the values is calculated as
\[
2 \times (70\text{LONG}+42\text{SHORT}) = 224.
\]
Since the more positive of ODG and the more negative of NMR mean that the sound quality is better, we estimated the sound quality degradation as the difference between the tested and the original sound and normalized them into positive values, namely, larger positive degradation stands for worse sound quality. As has been noted, the reduction of wordlength of the values has little influence on sound quality. Even if the wordlength of the values is decreased to only 5 bits, the sound quality just degrades a little, which is almost not perceivable. When the wordlength is 5 bits, the area of table is only 3808 bits, which is only 3.6% of the reference, which is assumed stored in a RAM or ROM with 16 bit/word for a 16-bit DSP.

6. CONCLUSION

In order to reduce the computational complexity, calculation of spreading function in an AAC encoder is replaced with a look-up table, which is considerably large in the data RAM or ROM on a programmable processor such as DSP for portable devices. In the paper, we employed two methods to reduce the size of the table of the spreading function. Referring to the MP3 software encoder LAME, we employed a linear-based array to store the non-zero values and an array to store the indices, by which only one-third of the original size was required. This efficiently reduces the memory usage on the programmable processor. Moreover, for dedicated hardware implementation, we further reduced the memory size by word length analyses of the table values. The size (at sampling rate 44100 Hz) was reduced to only 3.6% compared to that for a 16-bit DSP while audio quality was still maintained.

<table>
<thead>
<tr>
<th>Word length (bit)</th>
<th>LONG</th>
<th>SHORT Array of value</th>
<th>Array of index</th>
<th>Area (bit)</th>
<th>Area (%)</th>
<th>Quality degrade (ODG)</th>
<th>Quality degrade (NMR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 (Ref.)</td>
<td>4900</td>
<td>1764</td>
<td>6664</td>
<td>0</td>
<td>106624</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>13</td>
<td>625</td>
<td>295</td>
<td>920</td>
<td>224</td>
<td>13528</td>
<td>12.7%</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>576</td>
<td>255</td>
<td>831</td>
<td>224</td>
<td>10709</td>
<td>10.0%</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>494</td>
<td>232</td>
<td>726</td>
<td>224</td>
<td>8102</td>
<td>7.6%</td>
<td>0.01</td>
</tr>
<tr>
<td>7</td>
<td>422</td>
<td>195</td>
<td>617</td>
<td>224</td>
<td>5887</td>
<td>5.5%</td>
<td>-0.02</td>
</tr>
<tr>
<td>5</td>
<td>305</td>
<td>143</td>
<td>448</td>
<td>224</td>
<td>3808</td>
<td>3.6%</td>
<td>-0.01</td>
</tr>
<tr>
<td>3</td>
<td>199</td>
<td>96</td>
<td>295</td>
<td>224</td>
<td>2453</td>
<td>2.3%</td>
<td>-0.01</td>
</tr>
<tr>
<td>2</td>
<td>169</td>
<td>91</td>
<td>260</td>
<td>224</td>
<td>2088</td>
<td>2.0%</td>
<td>0.04</td>
</tr>
</tbody>
</table>

7. REFERENCES

AN MDCT-BASED PSYCHOACOUSTIC MODEL CO-PROCESSOR DESIGN FOR MPEG-2/4 AAC AUDIO ENCODER

Tsung-Han Tsai, Yi-Wen Wang
Dept. of Electrical Engineering
National Central University
Chung-Li, Taiwan, ROC
{han|yiwen}@dsp.ee.ncu.edu.tw

Shih-Way Huang
DSP/IC Design Lab, Grad. Inst. of Elect. Eng.,
Dept. of Electrical Engineering
National Taiwan University, Taiwan, ROC
shihway@video.ee.ntu.edu.tw

ABSTRACT

The Psychoacoustic Model (PAM) is a very important role in MPEG-2/4 Advanced Audio Coding (AAC) encoding. It determines sound quality of a given encoder and influences a lot in computational complexity. This paper presents a new architecture design for MDCT-based PAM co-processor. This work is based on the dedicated hardware design which is different from traditional programmable approaches. Moreover, to reduce the complexity, we replace the calculations of spreading function with reduced fixed coefficients, and decrease the transform kernels from three to one unit.

1. INTRODUCTION

The AAC is an international standard, which is first created in MPEG-2 AAC (ISO/IEC 13818-7), and then become the base of MPEG-4 general audio coding [1]. It is applicable for a wide range of applications from Internet audio to digital audio broadcasting. It also achieves high compression ratio and makes high quality performance due to an improved time-frequency mapping as well as some new coding tools.

The original audio encoding process is in Figure 1. It comprises PAM, Modified Discrete Cosine Transform (MDCT), Spectral Processing (SPP) and Quantization loop (Q loop). PAM calculates a masking threshold, which is the maximum distortion energy masked by the signal energy for each coding partition. Meanwhile, MDCT transforms input audio samples in time domain into spectrums in frequency domain. The frequency spectrums then transfer to SPP, which removes their redundancies by Temporal Noise Shaping (TNS) and joint coding. Afterward, the spectrums are non-uniformly quantized based on the masking threshold and available number of bits to minimize audible quantization error and then noiselessly coded in the Q Loop. While PAM calculates masking thresholds, SPP and Q loop processes at the same time. Because PAM is deterministic, a dedicated architecture can be implemented. Besides, SPP and Q loop are non-deterministic and a programmable architecture could be an approach for implementation.

Due to the complex calculation and high barrier domain knowledge in PAM, traditional implementations for PAM are mostly direct implementation and porting on high performance programmable approaches such as DSP or general purpose processors [2]. Our previous work has proposed a MDCT-based PAM algorithm with several advantages [3]. In this paper, we propose a dedicated hardware for MDCT-based PAM. According to our architecture, we can have the main advantage of MDCT-based algorithm to replace the complex-FFT computations with MDCT computations. Furthermore, the MDCT kernel used in analysis subband in encoding can be combined with the same MDCT hardware used in PAM. Thus only one transform kernels can be achieved compared with original standardized three kernels approach, one for analysis subband and two for PAM. Therefore, a dedicated architecture of PAM is achieved and can be applied as a co-processor or intellectual property (IP) in system-on-chip (SOC) design.

2. ALGORITHMS OF MDCT-BASED PAM

In Figure 2, we show the PAM algorithms for simplicity based on the 13 steps [2]. In steps 1-2, PAM normalizes the time-domain samples as input and transforms them into frequency-domain spectrum of real part r(w) and imaginary part i(w) by FFT. Therefore, when real-part spectrums result in the calculation of partitioned energy, imaginary-part spectrums give rise to the calculation of the unpredictability measure c(w) in steps 3-4. In step 5, both partitioned energy and unpredictability are convolved with the spreading kernels (TNS) and joint coding. Afterward, the spectrums are non-uniformly quantized based on the masking threshold and available number of bits to minimize audible quantization error and then noiselessly coded in the Q Loop. While PAM calculates masking thresholds, SPP and Q loop processes at the same time. Because PAM is deterministic, a dedicated architecture can be implemented. Besides, SPP and Q loop are non-deterministic and a programmable architecture could be an approach for implementation.
function in order to estimate the effects across the partitioned bands. Tonality index is calculated in step 6 to determine if a signal is tonal-like. Signal-to-Noise Ratio (SNR) is calculated in step 7 and then the masking partitioned energy threshold \( n(b) \) is estimated in steps 8-10. Hence, the masking curve is calculated. Perceptual Entropy (PE) is calculated in steps 11-12 to determine the block type. Finally, Signal-to-Masking Ratio (SMR) is estimated in step 13 as output. The \( w, b, \) and \( n \) indicate indices in the spectral line domain, the threshold calculation partition domain, and the coder scalefactor band domain respectively.

In [3], the MDCT-based PAM is presented. One of the contributions is the reduction on spreading function calculation. In step 5, the calculation of spreading function is composed of a series of complex functions such as comparisons, square roots, power of ten, squares, and divisions. Because the calculation of the spreading function are only dependent by the sampling rates and block type used. Therefore, it can be replaced with look-up tables. The other important contribution is the implementation on MDCT kernel instead of the original FFT kernel. In steps 2-4, there are some special functions such as arctangent, sine/cosine, square root, etc. They have high computational complexity and are hard to be implemented. MDCT-based flow replaced the steps 2-6 with both MDCT and Spectral Flatness Measure (SFM), which are easier to be implemented. According to the analyses, the two mentioned methods can reduce the computational complexity by 80%.

### 3. ARCHITECTURE DESIGN OF MDCT-BASED PAM

The overall MDCT-based PAM is described in Figure 3. In the original approach of standard, there are two FFT in PAM. One is used to calculate FFT with Long block type (2048 points), and the other is used to calculate FFT with Short block type (256 points). Now we replaced Long FFT and Short FFT with MDCT. MDCT should calculate one of the four types according to the block type. Because only one MDCT unit is employed, MDCT and Threshold Generator (TG) require calculating twice a frame. The first time is to estimate the block type and the second time is to calculate the output spectrums. However, TG calculates only once and the Inverse-Signal-to-Masking Ratio (ISMR) is stored in TG delay when the block type is not Short in order to reduce the computational complexity.

Detailed block diagram of MDCT and TG is illustrated in Figure 4. It is decomposed into four parts: Part I consisting of MDCT, Part II consisting of SFM and corresponding calculation of tonality (new\(_{tb}\)), Part III consisting of Spreading Function, and Part IV consisting of steps 7-13. TG comprises Part II, Part III, and Part IV. When compared to Figure 2, Figure 4 has some modification. Because MDCT replaces steps 2-4, there is no phase-information. Only the partitioned energies are convolved with the spreading function and SFM is used to generate the tonality index \( \text{new}_{tb} \) from the MDCT spectrums.

The Table I shows specification for input and output data types of MDCT-based PAM. According to Figure 4, the overall architecture design is shown in Figure 5. It also shows the details for the word length of the sign, integer, and fraction part in each block.

![Figure 3: Block diagram of MDCT-based PAM](image)

![Figure 4: Detailed block diagram of MDCT and TG.](image)

![Figure 5: Overall architecture of MDCT and TG.](image)

**Table I: Specification of input and output data types (frame).**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample rate: 44.1kHz</td>
<td>ISMR: 49x48b</td>
</tr>
<tr>
<td>Frame size: 1024x16b</td>
<td>ISMR: 14x48b</td>
</tr>
<tr>
<td>Long</td>
<td>Short</td>
</tr>
<tr>
<td>Block type: Long</td>
<td>Block type: Long</td>
</tr>
<tr>
<td>Spectrums: 1024x24b for Long</td>
<td>Spectrums: 128x8x24b for Short</td>
</tr>
<tr>
<td>Start</td>
<td>Start</td>
</tr>
<tr>
<td>Stop</td>
<td>Stop</td>
</tr>
</tbody>
</table>

---

---
3.1. MDCT

The FFT-based fast algorithm in [4] is adopted for the implementation of MDCT because it can reduce large computational complexity. The processing flow of the fast algorithm is shown in Figure 6. First, the multiplication by pre-twiddle coefficients is applied. Then, the N/4-points complex FFT transform is calculated. In FFT, we use radix-2 butterfly for implementation because of its regularity for dedicated circuits design. N=2048 is for Long type, and N=256 is for Short type. After the N/4-points complex FFT block, 512-points real and imaginary part for Long type, and 64-points real and imaginary part for Short type are obtained respectively. Then they are multiplied by post-twiddle coefficients. Finally, 1024-points real and imaginary part for Long type and 128-points for Short type are obtained as output Y(n).

The implementation of MDCT is shown in Figure 7. The MDCT is implemented and configured as four types, including Long type, Start type, Stop type, and Short type. The main architecture performs multiplication, addition and subtraction to complete MDCT algorithm. The block of MPY involves four multiplications, one addition and one subtraction to finish a complex operation. On the left of Figure 7, those ROM blocks save the coefficients for pre-twiddle, butterfly, and post-twiddle.

3.2. SFM and new tb

SFM and tb algorithms are defined as follows:

- **SFM**
  \[
  Gm = 10 \log_{10} \text{Am} = 10 \log_{10} \frac{\sum b P(1) + P(2) + \ldots + P(b)}{P(1) + P(2) + \ldots + P(b)}
  \]

- **new tb**
  \[
  \text{new tb} = \frac{SFM}{60}
  \]
Table II: The hardware cost of MDCT-Based PAM Co-processor.

<table>
<thead>
<tr>
<th></th>
<th>MDCT buffer</th>
<th>MDCT &amp; TG</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Size</td>
<td>Number</td>
</tr>
<tr>
<td>4</td>
<td>1024x16b</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>ROM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>MUL</td>
<td>--</td>
<td>7</td>
</tr>
<tr>
<td>ADD</td>
<td>--</td>
<td>38</td>
</tr>
</tbody>
</table>

The implementation of \( SFM \) and new \( tb \) is shown in Figure 8. Except some basic computation blocks, Log block is the key module in \( SFM \) and new \( tb \). Unlike general DSP approach which takes many cycles to complete a Log operation, we use a pipelined architecture for Log implementation with throughput one cycle. According to [5], we can use 16 stages to approximate Log value. The related Log architecture design is shown in Figure 9. There are some correlations between the stages. The final stage L17 will determine Log value.

3.3. Spreading Function (SF)

Spreading function algorithm is defined as:

\[
en(b) = nrom \ast \left( \sum e(b) \otimes sprngf(b) \right)
\]  

where \( \otimes \) is a convolution operator.

Originally, SF coefficients table is composed of a 70x70 matrices (ex. LONG type). We find the coefficients along the diagonal are non-zeroes and the others are all zeroes except bottom right part. Based on this property, word length is reduced largely. Accordingly, the size of the SF table can be reduced by over 90%.

3.4. Steps 7-13

Steps 7-13 algorithms are described in details as follows:

- \( SNR(b) = tb(b) \ast 12 + 6 \) ...step 7
- \( \log bc(c) = -SNR(b) \ast 10 \) ...step 8
- \( \log n\'b(b) = \log en(b) + \log bc(b) \) ...step 9
- \( \log nb(b) = \max(\log gsth(b), \min(\log n\'b(b), \log nb\_1(b) + \log(replev)), \) where \( replev \) is 1 for SHORT and 2 for LONG ...step 10
- \( PE = PE - (BW \ast \log nb(b) - \log e(b)) \) ...step 11
- Block Type Decision. ...step 12
- \( SMR = \frac{\sum r(w)^2}{\min(hr(w\_low)...(w\_high)) \ast BW} \) ...step 13

There are several complicated operations such as power, log, and multiplication in the original steps 7-13 of [1]. Referring to [6], which use Log scaling to reduce word length of data such as thresholds, we modify the flow of log and reuse the techniques in Section 3. Note that the logarithmic thresholds are converted by power to calculate SMR in Step 13. We also used the pipelined architecture in Log and Power calculation.

4. RESULTS

The hardware cost of the whole MDCT-Based PAM is estimated in Table II. MUL and ADD stands for the number of processing units of multipliers and adders. Most of the hardware is utilized by memory. In our experiments, we simulated the proposed design by Verilog with sampling rate 44100 Hz, it requires 36248 processing cycles per frame. Therefore, the co-processor can be operated about the clock rate 2 MHz to encode stereo channels in real-time.

5. CONCLUSIONS

In this paper, we present a dedicated hardware design of MDCT-based PAM co-processor in MPEG-2/4 AAC encoder. Unlike the traditional programmable approaches, four parts of PAM are optimized to reduce the hardware complexity. This approach can save the transform kernels for three to one unit. Besides, Spreading function is calculated with a reduced size lookup. We also use the pipelined and parallel architectures to improve the performance.

6. REFERENCES


SYNTHESIS BY MATHEMATICAL MODELS:
ELLIPITIC FUNCTIONS AND LACUNARY SERIES

Vittorio Cafagna and Domenico Vicinanza
Musica Inaudita
Dept. of Mathematics and Computer Science
University of Salerno, Italy
cafagna@unisa.it dvicinanza@unisa.it

ABSTRACT

Sound synthesis methods can be interpreted, from a mathematical point of view, as a collection of techniques of selecting and conceptually organizing elements of a Hilbert space. In this sense, mathematics, being a highly structured and sophisticated system of classification, modeling and categorization, seems to be the natural tool to describe existing synthesis methods and to propose new ones. Because, from this perspective, one can think of any available (or theoretically predictable, or imaginary) synthesis method as a collection of procedures to deal with meaningful parameters, with the term "synthesis by mathematical models" we mean an extensive use of the modeling and categorization power of mathematics applied to the world of sounds.

In this paper we give a few examples of sound synthesis techniques, based on mathematical models. After reviewing shortly FM synthesis and synthesis by nonlinear distortion, and suggesting some, to our advice, interesting open problems, we propose two different new methods: synthesis by means of elliptic functions and synthesis by means of nowhere (or almost-nowhere) differentiable functions and lacunary series.

The resulting waveforms have been produced using Csound as an audio engine, driven by Python scripts.

1. INTRODUCTION

A sound synthesis method is mainly an organization technique: each method selects a parametrized collection of sounds among all the elements of a Hilbert space. To every choice of a frequency \( \nu \) it is naturally associated the Hilbert space of functions of period \( T = \frac{1}{\nu} \), interpreted as the collection of all possible synthesizable sounds with that fundamental frequency. From this point of view, a synthesis technique is a selection strategy. FM synthesis, for example, selects sounds by means of three parameters: the carrier frequency \( \omega_c \), the modulator frequency \( \omega_m \) and the modulation index \( I \). The celebrated formula

\[
\sin(\omega_c t + I \sin(\omega_m t)) = \sum_{k \in \mathbb{Z}} J_k(I) \sin(\omega_c t + k \omega_m t)
\]

which is at the basis of the FM miracle, allows us to think of an FM sound as an element of a Hilbert space \( H \) of functions of a certain period determined by \( \omega_c, \omega_m, I \). When \( \omega_m \) is a multiple of \( \omega_c \), for every value of \( I \) the sound produced stays in the same Hilbert space of functions of period \( T = \frac{1}{\omega_c} \). The mathematical model is a curve, for which we propose the name of Bessel curve, \( \beta : \mathbb{R}_+ \rightarrow H \) defined by \( \beta(I) = \{ J_k(I) \}_{k \in \mathbb{Z}} \). Leaving \( \omega_c \) fixed, changing \( \omega_m \) causes your curve wandering around different Hilbert spaces, whose period is determined by \( \omega_m \) and \( I \).

Therefore the complete model of FM synthesis can be interpreted geometrically as a surface (the Bessel surface) in a mathematically complicated object, defined as the parametrized family of all Hilbert spaces of functions of positive periods. Adding variability in \( \omega_c \) generates a three-dimensional manifold (the Bessel 3D manifold).

An obvious question to ask is how many sounds you get employing this procedure. Fourier’s theorem essentially says that you can produce every possible sound (with a given fundamental frequency) by additive synthesis. It is unlikely to expect to obtain with just two oscillators (as opposed to infinitely many) the same result. In geometrical terms, this would mean that the Bessel curve is a space-filling curve, what for many good reasons we are tempted not to believe. Still, the question remains. Experience suggests that the number of FM sounds is huge, which is akin to say that the Bessel curve goes around a lot. The mathematical question is what does a lot mean. Maybe, even if it does not pass through every point of the Hilbert space, the Bessel curve might eventually pass near each. Then the question becomes what does near mean. Analogous questions can be asked for the Bessel surface and the Bessel 3D manifolds. We think of these as interesting open problems.

More generally, FM synthesis can be seen as an example of nonlinear distortion. You take a sound (the carrier) and you modulate it with the help of a function (the modulator). The abstract framework is to take a function \( f : \mathbb{R} \rightarrow \mathbb{R} \) and define a nonlinear operator \( \hat{f} : H \rightarrow H \) by the obvious formula \( \hat{f}(s)(t) = f(s(t)) \), \( \forall s \in H \). Taking into account some technical hypotheses on the function \( f \), which we are not going to discuss here, this provides you with a good (in some suitable sense) nonlinear operator on the Hilbert space \( H \) (mathematicians call it a Nemitsky operator). FM modulators are an example, others are Chebyshev polynomials and operators defined in terms of finite summation formulas. Here again, there is a vast amount of pure-mathematical questions to be answered, which are likely to be meaningful for the world of synthesis. Are there other examples of Nemitsky operators with a predictable behaviour from a spectral point of view? In mathematical terms, can you describe the action of some Nemitsky operators \( \hat{f} \) on the action on some bases of the Hilbert space? Maybe bases others than Fourier’s, like Bessel’s or Gabor’s or wavelets, or whatever. Or: what happens if you perturb a nice-behaved operator (like, for example, a Chebyshev polynomial)? And so on. We think of these as a sample of many interesting (very) open prob-
lems.

In this paper we propose two examples of sound synthesis methods based on well-known and highly developed pieces of mathematical knowledge, the theory of elliptic functions and the theory of lacunary series, with its cornucopia of nowhere or almost-nowhere differentiable functions.

Elliptic functions are complex functions of a complex variable, doubly-periodic and meromorphic. Double-periodicity is the main reason of interest for the synthesis of sounds. The class of doubly-periodic functions contains as a particular case the class of simply-periodic ones. But, as is frequently the case in mathematics, the passage from real to complex world allows the discovery of deep relations within objects, invisible from a one-dimensional perspective. Think, as a perspicacious metaphor, of the relations between trigonometric functions and the exponential, which demand the introduction of complex numbers to become apparent. One of the most important results of the theory is that any two elliptic functions are connected by an algebraic relation. This is patently false for simply-periodic functions: think of $e^x$ and $e^{2x}$. The implications are that you can, in principle, transform any sound into any other by manipulating a few coefficients of two polynomials.

As a consequence, you can express any elliptic function as a rational function of some chosen simple ones, which play, so to say, the role of a basis, the most historically established choice being that of the Weierstrass $\wp$ and its derivative $\wp'$. Meromorphy, which is the cause of the beautiful formulas, brings in singularities. In an expressive way (see [4]), one could say that noise is the price one has to pay in order to have an algebra of sounds. But noise appears in a very controlled and structured manner. Sounds produced by elliptic functions are very rich, the richness coming from the fine structure of the spectrum produced by the existence of singularities.

The interest for synthesis of continuous nowhere and almost-nowhere differentiable functions lies in the fact that their graph is incredibly jagged. Again, this gives rise to interesting sounds. The surprising fact is that this wild behaviour can be described very simply in terms of Fourier coefficients. Here are two examples, discussed later on:

$$R(x) = \sum_{k=1}^{\infty} \frac{1}{k^a} \sin(k^b x)$$

which is called the Riemann function, and

$$W(x) = \sum_{k=0}^{\infty} a^k \cos(b^k \pi x)$$

where $0 < a < 1$, $b > 1$, $ab > 1 + \frac{3\pi}{2}$, $b$ an odd integer, which is the Weierstrass function. In both cases one has gaps between the frequencies of two adjacent terms of the series (the reason these series are called lacunary), gaps which increase as $k$ becomes larger. Here the situation seems to be opposite with respect to the former case. While spectra of sounds produced by elliptic functions are very rich, due to the presence of zones with a high density of spectral lines, in this case the spectral lines are sparse. Still, the philosophy is the same: to produce a large organized class of interesting sounds, by manipulating few well-understood parameters. At the end of the paper, we reproduce the image a simple interface in Csound to play around with lacunary series and to experiment the dramatical changes in timbre due to small variations in the parameters. The Csound files, plus an archive of sounds, waveforms, spectra, Python scripts and Mathematica notebooks related to the items discussed in this article can be downloaded from the site http://www.musicainaudita.it.

2. COMPUTATION AND SOUND SYNTHESIS

All the waveforms described in this article have been computed using Python and Csound. The main strategy is to perform all the computations using free software available for different platforms (to have a tight control both at computation and rendering level). The idea to couple an audio engine (the acoustic compiler Csound) with a scripting language (Python) is motivated by our main goal: high flexibility and efficiency, together with high quality audio files.

Among the possible choices of acoustic compilers and scripting languages, there were many reasons to choose Python and Csound. First of all, Csound is free and available for all platforms. Its file format (a couple of ASCI files, named Orchestra and Score) is a well known standard with thousands of users spread around the world. Finally Csound could run offline on a personal computer, on a workstation or even on a linux cluster.

About the scripting languages, our interest was addressed to Python, since it is free, cross-platform and object-oriented. Python allows in natural way the creation of efficient and easy-to-maintain software environment; moreover, it allows the integration into websites, using CGI.

Python has been used to produce the right Csound files to drive the acoustic compiler, according to the mathematical model taken into account. In some cases, thanks to the mathematical libraries available (math for usual trigonometry and cmath for complex analysis), it has been used to quickly develop numerical calculation packages to compute actual values of the audio samples. After the computations, Python scripts prepare the score and orchestra files to generate the waveform according to the sampled values obtained. In a certain sense, we can say that Csound is playing the role of digital-to-analog converter.

In other situations (for example, when computing with the Riemann and Weierstrass nowhere differentiable functions), audio files have been created by Csound by additive synthesis. In this case Python has been used to set up the instrument, using Csound opcodes.

Coupling an acoustic engine and a scripting language which includes a complex and wide set of math libraries, opens up a wide range of possibilities, which run from algorithmic composition using Csound instruments and scores, to the complete low-level control of audio samples.

Sound synthesis by mathematical models becomes, in this way, the combination of two well distinct activities, each of them performed by an optimized system: numerical calculations (using Python scripts) and sound production (Csound). The two phases could be carried on independently, even at different times and on different platforms.

3. SYNTHESIS BY MEANS OF ELLIPTIC FUNCTIONS

3.1. Elliptic functions

Synthesis by means of two-variables functions has been investigated in many different ways (see, e.g., [1], [2]) since the pioneering article of Mitsuhashi ([3]). The usual point of view is to choose a surface, a closed curve (orbit) on the surface, and to produce a waveform sampling the function on the curve. Orbits play a very
important role in $n$-variable synthesis, their shape and geometrical characteristics affecting the final waveform. Closed orbits return periodic waveforms and open orbits (such as spirals) seem to be a promising tool to explore time-evolving sounds.

With respect to the existing literature, we decided to use complex functions of a complex variable, as opposed to real functions of two real variables. The main reason for this choice, alluded to already in the Introduction, is that complex analysis is a very rich and successful mathematical theory, with a strong tendency to bring to the surface unseen relations between real objects and to organize sparse arguments in a very articulated structure.

Let us come to the definitions.

A function $f : C \to C \cup \{\infty\}$ is doubly periodic with periods $\omega_1$ and $\omega_2$ if $\omega_1$ and $\omega_2$ are linearly independent over $\mathbb{R}$ and if $f(z + \omega_1) = f(z) = f(z + \omega_2)$ for all $z \in C$. An elliptic function is a meromorphic doubly periodic function. For $\omega_1$ and $\omega_2$ fixed, the corresponding elliptic functions form a field. The parallelogram with vertices $0, \omega_1, \omega_2, \omega_1 + \omega_2$ containing the sides adjacent to the origin, but not containing the other two, is called the fundamental parallelogram $\Pi$. Being periodic with respect to the lattice $L$ in the complex plane defined by the periods, one can think of elliptic functions as functions on the torus $C/L$. This has not only the structure of a two-dimensional smooth manifold, but also an intrinsic structure of a Riemann surface, determined by the periods $\omega_1$ and $\omega_2$. Varying the periods you get a different complex structure on the same (on a topologically equivalent) underlying real torus. You associate sounds to elliptic functions with the same method as in every $n$-dimensional synthesis: you choose a closed curve on the torus and restrict the function to the curve to obtain a couple of periodic waveforms, the real and imaginary parts. It is tempting to think of the couple as a stereophonic sound, and try to create auditory images of complex analysis. Anyway you can enjoy the freedom to experiment with any algebraic (even non-algebraic, for that matter) combination of the two. It is worthy to point out that, as opposed to the real case, you do not have a single sound associated to a complex curve and a closed curve, but an infinite family. Of course, the shape of the chosen closed curve plays also a dominant role in the sonic results. To begin with, one can distinguish between two big families: shrinkable curves and non-shrinkable (curves which wind around the holes of the torus.) Within the former class, the meaningful distinction seems to be between curves which go around a pole (and how many times), and curves which do not. In the latter class the meaningful parameter is probably the number of tours the curve makes around a hole (winding number). The simplest family of curves of the second class are helices, obtained by projecting on the torus $C/L$ straight lines in the complex planes. Extensive experiments have been carried on, using the slope (which affects the winding number in a readable way) as a varying parameter. We refer again to the quoted site of Musica Inaudita for the records.

There is an entire zoo of famous elliptic functions, connected to each other by a variety of well-known (and less well-known) formulas. We refer to [5] and [6] for an extensive treatment of the subject. In this paper we give just one example, the Weierstrass $\wp$, defined as follows:

$$\wp(z) = \frac{1}{z^2} + \sum_{\omega \in L'} \left( \frac{1}{(z - \omega)^2} - \frac{1}{\omega^2} \right)$$

(3)

where the sum is taken over the set of all non-zero $\omega$, denoted by $L'$. This series converges uniformly on compact sets not including the lattice points. What makes elliptic functions so interesting for audio synthesis is the fact, already recorded in the Introduction, that you can describe any of them in terms of some chosen collection of special functions. Electing, as we did, the Weierstrass function as our main character, the significant theorem becomes:

**Theorem.** Any even elliptic function is a rational function of $\wp$. Any elliptic function $f$ can be written uniquely in the form

$$f(z) = g(\wp(z)) + \wp^j h(\wp(z))$$

where $g$ and $h$ are rational functions.

The existence of such a result permits a systematic exploration of the sound quality of different elliptic functions, exploiting the different possible formulas. To fix the ideas, one can study the additive synthesis formulas offered by the theorem, starting with $\wp$ and $\wp'$, and producing rational functions of increasing degree in $\wp$ and $\wp'$:

$$\wp(z) + \wp'(z)\wp(z)$$

$$\wp^2(z) + \wp'(z)\wp(z)$$

$$\wp^3(z) + \wp'(z)\wp^2(z)$$

$$\wp^4(z) + \wp'(z)(\wp^3(z) + \wp(z))$$

$$\vdots$$

$$\wp^n(z) + \wp'(z)\left(\frac{\wp^n(z) + \cdots + \wp(z) + 1}{\wp(z) + \cdots + \wp(z) + 1}\right)$$

$$\vdots$$

### 3.2. Sound synthesis

Audio samples obtained using elliptic functions have been generated by Csound using the sampled values calculated by a Python script. For the computations, we used the following formula which expresses Weierstrass’s $\wp$ as the sum of an infinite series

$$\wp(z) = \frac{\pi^2}{4k^2} \sum_{k=-\infty}^{\infty} \frac{1}{k} \sum_{\omega \in L} \frac{\csc^2 \left( \frac{\pi}{2k} \omega \right)}{2k^2}$$

Thanks to the complex analysis Python libraries, all the computations have been performed without any need for external programs or resources. The same script has been used to first compute the values of the elliptic functions, at regular steps, on the given orbit, and then to produce a couple of score and orchestra files, processed by Csound and converted into an audio file.

The **revolution frequency** of the orbit has been chosen in the audio range, namely 440 Hz; to have a perceivable audio signal:

$$x = \cos(440 \cdot 2\pi t), \quad y = \sin(440 \cdot 2\pi t)$$

(4)

In Figures 1-5 we list some waveforms obtained from Weierstrass’s $\wp$, using different closed orbits.

### 4. SYNTHESIS BY MEANS OF CONTINUOUS NOWHERE DIFFERENTIABLE FUNCTIONS AND LACUNARY SERIES

#### 4.1. Introduction

In the previous Section we have presented a sound synthesis technique articulated in two phases: sample computing and sound realization (or better, conversion, since Csound works in this case as
Figure 1: Waveform of $\mathcal{R}_\phi$ calculated on the circular orbit $x = \cos(440 \, 2\pi t), \ y = \sin(440 \, 2\pi t)$

Figure 2: Waveform from $\mathcal{R}_\phi$ calculated on the open orbit $x = \sin(440 \, 2\pi t), \ y = t(1 + \cos(4402\pi t))$

Figure 3: Waveform from $\mathcal{R}_\phi$ calculated on the open orbit $x = \sin(440 \, 2\pi t), \ y = t(1 + \cos(4402\pi t))$

Figure 4: Waveform from $\mathcal{R}_\phi$ calculated on the open orbit $x = t, \ y = 3t \ (\text{helix})$

Figure 5: Waveform from $\mathcal{R}_\phi$ calculated on the open orbit $x = t, \ y = 50t \ (\text{helix})$

a digital-to-analog converter). For the sound generation procedure described in this Section we take full advantage of the capacities of Csound as an additive synthesizer. We again used Python scripts to prepare the Csound instruments, adding the right number of elementary oscillators (described by the \texttt{oscill} opcode) taking into account the frequency and amplitude relations. Once more, the main advantage in using this kind of organization is a high flexibility and the possibility to easily create a collection of Csound orchestras and scores, according to a certain principle (performing a systematic exploration of a parameter) and then run all the sound processing together afterwards.

4.2. The Riemann function

The Riemann function $R : \mathbb{R} \rightarrow \mathbb{R}$ is defined as follows (we refer to [7] for an accurate description of the world of nowhere differentiable functions):

$$R(x) = \sum_{k=1}^{\infty} \frac{1}{k^2} \sin(k^2 x)$$

(5)

The Riemann function $R$ is continuous on all of $\mathbb{R}$ but differentiable only on a set of points of measure zero (i.e. $R$ is non differentiable on a dense subset of $\mathbb{R}$.)

It has been proved, by Gerver [8] [9], Hardy [10] and Smith [11], that, despite the fact that $R$ is almost nowhere differentiable, it has a finite derivative at points of the form:

$$x_0 = \frac{2p + 1}{2q + 1}, \ p, q \in \mathbb{Z} \quad R(x_0) = -\frac{1}{2}$$

(6)

It might be interesting to observe that the values of $R$ can be explicitly computed at the numbers $x = p/q, \ p, q \in \mathbb{Z}$:

$$R \left( \frac{p}{q} \right) = \frac{\pi}{4q^2} \sum_{k=1}^{q-1} \sin \left( \frac{k^2 \pi p}{q} \right)$$

(7)

4.3. The Weierstrass function

The Weierstrass function was the first published example of a continuous nowhere differentiable function (1875):

$$W(x) = \sum_{k=0}^{\infty} a^k \cos(b^k \pi x)$$

(8)

$$0 < a < 1, \ b > 1, \ ab > 1 + 3\pi/2, \ b \text{ an odd integer.}$$
4.4. A graphical interface for lacunary series

Both Riemann’s and Weierstrass’s functions are examples of the large class of functions defined by convergent lacunary series. Let \( \{n_k\} (k \in \mathbb{N}) \) be a strictly increasing sequence of natural numbers such that \( n_{k+1}/n_k > q > 1 \). The series
\[
\sum_{k=1}^{\infty} a_{n_k} e^{i n_k t}
\]
(9)
is called lacunary.

Under suitable conditions on coefficients and frequencies (Hardy conditions) the lacunary series converge rapidly, offering the possibility to easily obtain interesting sonic results even using a limited number of oscillators. In this Section we discuss an implementation of an interactive interface for sound synthesis by means of lacunary series using eight oscillators.

The formula which describes the waveforms obtainable with this software tool is
\[
\sum_{n=1}^{8} \alpha_n \sin(n^2 \pi ft)
\]
(10)

The interface has been developed using MacSounds [12] by Matt Ingalls. The eight sliders on the left act on the Fourier coefficients \( a_1, \ldots, a_8 \). The sliders on the right (labeled, respectively as exponent and base frequency) act in a self-explanatory way. The choice \( k = 1 \) gives a full Fourier series. Choosing an integer \( q > 1 \) gives rise to lacunas in the spectrum. One can experiment in real time the enrichment of the sound produced by the migration of the eight spectral components towards the high frequencies. Figure 8 shows a screenshot of the program while running and the Figure 9 displays the waveform obtained from the parameters corresponding to the slider configuration of Figure 8. A last word on the slider \( k \). The slider moves continuously and therefore passes through every (ideally) real number, while the symbol \( k \) appearing in the formula stands for natural numbers. So, this little toy produces much more than sounds obtained by lacunary series. In fact the action obtained on a sound taking real exponents might be described as a nonlinear distortion, defined on a basis of the Hilbert space of functions with the chosen base frequency. This is related to the regular transformations defined by MacAdams [13]. To our advice, it seems a very interesting subject to investigate.

5. CONCLUSIONS

We have proposed two (as far as we know) new methods of synthesis, based on mathematical models. Our belief is that the parameters, which are significant for the synthesis are also meaningful from a pure-mathematical point of view. The hope is that, once the correspondence synthesis-mathematics established, the whole organizing power of mathematical theories can be applied to obtain an analogous conceptual frame by which interpreting sounds.

6. REFERENCES


SOUND TEXTURE MODELING AND TIME-FREQUENCY LPC

Xinglei Zhu
Institute for Infocomm Research (I2R) & National Univ. of Singapore
xzhu@i2r.a-star.edu.sg

Lonce Wyse
Institute for Infocomm Research (I2R)
lonce@zwhome.org

ABSTRACT

This paper presents a method to model and synthesize the textures of sounds such as fire, footsteps and typewriters using time and frequency domain linear prediction coding (TFLPC). The common character of this class of sounds is that they have a background “din” and a foreground transient sequence. By using LPC filters in both the time and frequency domain and a statistical representation of the transient sequence, the perceptual quality of the sound textures can be largely preserved, and the model used to manipulate and extend the sounds.

1. INTRODUCTION

Sound textures are sounds for which there exists a window length such that the statistics of the features measured within the window are stable with different window positions. That is, they are static at “long enough” time scales. Examples include crowd sounds, traffic, wind, rain, machines such as air conditioners, typing, footsteps, sawing, breathing, ocean waves, motors, and chirping birds. Using this definition, at some window length any signal is a texture, so the concept is of value only if the texture window is short enough to provide practical efficiencies for representation. Since all the temporal structure exists within a determined window size, if we have a code to represent that structure for that length of time, the code is valid for any length of time greater than the texture window size.

A “sound model” is a parameterized algorithm for generating a class of sounds. Sound models can provide extremely low bit rate representations, because only model parameters need to be communicated over transmission lines. That is, if we have class-specific encoder/decoder pairs, we can achieve far greater coding efficiencies than if we only have one pair that is universal [1]. An example of using a class-specific representation for efficiency is speech coded as phonemes. The problem is that we do not yet have a set of models with sufficient coverage of the entire audio space, and there exist no general methods for coding an arbitrary sound in terms of a set of models. The process is generally lossy and the “distortion” is difficult to quantify. However, there are specific application domains where this kind of model-based codec strategy can be very effective. Wyse, Wang and Zhu [2] describe a packet loss recovery method for transients in music using a “beat” model that vastly reduces the amount of necessary redundant data for error recovery. Another example might be sports broadcasting where a crowd sound model could be used for low bit-rate encoding of significant portions of the audio channel.

If generative sound models are used in a production environment, the same representation and communication benefits exist.

Ideally, all audio media could be parametrically represented just as music is currently with MIDI (musical instrument digital interface) control and musical instrument synthesizers. In addition to coding efficiency, interactive media such as games or sonic arts could take advantage of the interactivity that generative models afford. Sound textures are an important class of sounds for interactive applications, but in a raw or even compressed audio form they have significant memory and bandwidth demands that restrict their usage.

If a statistical description of features is valid, (e.g. the density and distribution of “crackling” events in a fire), the variance in the instantiations for a given parameter value would be semantically equivalent, if not perceptually so. That is, one might be able to perceive the difference between two reconstructed texture windows since the samples have a different event pattern, but if density is the appropriate description of the event pattern, then the difference is unimportant. We must thus identify structure within the texture window that can be represented statistically as well as structure that must be deterministically maintained.

2. SOUND TEXTURE MODELING

Texture modeling does not generally result in models that cover a particularly large class of sounds. It is more appropriate for generating infinite extensions with semantically irrelevant statistical variation than it is at providing model parameters for interactive control or for exploring a wider space of sound around a given example.

In this paper, we focus on synthesizing continuous, perceptually meaningful audio stream based on single audio example. The synthesized audio stream is perceptually similar to the input example and not just a simple repetition of the audio patterns contained in the input. The synthesized audio stream can be of arbitrary length according to the needs.

2.1. Time Scale in Modeling Sound Texture

Generally, different time frames are used for texture analysis. The texture window length is signal-dependent, but typically on the order of 1 second. If the window needs to be longer in order to produce stable statistics when time shifted, then the sound would be unlikely to be perceived as a static texture. An LPC analysis frame is typically on the order of 10 or 20 ms. The frequency domain LPC (FDLPC) technique, which is an important part of our system, is called “temporal wave shaping” in its original context [3], and it specifies the temporal shape of the noise excitation used for synthesis on a sub-frame scale.
Tzanetakis and Cook [4] use both analysis and a texture window. In recognition that a texture can be composed of spectral frames with very different characteristics, they compute the means and variances of the low-level features over a texture window of one second duration. The low level features include MFCCs, spectral centroid, spectral rolloff, the frequency below which lays 85% of the spectral “weight”), spectral flux (squared difference between normalized magnitudes of successive spectral distributions) and time-domain zero crossing. Dubonov [5] used a wavelet technique to capture information at many different time scales. St. Arnaud [6] developed a two-level representation corresponding roughly to sounds and events, analogous to Warren and Verbrugge’s “structural” level [7] describing the object source and the “transformational” level corresponding to the pattern of events caused by breaking and bouncing.

2.2. TFLPC Modeling

One of the objectives in model design is to reduce the amount of data necessary to represent a signal in order to better reveal the structure of the data. The TFLPC approach achieves a dramatic data reduction with minimal perceptual loss for a certain class of textures. Athineos and Ellis [8] used this representation to achieve excellent parameter reduction with very little perceptual loss using 40 Time Domain LPC (TDLP) coefficients and 10 Frequency Domain LPC (FDLPC) coefficients per 512-sample or 23ms frame of data resulting in a 10x data reduction. In this process, the compression is lossy although perceptual integrity is preserved and the range of signals for which this method works is restricted. This is a coding method rather than a synthesis model, although it achieves excellent data reduction. We cannot, for example, generate perceptual similar sounds of arbitrary length using this method, which greatly restricts the applications.

To construct a generative model, we want to connect the Time domain (TD) signal representation to a perceptually meaningful low-dimensional control. We have hope of doing this because the signal representation is already very low dimensional. We still need to “take the signal out of time” by finding the rules that govern the progression of the frame data vectors.

3. SYSTEM FRAMEWORK

The framework of the system is shown in Figure 1. There are five basic steps in the framework: frame-based TFLPC analysis, event detection, background sound separation, TFLPC clustering in reflection domain and resynthesis. The first four steps are the process of modeling the sound texture, and the last step is to synthesize sound of arbitrary length.

3.1. Frame-based TFLPC Analysis

A frame-based time and frequency domain LPC analysis is first applied to the sound for further event extraction and reflection domain clustering, as shown in Figure 2. Such an analysis is essentially the same as the method in [8]. Each frame in the signal is first multiplied by a hamming window. Following the time domain linear prediction (TDLP), 40 LPC coefficients and a whitened residue are obtained. Then the TD residue is multiplied by an inverse window to restore the original shape of the frame. We use a discrete cosine transform (DCT) to get a spectral representation of the residue and then apply another linear prediction to this frequency domain signal. This step is called frequency domain linear prediction (FDLPC), which is the dual of TDLP in frequency domain. We extract 10 FDLPC coefficients for each frame.

3.2. Event Detection

The detection of events is shown in Figure 3. The gain of time domain LPC analysis in the frame-based TFLPC indicates the energy of frames so that it can be used to detect events. The gain is first compared with a threshold (20% of the average of the gain over the whole sound sample) to suppress noise and small pulses in gain. A frame-by-frame relative difference is calculated and the peak position of the result is recorded as the onset of an event. To detect the offset of each event, we use the average of the gain between adjacent event onsets as an adaptive threshold. When the event gain is less than the adaptive threshold, the event is considered as over. The length of most events in our collection of fire sounds vary from 5-7 overlapped frames, or 60-80ms.

The event density over the duration of the entire sound is calculated as a statistical feature of the sound texture and this density is used in synthesis to control the occurrence of events.

3.3. Background Separation

After we segment out the events, we are left with the background sound we call “din” containing no events. We concatenate the individual segments and apply a 10-order time domain LPC filter to this background sound to model it. The TDLP coefficients we
obtain here are used to reconstruct the background sound in the resynthesis process.

3.4. TFLPC Coefficients Clustering

In this step, we cluster the TD LPC coefficients and FDLPC coefficients to further reduce the data amount. The process is as follows.

1) We first transform each of the TD LPC coefficient (TDLPCC) and FDLPC coefficient (FDLPCC) vectors into the reflection domain. The filters represented by the LPC coefficients are not generally stable under perturbation [9], so such a transform is necessary.

2) Then we determine the number of clusters of TDLPCC and FDLPCC separately. This is an issue of validity in unsupervised clustering. Here we use the K-means method in clustering and the criterion function of minimization of ratio of within-cluster scatter matrix’s norm and total scatter matrix’s norm [10] to determine the proper cluster number.

The criteria function is defined as:

$$ F = \frac{\det(S_c)}{\det(S_i)} $$  \hspace{1cm} (1)

where

$$ S_c = \sum_{i=1}^{c} \sum_{x \in X_i} [x - m_i]^T $$

is the within-cluster scatter matrix, $X_i$ is the $i$-th cluster, $c$ is the total number of clusters,

$$ m_i = \text{mean}(x | x \in X_i) $$

is the mean vector of the $i$-th cluster,

$$ S_i = \sum (x - m)(x - m)^T $$

is the total scatter matrix and $m = \text{mean}(x)$ is the mean of all the vectors. We limit the number of cluster to be in the range from 2 to 20 and then calculate the criterion function $F$ for different candidate cluster numbers in this range. Then we calculate the change rate of $F$ with increase of cluster number $c$. When the change rate is very small (less than 1/1000), which means the criteria function changes slowly, the current number is considered as the optimal one.

3) The center vector and variance of each cluster is calculated and recorded for resynthesis. Based on the assumption that each dimension of the LPCC vector is independent, we calculate the variance of each dimension separately so that we get a variance vector for each cluster. Instead of the original frame-based TFLPCC sequence of each event, the cluster index of each sequence and the cluster center and variance are recorded. We also record the time domain LPC gain sequence and the cluster number sequence of each event for resynthesis. We record these parameters to preserve the original order of frames, which is critical in our system.

3.5. Resynthesis

In the resynthesis process, we generate the background sound and event sequence separately and mix them together in the final step.

Given a desired sound length, we use a noise excited 10-order background TD LPC filter to generate the background sound.

For the foreground sound, the resynthesis process is shown in Figure 4 and described below.

1) Use the event density, which is the average number of events per second, as the parameter of a Poisson distribution to determine the onset position of each event in the resynthesized sound.

2) Randomly select an event index. According to the TFLPCC sequence, use the reflection domain TFLPCC cluster centers and the corresponding variance as the parameters to a Gaussian distribution function in each dimension to generate the reflection domain TFLPCC feature vector sequence for the event.

3) Transform the reflection domain coefficients into the LPC domain.

4) Do the inverse TFLPC. This is just a reverse process of the TFLPC analysis, as shown in Figure 5. We first get the DCT spectrum of the excitation signal and then filter it using the FDLPC...
coefficients to get the excitation signal in the time domain. Figure 6 shows the residue and the regenerated excitation in time domain. FDLPC captures the sub-frame contour shape well. Then we filter the time domain excitation using the TDLPC filter to get the time domain frame signal.

5) Repeat step 4 for all the frames inside one event and then overlap and add to reconstruct the event.

6) Repeat 2-5 until we generate events for all the event positions.

7) Mix the synthesized events and the background sound together to get the final result. The result is shown in Figure 7.

4. EVALUATION AND DISCUSSION

Informal listening tests show that the regenerated sound is quite similar to the sample audio clip. By using frame level contour extraction and TFLPC analysis, both the spectral and fine temporal characteristics of the sound are captured. To listen to and compare the original sound with the generated one, see our website http://www.zwhome.org/~lonce/Publications/dafx2004.html

The error for each generated transient event comes from two sources: one is the error between the excitation signal and the original residue; another is the difference between the generated LPCC and the original one due to the clustering. It is not easy to quantitatively measure the dissimilarity between the generated sound and the sample audio principally due to the statistical variation in the model.

4.1. Properties of reflection domain clustering

In the clustering of the TFLPCC, we use the reflection domain coefficients instead of LPC domain coefficients. The reflection domain coefficients have several advantages compared to the LPC domain coefficients [9]. Some of the advantages are:

1) the all pole filter is stable under perturbation provided that the corresponding reflection coefficients all lie between -1 and +1,

2) interpolating between two of reflection coefficients yields a smooth change in the frequency response.

Figure 8 shows how the frequency response changes when we scale the reflection domain coefficients. The first plot is the frequency response of a time domain reflection coefficients. The second plot is the frequency response of the normalization of the coefficients whose norm is 1. The last plot is the frequency response of scale factor 0.01 multiplies the original coefficients. The figure shows that when the maximum component of reflect coefficient vector is much smaller than 1, rescaling the coefficient vector does not change the frequency response of the LPCC much. In other words, such a change in the frequency response is acceptable.
and our clustering algorithm can be independent of the vector magnitude.

4.2. Comparison with Other Methods

4.2.1. An HMM Method

In the framework in Section 3, we cluster the reflection domain TDLPC and FDLPCC into clusters separately and record the TDLPC and FDLPCC cluster index sequences for each event to preserve the original order of the frames. By such a clustering we get two “codebooks” of the TDLPC and FDLPCC separately and greatly reduce the amount of information in reconstruction. However preserving the specific cluster number sequence for each event also restricts the flexibility of modeling. To gain more flexibility, we train a Gaussian Mixture Model to capture the order pattern.

After we get the reflection domain TDLPC and FDLPCC sequence for each event as we do in Section 3, we use these TDLPC sequences and FDLPCC sequence of all events to train two Gaussian mixture HMMs for TDLPC and FDLPCC separately. In resynthesis, we use these two HMMs to generate the reflection domain TDLPC sequence and FDLPCC sequence for regenerated events. However, result shows that such a system does not work well. In the Gaussian Mixture HMM, there are several possible Gaussian distributions for each state. When we generate coefficient vectors using the HMM, these distributions are chosen according to some probability. The randomness in the cluster sequence has a significant detrimental affect on the perceptual quality of the regenerated sound.

4.2.2. Comparison with Event-Based Method

As another approach to reduce the amount of data, we implement a system using TFLPC analysis to entire events instead of overlapped frames. First the energy of each frame is calculated and then we extract events from the energy sequence of the whole sound as we do to the gain sequence in Section 3.2. Next we apply TFLPC analysis to individual events instead of frames so that we have only one TFLPC vector and one FDLPCC vector for each event compare with frame-based method’s two vector sequences. The data amount is further reduced. However, there is a dramatic quality decrease when the event length is long. The reasons are as follows:

1) When the event length increases, the modeling ability of LPC decreases. We can use a greater filter order, but the quality is still worse than the short window case.

2) The limited amount of data affects the parameter extraction for the Gaussian distribution of each cluster. We get only two LPC vectors for each event instead of two LPC coefficient vector sequences, so there is not enough data to estimate the proper Gaussian distribution parameters for each cluster.

Based on these reasons, among the several methods we implemented in our experiment, the frame-based TFLPC analysis method which is introduced in the system framework section worked the best.

5. FUTURE WORK

We have demonstrated a method for modeling certain classes of sound textures. The method involves analysis at different time scales to preserve perceptually relevant information for synthesis and resynthesis. Future work will focus on improvement of quality and generalization of this method to a wider class of sounds.

Currently we use a frame-based TFLPC analysis. If we could capture the order pattern of the frames inside events, we could build pattern models to gain more flexibility.

In the current system we assume all the events are of the same kind and use a single Poisson distribution to simulate the occurrence of the events. This assumption may be violated for some sounds, such as the sound from tennis game containing the players’ footstep sound and the driving-ball sound. By classifying the events into different classes and using different statistical distributions for sequencing them, we can build a better model for the sounds containing more than one kind of event.

Sounds with both broadband noise and densely-packed micro-transients are very difficult to segment into individual transient events from the residual information. It is difficult to get global statistical features such as event density to control the resynthesis and the frame by frame method loses flexibility. Segmentation of such complex sounds should also be explored to generalize this method for flexible resynthesis.

6. REFERENCES

MAKING SOUND WITH NUMBERS, SIX YEARS LATER

Nicola Bernardini, Damien Cirotteau, Free Ekanayaka and Andrea Glorioso

ABSTRACT
In the first edition of the DAFx Conferences an extensive tutorial on professional and research software devoted to sound and music making was presented. The present paper attempts a revision of the concepts expressed in that tutorial, focusing particularly to the aspects related to research and innovation fostered by a strong paradigm shift that has happened in the mean time: that of Free Software development. Of course, this paradigm shift has also had its difficulties and harsh spots, requiring many extra efforts in order to overcome them. This paper will try to describe these as well as to outline the current state-of-the-art in the field.

1. INTRODUCTION
In the first edition of the DAFx Conferences, DAFx'98, one of the authors of this paper, along with Davide Rocchesso, presented an extensive tutorial on professional and research software devoted to sound and music-making was presented. Six years have gone by since then, and a revision of that paper is indeed necessary: while most of the concepts expressed therein are still completely valid, a major change has occurred in the general software infrastructure that underlies most professional and research sound and music endeavors – namely, the expansion of Free Software to an extent which was quite difficult to predict at that time. True, that paper was indeed aware that “a good 70% of that software is actually Free Software” (cf.[1, p.200]) and a re-edited version of it of the software mentioned in this document belongs to the open-source public domain” (cf.[1]) certainly, Free Software was already very much alive and widespread in 1998; however, it was difficult to foresee that “a good 50% of that software is actually Free Software” (cf.[2] p.200) and a re-edited version of it published in the Journal of New Music Research did specify further that “a good 50% of that software is actually Free Software” (cf.[2]) certainly, Free Software was already very much alive and widespread in 1998; however, it was difficult to foresee that the following conditions

1. users may choose liberally to use consumer-grade or professional-grade applications according to their own needs, skills and capabilities; no artificial commercial barriers are set to separate “domestic users” from “professional” ones;
2. users are allowed (and as a matter of fact, even encouraged) to use these tools in a creative way, adapting them to their needs instead of accepting passively the features (or lack thereof) of each application; this is a particularly relevant feature in the sound and music domain, where creativity is often at stake in a dialectic relationship with the available tools;
3. users may easily mingle directly with developers on the many communication channels offered by the Free Software community at large (mailing lists, IRC channels, etc.), thus promoting a knowledgeable use of tools and applications;
4. users may contribute actively to steer development of tools according to their needs, thus accumulating knowledge and skills in a much faster process than what is usually achievable in non-free software environments;
5. bugs and problems are reported and disclosed to public inspection; as such, not only they get found and fixed with shorter delays, but they allow users to build a much more trusting attitude towards their tools (something that has gone completely lost in non-free software environments — now caught instead in a sort of fatalistic “reboot” attitude).

The AGNULA Project
http://www.agnula.org
{nich|damien|free|sama}
@agnula.org

Media Innovation Unit
Firenze Tecnologia
Palazzo Borsa Merci
Volta dei Mercanti, 1
I-50122 Firenze
Tel: +39335/706 3282
Fax: +3906/233 200 750
http://miu.firenzetecnologia.it
{nich|damien|free|sama}@miu-ft.org

Incidentally, this addition is a clear example of the language shift that happened in these last years giving unprecedented attention to the specific licensing aspects of software distributed over the Internet. Nowadays, a much more precise terminology has been developed, and care must be used when adopting it (cf. http://www.gnu.org/philosophy/philosophy.html#Terminology andDefinitions ).
But, of course, this paradigm shift has also had its difficulties and harsh spots, requiring many extra efforts (i.e. other than writing software itself that is) in order to overcome them.

2. WHERE HAS ALL THE NON-FREE SOFTWARE GONE?

The Free Software paradigm shift is only one side of the story. In the previous DAFx paper, many non-free software packages devoted to sound processing and control were mentioned. In fact, in 1998 Free Software in audio was still a small niche in the complete picture. The situation depicted in Sec. 1 may hint at the fact that the current Free Software ruling may have succeeded or will succeed in putting non-free software out of business.

This is strictly non-true. The first reason is that Free Software is not here to put software houses out of business: rather, it promotes fair and liberal competition among equally fitted individuals and institutions.

The second reason is that non-free software devoted to music and sound has been put out of business by its own practices. The music and sound domain chronicles of the past 4-5 years have been filled with gory stories of buyouts and subsequent disappearances of most software houses operating in this field. One prominent example was the Emagic (producer of the Logic sequencer) buyout by Apple and subsequent discontinuation of the software production line for the Windows production line in 2002\(^1\) the impact of the decision apparently affected several thousands Windows users, who discovered how Emagic would “cordially invite all Logic Windows users to join us” [i.e. Emagic] on the Macintosh\(^2\) Windows users were so upset by this decision that they went as far as signing a petition against it\(^3\) to no avail whatsoever.

This is just one of the many examples available which cannot all be listed here for lack of space. For the same reason we will avoid the issue of all the software houses put out of business by impossible deals contracted with their reference operating system or hardware provider as well as that of the changing hands of software houses usually meaning vast changes in marketing targets, etc. At any rate, there have been so many cases now for non-free sound and music software brutally dumping their user communities that it is fair to assume that this has been the one of the most powerful motivations to push musicians and sound software users towards a conscious and aware use of Free Software. Certainly, making sound with numbers is possible today using Free Software exclusively. This paper strives to show why and how.

3. HISTORICAL BACKGROUND

As we already wrote, at around the same time that\(^1\) was being conceived and written, the situation of sound/music Free Software applications had already reached what could be considered well beyond initial pioneering stage. A website, maintained by musician, composer and GNU/Linux enthusiast Dave Phillips, was already collecting all possible sound and music software running on GNU/Linux architecture\(^4\) That collection would already count hundreds of applications covering all possible sound/music fields and needs and particularly the broad categories already expressed in\(^1\): languages for sound processing, in-line sound processing, time domain graphical editing and processing, analysis/resynthesis packages, interactive graphic building environments, pedagogic software and processing libraries, plugins and toolkits.

At that time, the biggest problem was that all these applications were dispersed over the Internet: there was no common operational framework and each and every application was a case-study by itself. Free Software binary distributions were young: a quick glance over the Internet shows that full binary distributions of Free Software date back to the summer of 1993, Slackware being the first well-known one\(^5\) in July 1993. Debian came second, created in August 1993, while Red Hat was created one year later\(^6\).

As a time-line reference, the GNU project startup declaration by Richard Stallman dates back to September 27, 1983\(^7\) while Linux was announced for the first time by Linus Torvalds on October 5, 1991\(^8\). Thus, in 1998, while\(^1\) was being written, no binary distribution was providing a solid coverage of multimedia applications and every sound/music fanlover/enthusiast/researcher/ professional had to get hold of Free Software sources, match dependency requirements (i.e. library versions, etc.), compile them, possibly debug/port them and then finally use them for whatever purpose.

A natural development followed\(^1\) shortly after\(^9\) musician, composer and programmer Marco Trevisani proposed to a small group of friends (Nicola Bernardini, Maurizio De Cecco, Davide Rocchesso and Roberto Bresin) to create LAOS (the acronym of *Linux Audio Open Sourc*ing\(^10\)), a binary distribution of all essential sound/music tools available at the time including website diffusion and support. LAOS came up too early, and it did not go very far.

In 2000, No Starch Press published a book by Dave Phillips devoted to sound/music in the GNU/Linux environment\(^11\). Besides providing excellent tutorial guiding, Phillips’ book developed at greater length the taxonomy introduced by\(^1\) concentrating mainly on Free Software applications: times were ripe for a successful attempt. And indeed, when Marco Trevisani proposed (this time to Nicola Bernardini, Günter Geiger, Dave Phillips and Maurizio De Cecco) to build DeMuDi *(Debian Multimedia Distribution*)\(^12\) an unofficial Debian-based binary distribution of sound/music Free Software.

Nicola Bernardini organized a workshop in Firenze, Italy at the beginning of June 2001, inviting an ever-growing group of supporters and contributors (including: Marco Trevisani, Günter Geiger, Dave Phillips, Paul Davis, François Déchelle, Georg Greve, Stanko Juzbasic, Giampiero Salvi, Maurizio Umberto Puxeddu and Gabriel Maldonado). That was the occasion to start the first concrete DeMuDi distribution, the venerable 0.0 alpha which was


\(^{2}\)This text appeared in a news originally on [http://www.emagic.de/english/news/](http://www.emagic.de/english/news/) of course, Emagic has deleted this embarrassing news a long time ago. However, the text in its integrity was pasted into the [http://www.beatmode.com/news/emagic/](http://www.beatmode.com/news/emagic/) page, so it is still available for evaluation.


\(^{4}\)http://www.sunsite.univie.ac.at/Linux soundapp/top.html, while the current edition of it may be found at [http://linux-sound.org/top.html](http://www.sunsite.univie.ac.at/Linux soundapp/top.html).

\(^{5}\)http://www.blackware.com/announce/1.0.php

\(^{6}\)http://www.debian.org/intro/about#history


\(^{8}\)http://www.gnu.org/gnu/initial-announcement.html

\(^{9}\)http://groups.google.com/groups?selm=1991Oct5.054106.11@sunsite.univie.ac.at

\(^{10}\)cf. [http://groups.google.com/groups?selm=1993Oct5.054106.467%40kaisaav.kaisaav.helsinki.fi](http://groups.google.com/groups?selm=1993Oct5.054106.467%40kaisaav.kaisaav.helsinki.fi)

\(^{11}\)The earliest evidence being a personal mail dating back to November 1st 1998

\(^{12}\)Personal mail exchanges on September 30th and October 23rd, 2000.
then quickly assembled by Günter Geiger with help from Marco Trevisani. A bootable CD-version was then burned just in time for the ICMC 2001 held in La Habana, Cuba, where Günter Geiger and Nicola Bernardini held a tutorial workshop showing features, uses and advantages of DeMuDi. Practically at the same time, Fernando Lopez-Lezcano was giving life to the PlanetCCRMA initiative. PlanetCCRMA was initially conceived as an internal CCRMA service to promote the usage of GNU/Linux running on Red Hat boxes in computer music courses. Its success quickly surpassed the walls of Stanford University to be adopted by many users worldwide, thus providing a strong alternative (albeit very different in scope and intentions) to the DeMuDi distribution.

On November 26, 2001 the European Commission awarded the AGNULA (A GNU/Linux Audio distribution) Consortium (composed by the Centro Tempo Reale, IRCAM, the IUA-MTG at the Universitat Pompeu Fabra, the Free Software Foundation Europe, KTH and Red Hat France) with consistent funding for an accompanying measure lasting 24 months (IST-2001-34879). This accompanying measure, which was terminated on March 31st 2004, gave considerable thrust to the AGNULA/DeMuDi project providing scientific applications previously unreleased in binary form and a Red Hat based distribution parallel to the Debian (termed AGNULA/ReHMuDi). After the funded period, Media Innovation Unit, a component of Firenze Tecnologia (itself a technological agency of the Chamber of Commerce of Firenze) has decided to partly fund further AGNULA/DeMuDi developments. AGNULA has constituted a major step in the direction of creating a full-blown Free Software infrastructure devoted to audio, sound and music, but there’s much more to it: it is the first example of a European-funded project to clearly specify the complete adherence of its results to the Free Software paradigm in the project contract, thus becoming an important precedent for similar projects in the future.

Nowadays, Free Software multimedia distributions are enjoying an outstanding success: new attempts of different size and purpose are started all over the world, and the future of Free Software for sound and music does indeed look bright. Among these, it is worthwhile to mention the dyne:bolic [14] distribution, another 100% Free Software distribution which enjoyed some European Community funding too through the sponsorship by the PublicVoiceXML project (IST-2001-34546), and the Medialinux [15] distribution from the Open Source Lab of Virtual Reality & Multi Media Park in Torino, Italy.

4. CURRENT STATUS

Given the successes of Free Software in the field of sound and music, we will attempt to give a short overview of the best Free Software available in each of the categories outlined in [1]. However, the reader must be aware that this is only a “best of” choice: distributions such as AGNULA/DeMuDi hold today a much larger variety of applications in each and every domain (cf. Figure 1).

---

Figure 1: A typical AGNULA/DeMuDi desktop

Figure 2: The Audacity editor

Sound Processing Languages: the Sound Processing Languages domain provides an exemplary illustration of the paradigm shift explained in Sec.4. While standard customary applications did not change much in six years, it was rather the paradigm shift in licensing schemes to provide for real innovation here. Whether or not as a consequence of the creation of Free Software distributions for sound and music, some of the better known and used non-free applications of the past shifted to Free Software licenses. Two brilliant examples of this trend were SuperCollider, previously non-free and running on PowerPC platforms exclusively, which was re-licensed under the GNU GPL (GNU General Public License [16]) in September 2002, while the venerable Csound [17] music compiler was re-licensed under the GNU LGPL (GNU Lesser General Public License) around the month of May 2003.

---


15http://www.dynebolic.org

16http://www.publicvoicexml.org

17An explained list of Free Software licenses may be found at http://www.fsf.org/licenses/licenses.html#LicenseList

http://www.csounds.com
Time-domain Graphical Editing and Processing: Free Software graphical editing in the time domain currently sports among the best applications in the field: Audacity (cf. Figure 2), Snd (cf. Figure 3). A major enhancement was offered with the large scale multi-track Free Software hard-disk recorder Ardour (cf. Figure 4) which is directly competing with non-free software suites whose cost runs around the three-four figures.

Analysis/Resynthesis Packages: Most of the functionalities of AudioSculpt, IRCAM flagship non-free application described in [1] have been replicated into Ceres3 (cf. Figure 5), a powerful Free Software editors in the spectral domain. Another powerful Analysis/Resynthesis package is CLAM [6], developed at Universitat Pompeu Fabra over the work previously done to invent and create the spectral modeling system SMS [7].

Interactive Graphical Building Environments: Free Software is very strong in this field with two well developed applications which have been enjoying a tremendous success for years: jMax (cf. Figure 6), direct incarnation of the Max/MSP environment, and Pure Data (better known as PD) [8]. Still, the creation of exclusive Free Software distributions such as AGNULA/DeMuDi has allowed to face and clarify licensing problems with both applications [9].

5. TECHNICAL DETAILS

All multimedia systems require very low-latency response in order to achieve real-time, and proper application precedence. While not particularly designed for real-time low-latency tasks, the Linux kernel, maintained by probably the largest development group in the world, made steep progresses in this field since 1999 onwards. A number of patches to the kernel were issued during this time, getting it to be a truly pre-emptive kernel with very low latency.

---

Figure 3: The Snd editor

Figure 4: Ardour, the hard-disk recorder

Figure 5: Ceres3, an Audiosculpt-like application

Figure 6: jMax
achieving sub-2 msec latencies under most conditions. Obviously, multimedia distributions currently offer patched kernels to achieve such latencies, effectively leading GNU/Linux to be the system of choice for multimedia endeavors.

But kernel latencies are only one of the innovative aspects of GNU/Linux system. Perhaps an even stronger aspect comes from the invention of Jack by Paul Davis, early professional audio GNU/Linux supporter. Jack is a user-space low-latency audio server, written for POSIX conforming operating systems such as GNU/Linux. It can connect a number of different applications to an audio device, as well as allowing them to share audio between themselves. Its clients can run in their own processes (i.e. as normal applications), or can they can run within the JACK server (i.e. as a “plug-in”). Jack was designed from the ground up for professional audio work, and its design focuses on two key areas: synchronous execution of all clients, and low latency operation. Figure 8 shows a graphical connection client which hooks up to the Jack server. The development of Jack spurred a number of innovative applications such as the Jack Rack (cf. Figure 9), essentially a host for LADSPA plug-ins, which can then be applied to all audio paths within the system. Another brilliant application spurred from Jack is Jamin, a module that was designed to perform high grade audio mastering of stereo input streams. Being a Jack client, it is easy to integrate it with the rest of the audio chain to allow the user to be able to change relevant parameters of the chain such as the mixing. As shown in Figure 10, the main components of this interface are a 1024 band hand drawn EQ, a 30 band graphic EQ, a 3 band peak compressor and a lookahead brickwall limiter. Furthermore, it allows smooth transition between different user’s presets to help the mastering process.

6. CONCLUSIONS

We hope to have shown convincing evidence that the past six years have witnessed a paradigm shift (from non-free to Free Software for professional sound and audio applications) that will probably lead to many changes in habits and practices in sound research and music making. While many difficulties have been overcome by the amazing work of voluntary developers, it is clear that many more are still waiting to be solved. As a conclusive statement, we can try to list some of them:

1. there is still a lot of undergoing duplicate work: while projects such as AGNULA, PlanetCCRMA or MediaLinux are a good step towards a higher coordination of forces, many different programs basically performing similar tasks; this problem is actually double:
   (a) developer’s resources get wasted;
   (b) users must perform the additional task of selecting the proper application to suit their needs; this is often
a hard and time-consuming job which musicians do not like to do;

2. most musicians tend to stick to their practised habits; thus, inertia to actually achieve a complete migration of most musicians is a much harder task than just building Free Software-based multimedia distributions;

3. in the research domain, a set of reference guidelines on the tools to be used and on the practices to follow still remain to be discussed and defined;

4. the professional music performance domain still lacks a common infrastructure that would lower the complexities of such projects; this infrastructure is missing in the non-free software realm too, where it is probably impossible to be achieved. Free Software constitutes a very good starting point to design and constitute this infrastructure – but this is something that is still to be done.

All in all, while unexpectedly outstanding leaps have been accomplished in the past six years, probably more achievements are ahead of us.

7. ACKNOWLEDGEMENTS

As the reader may expect, projects such as DeMuDi and AGNULA are the result of the common effort of a very large pool of motivated people. And indeed, giving credit to any deserving individual that contributed to these projects would probably fill completely the space allotted for this paper. Therefore, we decided to make an arbitrarily small selection of those without whose help DeMuDi and AGNULA would not probably exist. First of all, we would like to thank Richard Stallman, without whose effort Free Software would not exist at all; Linus Torvalds, who contributed the operating system we all got to depend on in the last dozen of years; and Ian Murdock who started Debian, the wonderful packaging infrastructure AGNULA/DeMuDi is based on. Then, Marco Trevisani, who has been pushing the envelope of a Free audio/music system for years, Dave Phillips, Günter Geiger, Fernando Lopez-Lezcano, François Déchelle and Davide Rocchesso: all these people have been working (and still work) on these concepts and ideas since the early days. Georg Greve, the president of Free Software Foundation Europe, has guided us through the difficulties of carrying out a rigorous Free Software project: may his patience be awarded by the warmest thanks. Other people that deserve our gratitude are: Philippe Aigrain and Jean-François Junger, the European Commission officials that have been promoting the idea that AGNULA was a viable project against all odds inside the Commission itself; Dirk Van Rooy, later AGNULA Project Officer, Marc Leman and Xavier Petrot, patient AGNULA Project Reviewers; Luca Mantellazzi and Giovanni Nebiolo, respectively President of Firenze’s Chamber of Commerce and CEO of Firenze Tecnologia, for their support: they have understood the innovative potential of Free Software much better than many so-called open-source evangelists. Finally, last but not least, Anna Meo and Nicola Giosmin, close collaborators for many years in all these (and other) endeavors, deserve all our endless esteem and gratitude: without them, a lot of these achievements would have made our life much harder than it has been.

8. REFERENCES


COMPUTER INSTRUMENT DEVELOPMENT AND THE COMPOSITION PROCESS

Victor Lazzarini
Music Technology Laboratory
NUI, Maynooth, Ireland
Victor.Lazzarini@may.ie

ABSTRACT
This text looks at the computer instrument development work and its influence on the composition process. As a preamble to the main discussion, the different types of software for sound generation and transformation are reviewed. The concept of meta-themes is introduced and explored in the context of contemporary music. Two examples of the author’s computer music work are used to discuss the complex relationship between software development and composition. The first piece provides an example of such relationships in the context of ‘tape’ music. The second explores the use of computer instruments in live electroacoustic music. The activities of composition and instrument creation will be shown to be at times indistinguishable and mutually dependent.

1. INTRODUCTION
Computer music instruments can be generally defined as programs that allow computers to generate or transform sounds, performing as an extended musical instrument. In the past fifteen years or so, composers have been greatly interested in the use of computers in live electroacoustic music. This has been mostly due to two factors: (i) the advances in microcomputer technology; (ii) the development of graphic music programming languages, from the early Max/FTS system [1] on the IRCAM workstation, to the latest Macintosh and Windows-based software synthesis environments. The former is related to the availability of increasingly better and cheaper hardware platforms. The latter factor made programming more accessible to composers less keen on tackling text-based languages to create customised computer instruments.

In general, much of the enthusiasm generated by these developments can be attributed to the fact that computers can be treated as a multi-purpose maker of instruments, rather than a device with a limited number of applications. Trevor Wishart, a composer who also defines himself as an instrument-maker, states:

“Information Technology allows us to build sound-processing tools of immense generality and flexibility (...). The “instrument” is no longer definable (if subtle) closed universe, but a groundswell of possibilities...” [2]

In response to that, the focus of the work for some composers has shifted from the traditional score-writing composition activities to incorporate the task of instrument-making into the process. In a way, electroacoustic music composers have been performing, to variable extents, the job of instrument development ever since the early days of the Musique Concrète [3] [4]. Nevertheless, it was only with the advent of Computer Music and Max Mathews’ MUSIC series of programs [5] that the task of instrument-making was made indistinguishable from the act of composition itself. It is impossible for instance to say where instrument development stops and where ‘composition’ starts in Risset’s Mutations of 1969 [6].

This article will explore how these two activities are entwined. The concept of meta-thematic processes as an important link between them is discussed. Two of the author’s computer music works, Mouvements and The Trane Thing, are used as practical examples. As a background to this discussion, different types of computer software systems are examined in terms of what they offer for composers interested in instrument-making.

2. SOFTWARE FOR SOUND GENERATION AND TRANSFORMATION
The range of available computer music software for sound generation and transformation is vast. We can separate them into three levels, in terms of flexibility, programmability and generality. At the first level, we have the ‘hard-wired’ and semi-‘hard-wired’ programs: graphic software synthesizers, audio recording software plug-ins and similar systems. These tend to offer a pre-set range of functions, some provide limited programmability in terms of ‘patches’, mimicking outboard equipment such as old modular synthesizers. Nevertheless, the majority of these tend to limit the composer to a number of choices dictated by the musical concepts of the software designer. Their use as computer instruments is limited. Many of these programs are responsible for the dissemination of clichés and predictable uses that plague much of the music made with computers today.

At the middle level, we find the programming languages, such as MaxMSP, SuperCollider [7], csound [8], Nyquist [9], etc. These are quite distinct from the programs discussed above by the fact that they are reasonably open-ended and programmable. They do not offer a fixed set of applications, but can be used to create computer instruments, by providing building blocks commonly known as unit generators (ugens). The level of programmability can vary quite a lot between these systems. At the higher level, we find the graphic programming packages such as MaxMSP, Pure Data and jMax. These are loosely based on the object-oriented programming model, offering classes from which objects can be instantiated and some extensibility by the use of class composition. At the middle level, we have the text-based language csound, which offers some more programmability and an extensive collection of ugens. Csound suffers slightly from the fact that it uses some outdated programmability and an extensive collection of ugens.
3. META-THEMES AND CONTEMPORARY MUSIC

Contemporary music, since the early part of the 20th century has been characterised by the use of meta-thematic processes. In fact, this is one of the defining elements of modern and post-modern music, which sets them apart from the so-called period of common practice (Baroque to Romantic). While the traditional approach had previously been focused on thematic composition, since Schoenberg, meta-thematic techniques have dominated much of contemporary music.

Meta-themes are generative principles rather than explicit musical statements. They can be instantiated in thematic or non-thematic forms. Elaboration of meta-themes often takes place at a pre-compositional stage; however, in many cases they can arise as part of the music writing process. Examples of meta-thematic procedures are found in different styles and genres, from dodecaphonic music to Messiaen’s parametrisation of musical material to aleatoric and process composition.

A composition can be based on a single meta-theme, or on a series of meta-thematic elements. For instance, in the case of serial music, there is often a single permutational principle as a generative meta-theme. A piece can also have elements with a clear meta-thematic origin combined with others that are not have such derivation. In certain examples of process music, the basic melodic material does not have any particular provenance (apart from being generally tonal or modal), while the rhythmic manipulation and formal development is derived from a generative principle.

Particularly important is the role that meta-thematic processes play in Computer Music. An early example of this is found in the already mentioned Mutations by Risset. In that piece, the guiding principle, according to the composer is that

“Mutations refers to the gradual transformation which occurs throughout the piece, and to the passage from a discontinuous pitch scale, at the beginning, to the pitch continuum at the last part.”[12]

This meta-thematic idea is further refined prior to the realisation of the sonic structures that compose the piece. Moreover, it serves as the support for the development of the computer instruments used in the piece. It is at the meta-thematic level that we find the basic design for both composition and instruments. This is indeed, typical of Risset and other Computer Music composers, in that the roles of composer and instrument designer become entangled. Meta-themes can operate as a link between the two activities.

A recent work by Rajmil Fischman [13] illustrates the point even more completely. In his article, a complete description of meta-thematic procedures leading to their sonic realisation through computer instruments is shown. The musical objective was to deal with the application of Quantum physics concepts in the generation of stochastic music structure with granular synthesis. A single equation (Schroedinger’s Equation) was used as the basic generative principle for instrumental (synthesis & processing), sonic and musical development. This work is also significant that it points to generalised connections between a particular meta-thematic idea and their concrete instrumental (sound-generative) and musical (structure-generative) instances.

From the above discussion, it might be inferred that the use of meta-themes indicate a bias towards a total-organisation approach to composition. This is, however, not true, as it is equally possible to detect such elements in music not characterised by structuralist methods. In both Fischman’s and Risset’s examples, in fact, structure is dictated by sonic preoccupations rather than purely organisational ones. In addition, as hinted before, meta-thematic coherence can arise from the compositional process, even when it is not explicitly sought. Many examples of this are found in the music of the acousmatic school.

4. COMPOSITION EXAMPLE 1: MOVEMENTS (FOR 8-CHANNEL TAPE)

The piece Movements is an example of the use of csound in sound and structure composition. That system was used to generate all the synthesised sounds, which were subsequently put together using a multitrack program. The fact that csound is a quite flexible tool for creating sounds was crucial in the composition process. This Section will outline the main elements involved in the development of the piece.

This piece is structured around a single principle, its meta-theme, which governs the synthesis procedures used, resulting in music elements that are linked together by a certain audible trait. This generative principle (and its different implementations) was developed step-wisely. The basic form for it was elaborated by modifying a Shepard-tone instrument design [13]. This instrument creates the acoustic illusion of a glissando that continuously descends or ascends without ever reaching an end point. This particular design created these glissandos by using ten sine-wave oscillators tuned in octaves, constantly sliding down ten octaves from the top-most frequency. The illusion is created by using an
overall spectral envelope, in the shape of a Gaussian window, which attenuates the lower and higher ends of the glissando. The first modification made to the instrument was the elimination of the frequency glide, which created a pedal tone constantly changing in octave. In addition, the interval between each oscillator was made variable. With this modification, they could be separated by any frequency ratio, not only octaves. This instrument design became the basis on which all other instruments were created for the piece. Because they share the same principle, their sonic output will also share the same gestural and textural traits.

1) The original one with 10 sinewave oscillators for sound generation.
2) Using a 10-piece filterbank with white noise as input; the evenly spaced frequencies were re-interpreted as the filter centre frequencies.
3) Using 10 FOF generators: This uses the base frequency ($f_0$) as the fundamental, above which 10 formant frequencies are placed, following the original spacing rule (thus making the highest 11 times the interval over the fundamental).
4) Same as (1) except that the interval ratio is made to vary randomly 10 times/sec.
5) Same as (2) except that the amp envelope table lookup position for each signal generator is controlled by a random number generator. In (2) [and all other instruments], this was controlled by a constantly cycling variable going from start to end or start of the envelope.
6) Using a 10-carrier FM set-up. The evenly spaced frequencies are interpreted in two ways: (i) for carriers 2-5, they are effectively formant frequencies; (ii) for carriers 6-10 they are taken as the carrier frequency, directly.
7) Same as (6), except that only carriers 2-3 are using formant frequencies.
8) Same as (6), except that all carrier frequencies are set directly to the evenly spaced frequency values.
9) Same as (3) with same randomised and variable parameter (FOF grain size and ‘local’ envelope).
10) Same as (9) with a different method of randomisation.
11) Same as (10) with some more variable parameters.

The different implementations listed above were developed in response to musical needs. For instance, as the start gesture for the piece was being composed, it was felt that a richer spectral content would be more suitable for it. This was then realised by using filtered white noise instead of sineswages, giving origin to the first variant on the basic model. In the middle section, the continuous textures were supposed to be interrupted, so an instrument that could generate more percusive sounds was developed. By keeping to the general design and varying the sound synthesis technique, a variety of interconnected musical elements were generated. Depending on the choice of parameters (spectral envelope shape, sweep rate, etc.) and instrument design, different types of gestures and textures can be created: spectral ‘arpeggios’, glissandos, drones, chords, timbral mutations, etc.

Here, it is clear that instrument design is crucial to the composition process, being, in fact, seamlessly integrated to it. It not only responds to structural needs, but also informs the creative activities by providing the different ways of articulating ideas. The use of csound was also decisive, because it provided a flexible platform for the development of these instruments.

In terms of its overall structure, this piece uses conventional composition techniques such as variation/development in an extended manner. There is no such thing as a theme, but a working principle, a meta-theme which that links all sounds, textures and gestures in the piece. It is used to generate an overall sense of unity, without recourse to traditional compositional methods. Structural ideas similar to this one are found in many examples of computer music. In fact, this method of work was probably born out of the possibilities brought on by computer instrument development. Mouvements was composed as a tribute to J C Risset.

Figure 1: The four window shapes used

This basic design was used as the starting point for the composition of Mouvements. Some further modifications were made to it. The single-shape spectral envelope was substituted with a choice of four window types: Gaussian (the original), Bartlett (triangle), Blackman-Harris and Kaiser (Fig. 1). It was also made possible to vary the base frequency and the period of the envelope cycle in time, so that a moving gesture could be articulated. The sound output of each oscillator was also panned between pairs of speakers in parallel with the amplitude envelope. In addition, the direction of the envelope cycle (upwards/downwards) and a sound fade in/out time could also be set. This became the first implemented instrument used in the piece. The overall design, which will be shared between instruments, can be described as having the following characteristics (Fig. 2):

- 10 parallel audio signal generators with evenly spaced, but variable, frequency intervals;
- An overall spectral envelope (determined by the window shape) that cyclically sweeps from one end of the spectrum to the other at different speeds, emphasizing/de-emphasizing the sound of each signal generator;
- Individual spatial placement of the spectral components (also controlled by the envelope cycle).

Eleven different implementations of this design model were developed. They have the following characteristics:
5. COMPOSITION EXAMPLE 2: THE TRANE THING (FOR TENOR SAXOPHONE AND COMPUTER)

The composition of The Trane Thing is an example of how research activities in computer music can be linked to an artistic output. The piece was composed as a duet for saxophone and computer instrument. This was developed as a stand-alone piece of software created using the Sound Object (SndObj) Library, a set of programming tools developed as part of the author’s research. The library was used to provide the audio processing aspect of the instrument. The graphical user interface was built using the Microsoft© Foundation Classes framework. The instrument was custom-built to the needs of the composition. The use of software development tools for its creation was essential, because it provided a way to shape the instrument to match exactly the needed specification.

A basic instrument was created and these ideas were tested with some saxophone material. Slowly, the sonic shape of the piece started to crystallise. It was soon realised that more 'landscape' sounds were needed to involve the saxophone. Four sampling operators were then included to capture and replay some saxophone material. These completed the original goal of immersing the saxophone in a pool of sound. They also provided the means of capturing improvised gestures, something that became a compositional interest from the very start of the work. In fact, improvisation turned out to be one of the defining elements of the score, as this grew to become a tribute to the late American saxophonist John Coltrane (Figure 4). The contemplative, slightly modal, sound of the piece was certainly influenced by his music.

The instrument design is relatively simple and uses basically two main components, a string resonator and a sampling operator, as seen in Figure 3. The main idea was to create layers of pedal sounds behind the saxophone lines that would constantly change according to the notes that composed those lines. The string resonators were ideal for this purpose, because they could be tuned to structurally significant frequencies and would react to them and to their harmonics. A decision was made to limit the instruments to four strings, for the practical reason that more strings make the result more undistinguished. The use of several resonators creates a diffuse reverberation field. This, of course, was not the intended result, as the main job of the string resonators was to create drones and sustained harmonies.

The user interface was also designed in parallel to the piece composition, according to what actions were required by the score. A few extra elements (frequency and pitch adjustments etc.) were also included, but they did not find a specific use in the piece. The instrument window is shown on Fig. 5. An important point was to make it playable, so the controls were made very accessible and easy to use. MIDI control was not included, in order to keep the required set-up to a minimum, so that the piece could be performed more often. It only requires a computer (laptop or desktop) running Windows operating system and a reasonably good soundcard. The fact that it has been performed quite often (in Ireland and abroad) is in part due to these careful considerations.

This work shows an example of how instrument development fits into the composition process. On one hand, it influences how the music is composed by providing part of its sonic characteristics, on the other it is influenced by the requirements of the score. In fact, these two elements of the composition work, score writing and computer programming, are mutually dependent. As they evolved together, there is a special type of integration being
forged between the music and its instrumentation. This is something that would not be otherwise possible. The ability of a composer to perform both tasks, score-writing and programming, is essential. The musical result is then solely in his hands.

![The Trane Thing instrument and its main controls](image)

This piece was not written explicitly using a meta-thematic basis developed at a preliminary stage. However, arising from the composition process itself, there are meta-thematic elements that define the soundscape of the piece: use of Coltrane-like melodic fragments, the resonating environment, the use of open-end sections and audio capture/replay. Again, these form a groundwork that connects the activities, of computer instrument development and composition.

6. CONCLUSION

This text has discussed some aspects of the use of computer instruments in music composition. It was shown that an important element of contemporary music composition is the concept of meta-thematic development. In Computer Music, this can serve as basic link between computer instrument design and music structure.

Using two examples, the task of instrument development and its place in the composition process were explored. The first example discussed it in relation to tape music composition and the second in relation to live electroacoustic music. It was shown that the activities of composition and instrument creation could be, in some situations, indistinguishable and, in others, mutually dependent. This discussion also points to the emergence of a new kind of composer, versed not only in the traditional aspects of composition, but also in signal processing and computer programming.

There are many further issues relating to the relationship between instrument-making and composition. These range from technical to educational, aesthetic and analytic ones. Not much attention has been given, for instance, as to how computer instrument development should be taught in the ever-expanding third-level music technology area. Similarly, studies have so far only scratched the surface in relation to the aesthetic implications of a music practice that incorporates instrument-making by default. It is hoped that research in this area will be instigated by further work from music scholars, educators and composers. In such circumstances, it is expected that some of these questions will be addressed more thoroughly in future research.

7. REFERENCES

EVENT-SYNCHRONOUS MUSIC ANALYSIS / SYNTHESIS

Tristan Jehan
Massachusetts Institute of Technology
Media Laboratory
tristan@media.mit.edu

ABSTRACT
This work presents a novel framework for music synthesis, based on the perceptual structure analysis of pre-existing musical signals, for example taken from a personal MP3 database. We raise the important issue of grounding music analysis on perception, and propose a bottom-up approach to music analysis, as well as modeling, and synthesis. A model of segmentation for polyphonic signals is described, and is qualitatively validated through several artifact-free music resynthesis experiments, e.g., reversing the ordering of sound events (notes), without reversing their waveforms. Then, a compact “timbre” structure analysis, and a method for song description in the form of an “audio DNA” sequence is presented. Finally, we propose novel applications, such as music cross-synthesis, or time-domain audio compression, enabled through simple sound similarity measures, and clustering.

1. INTRODUCTION
Music can be regarded as a highly complex acoustical and temporal signal, which unfolds through listening into a sequential organization of perceptual attributes. A structural hierarchy [1], which has been often studied in the frequency domain (i.e., relationship between notes, chords, or keys) and the time domain (i.e., beat, rhythmic grouping, patterns, macrostructures) demonstrate the intricate complexity and interrelationship between the components that make music. Few studies have proposed computational models on the organization of timbres in musical scenes. However, it was shown by Deliège [2] that listeners tend to prefer grouping rules based on timbre over other rules (i.e., melodic and temporal) and by Lerdahl in [3] that music structures could also be built up from timbre hierarchies.

Here we refer to timbre as the sonic “quality” of an auditory event, that distinguishes it from other events, invariably of its change in pitch or loudness. From an auditory scene analysis point of view, by which humans build mental descriptions of complex auditory environment, an abrupt event is an important sound source separation cue. Auditory objects get first separated and identified on the basis of common dynamics and spectra. Then, features such as pitch and loudness are estimated [4]. Moreover, the clear separation of sound events in time makes music analysis and its representation easier than if we attempted to model audio and music all at once.

Segmentation has proven to be useful for a range of audio applications, such as automatic transcription [5], annotation [6], sound synthesis [7], or rhythm and beat analysis [8] [9]. Data-driven concatenative synthesis consists of generating audio sequences by juxtaposing small units of sound (e.g., 150 ms), so that the result best matches a usually longer target sound or phrase. The method was first developed as part of a text-to-speech (TTS) system, which exploits large databases of speech phonemes in order to reconstruct entire sentences [10].

Schwarz’s Caterpillar system [7] aims at synthesizing sounds with the concatenation of musical audio signals. The units are segmented via alignment, annotated with a series of audio descriptors, and are selected from a large database with a constraint solving technique.

Zils and Pachet’s Musical Mosaicing [11] aims at generating music with arbitrary samples. The music generation problem is seen as a constraint problem. The first application proposed composes with overlapping samples by applying an overall measure of concatenation quality, based on descriptor continuity, and a constraint solving approach for sample selection. The second application uses a target song as the overall set of constraints.

Lazier and Cook’s MoSievius system [12] takes up the same idea, and allows for real-time interactive control over the mosaicing technique by fast sound sieving: a process of isolating subspaces as inspired by [13]. The user can choose input and output signal specifications in real time in order to generate an interactive audio mosaic. Fast time-stretching, pitch shifting, and k-nearest neighbor search is provided. An (optionally pitch-synchronous) overlap/add technique is used for synthesis.

Few or no audio examples with these systems were available. Lazier’s source code is however freely available online. Finally, a real world example of actual music generated with small segments collected from pre-existing audio samples is among others, John Oswald’s Plunderphonics project. He created a series of collage pieces by cutting and pasting samples by hand [14].

2. AUDITORY SPECTROGRAM
Let us start with a monophonic audio signal of arbitrary sound quality—since we are only concerned with the musical appreciation of the audio by a human listener, the signal may have been formerly compressed, filtered, or resampled—and any musical content—we have tested our program with excerpts taken from jazz, classical, funk, pop music, to speech, environmental sounds, or simple drum loops. The goal of our auditory spectrogram is to convert the time-domain waveform into a reduced, yet perceptually meaningful, time-frequency representation. We seek to remove the information that is the least critical to our hearing sensation while retaining the important parts, therefore reducing signal complexity without perceptual loss. An MP3 codec is a good example of application that exploits this principle for compression purposes. Our primary interest here is segmentation (see Section 3), therefore the process is being simplified.

First, we apply a standard STFT to obtain a regular spectro-
gram. Many window types and sizes have been tested, which did not really have a significant impact on the results. However, since we are mostly concerned with timing accuracy, we favor short windows (e.g., 12 ms Hanning), which we compute every 3 ms (i.e., every 128 samples at 44.1 KHz). The FFT is zero-padded up to 46 ms to gain additional interpolated frequency bins. We now calculate the power spectrum, and then group and convert resulting bins into 25 critical-bands according to a Bark scale—see equation (1). At low frequencies, critical bands show an almost constant width of about 100 Hz while at frequencies above 500 Hz, they show a bandwidth which is about 20% of the center frequency [15].

\[
z(f) = 13 \cdot \arctan(0.00076f) + 3.5 \cdot \arctan\left(\frac{f}{7500}\right)^2 \tag{1}
\]

A non-linear spreading function is calculated for every frequency band with equation (2) [16]. The function models frequency masking and may vary depending on the refinement of the model. More details can be found in [17].

\[
SF(z) = (15.81 - i) + 7.5(z + 0.474) - (17.5 - i)\sqrt{1 + (z + 0.474)^2} \tag{2}
\]

where

\[
i = \min(5 \cdot |F(f)| \cdot BW(f), 2.0), \quad \text{and}
\]

\[
BW(f) = \begin{cases} 
100 & \text{for } f < 500 \\
0.2f & \text{for } f \geq 500 
\end{cases}
\]

Another perceptual phenomenon that we consider as well is temporal masking, and particularly post-masking. The envelope of each critical-band is convolved with a 200-ms half-Hanning (i.e., raised cosine) window. This stage induces smoothing of the spectrogram, while preserving attacks. The output merely approximates a “what-you-see-is-what-you-hear” type of spectrogram, meaning that the “just visible” in the time-frequency display (see Figure 1, frame 2) corresponds to the “just audible” in the underlying sound. The spectrogram is finally normalized to the range 0-1.

Among perceptual descriptors commonly exploited stands out loudness: the subjective judgment of the intensity of a sound. It can be approximated by the area below the masking curve. We can simply derive it from our spectrogram by adding the energy of each frequency band (see Figure 1, frame 3).

3. SEGMENTATION

Segmentation is the means by which we can divide the musical signal into smaller units of sound. When organized in a particular order, the sequence generates music. Since we are not concerned with sound source separation at this point, a segment may represent a rich and complex polyphonic sound, usually short.

We define a sound segment by its onset and offset boundaries. It is assumed perceptually “meaningful” if its timbre is consistent, i.e., it does not contain any noticeable abrupt changes. Typical segment onsets include abrupt loudness, pitch or timbre variations. All of these events translate naturally into an abrupt spectral variation in our auditory spectrogram.

First, we convert the spectrogram into an event detection function. It is obtained by first calculating the first-order difference function for each spectral band, and then by summing these envelopes across channels. The resulting signal contains peaks, which correspond to onset transients (see Figure 1 frame 4). We smooth that signal in order to eliminate irrelevant sub-transients (i.e., sub-peaks) which, within a 50 ms window would perceptually fuse together. That filtering stage is implemented by convolving the signal with a Hanning window (best results were obtained with a 150-ms window). This returns a smooth function, now appropriate for the peak-picking stage. The onset transients are found by extracting the local maxima in that function (see Figure 1, frame 5). A small arbitrary threshold could be necessary to avoid smallest undesired peaks, but its choice should not be critical.

Since we are concerned with reusing the audio segments for synthesis, we now refine the onset location by analyzing it in relationship with its corresponding loudness function. An onset would typically occur with an increase in loudness. To retain the entire attack, we search for the previous local minimum in that signal (i.e., usually a small shift of less than 20 ms), which corresponds to the softest moment before the onset (see Figure 1, frame 3). Finally,
we look in the corresponding waveform, and search for the closest zero-crossing, with an arbitrary but consistent choice of direction (e.g., negative to positive). This stage is important to insure signal continuity at synthesis (see Section 5).

4. BEAT TRACKING

Our beat-tracker was mostly inspired by Eric Scheirer’s [18] and assumes no knowledge beforehand. For instance, it does not require a drum track, or a bass line to perform successfully. However, there are differences in the implementation which are worth mentioning. First, we use the auditory spectrogram as a front-end analysis technique, as opposed to a filterbank of six sixth-order elliptic filters, followed by envelope extraction. The signal to be processed is believed to be more perceptually grounded. We also use a large bank of comb filters as resonators, which we normalized by integrating the total energy possibly contained in the delay line, i.e., assuming DC signal. A salience parameter is added which allows us to estimate if there’s a beat in the music at all. For avoiding tempo ambiguity (e.g., octaves), we use a template mechanism to select the faster beat, as it gives more resolution to the metric, and is easier to down-sample if needed.

For avoiding tempo ambiguity (e.g., octaves), we use a template mechanism to select the faster beat, as it gives more resolution to the metric, and is easier to down-sample if needed.

Figure 2: Beat tracking of a 27 sec. excerpt of Watermelon man by Herbie Hancock. [from top to bottom] 1) the waveform (blue) and the beat markers (red); 2) the tempogram; 3) the tempo spectrum after 16 sec. of tracking.

Figure 2 shows an example of beat tracking a polyphonic jazz-fusion piece at roughly 143 BPM. A tempogram (frame 2) displays the knowledge of tempo gained over the course of the analysis. First, there is no knowledge at all, but slowly the tempo gets clearer and stronger. Note in frame 1 that beat tracking was accurately stable after merely 1 second. The 3rd frame displays the output of each resonator. The strongest peak is the extracted tempo. A peak at the sub octave (72 BPM) is visible, as well as some other harmonics of the beat.

5. MUSIC SYNTHESIS

The motivation behind this preliminary analysis work is primarily synthesis. We are interested in composing with a database of sound segments—of variable sizes, typically ranging from 60 to 300 ms—which we can extract from a catalog of musical samples and pieces (e.g., an MP3 database), and which can be rearranged in a structured, and musically meaningful sequence, e.g., derived from the larger timbre, melodic, harmonic, and rhythmic structure analysis of an existing piece, or a specific musical model (another approach to combining segments could consist for instance of using generative algorithms).

In sound jargon, the procedure is known as analysis-resynthesis, and may often include an intermediary transformation stage. For example, a sound is analyzed through a STFT and decomposed in terms of its sinusoidal structure, i.e., a list of frequencies and amplitudes changing over time, which typically describes the harmonic content of a pitched sound. This represents the analysis stage. The list of parameters may first be transformed, e.g., transposed in frequency, or shifted in amplitude, and is finally resynthesized: a series of oscillators are tuned to each frequency and amplitude, and are summed to generate the waveform.

We extend the concept to “music” analysis and resynthesis, with structures derived from timbre which motivated the need for segmentation. A segment represents the largest unit of continuous timbre. We believe that each segment could very well be resynthesized by known techniques, such as additive synthesis, but we are only concerned with the issue of music synthesis, i.e., the structured juxtaposition of sounds over time, which implies higher level (symbolic) structures. Several qualitative experiments have been implemented, to demonstrate the advantages of a segment-based music synthesis approach over an indeed more generic, but still ill-defined frame-based approach.

5.1. Scrambled Music

This first of our series of experiments assumes no structure or constraint whatsoever. Our goal is to synthesize an audio stream by randomly juxtaposing short sound segments previously extracted from an existing piece of music—typically 2 to 8 segments per second with the music that was tested.

At segmentation, a list of pointers to audio segments is created. Scrambling the music consists of rearranging randomly the sequence of pointers, and of reconstructing the corresponding waveform. There is no segment overlap, windowing, or cross-fading involved, as generally the case with granular synthesis to avoid discontinuities. Here the audio signal is not being processed. Since segmentation was performed perceptually at a strategic location (i.e., just before an onset, at the locally quietest moment, and at zero-crossing), the transitions are artifact-free and seamless.

While the new sequencing generates the most unstructured music, the event-synchronous synthesis approach permitted us to avoid generation of audio clicks and glitches. This experiment is arguably regarded as the “worst” possible case of music resynthesis; yet the result is audiowise adequate to hearing (see Figure 3).

The underlying beat of the music, if any, represents a perceptual metric on which the segment structure fits. While beat tracking was found independently of the segment structure, the two representations are intricately interrelated with each other. The same scrambling procedure can be applied to the beat segments (i.e., audio segments separated by two beat markers).
A new list of pointers to beat segments is created for the beat metric. If a beat marker occurs at less than 10% of the beat from a segment onset, we relocate the marker to that segment onset—strategically a better place. If there is no segment marker within that range, it is likely that there is no onset to be found, and we relocate the beat marker to the closest zero-crossing in order to minimize possible discontinuities. We could as well discard that beat marker altogether.

We apply the exact same scrambling procedure on that list of beat segments, and generate the new waveform. As predicted, the generated music is now metrically structured, i.e., the beat is found again, but the harmonic, or melodic structure are now scrambled. Compelling results were obtained with samples from polyphonic african, latin, funk, jazz, or pop music.

5.2. Reversed Music

The next experiment consists of adding simple structure to the previous method. This time, rather than scrambling the music, the segment order is entirely reversed, i.e., the last segment comes first, and the first segment comes last. This is much like what we could expect to hear when playing a score backwards, starting with the last note first, and ending with the first one. This is however very different from reversing the audio signal, which distorts the perception of the sound events since they start with an inverse decay, and end with an inverse attack (see Figure 4).

The method has been tested successfully on several types of music including drum, bass, and saxophone solos, classical, jazz piano, polyphonic folk, pop, and funk music. It was found that perceptual issues with unprocessed reversed music occur with overlapping sustained sounds, or long reverb—some perceptual discontinuities cannot be avoided.

This experiment is a good test bench for our segmentation. If the segmentation failed to detect a perceptually relevant onset, the reversed synthesis would fail to play the event at its correct location. Likewise, if the segmentation detected irrelevant events, the reversed synthesis would sound unnecessarily granular.

The complete procedure, including segmentation and reordering, was run again on the reversed music. As predicted, the original piece was always recovered. Only little artifacts were encountered, usually due to a small time shift with the new segmentation, then resulting into slightly noticeable jitter and/or audio residues at resynthesis. Few re-segmentation errors were found. Finally, the reversed music procedure can easily be extended to the beat structure as well, and reverse the music while retaining a metrical structure.

5.3. Time-Axis Perceptual Redundancy Cancellation

A perceptual multidimensional scaling (MDS) of sound is a geometric model which allows the determination of the Euclidean space (with an appropriate number of dimensions) that describes the distances separating timbres as they correspond to listeners’ judgments of relative dissimilarities. It was first exploited by Grey [19] who found that traditional monophonic pitched instruments could be represented in a three-dimensional timbre space with axes corresponding roughly to attack quality (temporal envelope), spectral flux (evolution of the spectral distribution over time), and brightness (spectral centroid).

Similarly, we seek to label our segments in a perceptually meaningful and compact, yet sufficient multidimensional space, in order to estimate their similarities in the timbral sense. Perceptually similar segments should cluster with each other and could therefore hold comparable labels. For instance, we could represent a song with a compact series of audio descriptors (much like a sort of “audio DNA”) which would relate to the segment structure. Close patterns would be comparable numerically, (much like two protein sequences).

Thus far, we have only experimented with simple representations. More in-depth approaches to sound similarities and low level audio descriptors may be found in [20] or [21]. Our current representation describes sound segments with 30 normalized dimensions, 25 derived from the average amplitude of the 25 critical bands of the Bark decomposition, and 5 derived from the loudness envelope (i.e., loudness value at onset, maximum loudness value, location of the maximum loudness, loudness value at offset, length of the segment). The similarity between two segments is calculated with a least-square distance measure.

With popular music, sounds tend to repeat, whether they are digital copies of the same material (e.g., a drum loop), or simply musical repetitions with perceptually undistinguishable sound variations. In those cases, it can be appropriate to cluster sounds that are very similar. Strong clustering (i.e., small number of clusters compared with the number of original data points) is useful to describe a song with a small alphabet and consequently get a rough but compact structural representation, while more modest clustering (e.g., that is more concerned with perceptual dissimilarities), would only combine segment that are very similar with each other.

While modern lossy audio coders efficiently exploit the limited perception capacities of human hearing in the frequency domain [17], they do not take into account the perceptual redundancy of sounds in the time domain. We believe that by canceling such redundancy, we not only reach further compression rates, but since the additional reduction is of different nature, it would not affect “audio” quality per se. Indeed, with the proposed method, distortions if any, could only occur in the “music” domain, that is a quantization of “timbre”, coded at the original bit rate. It is obviously arguable that musical distortion is always worse than audio distortion, however distortions if they actually exist (they would not if the sounds are digital copies), should remain perceptually undetectable.
We have experimented with redundancy cancellation, and obtained perfect resynthesis with simple cases. For example, if a drum beat (even complex and poly-instrumental) is looped more than 10 times, the sound file can easily be reduced down to 10% of its original size with no perceptual loss. More natural excerpts of a few bars were tested with as low as 30% of the original sound material, and promising results were obtained (see Figure 5). The more abundant the redundancies, the better the segment ratio, leading to higher compression rates. Our representation does not handle parametric synthesis yet (e.g., amplitude control), which could very much improve the results. Many examples were purposely over compressed in order to generate musical artifacts. These would often sound fine if the music was not known ahead of time. More on the topic can be found in [22].

![Figure 5](image1.png)

**Figure 5:** [top] Original auditory spectrogram of 11 sec. of an african musical excerpt (guitar and percussion performed live), and its corresponding loudness function. [bottom] Resynthesized signal’s auditory spectrogram with only 10% of the original material, and its corresponding loudness function.

### 5.4. Cross-synthesis

Cross-synthesis is a technique used for sound production, whereby one parameter of a synthesis model is applied in conjunction with a different parameter of another synthesis model. Physical modeling, linear predictive coding, or the vocoder for instance enable cross-synthesis.

We extend the principle to the cross-synthesis of music, much like in [11], but we *event-synchronize* segments at synthesis rather than using arbitrary segment lengths. We first generate a source database from the segmentation of a piece of music, and we replace all segments of a *target* piece by the most similar segments in the source. Each piece can be of arbitrary length and style.

The procedure relies essentially on the efficiency of the similarity measure between segments. Ours takes into account the frequency content as well as the time envelope, and performs fairly well with the samples we have tested. A more advanced technique based on dynamic programming is currently under development. We have experimented with cross-synthesizing pieces as dissimilar as a guitar piece with a drum beat, or a jazz piece with a pop song. Finally our implementation allows to combine clustering (Section 5.3) and cross-synthesis together—the target or the source can be pre-processed to contain fewer sounds, yet contrasting ones.

The results that we obtained were inspiring, and we believe they were due to the close interrelation of rhythm and spectral distribution between the target and the cross-synthesized piece. This interconnection was made possible by the means of synchronizing sound events (from segmentation) and similarities (see Figure 6).

Many sound examples for all the applications that were described in this paper, all using *default* parameters, are available at: [http://www.media.mit.edu/~tristan/DAFx04/](http://www.media.mit.edu/~tristan/DAFx04/)

![Figure 6](image2.png)

**Figure 6:** Cross-Synthesis between an excerpt of Kickin’ back by Patrice Rushen (source) and another excerpt of Watermelon man by Herbie Hancock (target). [top] The target waveform, its auditory spectrogram, and its loudness function. [bottom] The cross-synthesized waveform, its auditory spectrogram, and its loudness function. Note the close timing and spectral relationship between both pieces although they are made of different sounds.
6. IMPLEMENTATION

The several musical experiments described above easily run in a stand-alone Mac OS X application through a simple GUI. That application was implemented together with the Skeleton environment: a set of Obj-C/C libraries primarily designed to speed up, standardize, and simplify the development of new applications dealing with the analysis of musical signals. Grounded upon fundamentals of perception and learning, the framework consists of machine listening, and machine learning tools, supported by flexible data structures and fast visualizations. It is being developed as an alternative to more generic and slower tools such as Matlab, and currently includes a collection of classes for the manipulation of audio files (SndLib), FFT and convolutions (Apple’s vDSP library), k-means, SVD, PCA, SVM, ANN (nodeLib), psychoacoustic models, perceptual descriptors (pitch, loudness, brightness, noisiness, beat, segmentation, etc.), an audio player, and fast and responsive openGL displays.

7. CONCLUSION

The work we have presented includes a framework for the structure analysis of music through the description of a sequence of sounds, which aims to serve as a re-synthesis model. The sequence relies on a perceptually grounded segmentation derived from the construction of an auditory spectrogram. The sequence is embedded within a beat metric also derived from the auditory spectrogram. We propose a clustering mechanism for time-axis redundancy cancellation, which applies well to applications such as audio compression, or timbre structure quantization. Finally, we qualitatively validated our various techniques through multiple synthesis examples, including reversing music, or cross-synthesizing two pieces in order to generate a new one. All these examples were generated with default settings, using a single Cocoa application that was developed with the author’s Skeleton library for music signal analysis, modeling and synthesis. The conceptually simple method employed, and audio quality of the results obtained, attest for the importance of timbral structures with many types of music. Finally, the perceptually meaningful description technique showed clear advantages over brute-force frame-based approaches in recombining audio fragments into new sonically meaningful wholes.

8. REFERENCES


MUSICAL COMPUTER GAMES PLAYED BY SINGING

Perttu Hämäläinen, Teemu Mäki-Patola
Telecommunications Software and Multimedia Laboratory
Helsinki Univ. of Technology, Espoo, Finland
{pjhamala|tmakipat}@tml.hut.fi

Ville Pulkki, Matti Airas
Laboratory of Acoustics and Audio Signal Processing
Helsinki Univ. of Technology, Espoo, Finland
{ville.pulkki|matti.airas}@hut.fi

ABSTRACT

Although voice has been used as an input modality in various user interfaces, there are no reports of using pitch of the user’s voice for real-time control of computer games. This paper explores pitch-based control for novel games for musical education. Mapping pitch to the position of a game character provides visual feedback that helps you to learn to control your voice and sing in tune. As demonstrated by two example games in this paper, the approach can be applied to both single and two-player games even with just one microphone.

1. INTRODUCTION

Voice has been used as an input modality in various user interfaces. Speech recognition is nowadays used in games and other applications for wide audiences. The technology has matured so that simple recognition of a limited set of spoken commands can be implemented with little technological expertise using tools like the Microsoft Speech Application SDK [1].

An alternative to recognizing spoken commands is using sound pressure level, pitch, and other features of voice for immediate, real-time control. Igarashi and Hughes call this approach “Voice as Sound” [2] and give examples, such as a television volume control where the user says “Volume up, ahhh” and the volume continues to increase as long as the “ahhh” continues. Hämäläinen and Höysniemi have used the approach in a multimodal perceptual game user interface, where you control a dragon that flies when you flap your hands like wings and breathes fire when you shout [3]. Earlier, similar features of sound have mainly been used to control musical user interfaces. For example, in the Swim Swan performance in 1993, the pitch and loudness of a clarinet were used to control musical synthesis [4]. Speech input without actual speech recognition has been in used toys, such as a parrot that repeats words [5], for which it is necessary only to segment the input signal into words or phrases.

This paper describes two computer games that apply the “Voice as Sound” approach in the context of musical edutainment (education and entertainment). The main idea is to use the pitch of your voice to control a game so that you learn to control your voice and sing correctly with help of immediate graphical feedback, created by mapping pitch to the position of an avatar or other visual objects in the game.

Pitch has been defined as the characteristic of a sound that makes it sound high or low or determines its position on a scale [6]. Human perception of pitch is a complex process, but in case of singing, pitch corresponds to the frequency at which the vocal cords vibrate. However, the concept of pitch and the related musical vocabulary are not clear for many people. For example, if you instruct a person with no musical background to sing “higher”, he or she may just continue to sing at the same pitch, but use a different vowel.

In the following, we first describe the two example games, shown in Figures 1 and 2. We then discuss the technological aspects and our experiences from testing the games.

2. THE HEDGEHOG GAME

The Hedgehog game in Figure 1 is a part of Soittopeli (PlaySingMusic), a musical edutainment CD-ROM released in Finland in 2000 by Elmox Ltd. PlaySingMusic also features other games played with a MIDI keyboard, but they are beyond the scope of this paper. The game is aimed at 5 to 10 year old children.

The game is designed to provide a fun way of learning to sing. A song is played and the scenery scrolls from right to left in sync with the music. The player has to sing so that the hedgehog stays on the path that twists along the melody. As the vertical position of the hedgehog corresponds directly to pitch, the player gets immediate visual feedback that helps in finding the right pitch.

Figure 1: A screenshot of the hedgehog game. The vertical position of the hedgehog is controlled by the pitch of the user’s voice. You sing the song correctly if the hedgehog stays on the path twisting along the melody. The words of the song are shown at the bottom of the screen.
The feedback helps particularly when the player has some idea of going up and down with the melody, but the intervals and key are not right. The game grades the player’s performance as the average distance from the correct notes and the player is awarded with different fanfares and prizes after the song. The expression of the hedgehog also changes according to the performance, shown in the bottom-right corner of the screen.

3. PITCH-CONTROLLED PONG

Pitch can be used to control practically any game with one-dimensional input. We created the pitch-controlled Pong game in Figure 2 to provide a simple game for learning the concept of pitch and to experiment with a two-player game. Pitch controls the vertically moving bats and the goal is to intercept a bouncing projectile with your bat, similar to the original Pong and other “paddle games” derived from it [7].

The history of computer games features several examples of multiplayer games played on a single computer. Generally speaking, you have to either use multiple control devices, or take turns using a shared control. Examples of the former are Tekken and other martial arts games played with several gamepads. Stewart et al. have also researched this kind of interfaces for collaborative work [8]. Having two microphones would make it easy to implement a two-player game, because the pitch detection usually outputs the pitch of the loudest voice. However, few users have the equipment needed to connect two microphones into the left and right input channels of a soundcard.

The main reason for choosing Pong as the starting point was that it combines both real-time and turn-based control so that it works using a single microphone. In two-player mode, pitch controls both bats so that they are always at the same vertical position. The players take turns in singing or humming so that once a player hits the ball, it’s the other player’s turn to react and sing his or her bat to intercept the ball. In one-player mode, the computer controls the other bat.

Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx'04), Naples, Italy, October 5-8, 2004

---

Figure 2: A pitch-controlled Pong-style game. The bats on the left and on the right move vertically according to the pitch of the user’s voice. The screenshot is from one-player mode, where the right bat is controlled by the computer.

---

Figure 3: A frame of the vowel ‘u’ and the corresponding autocorrelation. The strongest peak of the autocorrelation is located at the signal period of 124 samples.

---

4. PITCH DETECTION TECHNOLOGY

Pitch detection (fundamental frequency estimation, f0 estimation) has been researched widely since the 70’s. It is perhaps most widely used as part of speech recognition and coding. Estimating pitch can be implemented as estimating the periodicity of voice, as seen in Figure 3. A periodic waveform repeats itself after the period T, the reciprocal of the fundamental frequency f0. In Figure 3, the period of 124 samples is clearly visible.

There are several algorithms for pitch detection. We have mainly experimented with autocorrelation based methods. The approach was found suitable for speech signals already in the 70’s [9] and several improved variants of the approach have been developed, one of the latest being YIN by de Cheveigné and Kawahara [10], who report error rates three times lower than the best competing methods.

Comparing pitch detection algorithms is difficult, since they are often tested with different data. Recently, the Keele pitch detection reference database [11] has been gaining popularity. It has been used to evaluate YIN as well as methods by Wang and Seneff [12] and Tabrikian et al. [13] to name a few.

In autocorrelation based methods, sound input signal is divided into short frames and the period of a frame is estimated by computing its autocorrelation function (ACF), as shown in Figure 3. ACF estimates the similarity of a signal with the same signal delayed by some time lag. If the signal is periodic, the strongest peak of its ACF is usually located at the lag equal to the period. Note that the ACF computed from a frame of finite length is only an estimate, since the autocorrelation of a random process x(k) is defined as the expectation E[x(k)x*(l)], where * denotes conjugate. For wide-sense stationary (WSS) processes, autocorrelation depends only on the lag k−l[14].

Autocorrelation of a WSS process can also be defined as the inverse Fourier transform of the signal’s power spectrum (squared magnitude spectrum). In practice, the fastest way to compute the ACF of a signal frame is to assume stationarity within the frame, compute an estimate of the power spectrum using FFT and then apply inverse FFT.

In a pitch-controlled application, you often need to know whether there is voice or just background noise or non-voiced phonemes, such as ‘s’ or ‘t’. One suitable estimate for voicelessness
is the relative signal power at the signal period, estimated as the ratio of the ACF at the period and at zero lag [9].

4.1. Latency

For the game or interaction designer, it is important to note that no pitch detection method is perfect. The performance is affected by differences in people’s voices, the quality of the microphones and soundcards, and interfering sounds from the environment. Figure 4 compares YIN to a basic autocorrelation based method that simply finds the maximum value of ACF within a range of suitable periods. YIN performs better, but the output still has some glitches. The errors can be reduced, for example, by using a median filter or analyzing longer signal frames, but in general, more reliable methods cause more delay in the detection and thus result in a less responsive user interface.

![Figure 4: Comparison of pitch detection using basic autocorrelation based method and YIN. The test signal is male voice with sampling rate 11025Hz and frame size 512 samples.](image)

There are many studies on the effect on latency on user interfaces. A classical experiment conducted by Michotte and reported by Card, Moran and Newell [15] shows that users perceive two events as connected by immediate causality if the delay between the events is less than 50ms. Dahl and Bresin [16] suggest that over 55ms of latency degrades use of a percussion instrument without tactile feedback while playing along with a metronome. Finney has shown that delay in auditory response caused large errors in performance of pianists [17]. A study by Sawchuk et al. [18] showed that latency tolerance is highly dependent on the piece of music and the instrumentation. Collaborative playing over a networked system was researched. A bit surprisingly the performers tolerated 100ms of latency on a piano sound but only 20ms on an accordion sound in the same piece.

Less than 10ms latencies are often suggested for musical controllers, as professional piano players might already notice that much [19][20]. However, the amount of tolerable latency depends on the music played, the instrument and the presence or absence of tactile feedback. For an extreme perspective, let us remind that latencies as high as several hundred milliseconds are not rare for church organs and yet they can be played when also practiced with the same latency.

In our games, the user and the computer form a control system where the user tries to adjust his or her voice according to both aural and visual feedback. In general, the more delay there is in the feedback loop, the slower the control and the more there’s overshoots if pitch is adjusted rapidly. However, depending on experience and musical talent, the user does not rely entirely on feedback. When you sing, you are usually able to start a note roughly at the correct pitch, after which you make small adjustments according to the feedback. The main benefit of the visual feedback is that it helps you train your ear and singing.

Latency is not an issue in the Pong game, where the player has several seconds to search for the correct pitch. However, in the Hedgehog game, the tempos of the songs are limited by latency. Delays in the movements of the hedgehog are not disturbing as long as they are small compared to the lengths of the notes. There’s no problem with most slow and medium-paced songs, but the visual feedback can start to feel ambiguous in fast songs, such as Polly Wolly Doodle.

The total delay between voice input and graphical feedback consists of the following components:

- The pitch detection algorithm. The frame size of pitch detection should be at least double the longest signal period to be detected. If the lowest note to be expected is a2 (55Hz), this results in minimum frame size of 36ms. If the sampling rate is 11025Hz, the minimum power of two for the frame size is 512 samples (46.4 ms).
- Video hardware and drivers. Games are usually double-buffered, meaning that graphics are first drawn to an off-screen surface that is made visible on the next screen refresh. This is essential to prevent tearing in the scrolling scenery of the hedgehog game. Depending on the timing of screen refreshes and incoming audio frames, double buffering can cause a delay of one refresh. At a typical refresh rate of 70Hz this yields 14ms.
- Audio hardware and drivers. MacMillan et al. report audio input-output latencies in the range of 60-120ms in the Windows operating system using DirectSound drivers commonly supported by consumer audio cards [21]. Windows with ASIO drivers, Linux with ALSA drivers, and MacOS X with CoreAudio can provide lower latencies, but form a lot smaller target audience. In our experience, most of the latency is due to the sound input part, which is logical since games and other consumer software typically only require low output latency for sound effects and music. Our games run on Windows computers and we use DirectSound for the sound input.

If we take the song Polly Wolly Doodle as an example, the shortest notes in the melody are eighths. If the tempo is 180 beats per minute, the duration of an eighth is 1/6 s. According to our experience, the latency should be at most half that, resulting in 83ms. Thus, even the delays of the audio and video hardware and drivers alone can be too much and this should be taken into account when selecting the songs. The pitch detection algorithm is typically not the major cause of latency in a consumer PC system, if no filtering is applied after the frame-based analysis.

4.2. Octave Errors and Transposing

As seen from Figure 4, octave errors are common in f0 estimation, that is, ACF peaks at 2N f0 are selected, where N is an integer. This happens particularly with low quality microphones. Fortunately,
octave errors are not an issue in our games, since the detected pitch is transposed in octave steps into a suitable range.

In the hedgehog game, different players may want to sing the notes from different octaves so we first designed the game so that the visible tone scale equals one octave and the pitch is simply transposed inside that octave. People sing at different octaves and specifying the octave before the game would make the user interface more complex. Most of the songs children sing in elementary schools in Finland have tone scales less than one octave.

However, the simple transposing breaks the visual feedback system of the game, since the mapping from the pitch to the vertical position is many-to-one instead of one-to-one. For example, if the desired note is at the bottom of the screen, singing only a little below it causes the hedgehog swing to the top of the screen, that is, above the path. Another drawback of the simple transposing is that small fluctuations about the octave border result in the hedgehog appearing randomly at the top and bottom of the screen.

Fortunately, the problems can be solved by transposing the pitch into the octave closest to the desired note and then clipping the corresponding vertical position into the visible range. This way, the Hedgehog appears in the direction where it is closer to the path so that the feedback is correct in errors smaller than a half octave. Singing below a note at the bottom of the screen presses the hedgehog against the bottom so that you know to correct the pitch upwards. The improved transposing also allows wider tone scales than one octave.

The vertical range of the Pong game also equals one octave and pitch is transposed inside it. The transposing is required to allow players with differently pitched voices to play against each other.

5. DISCUSSION

During development, the hedgehog game was tested informally with several adults, 6 third-graders (9 to 10-year-olds), 3 preschoolers and 2 14-year-olds. Preschoolers seemed to need adult guidance in using the game, but were able to play using songs they already know, such as Mary had a little lamb. The third-graders all played the game successfully, but two non-musical children needed 4 to 5 times practicing until they figured out the mapping from their voice to the movement. It helped when we instructed them that the loudness of the voice has no effect and that “you should make your voice go up and down like the siren of a fire-truck”, giving a concrete example of what ‘up’ and ‘down’ mean.

In general, the game works best if the player already knows the song. Learning a new song based on the visual feedback is difficult and requires a really slow tempo or long notes so that you have time to find the notes.

The pitch-controlled Pong has been tested informally in several parties. The game can also be played in a web browser at the address http://www.elmorex.com/products/pong.html. Because the loudest voice dominates, it is easy to play dirty and disturb your opponent, but the same applies also to other turn-based games played with a shared input device. Engaging in a game usually means that you agree on the rules, but on the other hand, a shouting contest is often as entertaining as fair play.

The game seems to satisfy the goal of teaching the concept of pitch, but people sometimes need instruction despite the seemingly simple control. Since the game does not have a musical background to sing to, you learn the tone scale of the display only by experimenting. We found out that trying to make your voice to go higher without a clear target can result in problematic results. Instead of pitch, some people only change the vowel they are voicing, for example, from ‘a’ to ‘e’, which can be explained in terms of spectral changes that contribute to the perception of pitch.

Occasionally, people have made their voices go up a full octave, effectively making the bat stay at the same location because of the octave transposing. One solution for this is to lowpass filter the pitch before the transposing so that the bat can be seen going upwards and wrapping back down, but this makes octave errors visible and increases latency.

It can be questioned whether games controlled by singing might face the same doom as brainwave sensors and other interaction devices found uncomfortable and tedious in prolonged game use, as described by Crawford [22]. We agree with Crawford in that developing a new control method for an existing game is usually not a very fruitful approach, especially if you could play better with a traditional controller. The Pong game is an example of this and although described fun and comical by the users, it loses its appeal quickly. On the other hand, this can be attributed to the overall simplicity of the game and the new interface can be considered successful for reviving the game for party use.

In general, the control mechanism should not be viewed as a separate entity, but as a part of the user experience and gameplay. In the hedgehog game, the interface makes gaming a spectator sport — you can show off your musical skills much the same way as when singing karaoke or performing on stage in a concert. This way, it is related to the dancing games that first gained popularity in Asia and have become a hit also in the western world thanks to recent PlayStation 2 and X-Box titles, such as Dancing Stage Fever. People dance and sing for fun also without computers, and the games add a possibility for improving your skills and competing based on quantitative feedback.

Staraoke, a TV series based on a remake of the hedgehog game was broadcasted in Finland in 2003. In the show, schoolchildren competed in the game. Staraoke was the fourth most popular program on Finland’s commercial television channels among 4 to 9 year old children [23]. In an upcoming version, you can apply for the show by playing the game at home and then uploading your recorded performance to the show’s website.

Lately, interactive karaoke games in the style of the hedgehog game have been becoming a new game genre of their own. Time Magazine ranked Konami Karaoke Revolution as the best game of 2003 [24]. Instead of a story-based avatar and graphics, the game uses an arrow as a pitch indicator and horizontal bars as notes, similar to the piano roll views of many sequencers. In May 2004, Sony Computer Entertainment Europe released a rival karaoke game called SingStar, also with a piano-roll view.

6. CONCLUSIONS AND FUTURE WORK

We have introduced a novel approach to musical entertainment based on using pitch of the user’s voice as a game control mechanism. The games can help you to learn to sing and to control the pitch of your voice. In the future, we will carry out more rigorous testing to inspect the educational effects of the games. For example, it can be questioned whether the learning is only contextual, since the player can learn to rely on game graphics instead of hearing.

The pitch detection sometimes reacts to the background music so that in the hedgehog game, the music controls the hedgehog
when the user is not singing. This also prevents adding a reference melody for learning new songs, since the reference could make the hedgehog stay on path even if the user does not sing at all. There is no problem when using headphones or low output volume, but a general solution would be to use acoustic echo cancellation (AEC) methods to remove the music from the input signal. According to preliminary tests, basic approaches, such as LMS adaptive filters do not completely cancel the music and remove the errors. Echo cancellation methods often assume that there is not much double-talk (simultaneous output and input activity) and that the output and the desired input are not correlated. These assumptions do not hold in musical games, especially if the musical background includes melody and there are only few pauses. On the other hand, echo cancellation can be optimized because the games only need the pitch information and the processed sound does not have to be output for listening.

Considering other games that could be controlled by singing, it should be noted that pitch can also be mapped to other variables than vertical position. Using pitch together with loudness or other features could also enable control in two or more dimensions. This kind of interface could be used to teach dynamics and articulation in addition to singing in tune.

Increasing the dimensionality of control complicates the task of displaying the desired input in a predictable way. In the case of the hedgehog game, the basic method is to use an N+1 dimensional display for an N-dimensional input signal, using the extra dimension for time. However, we are also researching other methods.

7. ACKNOWLEDGEMENTS

This work has been partly supported by the HeCSE graduate school of the Academy of Finland. We also want to acknowledge the contributions of our colleagues at Elmorex, especially Jyrki Kohonen (CEO), Niina Saarelainen (music production), Tero Tolonen (audio technology), Erno Soinila (programming), and Samuli Siltanen (idea of the Hedgehog game with Ville Pulkki).

8. REFERENCES

DIGITAL AUDIO EFFECTS APPLIED DIRECTLY ON A DSD BITSTREAM

Josh Reiss and Mark Sandler

Centre for Digital Music
Queen Mary, University of London
Mile End Road, London E14NS, U.K.
{josh.reiss|mark.sandler}@elec.qmul.ac.uk

ABSTRACT

Digital audio effects are typically implemented on 16 or 24 bit signals sampled at 44.1 kHz. Yet high quality audio is often encoded in a one-bit, highly oversampled format, such as DSD. Processing of a bitstream, and the application of audio effects on a bitstream, requires special care and modification of existing methods. However, it has strong advantages due to the high quality phase information and the elimination of multiple decimators and interpolators in the recording and playback process. We present several methods by which audio effects can be applied directly on a bitstream. We also discuss the modifications that need to be made to existing methods for them to be properly applied to DSD audio. Methods are presented through the use of block diagrams, and results are reported.

Keywords: Sigma Delta Modulation, SACD, DSD, Digital Audio Effects, Bitstream Signal Processing

1. INTRODUCTION

One-bit signals are used throughout the audio recording, editing and playback process. Most analog to digital and digital to analog converters employ a sigma delta modulator that converts a signal to a bitstream. Digital audio is often stored during production in a single bit format. In addition, the high-end audio distribution format, SuperAudio CD, employs the single bit recording format known as Direct Stream Digital, or DSD.

The benefits of the DSD format are numerous. Improvements in the traditional pulse code modulation (PCM) format from higher bit rates and higher sampling rates have experienced diminishing returns. This is partly due to the difficulties in implementing accurate high bit quantisers, but primarily due to the losses incurred from filtering. PCM systems require steep filters at the input to block any signal at or above half the sampling frequency. Ideally, a brick wall filter should be used; passing all frequencies below the Nyquist frequency, and rejecting all above. Yet an ideal brick wall filter does not exist.

In addition, quantization noise is added by the multi-stage or cascaded decimation (downsampling) digital filters used in recording and the multi-stage interpolation (oversampling) digital filters used in playback. Increasing the sample rate, as with DVD-Audio, eases the difficulty of the brick wall filter, but does not correct the problems introduced by multi-stage decimation and interpolation.

This was the inspiration for a 1 bit audio format, as first proposed by Angus [1], and independently implemented as Direct Stream Digital (see Figure 1). As in conventional PCM systems, the analog signal is first converted to digital by 64x oversampling sigma delta modulation. The result is a 1-bit digital representation of the audio signal. Where conventional systems immediately decimate the 1-bit signal into a multibit PCM code, Direct Stream Digital records the 1-bit pulses directly.

The resulting pulse train has some remarkable properties. The oversampled signal has very high quality phase information, making phase vocoder-based effects effective at creating the new delay is far more precise. Furthermore, 1-bit audio effects can be applied on the DSD signal directly before or after processing and audio effect creation in the 1 bit domain is appealing for many reasons. The oversampled signal has very high quality phase information, making phase vocoder-based effects easier and more accurate. Effects using variable delays, such as chorus and flange, also benefit from oversampling since interpolation of the delay is far more precise. Furthermore, 1-bit audio effects can be applied on the DSD signal directly before or after processing and audio effect creation in the 1 bit domain is appealing for many reasons. The oversampled signal has very high quality phase information, making phase vocoder-based effects easier and more accurate. Effects using variable delays, such as chorus and flange, also benefit from oversampling since interpolation of the delay is far more precise. Furthermore, 1-bit audio effects can be applied on the DSD signal directly before or after processing and audio effect creation in the 1 bit domain is appealing for many reasons. The oversampled signal has very high quality phase information, making phase vocoder-based effects easier and more accurate. Effects using variable delays, such as chorus and flange, also benefit from oversampling since interpolation of the delay is far more precise. Furthermore, 1-bit audio effects can be applied on the DSD signal directly before or after
encoding, thus maintaining the simplified production chain as in Figure 1.

The goal of this paper is to describe how to develop standard audio effects on the DSD bitstream, while minimizing intermediate conversions to multibit format (thus destroying all benefits of DSD). Previous work [4-12] has already established that suitable IIR and FIR filters can be created, as well as some mixing tools. However, common audio effects, such as compressors, expanders, reverb, modulation, and so on, have not yet been developed. In the following Sections we will demonstrate how these effects can be applied directly on a bitstream without introducing unwanted artifacts, or significant degradation of audio quality.

2. PROPERTIES OF THE DSD BITSTREAM

There are several features of DSD which distinguish it from PCM. At its heart, DSD is specified as being a 1-bit format, with a sampling rate of 64*44.1 kHz, or 2.8224 MHz [13]. Little else is specified regarding the format, although constraints are imposed for the archiving of DSD on SuperAudioCDs and the playback of those CDs (notably, restrictions on noise levels, frequency response, peak levels and DC offsets). However, the specifications of DSD also note the following properties:

1. The 1-bit format is such that the 1 represents a positive output (+1) and the 0 a negative output (-1).
2. The 0 dB reference level has been set to 50% of the maximum theoretically possible modulation depth. At least 4 out of any 28 consecutive bits must be set to 1 (and similarly for 0). This maximum setting corresponds to 3.10 dB.
3. Silence patterns are defined as repeating bytes where each byte contains an equal number of 1s and 0s.

Unlike PCM, the DSD signal always has a power of 1 (the bits representing +1 and -1 levels). Thus any instantaneous measurement of signal level is meaningless. Furthermore, whereas PCM has a strict 0 dB maximum, the 0 dB limit for DSD has been imposed as a safety measure. In practice, this means that a DSD signal, when put through a sigma delta modulator, is unlikely to result in instability or severe clipping since its peak levels have already been restricted to within safe margins.

Silence patterns do not make sense in 44.1 kHz PCM since any repeating pattern would be ≤ 22.05 kHz and hence potentially audible. A constant DC level represents silence in PCM. But for a DSD signal, constant levels (i.e., all zeroes or all ones) are not allowed. A repeating pattern of 8 bits or less, on the other hand, only has frequency components above 176 kHz, i.e., far outside the range of human hearing. Thus whenever inaudible output is required, a silence pattern should be used. This is important in the construction of many audio effects, such as noise-gating.

3. TIME-DOMAIN AUDIO EFFECTS

Most time-domain based audio effects have well-established implementations [14]. The general design of these effects, when implemented on a DSD signal, can follow the design used for PCM signals. In this Section we describe those design modifications which are necessary for DSD.

3.1. Bitstream addition

Perhaps the most fundamental signal processing is the addition of two signals. O’Leary and Maloberti [15] demonstrated an elegant bitstream adder (Figure 2). The oversampled nature of the bitstream allows one to use a simple feedback loop whereby two bitstreams are added along with the sum bit from the previous iteration. When the bandwidth of the input signals is far below the sampling frequency, as is the case with DSD, the output carry bits are an excellent representation of the average of the two signals. This bitstream adder is remarkable because it requires no requantisation, and it has been shown to be highly effective for oversampled signals. The alternative, bitstream addition via the interleaving of bitstreams [16], suffers degradation of audio quality due to downsampling, phase shift and possible introduction of low-frequency noise.

However, although this bitstream adder does not explicitly perform requantisation, it amounts to the same effect. Thus it acts as a first order sigma delta modulator and introduces some noise and distortion into the audible band. The bitstream adder is suitable either for a limited duration, or when increased noise is acceptable. An alternative would involve summing the signals and then performing high order noise shaping.

3.2. Delay based effects

By using the bitstream adder in conjunction with multiple delays, it is possible to create a flanger or chorus effect entirely through simple logic operations on the bitstream. This is indicated in Figure 3 where BSA represents the bitstream adder from Figure 2. This implementation is very elegant and appealing because it requires no filtering, decimation, interpolation or requantisation. It deals solely with bit operations and delays. Furthermore, the delays can be set to any length, and due to the high sampling rate of DSD, there are far more options over the number of voices and their placement. To weight the delayed signals, a given delay time may be repeated in the inputs to the bitstream adders.

Figure 2: A bitstream adder.

Figure 3: Implementation of a basic flanger or chorus using the bitstream adder (BSA) of Figure 2.
However, it suffers serious limitations in that it allows for no mixing of signals other than additively. Furthermore, the number of signals mixed in this way must be a power of 2. Successive use of the bitstream adder in parallel and series may mimic the effect of a multiplier, but significant noise might then accumulate in the audio band, and it still does not allow for easy implementation of a gain control. A bit stream multiplier is essential for volume adjustment, or for versatile mixing of signals. Therefore, most effects will be implemented using conversion to a multibit domain, and then a sigma delta modulator in the final stage is used for requantisation to DSD. As shown in Section 3.3, this SDM can sometimes be incorporated into the effect processing stage.

### 3.3. Level detector

In order to implement many effects, such as noise gating, expansion, limiting and compression, a level detector is required. In PCM, this is trivial, since the instantaneous level is given by the quantised signal at any given time. For a bitstream, however, the instantaneous value is either 0 or 1, corresponding to a 1 or -1, for input over the range [-Max, Max] where the maximum absolute value of the input is some value Max < 1. Nevertheless, PCM level detection usually employs a time average of power of the signal and bitstream level detection can do the same. It is important however, that the time average be over roughly the same amount of time but not over the same amount of samples. The high oversampling rate demands this.

Time average level detection becomes even simpler for DSD signals. RMS estimation of power is unnecessary. One can simply count the bits. Over a window of size N, where M is the number of ones in the window, $P=N-2M/N$ gives an estimate of the power. A value between 0 and 1 for P can set the threshold. For most dynamic processing, standard techniques can then be applied. A variable gain can multiply the signal, with the additional requirement that the output is processed through a sigma delta modulator (and optionally, a low pass filter), to return the signal to DSD format.

For an accurate envelope detector, a simple moving average filter should not be used. A decimation filter is preferred since it more accurately represents the multibit level of the signal at any instance. It is important to note that under such a situation, decimation need only be used for level detection, and no additional decimation/interpolation is applied to the bitstream.

### 3.4. Modulators

Modulation involves the multiplication of an audio signal by some carrier signal, typically a sinusoid. To do this using entirely DSD signals would involve the multiplication of two bitstreams. Unfortunately, this is not as simple as the addition of bitstreams as in Figure 4. The product of two single-bit signals can be obtained with just one logical gate, an XNOR (or an AND if the signals were restricted to [0,Max]). However, this approach affects the noise-shaping characteristics. Multiplication in time domain corresponds to convolution in z-domain. Therefore, the resulting bitstream has four components: one from the convolution of the two signals, two from convolutions between one signal and the shaped noise of the other bitstream, and the last from the convolution of the two shaped noises. Since the last term has a flat frequency spectrum, the result of a multiplication of two noise-shaped bitstreams is a non noise-shaped waveform, whose in-band noise limits the accuracy of processing.

Currently, the only alternative is to perform multiplication of DSD signals via decimation to a multibit domain, and then reconvertin to DSD via upsampling and requantisation. This suffers severe drawbacks because of the introduction of low frequency noise.

### 3.5. Noise gating

An extreme noise gate operates simply as a threshold below which there should be no signal. A noise gate operating on a DSD signal has several important distinguishing characteristics which require modifications of the standard PCM noise gate in order to function. First, the level detector or envelope follower requires modifications, as mentioned in Section 3.3. Noise gating however, requires further modification. When the signal has been faded to zero, the output must correspond to DSD silence. It is conceivably possible that traditional techniques will produce a signal that, although representing the output of an SDM acting on zero input, will not be silent [17, 18]. This could occur due to small DC offsets or initial conditions of the SDM. This problem is especially serious because, rather than this signal being a very high frequency pattern, as DSD silence is defined, it may be a very low frequency pattern and hence audible.

For these reasons, when silence is required at the output, as may be the case in a noise gate, the output bitstream is replaced with a DSD silence pattern. If smooth transitioning between silence and low-level signal is required, then one of the switching techniques described in Section 3.6 can be applied during the fade-in and fade-out stages.

### 3.6. Smooth mixing and switching of bitstreams

It is well-known that switching of PCM signals can result in audible artefacts due to discontinuities in the output signal. This is avoided by strictly requiring that the PCM samples from the initial and replacement streams be identical at the point at which the switch is made. Samples around the switch should also be roughly identical to prevent abrupt changes in signal slope (and instantaneous frequency) as well.

But the DSD signal contains historical information. That is, the current signal is determined by a sequence of bits, and the next bit is a function of prior states as well as current input. Thus, sample matching is not sufficient. Smooth switching requires that the switch happen when the two bitstreams are synchronised.
In [19], Reefman and Nuitjen described an approach to synchronisation of bitstreams which allows for seamless switching. This approach involves the use of a sigma delta modulator acting on the mix of the two input bitstreams. However, this SDM must be synchronised such that it produces the bitstream A when acting just on A, and the bitstream B when acting just on B.

In order to produce synchronisation, the integrator states, or initial conditions of the SDM, must match those integrator states. This synchroniser can be implemented by using a least squares approach to find integrator states which minimise the difference between a DSD input signal and the resulting DSD output signal. Thus, editing is done as depicted in Figure 6. When synchronisation is ready, the switch is changed to the central position, and G is set to 1. G is slowly decreased to 0, then the output stream is resynchronised to input stream B, and the switch is set to the downwards position.

An alternative switching method is proposed in Figure 7. We note first that both input and output streams are low-pass filtered, and the application of a slowly changing gain and a first order SDM should not significantly change the bandwidth of the signal. Importantly, a first order SDM will have no effect on a DSD bitstream. The difference between quantization of a bit and the original bit is zero. Thus, when G is set to 1 in Figure 7, the output bitstream is A. As G is decreased, a cumulative error based on the difference between the two input signals is added to the quantiser input. As G approaches 0, the difference between the output and input bitstream B also approaches 0. Eventually, the feedback term approaches a constant (typically non-zero) and the output bitstream is identical to B. The only significant introduction of noise is the non-shaped noise due to the first order SDM acting on the sum of two bitstreams when the gain is in the region 0<<G<<1. However, this occurs over a relatively short period and is minimized since both inputs are already low-pass filtered.

The result of this switching scheme on input signals of frequency 1 and 2 kHz, is depicted in Figure 8. A switch is desired at 2 milliseconds. The example is particularly pernicious (and somewhat unrealistic) since the waveforms are very different; out-of-phase and with peak amplitudes of 0.2 and 0.9. The gain is changed linearly from 1 to 0 over 1,600 samples, or just over half a millisecond. Depicted are the analog input signals before conversion to bitstreams, and the output signal after decimation to multibit, 44.1 kHz using a sinc³ filter. The resulting transition at 2 msecs is smooth without abrupt changes in amplitude or slope. There is a slight and temporary increase in frequency, but this effect can be minimised through the use of a slower gain change or eliminated completely by using a detection scheme to find a more appropriate time to perform the edit.

Improvements to this method could also be achieved by using a more effective noise shaper (higher order SDM) instead of the first order SDM in Figure 7. However, with gain equal to 1, the output bitstream would not be identical to the input bitstream. To phase out the effects of requantisation, and resynchronize the output bitstream with the input stream A, we can slowly reduce the feedback coefficients of the modulator. As feedback coefficients approach zero, the modulator becomes lower order until it approaches a first order SDM, and as before, has no effect on the bitstream.

The output of the circuit from Figure 7. This is the worst case scenario, where the input bitstreams have differing amplitudes and opposing phases.
4. FREQUENCY-DOMAIN EFFECTS

Virtually all frequency-domain based audio effects, such as equalisers, wah-wah, and phasers, require the construction of FIR or IIR filters. A significant body of research exists on 1-bit filters. A full discussion of 1-bit filter designs is beyond the scope of this work. Here, we note the main research and how 1-bit designs differ from their PCM-based equivalents.

Angus [4] provided a means of implementing FIR and IIR filters on the DSD bitstream. This was based partly on prior work on FIR filters by Wong and Gray [5, 6] and Kershaw, et. al. [7] and IIR filters by Johns and Lewis [8, 9], and on his own work concerning the processing of one bit digital audio signals [10].

Equalisation is usually implemented by shelving filter design using first order filters. In [4], Angus demonstrated a bass cut/boost control filter which acts directly on the DSD bitstream. He reported roughly equivalent performance to PCM equalizers.

4.1. FIR Filters

Filters for DSD input and output signals have several design considerations which distinguish them from their PCM equivalents. The main alterations are not the same for IIR filters and FIR filters [11]. For a one-bit FIR filter acting on a 64 times oversampled DSD signal, the delay line consists of $z^{-4}$ delays rather than $z^{-1}$ delays. In effect, the taps are subsampled. This has the effect of zero-interleaving the impulse response by a factor of 64. The frequency response is replicated throughout the entire frequency range. This would thus demand a high order filter, except for the fact that this replicated response is outside the audible range. In general, the out-of-band frequency response is irrelevant. Whether the signal needs additional filtering is then dependent on the use of the filter and on the requirements for the high frequency content of the signal. Alternatively, one could redesign the filter using single delays and take into account the high sample rate and single bit input. This approach involves a combination of cascaded integrators and a sparse tap filter [4]. It is efficient, removes the high frequency noise and can achieve the desired frequency response.

4.2. IIR Filters

IIR filtering of a DSD signal, on the other hand, does not change the delays but changes the coefficients. The coefficients of the filter can be calculated in the same way as for PCM, but the oversampling implies that their values will be very different.

As has been mentioned, requantisations should be kept to a minimum. Thus, if the filtering consists of IIR/FIR filters, a noise shaping filter and a low pass filter, then these stages should be combined in such a way that there is only one requantisation in the final stage. Figure 9 depicts an IIR filter which incorporates an SDM-based requantiser. Although such a design is efficient and eliminates the multi-bit stage, it does not differ greatly from a cascade of one-bit filters followed by a remodulator.

Minimising decimation, interpolation, and requantisation is not a drawback. These filters add to system complexity and degrade performance. In addition, filtering in the oversampled domain is advantageous because it relaxes specifications on anti-alias and reconstruction filters at the analog interfaces, thus improving phase linearity [12].

5. CONCLUSIONS

This work concerned how to apply audio effects directly on a DSD bitstream. The general architecture of many effects is approximately the same. However, major modifications need to be made to level detection, noise gating, and switching methods. Conversions to the multibit domain, quantisations and filtering should be minimized. Thus, wherever possible, processing stages should be combined and a single requantisation step should be placed at the end.

One subject which has not yet been adequately investigated is an empirical comparison of audio effects implemented on PCM and DSD signals. All the effects methods discussed within were analysed via the use of simple SDMs for requantisation and a decimation filter allowing comparison with PCM effects. However, this introduces further noise and thus direct comparison is not easy. Development of sophisticated decimation filters and implementation of high order SDMs would allow for a more rigorous analysis. Also, proper analysis of audio effects on DSD signals requires listening tests comparing the signal before and after the effect is applied. However, DSD signals are hard to come by. A new audio format, DSDIFF, has been proposed for the exchange and storage of DSD-encoded audio [20]. As the format gains acceptance, DSD sample files will become available and direct comparison of audio effects on DSD and PCM signals will become possible.

6. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of Prof. James Angus for his comments and criticism concerning this work.

7. REFERENCES


www.superaudiocd.philips.com/InformationCenter/NO/FArticleSummary.asp?lnNodeId=3404
**Hierarchical Organization and Visualization of Drum Sample Libraries**

Elia Pampalk\*, Peter Hlavac

Austrian Research Institute for Artificial Intelligence (OeFAI)
Vienna, Austria

| elias@oefai.at |

Perfecto Herrera

Music Technology Group
Institut Universitari de L’Audiovisual
Universitat Pompeu Fabra, Barcelona, Spain

| pherrera@iua.upf.es |

**Abstract**

Drum samples are an important ingredient for many styles of music. Large libraries of drum sounds are readily available. However, their value is limited by the ways in which users can explore them to retrieve sounds. Available organization schemes rely on cumbersome manual classification.

In this paper, we present a new approach for automatically structuring and visualizing large sample libraries through audio signal analysis. In particular, we present a hierarchical user interface for efficient exploration and retrieval based on a computational model of similarity and self-organizing maps.

1. Introduction

Digitized drum samples are an important ingredient in music production for many styles of contemporary music. However, finding the best samples for a drum loop can be a difficult and very time-consuming task.

Countless sample CDs with drum sounds are available on the market. Each one uses its own way and criteria to organize and label the hundreds of samples it contains. Furthermore, an increasing amount of samples are available directly from the Internet. Each source has different naming conventions and supplies metadata (instrument type, diameter, settings, recording environment,...) of variable quality. This makes it difficult to integrate samples from different sources into a single collection with a content-based organization and practically limits the size of sample collections artists and producers work with.

Currently basically two approaches are used to organize samples from different sources. The first approach is to classify samples on the first level by the instrument (tom, bass drum, snare, hi-hat, cymbal, etc.) and on the second level by the CD source (name of the sample CD or manufacturer). The second approach is to classify samples by their source on the first level, and instrument on the second level. Usually both approaches are used in parallel.

Suppliers have recently started to address the difficulties of managing large sample libraries by integrating them into virtual instruments which offer advanced search mechanism combined with a graphical user interface (e.g. Stylus from Spectrasonics, Groove Agent from Steinberg, PLP 120 from Best Service). However, no system is available for drum sounds which supports similarity-based exploration and retrieval beyond metadata queries.

2. Related Work

A vast amount of research on organizing and structuring sounds has been published. Three main directions are relevant to our work. namely: 1) instrument identification and classification, 2) timbre spaces, and 3) user interfaces to sound collections.

Instrument identification and classification (for an overview see [1]) is relevant for several reasons. For one it would be a valuable extension to the work presented in this paper to automatically label sounds and use this information in the interface. Furthermore, most approaches to instrument classification can be generalized to classify according to almost any concept. Thus, in an ideal case drum sounds could be classified into bright, thick, heavy, or any other user defined categories. Furthermore, features extracted from an audio signal which are useful for classification are also likely to be useful in developing similarity measures which form the basis of our work. Very promising results on classification of drum sound have recently been published in [2,3]. However, since we rely on similarity measures, the results from instrument classification cannot be applied directly to our work.

Interesting models for the similarity of percussive instruments have been the outcome of research on timbre spaces and perceptual similarities in general. For example, Lakatos [4] suggests a 3-dimensional timbre space for percussive sounds. In particular, he suggests that the physical dimensions which are correlated with

\* Part of this work was done while the author was a visiting researcher at MTG-IUA.
the 3 most salient perceptual dimensions are log-attack time, spectral centroid, and temporal centroid. These physical dimensions are also used in the MPEG-7 description format as descriptors for timbre [5]. However, in first experiments we conducted the quality of these descriptors was insufficient to distinguish fine details in the samples as required for our task.

Another interesting aspect of research on timbre spaces is that the similarity relationships of the sounds are usually visualized in 2 or 3 dimensions using multi-dimensional scaling. One of the earliest approaches to use a self-organizing map (SOM) to study timbre spaces by visualizing sound collections was presented by Cosi et al. [7]. An auditory model is used to compute the similarity between 19 relatively different sounds (e.g., flute, oboe, piano, organ,...) with pitch C4. Although this previous work differs from our work with respect to the number and type of samples, the same principles apply.

Our approach is mainly inspired by the work of Feiten et al. [8,9] where a SOM-based interface to efficiently access sample collections was proposed. analogue to [7] an auditory model to compute the similarity of approximately 100 synthesized samples is used as input to a self-organizing map to visualize the collection. Despite differences in number and type of samples, our main contribution to this direction of research is an adapted similarity measure which was optimized using results from preliminary drum listening tests. In addition, we present a new hierarchical extension to the SOM-based interface to deal with large collections.

A quite different user interface to find sounds is the Sonic Browser [10]. The main idea of the Sonic Browser is to browse audio files by listening to them simultaneously in a stereo-spatialized sound scape. However, feedback from our targeted users indicated that even small overlaps (e.g. open hi-hat fading out while snare starts playing) are irritating and should be avoided.

### 3. SIMILARITY MEASURE FOR DRUM SOUNDS

In general it is difficult to predict when a human listener will consider two drum sounds to be similar. Similarity depends on the context, which aspects the listener is focusing on, and subjective impressions. Sounds can be described in measurable dimensions such as attack time or spectral centroid but are commonly described with vaguely defined adjectives such as dark, fat, punchy, deep, thick, crispy, etc.

However, compared to other instrument sounds there are several simplifications which make the similarity easier to compute. First of all, drums allow fewer variations in pitch than instruments in general. Thus, we consider changes in frequency equally important to changes in the loudness envelope over time. Secondly, the temporal loudness contour usually has a relatively simple shape. In particular, to some extent we can ignore effects such as vibrato or other loudness modulations which are very common for other types of instruments.

Our approach to compute the similarity of drum sounds is based on [8] where the main idea is to interpret sonograms as vectors and use a distance metric to compute distances in the vector space. Sonograms are computed taking some aspects of the auditory system into account. As input we use 44kHz mono samples. Samples longer than 500ms are truncated. We use a FFT with 23ms windows, weighted with a Hann function, and 12ms overlap to obtain the spectrogram. To model the frequency response of the outer and middle ear we use the formula proposed by Terhardt [11].

$$A_{db}(f_{stim}) = -3.64f^{-0.8} + 6.5\exp(-0.6(f - 3.3)^2) - 10^{-3}f^4.$$  

The main characteristics of this weighting filter are that the influence of very high and low frequencies is reduced while frequencies around 3–4kHz are emphasized (see Figure 1).

![Figure 1: Terhardt’s outer and middle-ear model. The dotted lines mark the center frequencies of the 24 critical-bands.](image)

Subsequently the frequency bins of the STFT are grouped into 24 critical-bands according to Zwicker and Fastl [12]. The conversion between the bark and the linear frequency scale can be computed with,

$$Z_{crit}(f_{stim}) = 13 \arctan(0.76f) + 3.5 \arctan(f/7.5)^2.$$  

The main characteristic of the bark scale is that the width of the critical-bands is 100Hz up to 500Hz, and beyond 500Hz the width increases nearly exponentially (see Figure 1 where the dotted lines appear almost equally spaced beyond 500Hz on the log-scaled frequency axis).

We calculate spectral masking effects according to Schroeder et al. [13] who suggest a spreading function optimized for intermediate speech levels. The spreading function has lower and upper skirts with slopes of +25dB and −10dB per critical-band. The main characteristic is that lower frequencies have a stronger masking influence on higher frequencies than vice versa. The contribution of critical-band $z_i$ to $z_j$ with $\Delta z = z_j - z_i$ is computed by,

$$B_{db}(\Delta z_{crit}) =$$

$$= 15.81 + 7.5(\Delta z + 0.474) + 17.5 \left(1 + (\Delta z + 0.474)^2\right)^{1/2}.$$  

We calculate the loudness in sone using the formula suggested by Bladon and Lindblom [14].

$$S_{sone}(l_{db,spl}) =$$

$$= \begin{cases} 2^{(l-40)/10}, & \text{if } l \geq 40\text{dB}, \\ (l/40)^{2.642}, & \text{otherwise.} \end{cases}$$

After these steps each sample is described by a sonogram in the dimensions time ($f_s = 86$ Hz), frequency (24 critical-bands with the unit bark), and loudness (measured in sone) with a maximum length of 500ms. Examples for sonograms are shown in Figure 2.

The use of different metrics to compare sonograms was studied in [15] where based on data from listening tests the authors came to the conclusion that a Minkowski metric with $p = 5$ produces best results of synthesized harmonic samples. In first experiments we could not confirm these these findings for drum samples, thus, we have resorted to the use of the Euclidean distance.
A main problem when using sonograms is the sensitivity to time shifts which requires some sort of temporal alignment. For example, comparing 2 versions of the same sample where one version is shifted by 20ms could yield a distance larger than the distance between two perceptually different samples. In [9] an approach is proposed where a SOM is trained on steady state sounds (with approximately 6ms duration) extracted from the samples. Subsequently, the samples are represented by trajectories on the SOM. The sounds are then compared by computing the distance of their trajectories using the city-block distance and aligning them temporally so that the distance is minimal. The main advantage of using the SOM is to optimize the computations, however, the computing power available today allows us to directly align the sonograms.

To align two sonograms we compute the distances between them while shifting them against each other in the range of 50ms. We then take the minimum of these distances as the distance of the samples. Figure 3 gives an example for such a direct alignment.

**Figure 3:** Illustration of the temporal alignment assuming there is only one frequency band. B is aligned with A which results in B’ with a minimum distance to A.

### 4. SELF-ORGANIZING MAP

The SOM [16] is a useful algorithm mainly to visualize very high-dimensional data. In previous work we have applied it to organize and visualize music collections [17][18]. In this paper we use 1-dimensional SOMs to hierarchically structure the samples and a 2-dimensional SOM for visualization. The SOM consists of units which have a topological order (usually a 2-dimensional rectangular grid, referred to as map). Each of these units is assigned a model vector in the data space. The model vectors are not equally spaced. In particular, especially in sparse areas some units might represent no data items. The number of data items mapped to each unit is not equal. Especially in sparse areas some units might represent no data items. Second, the model vectors are not equally spaced. In particular, in sparse areas the adjacent model vectors are relatively far apart while they are close together in areas with higher densities.

The main objective of the SOM is to map similar data items (i.e., sound samples) to units close to each other. This is achieved by iteratively optimizing the topology and quantization error. The quantization error is optimized by adapting the model vector of each unit so that it better represents the samples assigned to it. This is identical to k-means clustering. The topology is preserved by taking the neighborhood of each unit into consideration when adapting the model vectors. The model vector of each unit is adapted not only to fit the directly assigned samples, but also the samples of neighboring units. The size of the neighborhood which is taken into consideration is decreased gradually during training. The final size of this neighborhood together with the number of map units (map size) are the two main parameters of the SOM which control how much freedom the SOM has to adapt to the data.

**Figure 4:** Illustration of the SOM. (a) The probability distribution from which the sample was drawn. (b) The model vectors of the SOM. (c) The SDH and (d) the U-matrix visualizations.

Figure 4 illustrates some important characteristics of the SOM. Samples are drawn from a 2-dimensional probability density function. A 2-dimensional (8×6) SOM is trained so that the model vectors adapt to the topological structure of the data. There are two important characteristics of the non-linear adaptation. First, the number of data items mapped to each unit is not equal. Especially in sparse areas some units might represent no data items. Second, the model vectors are not equally spaced. In particular, in sparse areas the adjacent model vectors are relatively far apart while they are close together in areas with higher densities.

Both characteristics can be exploited to visualize the cluster structure of the SOM using smoothed data histograms (SDH) [19] and the U-matrix [20], respectively. The SDH visualizes how many items are mapped to each unit. The smoothing is controlled by a parameter. The U-matrix visualizes the distance between the model vectors. The SDH visualization (Figure 4(c)) shows the cluster structure of the SOM. Each of the 5 clusters are identifiable. The U-matrix mainly reveals that there is a big difference between the clusters in the lower right and the upper right.

### 5. USER INTERFACE

The HTML-based user interface (see Figure 5) that we have developed consists of two parts. The upper part is only text-based and the lower part is mainly graphical. In the following we describe the ideas and concepts behind both of them. The main idea is to give the user first a overview of the different samples available and then rapidly narrow down the search with each input from the user. A demonstration is available online without the audio files due to
5.1. Text Interface

The intention of the text-based interface is to create a very simple interface which would allow the user to navigate in the sample collection with closed eyes using only a few keyboard keys. The basic functionality would be using up and down keys to listen to the next or previous sample, right to listen to more similar and left to listen to less similar samples. However, in the HTML interface this functionality is available only for usage with the mouse.

In Figure 5 the four columns represent the four levels of the hierarchical structure. The first level is the leftmost column, the fourth level the rightmost. In this case, the user has selected the first sound on the first level (requesting more of this kind). Each of the 9 choices are typical sounds for the sub-branches they represent. On the second and third level the 5th sample was selected leaving a final set of 5 samples in the fourth column. If the user is not satisfied with this set it is always possible to make different choices at higher levels in the hierarchy and explore other branches.

The first level is a rough summary of the collection based on 9 samples. The number 9 was arbitrarily chosen manually. It is a trade-off of using as many samples as possible to describe the collection as accurately as possible on one side, and using as few samples as possible to create a good summary. In future work it might be interesting to investigate determination of this number automatically for each node in the tree.

The 9 samples are determined using a 1-dimensional SOM with 9 units. The motivation for using a SOM is to order the samples in a meaningful way. In particular, adjacent neighbors on the list should be similar to each other. In this case we have a rough order of toms (1,2,4,5), snares (3,6,7), hi-hat (8), and cymbals (9). Alternatively, k-means clustering could be used in combination with a traveling salesman algorithm to sort the clusters.

Each of the 9 (parent) samples on the first level represent a subset of the collection (children). Each subset includes all samples which are best represented by the respective parent. The number of these children is displayed in brackets next to the name of the parent. Furthermore, each parents subset is enlarged by 50% to include children which are not best represented but are nevertheless similar. For example, in Figure 5 on the first level 150 samples are best represented by the first parent. The subbranch includes these 150 plus an addition 75 which are best represented.

http://www.oefai.at/~elias/dafx04
by the second parent but are located on the boundary to the first parent. This overlap in the hierarchy ensures that samples which are similar to two parents can be found more easily.

For each set of children a SOM is trained until the number of children in a set is smaller than 13. Thus, the depth of the branches of the tree are not fixed to a specific number. In the experiments we will discuss later the depth reached a maximum level of 4 and a minimum level of 3.

There are several alternatives to create hierarchical structures. In previous work [21] we have used the Growing Hierarchical SOM (GHSOM) [22]. The main reason for not using the GHSOM in this work is because it would not be possible to create the visualizations described below. However, the GHSOM and variations [23] use heuristics which would automatically adjust the number of parent samples. These heuristics might be suitable choices for future work.

5.2. Graphical Interface

The intention of the graphical user interface is to give the user more information on the automatically created organization. The GUI is tightly coupled with the text-based interface. The visualization is based on one large 2-dimensional (48 × 12) SOM which is trained on the whole collection. The location of the 9 samples (determined in the text-based part) are marked on the map with numbers. The user can move the mouse over the samples to get a tooltip with the sounds filename displayed and hear the sounds. If the user wants to find more of the same then it is possible to click on the sounds to descend to the next level.

The image in the lower part of Figure 5 is the SOM where the 9 samples of the first level marked. Note the order which is created by the 1-dimensional SOM. The gray-shadings in the background are a smoothed data histogram and indicate how many items are located in the different areas. The SDH smoothing parameter is automatically adjusted for each hierarchy level to create rough summaries on higher levels, and more detailed summaries on lower levels. In particular, the parameter was calculated based on the square root of samples on the respective level.

In previous work we used the SOM and SDH to visualize music collections using a metaphor of islands of music [17]. Clusters found by the SDH were visualized as islands. In this work, we use the SDH mainly to visualize which part of the collection is currently being considered in a branch of the hierarchy.

Figure 5 illustrates how the set of samples is narrowed down as the user makes selections. On the first level (see map in Figure 5) the whole collection is visualized on the map and the parents are spread across it. On the second level the area only covers about one quarter of the whole collection. Finally, on the fourth level the parents are so close to each other that they overlap in the figure.

6. FIRST EVALUATIONS & DISCUSSION

For the experiments we used (mostly dry, or mixed with some ambience or reflection room) drum samples from 2 sample CDs from different vendors. In total the collection had 817 distinct samples. We computed the HTML interface described above and informally demonstrated it to 5 prospective users who use and search for drum samples on a daily basis. In general feedback was very positive. We observed the following.

1) Given the choice the users always preferred the graphical interface. One user explained that producing music is a creative process. Unconventional search mechanisms are more likely to come up with something unpredictable. However, the text interface was also considered to be very useful. Mainly because some of the displayed file names contained very useful information (e.g., SN 6” mixed with reflection room rimshot hard). The request was made to better integrate this information into the graphical interface.

2) Generally, the better integration of metadata was a major request. Especially filters to focus only on one instrument are missing. However, filtering instruments might be a suboptimal solution in cases where samples from different instruments can sound very similar such as a bass drum mixed with a reflection room and snares. One solution might be to use a color coding to indicate the instrument type of the parent in the GUI. Another option might be to use component planes analogue to the weather charts in the islands of music metaphor [17]. The weather charts are laid over the SOM and display in which areas there is a high or a low concentration of a specific property.

3) A very interesting point which was brought up is that 3 hierarchical levels would be sufficient instead of 4. Mainly this can be explained by weaknesses of the similarity measure we use. Although, the correlation of the similarity measure and user ratings from preliminary A-B drum listening tests is around 0.8, the similarity measure is far from optimal. In particular, the level of detail which our organization creates is not supported by the similarity measure. For example, in Figure 5 in the 4th column there is a mix of toms, bass drums, and even one snare drum. Another reason why 3 levels could be sufficient is that in manually created organizations it is very common to have significantly more than 12 samples in the lowest levels of the hierarchy (some users mentioned they usually have 30 or more samples in directories on the lowest level).

4) Another important point brought up in the interviews was that every artist uses his own vocabulary to describe samples. Although it would not be necessary to adapt to each artists individual vocabulary it would be very useful to classify samples according to words such as dark, thick, crispy, etc. even if the meaning of these words needs to be more or less arbitrarily predefined. This additional information could be visualized as mentioned in 2).

Other points brought up were 5) the lack of a zoom function, 6) the restriction to mono sound, 7) the selected parent should be marked on the lower level hierarchy, 8) a good system should allow very fast browsing using as few clicks as possible to support the creative process.
7. CONCLUSIONS

We have presented a system to automatically organize and visualize drum sound collections hierarchically. The similarity measure we use has several limitations, however, it seems to be sufficient for drum sounds and their specific characteristics. First feedback we got from prospective users was very positive although some modifications were suggested. In future work will focus on improving the similarity measure. Currently we are conducting listening tests for drum sounds to gather data to optimize the various parameters in the similarity measure. Furthermore, we plan to implement a VST plugin to integrate the system into sequencers such as Cubase or Logic.

8. ACKNOWLEDGEMENTS

This research was supported by the EU projects HPCRN-CT-2000-0015 MOSART\(^1\) and FP6-507142 SIMAC\(^2\). The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Education, Science, and Culture and by the Austrian Federal Ministry for Transport, Innovation, and Technology. Furthermore, the authors wish to thank the researchers and students at the MTG for participating in the drum listening tests and Marx Zsolt for valuable feedback.

9. REFERENCES


http://www.diku.dk/musinf/mosart/
http://www.semanticaudio.org
1. INTRODUCTION

The most common mode for human-computer interaction, the visual mode, is dominating user interfaces, but exploiting other modes of interaction becomes more important. This has various reasons, firstly, the content of interfaces is growing and becoming more complex. Secondly, in more and more applications the visual mode is restricted by form factors, the mobility of the user or simply by being occupied for other tasks. Thirdly, the fact that computers play a more central role in our society nowadays, builds up the awareness that they must be available for all parts of society. People with visual disabilities have major disadvantages in accessing computers because of the lack of efficient non-visual user interfaces.

The paper presents the work conducted on the use of spatialised audio in human-computer interaction. It includes the building of a prototype presenting a sample application and its evaluation with a mixed group of visually impaired and normal sighted participants.

The following Sections will briefly describe the motivation for this work and provide information about some previous work. Section 2 describes in detail the built prototype including the sample application and its implementation. Subsequently, Section 3 describes the conducted evaluation process covering the tasks to be completed and the whole test setup. In Section 4 the results of the test are presented and the most important findings are pointed out. Finally, Section 5 draws some conclusions and looks on future research.

1.1. Motivation

To develop audio interfaces can improve the usability of human-computer interaction for a wide range of users and may be applicable on very different systems.

Visually impaired and blind people are not capable of using computers as they come out of the box. Every customary PC comes with a screen device which is of no use for this user group. The state of the art access for visually impaired people are screen-readers and braille-lines, both available only for significant additional costs to raise for the target group to have only inferior access to modern information technology. It is our social responsibility not to exclude such a big user group (approx. 190 million people with visual disabilities in the world [1]) from the developments in modern information technology by providing them adequate access to it.

Mobile computing is being considered as one of the key applications in present and future information technology. Devices like cellular phones, personal digital assistants (PDAs) and notebooks with wireless connection capabilities for information transfer are already widely used. Wearable computing or devices providing ambient intelligence are the very next logical step, connecting even more compact devices with the networks of the future. More complex user interfaces need to be presented with devices unsuitable for the use of conventional graphical displays.

Due to these facts new modalities must be introduced to HCI and for several compelling reasons audio is a good candidate. The most obvious advantage is that "...audio display space is not wed to the disappearing resource of screen space" [2]. Audio interfaces can be potentially very large without the need of large interface devices.

Another scientific field which may benefit from high performance auditory displays is the sonification of complex data. Spatialisation techniques have the potential to widen the possible display area for sonification and may therefore be very useful in depicting big multi-dimensional data sets.

1.2. Previous Work

Three dimensional, surrounding interfaces have the advantage of a much bigger display area than two dimensional screens or even...
one dimensional screenreaders. But they need to overcome the problem of concurring information and its interference [3, 4].

The segregation of sound sources in a virtual 3D environment is a necessary prerequisite of presenting information parallel to the user without confusion. Research has shown that sound source discrimination is a complex process. From the physiological point of view segregation depends on localisation cues like monaural spectrum changes or binaural sound wave differences [5]. But as stated in [6] sound source segregation is not only a question of localising the sources, it is heavily dependent on psychoacoustic effects.

The content of the presented audio streams is crucial for the user to be able to segregate them. This has been shown for non-speech sounds [7], earcons [8] and speech sources [6]. Other psychoacoustic effects like informational masking [9] can additionally influence the presentation.

Spatial audio was used for auditory displays as soon as the simulation of sound fields became possible in real-time [10]. The interactive nature of user interfaced required "online" authoring of acoustic scenes so that pre-recorded audio was not satisfactory. With the improvement of audio rendering methods and the increasing knowledge about psychoacoustic effects virtual environments were able to present more complex information [11, 3].

However, the use of high definition audio rendering in human-computer interaction is still rare because little is known about how to design scenes effectively. The major problems identified were the users mental load [3] in combination with disorientation [11] and insufficient accuracy and naturalism in acoustic simulation. The presented work tries to address these issues with combining usability engineering methods with sophisticated real-time soundfield recreation techniques. Key idea is to consider semantic information of the user interface during the depiction process to find working acoustic representation for interaction tasks [12]. The presented work should provide the basis for further investigation on this field.

2. THE SYSTEM

The following Sections describe in detail the prototype system and the sample application to present.

2.1. The Application

The application to be presented in the auditory display was chosen to be a simple food market. It includes selecting food articles for a shopping list and an ordering process with payment and delivery information. The application was chosen because it combines important interaction tasks occurring in HCI, but is on the other hand abstract so that participants are most likely not very familiar with similar programs.

The application is compiled of the following interaction tasks: navigation in a structured menu, text input, exclusive and non-exclusive selection out of a list, confirmation of selections and confirmation of notifications or alerts. In a typical graphical user interfaces these tasks would be represented by menus, lists, text fields, buttons and check boxes. However, it is important to state that interaction tasks are mode independent. We were not looking for acoustic mappings for graphical user interface elements, but presenting the interaction task in the auditory domain.

Figure 1 illustrates the menu structure of the application. The menu is structured in 3 separate title menus with different submenus. Each sub menu leads to a corresponding dialog, where different actions can be performed by the user. In the dialog New list, personal data (name, street, village, date of delivery) can be entered. The Search menu naturally hosts a search function. With Close the user is asked for confirmation before leaving the program. Each of the Goods dialogs includes a list with 4 different products (e.g. milk products: milk, cheese, yoghurt, cream cheese) and each may be selected for the shopping list. In the Delivery dialog users can choose between picking up the items from the shop or if they should be delivered. In the Payment dialog, the user can pay cash or with credit card. Additionally, a few alerts and notifications were implemented, e.g. if one of the text fields was left empty, an error message appears. If the search function finds an item, there is a corresponding notification.

Figure 1: Menu structure of the shopping application.

2.2. Auditory Representation

The auditory display is set in a virtual room with the dimensions 20m x 20m x 7m (width, length, height). The items presented are arranged in a semicircle in front of the user and their number is restricted to 6. Interaction with the application is performed through the participants head position and a conventional keyboard. The virtual room is divided into several active areas (a region of +/-10° around each Icons position). If the user turns its head towards one of the icons, it becomes active as soon as he looks into one of the active areas. This means that any input via keyboard will apply to the active icon only. The total range of the display is adjustable and after pretesting the system the overall region was scaled to +/-60°. The spacebar is used for clicking (as substitute for the mouse which cannot be utilised by blind users). The F1 and F2 keys provide context sensitive help: F1 further information about the active area, F2 the context of the menu or dialog the user is in. The Esc-
button leads back to the main menu. Dialogs have to be left with OK or Cancel.

To mimic the visual ability of focusing on a certain area of the monitor, an acoustic zoom function is implemented. This is done by weighting the input gain of the sources according to the relative position to the user. As weighting function a Gaussian window with variable width is used so that users may widen or narrowing their focus. However, the weighting function does not affect the room model used, the perception of room size does not shrink with the use of the zoom function.

The user interface items were modelled with auditory icons. Table 2.2 provides an overview over the used sounds. For some of the general sounds an “iconic language” was created. A vocabulary of sounds was created which were combined according to a syntax. For example, OK was always represented by a gong, but in dialogs was braided with the auditory icon of the dialog it was intended to confirm.

All Auditory Icons were played in a loop, but as all icons were natural sounds their gain envelope is fluctuating making localisation more difficult. As an additional help, to each source position a continuous musical tone was added. The pitch of the 6 musical tones was increasing from the left to the right. Together, the musical tones form a C5 7- chord). The gain was kept lower than the gain of the icons. Each musical tone was a combination of 4 harmonic partial tones. Additional, a slight amplitude vibrato was added to make the tones easier to distinguish.

The test was performed by a group of 10 test participants. Among them were 4 participants who were blind or visually impaired. All participants got the same introduction before the test: The functionality of the program and the menu structure was presented, and the participants heard the auditory icons one time via loudspeakers. Afterwards, the participants had 15 min of free training with the application. In this time, all questions were answered and the participants were encouraged to try the help and the zooming functions. For equal conditions between participants with normal seeing ability and the visually impaired ones, no optical cue was presented on the monitor.

All Auditory Icons were played in a loop, but as all icons were natural sounds their gain envelope is fluctuating making localisation more difficult. As an additional help, to each source position a continuous musical tone was added. The pitch of the 6 musical tones was increasing from the left to the right (fground = 392, 523, 659, 784, 932, 1046 [Hz] or g1, c2, e2, g2, b2, and c3. Together, the musical tones form a C5 7- chord). The gain was kept lower than the gain of the icons. Each musical tone was a combination of 4 harmonic partial tones. Additional, a slight amplitude vibrato was added to make the tones easier to distinguish.

The following tasks were asked to be completed from the participants:

**Task 1:** Opening the menu File→New List and entering the personal data in 4 text fields (name/street/village/date). Correct result: 4 text fields filled in.

**Task 2:** In Order, the terms of delivery and payment had to be chosen among a list of two possibilities for each. The meaning of the list elements head to be found via F1- function. For payment, a fictitious credit card number had to be entered. Correct result: 2 selections and 1 correct number entered in a text field.

**Task 3:** From the Goods sub menu participants had to choose 8 correct items out of 4 lists (fruit / milk products / beverages /bread). Correct result: 8 selections.

**Task 4:** Participants had to enter 2 specific items into the search function and, if found, add them to their shopping list Correct result: 2 search results added to list.

**Task 5:** Finally, participants were asked to quit the application. Correct result: 1 decision, leaving the application.

### 3.3. Collected Data

In a background questionnaire all participants were asked about their personal data, their education and about their background in using computers. After conducting the test the participants were interviewed to collect subjective data about the users’ view of the system.

During the test bottom-line data was recorded. Firstly, regarding the task results meaning the filled in text in most cases. Secondly, all navigation and events were recorded. Head rotation was recorded every 100ms, any keyboard events including time stamps as they occurred.

Additionally, all sessions were taped on video and the test administrator attended the test with a pair of extra headphones and

---

1StAX SR-007
2Ascension Tech. Corp., Flock of Birds

---

**Proc. of the 7th Int. Conference on Digital Audio Effects (DAFx’04), Naples, Italy, October 5-8, 2004**
took notes on anything remarkable uncovered by any logging facilities.

4. ANALYSIS

For analysis, the participants are divided into two groups: six participants with normal seeing ability (participant 5 to 10: group S for seeing) and four visually impaired or blind participants (participant 1 to 4: group B for blind). All members of group B need additional tools for working with a PC. Three of them use screen-readers (JAWS4), two of them in combination with a Braille-lines. Only one member of this group is able to work with spectacles and at a very high resolution without additional tools. All members work in an office job with the aid of a PC and mainly use the MS Office package. Three of them have a lot of working experience and have passed the ECDL (European Computer Driving License). The 4th participant has little working experience and is currently absolving the ECDL. Group S consists of five students of different fields and one employee of the institute. All of them have a lot of computer experience and spend several hours each day with a PC. They use different operating systems (Windows, Linux, Apple) and different programs.

4.1. Bottom-line Data

As stated above all head movements and user events were recorded during the sessions. Figure 3 shows a typical result of this data. The azimuth angle on the x-axis cuts into active and inactive areas like illustrated in Figure 2. The y-axis shows the time elapsed during a task. In this case participant number 2 was completing task 2 (from minute 6 to 9 in his test session). The chart illustrates the navigation through the menus including all selection events and provides additional information of the zoom factor active at each time. Remarkable are the straight head movements from one menu to the other and the comparatively little focusing effort of the user when selecting an item.

Participant 2 was at the time of conducting task 2 already very advanced in using the system. However, Figure 4 shows that in average the focusing effort of users resulted in a high ratio of hits. Four out of five attempts to select an item were successful. This shows that the quality of the directional sensation and the chosen size of the active areas were sufficient to present six items in the auditory display. Another remarkable result is that there was no significant difference between group S and group B in terms of successful hits. This means that there could not be any difference determined in the accuracy of sound source localisation between blind and sighted users.

The equality in performance was also shown from the fact that the selection events per time showed no difference between the groups. Figure 5 shows that the hits and misses per minute are equally distributed among the groups. With an average of one successful selection every 15 seconds the performance in general is surprisingly near to graphical applications.

Another interesting result is the relation of the angel covered by head rotation and the number of events. Figure 6 shows the average angle covered for one event including help, zoom and selection events. The angle represents the average head movement of the user to reach a certain goal. Figure 7 illustrates the same relation as above, but considering only successful selection events. Both charts show that the needed head movement is in the range of the presented area of auditory icons. However, it is remarkable that half the way is wasted with either help events or misses. As in the charts above there is no significant difference identifiable between the group S and B. Although it seems that sighted users are
slightly more effective in this figure, the difference lies within the variance of the data and has therefore no clear outcome.

The use of the F1 help function over the time of a user’s session is illustrated in Figure 8. It does not show that people used it less to the end of the test as it was expected. However, it is remarkable that it seems that blind people used it less than their sighted colleagues.

This also is true for the use of the zoom function. It seems that sighted participants used the zoom function more often than the blind. In contrast to the usage of the F1 help function the zoom function was used less frequently the longer the test lasted. This points out that the zoom function may not be a very useful interaction or that at least the handling of it was not very valuable for the participants.

4.2. Subjective Evaluation

From the questionnaire after the test held in form of an informal interview the following key points could have been extracted:

From ten participants, two stated that they had problems with the spatial segregation in the beginning but improved with practice. Two of them explicitly said that the zoom-function was helpful, and two of them found the active areas around the sources to small, so that they often missed the target.

Seven participants said that they had a good orientation within the menu structure, and two of them explicitly reported a good mental image of the menus. Two participants found the F2-function a good idea, only one participant wanted more help-functions with regard to the menu structure. Three participants found the menu structure rather confusing, one had problems when entering a new room or menu.

The participant’s description of the virtual room were very different. Five participants had a clear impression of a big room and could describe it afterwards, four participants did not have such an impression and one participant did not concentrate on the room at all.

The musical tones in the background were found helpful by four participants, two participants did not even notice them, only one found them too loud and one participant suggested that they should decrease in loudness when zooming out.

From ten participants, only one had problems remembering the meaning of the icons, because the time for learning them was too short. The other participants found them easy to remember, and three of them said that the F1 function was very helpful, if
they had forgotten one of the icons.

Working with headphones was no problem for seven participants, two participants did not like working with headphones, and one said to prefer speakers.

5. CONCLUSIONS

The paper showed the building of a prototype auditory display using high-definition audio rendering. The evaluation revealed some remarkable findings above all that there is no evidence that there is any difference in performance between sighted users and users with visual disabilities.

Furthermore, it showed that relatively complex applications can be interfaced with spatial auditory displays easily and reach remarkable performance. Methodologies and approaches in this first prototype are nevertheless in their fledgling stages. For example, for practical work the selection with head rotation is definitely not a convenient way.

However, the development of “iconic languages”, the use of high definition audio rendering and complex room models, zoom functionality and additional orientation cues are promising approaches towards a usable spatial auditory display.

In our opinion future research in this scientific field should focus on the development of a generic depiction process for user interfaces. This ranges from mode independent descriptions of interaction tasks within user interfaces to their acoustical representation in virtual environments [12]. Auditory displays must not be simple mappings of graphical user interfaces. Furthermore, a lot more must be known about the psychology of human hearing and information retrieving to be able to exploit the major advantage of the virtually huge display area of spatial auditory displays.

6. ACKNOWLEDGEMENTS

Many thanks to all who volunteered to participate in the hearing test. We also would like to thank the ISIS (Integration, Service, Information and Schooling for the blind) group with which we collaborate since two years. They helped us to acquire test participants and helped us understanding the needs of the people concerned. ISIS is a project of the Berufsförderungsinstut Styria, Austria.

7. REFERENCES

VISUALIZATION OF SOUND AS A CONTROL INTERFACE

Anne Sedes, Benoît Courribet, Jean-Baptiste Thiébaut
Centre de recherche Informatique et Création Musicale
Université de Paris VIII
Maison des Sciences de l’Homme Paris Nord, Saint-Denis, France
{asedes|bcourribet|jbthiebaut}@mshparisnord.org

ABSTRACT

We here introduce the opportunity of using visualization of sound as a control interface, for artistic live performance as well as for new digital audio effects developments. Two different approaches are exposed. The first access consists in using video matrices for mapping the variables parameters of sound processing, with the coordinates of any controller in a 2D plane or 3D space. The second access proposes a visualization of sound that modifies sound data by processing the data of the image itself with its own graphical properties.

Exploring this kind of « transducting » relation between visual and audio may be interesting for artistic creation domain using virtual surroundings; it may cause an interest for the real time digital audio, for audiovisual mixing and new interfaces for sound design. Besides, it points the opportunity of developing 3D control interfaces for audio and visual processes.

1. INTRODUCTION

Beyond scientific graphical representations of sound, such as waveforms or spectrograms, there is no objective representation of sound phenomenon, no machine to visualize sound. There are only subjective, more or less metaphorical representations, in heterogeneous sound or musical context. Visualizing the sound phenomenon, after the real time digital audio flow coupled to real time video data flow, the way a programming software such as Max/MSP/Jitter [1] can offer, consists as far as a representation, to visually mark some of the sound qualities, to offer a better listening, a better perception, and real time operation on sound. Hence, we are inverting the ordinary relation between the audio and visual domains, by using visualization of sound as a control interface of sound perceptive parameters, indeed exploiting the graphical properties of sound visualization to process sound itself.

2. USING TERRAINS AS VISUAL INTERFACES

Considering our previous research on self-centered perception of sound spaces [2][3][4][5], we have decided to link some of its characteristics with our study of visualization of sound as an interface. Assuming that a musical piece can be seen as a set of parameters varying in time, a deterministic musical piece can be represented as in Figure 1. There, the time line is explicit, and proposing this type of visualization modifies our perception of the structure of the piece (because all evolutions past and future can be seen at the same time). Thus, we have decided to let the user free to imprint his own timeline to the piece. Referring conceptually to the Wave Terrain Synthesis [6], we have decided that each parameter would be represented by a terrain. The terrain will appear on screen as a monochrome picture, the value of the parameter being mapped with the intensity of the colour. A terrain set (a superposition of terrains) would result, on which the user could move, outputting the parameters values at the point where he is located.

Thanks to the Max/MSP/Jitter programming environment, we have developed software to illustrate these concepts, using the OpenGL set of Jitter objects. We are now going to describe successively both the writing and the visualization interface.

The main goal of the writing interface is to create quickly a terrain set by drawing coloured shapes within a reduced matrix (a tenth of the final matrix). One has to select one of the three planes (red, green and blue) and set the intensity of the corresponding colour and then draw directly onto the terrain window (the left pane in Figure 2). It is possible to use an interpolating filter in order to smooth the terrain.
for the final matrix (the result of the interpolation appears in the right window in Figure 2). Thumbnails for each plane are available on the left side. When done, the resulting matrix has to be exported in the Jitter matrix format in order to be loaded later in the visualization interface.

The visualization interface is a minimal 3D environment consisting in a textured ground on which it is possible to move thanks to the mouse. The 3D engine is the same as the one used in prior works Egosound [7] and La Terre ne se meut pas [8].

For an \((x,y)\) set of cartesian coordinates corresponding to the location of the user on the ground, we have a set of parameters \(R(x,y), G(x,y)\) and \(B(x,y)\) according to the matrix \((x,y)\) cell values. From now, we will only consider one parameter \(P(x,y)\), because of the grey scale constraint of the presentation. In Figure 3 we can see a terrain (one parameter) together with its projection in the visualization interface.

From a trajectory \(T : t \rightarrow (x(t), y(t))\) (a set of \((x,y)\) coordinates varying in time), a time-varying \(P(t)\) will result. We just have seen in a very simple way how time and space are linked, and how we can recover a time-varying expression of a parameter from its spatial distribution. The screenshots in Figure 4 show the same terrain, on which are drawn two different trajectories (clockwise). Thus, we have two different evolutions of the parameter, represented first with colour intensity, then as usual amplitude / time graphs. Therefore, we can easily imagine that, given a terrain there is an infinity of possible dynamic evolutions for its associated parameter. Of course, given a fixed trajectory, depending on the variation of the speed along this trajectory, many different results can be obtained (see, e.g., Figure 5).

Moreover, this has been discussed in the case of a single parameter. Although there might be an infinity of musical results associated with a single terrain set, it doesn’t mean that the piece (even if interactive) escapes to the composer intentions. On the contrary, we will show briefly how the design of a terrain can determine global trends. The only thing known for sure is that the movement on the terrain is continuous: the \((x,y)\) pairs will evolve contiguously (the user cannot “jump” from one point to another). Here are two examples of strategies for terrain design. In the left screenshot of Figure 6 the terrain has been designed with care of continuity, therefore, whatever trajectory or speed will be used onto it, the associated result will be a continuously evolving parameter, while in the right screenshot no interpolation has been used and small discontinuities have been drawn; the parameter will then evolve with discontinuities. Thus, one can determine trends in a certain way and leaving room for variations.

One of the main interests of seeing parameters as terrains is that we can use many different software packages to create 2D pictures to be considered as terrains. The one we have presented has been developed for an almost didactic purpose and has the advantage of being integrated in the Jitter environment. Of course, we could use software such as Photoshop [9], Scilab [10] or terrain editors like the Parametric Terrain Editor (PTE) [11] which has been especially developed for terrain design in 3D software. An example of terrain designed with PTE is shown in Figure 7.

Though the visualization interface is based on an immersive 3D environment, we have only dealt with 2D geometry. It is possible to imagine that the value of a parameter might be used to set both the colour intensity and the \(z\) coordinate of the corresponding point on the terrain (see Figure 8). However, we are working on the realization of tools for “real” 3D design, which implies many parameters.
conceptual problems (though the visualization interface is ready by now for 3D movements). We can also imagine time-varying terrains (videos instead of still images) setting a global timeline for the interactive piece, so that sequences might not be experienced twice, for example.

3. GRAPHICAL TRANSFORMATION OF AN ANALYSED SOUND

We have presented visual interfaces that allow the user to control high-level sound parameters. We will now present a visual interface based on a formal representation of sound. Our aim is to modify the sound helped by a graphical interface of the sound itself after analysis. In order to manipulate the sound graphically, we apply a STFT to a stream of input samples and draw it on a visual window. Scaled to a logarithmic view, the sonogram is then graphically modifiable. The resulting matrix of this transformation is then synthesized (see Figure 9).

The window representing the sound is a time-frequency-amplitude representation. Graphical transformation would necessarily transform sound according to these three parameters. Considering the visual window as a drawing interface, the user modifies the STFT synthesized in real-time. We developed mainly two ways of manipulating the visual STFT.

3.1. Global transformations

We present the different modifications according to the main transformation they concern between time, frequency and amplitude. Some similar operations have been described [8]. They are usually obtained by a phase vocoder, with functions that modify partials. Regarding sound as an image allows us to apply graphical transformations that have not been implemented within a phase vocoder. The interface is transparent due to the formal relation between the image and the sound. Graphical transformations are obtained with standard externals of the Jitter library. Based on Musical transformations using additive analysis / resynthesis of the Computer Music Tutorial [6], we describe the transformations obtained with a graphical control.

3.1.1. Time-based transformations

Considering that the abscissa axis represents the time, all the graphical modifications along this axis will modify the sound.

Time scaling (see Figure 10): Zoom in or zoom out. A zoom in causes a graphical interpolation that sounds like an ordinary deceleration. A zoom out causes a graphical reduction that sounds like acceleration. Due to the time limit of the matrix, in the zoom transformation loss of information occurs. While zooming in, output parts of the matrix are lost. While zooming out, the image is reduced, thus data are lost too.

Non-linear synthesis (see Figure 11): The user can control the time position of the STFT synthesis. The user controls the index of a vector of the matrix, corresponding to a single vertical line of the image. The synthesis thus concerns only 512 samples (0.011s).

The time line can be modified by a control of the index. For example, a broken time line can be used to control the STFT synthesis. In the particular case of the synthesis of a single line, the repetition of 512 samples is a very textured, continuous sound.
Figure 11: The ordinate axis represents the index of the synthesized vector. A non-linear curve causes a non-linear time-play of the sound.

Figure 12: The left image is the STFT of a digital click. Transient attack becomes continuous sound on the right image after a \( \pi/2 \) rotation.

3.1.2. Frequency-based transformations

Considering that the ordinate axis represents the frequency, all the graphical modifications of this axis will modify the sound.

**Spectrum shifting:** Shifting the image up or down. The modified sound consists of the addition of a factor \( n \) to all partials. This shifting can sounds differently according to the algorithm used for scaling the STFT.

**Spectrum inversion:** Vertical inversion of the image. Obtained by a vertical zoom of a factor \(-1\) (low frequencies take place of high frequencies, and vice versa)

**Spectrum zoom:** Zoom in or zoom out of the ordinate axis. In a zoom in operation, the frequency domain zoomed is scaled to a larger frequency domain. Border information and precision are lost due to the finite definition of the STFT. In a zoom out operation, the frequency domain is restricted. Data are scaled to a lower domain.

**Spectrum rotation** (see Figure 12): Rotation of the image based on an anchor point. It causes a linear frequency variation, ascendant or descendant according to the sense of rotation. For a \( \pi/2 \) rotation, a transient attack becomes a continuous sound, and vice versa.

3.1.3. Amplitude-based transformations

Considering that amplitude is a function of the luminance of the image, all the graphical transformation on luminance would influence the sound.

**Kind of reverb:** Apply a blur transformation to the matrix.

**Saturation** (see Figure 13): Saturation / desaturation of light. The matrix is multiplied by a factor \( n \).

The limits of this approach are that this sonic design can be reduced to the drawing of an FFT. Creating sounds with this process encounters graphical needs imposed by the time-frequency representation of the FFT.

3.2. Local transformations

In order to apply local transformations, we use a transfer matrix that allows the user to draw graphic controls, as shown in Figure 14. The transfer matrix uses the \( R, G, B \) layers as different planes of parameters. While the user draws on the transfer matrix, the STFT is processed with each plane of parameters, according to different ways that we will develop now.

We consider that the abscissa axis of the transfer matrix represents time the same way it is represented in the STFT. All the transformations we draw on this interface will affect the sound in time.

The first transformation we program concerns amplitude. We...
use the red plane of the matrix transfer as a frequency filter. The value of each cell, between 0 and 255, is scaled to a multiplying $k$ factor between 0 and 1. The STFT matrix is then multiplied by the red matrix to obtain an amplitude-filtered matrix that can be synthesized (see Figure 15).

The green plane is used for a spectral delay. Each cell of the matrix, also scaled to the interval $[0,1]$, will process the STFT according to a jitter external, $\text{jit.scanslide}$. The STFT bins will be delayed graphically according to the green bins of the transfer matrix. This causes a spectral delayed signal (see Figure 16).

These transformations have been implemented in the Sonos software. The blue plane has not yet been devoted to a specific effect. This approach of sound transforming by a mapping of graphic properties to a musical purpose shows a great potential. Image processing used by video-jokeys could control musical parameters via this interface.

4. REFERENCES

[1] [www.cycling74.com](http://www.cycling74.com)


[7] Egosound is a software downloadable from [www.mshparisnord.org](http://www.mshparisnord.org)

[8] *La terre ne se meut pas* is an artistic virtual environment, for the exhibition “Créer du sens à l’ère numérique”, *H2PTM’03*, Saint-Denis, France.

[9] [www.adobe.com](http://www.adobe.com)

[10] [www.scilab.org](http://www.scilab.org)

Authors' Index

Abel, Jonathan 95
Airas, Matti 367
Alonso, Asier 307
Alvarez, Esteban 27
Antonacci, Fabio 274
Arvidsson, Per 169
Avanzini, Federico 113
Bank, Balázs 89
Barbosa, Alvaro 519
Barry, Dan 240
Batke, Jan-Mark 189
Beeson, Mark 268
Bello, Juan Pablo 207
Beracoechea, Jon Ander 212, 245
Berg, Jonathan 228
Bernardini, Nicola 350
Bilbao, Stefan 119
Bokesoy, Sinan 70
Bonada, Jordi 49, 291
Burnett, Ian 280
Catagna, Vittorio 359
Camurri, Antonio 1
Cano, Pedro 519
Casajús-Quirós, F. Javier 212, 245
Caulkins, Terence 250
Chen, Liang-Gee 351
Cirotteau, Damien 350
Corteel, Etienne 250
Courrèges, Benoît 390
Coyle, Eugene 83, 185, 240
Daudet, Laurent 22
Davies, Mike 207, 256
de Götzen, Amalia 5
De Sena, Antonio 65
Depalle, Philippe 296
Dixon, Simon 193
Dorran, David 85
Driessen, Peter 17
Duxbury, Chris 207
Eisenberg, Gunnar 189
Ekanyaka, Free 350
Ericsson, Johan 262
Erkut, Cumhur 107
Esquef, Paulo A. A. 107
Evangelista, Gianpaolo 145
Every, Mark 197
Fabig, Lars 70, 319
Faller, Christof 151
Foco, Marco 274
Fontana, Federico 113
Frauenberger, Christopher 384
Gainza, Mikel 185
Garcia, Begona 303, 307
Glorioso, Andrea 350
Gouyon, Fabien 193, 319
Herre, Jurgen 157
Herrera, Perfecto 378
Hlavac, Peter 378
Howard, David 311
Huang, Shih-Way 331, 335
Huvilainen, Antti 61
Hämaäläinen, Perttu 11, 367
Holdrich, Robert 384
Jamer, Jordi 70
Jang, Dae-young 315
Jehan, Tristan 361
Jung, Hoe-Kyung 315
Kang, Kyeongok 315
Karjalainen, Matti 107
Kauppinen, Ismo 45
Keiler, Florian 285
Kimm-Bolte, David 42
Klimek, Jaroslav 38
Kling, Garry 33
Kollegger, Michael 57
Koppenberger, Markus 319
Krach, Bernhard 101
Langwagen, Guillermo 27
Lawlor, Robert 83, 185, 240
Lazzarini, Victor 53, 356
Leistikow, Randal 228
Livshin, Arie 222
Loscos, Alex 49, 319
Loyo, Estivaliz 207
Lysaght, Thomas 177, 181
Mac Manus, Lorcán 177, 181
Marchand, Sylvain 76
Maroil, Matija 217
Mendez, Rafael 27
Merimaa, Juha 159
Miklavic, Stanley J. 169, 262
Mitranouidis, Nikolaos 256
Moon, Han-gil 315
Mullen, Jack 311
Murphy, Damien 268, 311
Mäki-Patola, Teemu 11, 367
Ortiz-Berenguer, Luis I. 212, 245
Pakarinen, Jyri 107
Pampalk, Elias 578
Peissig, Jurgen 285
Petrusch, Stefan 101
Potard, Guillaume 280
Pulkki, Ville 159, 367
Purnhagen, Heiko 163
Putz, Veronika 384
Rabenstein, Rudolf 101
Rajmic, Pavel 58
Raspaud, Martin 76
Rath, Matthias 175
Reiss, Josh 372
Roads, Curtis 33, 302
Rocchesso, Davide 65, 113
Rodet, Xavier 222
Roth, Kari 45
Rui, Ibon 307
Röbel, Axel 254
Sandler, Mark 207, 372
Sarti, Augusto 274
Schoenwiesener, Marc 177
Schwarzbacher, Andreas 181
Sedes, Anne 390
Sikora, Thomas 189
Smith III, Julius O. 95, 228
Smyth, Tamara 95
Sporer, Thomas 133
Sturm, Bob L. 323
Sujbert, László 89
Szymanski, John 197
Theile, Günter 125
Theilemann, Henning 201
Thiébaut, Jean-Baptiste 390
Thomburg, Harvey 228
Timoney, Joseph 177, 181
Tonelli, Massimiliano 256
Torres-Guijarro, Marisol 212, 245
Trautmann, Robert 57
Tsai, Tsung-Han 331, 335
Tubaro, Stefano 274
Tzanetakis, George 17
Verfaillie, Vincent 296
Vicente, Javier 305, 307
Vicinanza, Domenico 359
Viste, Harald 145
Valimaki, Vesa 107
Wack, Nicolas 195
Walsh, Rory 53
Wang, Yi-Wen 335
Warschel, Olivier 250
Wyse, Lance 345
Yeh, Chunghsin 234
Zhu, Xinglei 169
Zita, Andreas 169
Zöller, Udo 285