AutoRhythm: A Music Game With Automatic Hit-Timing Generation and Percussion Identification

Tzu-Chun Yeh and Jyh-Shing Roger Jang, *Member, IEEE*

***Abstract*—This article describes a music rhythm game called *AutoRhythm*, which can automatically generate the hit timing as game contents from a given piece of music, and identify user-defined percussion of real objects in real time for gameplay. More specifically, *AutoRhythm* can generate the hit timing of a piece of music based on onset detection, so the user can use any music from their own collection for the rhythm game. Moreover, to make the game more realistic, *AutoRhythm* also allows the user to interact with the game via any object that can produce percussion sounds, such as a pen or a chopstick hitting against a table. *AutoRhythm* can identify the percussions in real time to replace tapping on the screen. This real-time user percussion identification is achieved based on the frame-based power spectrum of the filtered recording after background music reduction, which is performed based on the concept of active noise cancellation, with the estimated noisy playback music being subtracted from the original recording. Based on a test data set of 100 recordings, our experiment indicates that our system can achieve an *F*-measure of 78.22%, which outperforms other well-known classifiers and is quite satisfactory for the purpose of gameplay.**

***Index Terms*—Active noise cancellation, automatic hit-timing generation, music game, percussion identification, rhythm game.**

I. Introduction

MUSIC-BASED games are quite popular on mobile platforms since they can create close and intense interactions between the user and the device. Recently, music rhythm games have attracted considerable attention due to a wide variety of game types and rich interactivity. During such rhythm games, users usually respond to the onset events of a piece of music by pressing a button or tapping on the screen directly. Users can even compete with each other through LAN or internet. At the end of the game, the user is given a score and its ranking based on his/her accuracy and consistency in timing when compared with the music's predefined "hit timing."

As smartphones are becoming widely available, music rhythm games have appeared to be one of the most popular game genres due to the high interaction and immersion during gameplay. Starting from *Tap Tap Revenge* [1], a number of music rhythm games have been developed and widely spread since then, such as *Cytus* [2], *Deemo* [3], *Jubeat Plus* [4], *Piano Tiles* [5], *Dream Piano* [6], and *Tiles Hop* [7]. Recently, several music rhythm games claim to have automatic content generation from users' music collection to deal with the lack of music for game contents, including *BeatMp3* [8], *TapTube* [9], *Melobeat* [10], and *Musiverse* [11] These games can take user-owned mp3 files or YouTube music to generate the contents for gameplay. However, some of these games generate random hit timing with no correlation to music. On the other hand, some of the games do generate the hit timing based on the detected onsets in the music, but the assignment to tracks is randomly permutated. In an unformal discussion in Chinese,[1] this issue truly affected the players' user experience since most music game players expected the hit timing to be permutated in a "reasonable" way, and most of the existing rhythm games which can automatically generate the game contents cannot satisfy these game players.

Another trend of music rhythm games is to increase the interaction with users in order to create immersive experience. Instead of press buttons by fingers, this kind of rhythm games tend to interact with users by dummy instruments or body moves. For instance, *Dance Dance Revolution* [12] interacts with users by their footsteps. *Samba de Amigo* [13] lets users play the rhythm game through dummy maracas. *Drummania* [14] was developed with a whole dummy drum set to create an immersive environment that lets users feel like playing true drum sets in the game. The latest version of *Guitar Hero* [15] allows up to 4 players to play dummy keyboard, guitar, bass, and vocal simultaneously to render a complete song. With VR technology, *BeatSaber* [16] is the first successful rhythm game in the virtual space in which a user can hold two virtual lightsabers to slash the incoming notes and scores. Audioshield [17] is another VR music rhythm game that shares similar idea to BeatSaber. Moreover, there are several games combining the concepts of rhythm games and shooting games. *Beat Hazard* and *Symphony* [18], [19] were two representative indie games of this kind, which also support automatic content generation. These games indicate the importance and necessity to create a variety of interactions with users such that a seamless immersive game environment can be rendered.

In this article, we describe an innovative rhythm game called *AutoRhythm* that supports advanced content generation as well as use of physical objects for gameplay. More specifically, the

Manuscript received September 5, 2018; revised April 25, 2019; accepted July 12, 2019. Date of publication August 20, 2019; date of current version September 15, 2020. *(Corresponding author: Tzu-Chun Yeh.)*

T.-C. Yeh is with the Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan (e-mail: kenshincs@mirlab.org).

J.-S. R. Jang is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: jang@mirlab.org).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TG.2019.2936033

[1]The original discussion is in Zhihu, a famous question-answering website in China. Link: https://www.zhihu.com/question/26133992

game not only generates playable hit timing for any music but also groups the generated hit timing by its acoustic features. This grouping method is based on *k*-means clustering to assign onsets of similar timbre to the same track to create a more "structured" game contents. Moreover, the game supports real-time user percussion identification (UPI) that can recognize different kinds of users' specific percussion input on the fly. The feature of the game creates a different interaction scenario, which allows the user to take two objects (such as a pen and a chopstick) to hit on a hard surface (such as a table or a plastic box) to generate the percussive inputs for gameplay. Furthermore, we propose a novel background music reduction (BMR) method based on the concept of active noise cancellation (ANC) to strengthen the reliability of UPI.[2]

The rest of this article is organized as follows. Section II introduces work related to the underlying approaches of the proposed system. Section III gives an overview of the proposed system. Section IV describes the technical details involved in the system, including the method for automatic hit-timing generation, the approach for music reduction in the recording, and the features and classifier for real-time UPI. Section V discusses experimental results and error analysis. Section VI concludes the paper with possible directions for future work.

II. Related Work

In Section I, we have introduced a number of music rhythm games with various characteristics. Recently, more and more music rhythm games support game content generation based on the user's music collection or online music source like YouTube. This procedure of automatic content generation, also known as procedural content generation [20], has been widely used in different game genres. Jordan *et al.* [21] brought the concept of procedural content generation into the music-based game called *BeatTheBeat*, which utilized music's audio features in creating game boards and game contents for various mini games such as rhythm games and tower defense, etc. However, no detailed experiments or analysis was reported regarding the quality of the generated contents, especially the generated rhythms and onsets. To generate an accurate list of hit timing, we use onset detection to identify the onset positions for a given piece of audio music. Traditional onset detection is based on spectral analysis [22]–[24]. Böck *et al*. surveyed the online capabilities of several onset detection methods [25]. More recently, Schlüter and Böck proposed the use of convolutional neural network for onset detection [26], which outperforms traditional one. Our system employs a fast yet reliable spectral-flux-based onset detection method, which will be briefly introduced in Section IV-A.

UPI is an essential part of the proposed *AutoRhythm* in order to support real-time identification of two types of percussion generated by the user via physical objects. Here we need to design a relatively fast and reliable method for UPI to achieve better user experience during gameplay. Wessel and Wright mentioned that a reasonable latency of interaction when using computational musical instrument is under 10 ms [27]. Moreover, since the user may change their percussion objects right before each gameplay, we cannot use a pre-trained model for UPI to fit all users and all games. Instead, we need to invoke an efficient pre-game training on the device before each gameplay. Work on UPI is scarce in the literature. The most similar work is on drum sound identification in the literature, as explained next.

1) Herrera *et al*. [28] compared various combinations of features and classifiers for drum sound identification. Schloss [29] used the power of selected frequency bins and music structures to transcribe percussive music. However, both studies [28], [29] can only classify monosyllabic and clear percussive sounds without background music.
2) Gouyon *et al*. [30] grouped percussive sounds into two classes (snare-like and bass-drum-like) via agglomerative clustering, with features related to attack time, spectrum, and zero-crossing rate. Yoshii *et al.* [31]–[33] used an iterative adaptation algorithm to obtain the adapted template of the power spectrum of percussive sounds in polyphonic signals. However, the methods employed in these studies require length computation and thus not suitable for online applications.
3) Dittmar and Gärtner [34] utilized a nonnegative matrix factorization-based method to separate the signal source to distinguish high-hat, kick, and snare in real time. However, since the model must be pre-trained in advance and thus not suitable for our game scenario which requires fast pre-game training before each gameplay.

Instead, our system can train the detection model for the sounds of user-specific percussion objects in 3–5 s on the device before each gameplay, and identify the given percussion sounds in around 2–5 ms during gameplay, for a common mobile phone. The details of the algorithms for our system will be described in Section IV-B

ANC reduces noise by subtracting the noise source after passing it through an appropriate transfer function implemented as a filter. To use ANC, we need to be able to measure the noise source precisely, and then use system identification (SID) techniques to identify the transfer function, where the desired output is taken as the mixture of the clean signals and the transformed noise. Since its debut in the 1930s [35], ANC has been widely applied in various scenarios, such as to reduce engine noise within helicopter cockpits [36], or to differentiate fetal heartbeats from mothers' [37]. Most of the recent research on ANC is focused on hardware implementation, such as headphones [38]–[40] with ANC. Our study, on the other hand, is to utilize the concept of ANC to enhance the recognition rate of UPI from our previous work [41]. In particular, we use ANC to reduce recorded playback music such that the user's percussion input can be enhanced. In other words, the original music is viewed as a noise source, requiring a transfer function to convert the original music to the recorded music. Once the transfer function is obtained via system identification, we can subtract the predicted component of the transformed music from the recording to enhance the

[2]An example video that introduces this game is available through: https://www.youtube.com/watch?v=lkjEa1iV4P0

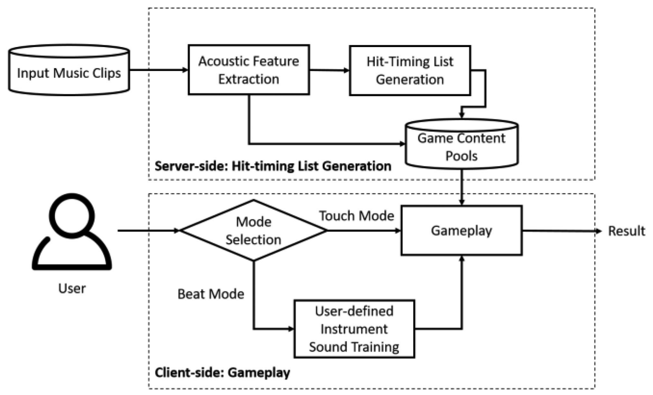

Input Music Clips
Acoustic Feature Extraction
Hit-Timing List Generation
Game Content Pools
Server-side: Hit-timing List Generation
User
Mode Selection
Touch Mode
Gameplay
Result
Beat Mode
User-defined Instrument Sound Training
Client-side: Gameplay

Fig. 1. System overview of our game. On the server side, the hit-timing list is generated by the extracted acoustic features, and stored in the Game Content Pool. On the client side, once the music is determined, the corresponding game contents (either generated or cached at the server) are downloaded from the server. Moreover, a pregame training for UPI is invoked on the client device for identifying the models for users' percussion. Once the game contents and the UPI models are ready, the use can proceed with the gameplay on the mobile device.

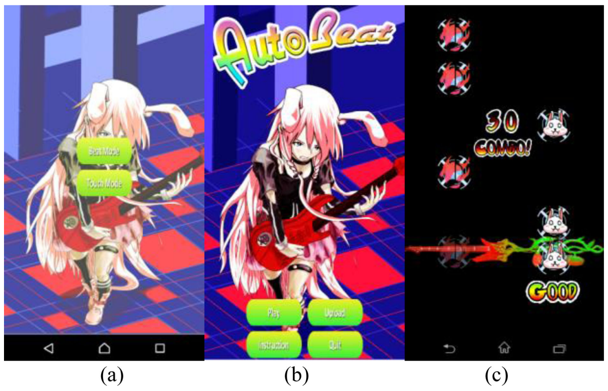


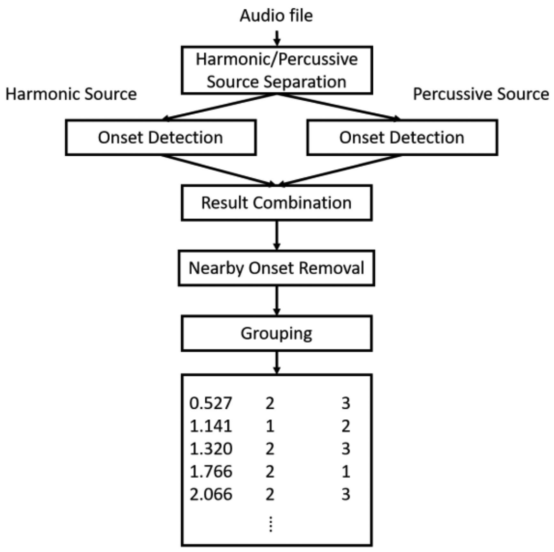
Fig. 2. Screenshots of *AutoRhythm* during mode selection and gameplay. (a) Mode selection. (b) Main screen. (c) Gameplay.

user's percussion input. Section IV-C describes the proposed method in detail.

III. System Overview

Fig. 1 shows the basic blocks of the proposed system. This game system is developed in an Android 4.4.2 OS with Java. A user can upload any music to our server to generate the hit-timing list. User can then start playing the game in two different modes (to be explained later). Fig. 2 shows a screenshot of *AutoRhythm*, with two streams of down falling icons, where the hit timing coincides the time the icon meets the horizontal bar.

AutoRhythm provides two modes for playing the game. In the traditional playing mode, the user needs to tap the horizontal bar on the screen when an icon hits the bar. To make the game more realistic and fun, the proposed system allows players to tapping on any solid surface instead of merely tapping on the screen. For example, the user can create the percussion by tapping a pencil on a desk. The game takes two different percussion sounds obtained from two different objects (tapping on potentially different surfaces) to account for the tapping on the left and right streams of icons. These sounds are recorded by the user before game playing, and the proposed system can construct a classifier for real-time identification of each percussion (see Section IV-A for details).



0.527	2	3
1.141	1	2
1.320	2	3
1.766	2	1
2.066	2	3
	⋮	

Fig. 3. Flowchart of hit-timing generation. The input audio file is separated into harmonic and percussive sources in the HPSS block. Then we perform onset detection on both sources, combine the result, remove redundant onsets, and group remained onsets into two tracks with different degree of difficulty for gameplay.

IV. Features and Method

A. Hit-Timing Generation

Fig. 3 illustrates the flowchart of hit-timing generation in *AutoRhythm*. The hit timing of a given piece of music usually coincides with onsets of either harmonic sources (such as string instruments or human voices) or percussive sources (such as drums or cymbals). As a result, in order to achieve a better accuracy of onset detection, we need to separate the music into harmonic and percussive sources by harmonic/percussive source separation (HPSS) [42]. This step is especially important for identifying soft onsets of harmonic sources. After HPSS, a spectral-flux-based onset detection method is applied to both sources to find the onsets. In particular, we partition the spectrogram into four equal-divided parts according to their frequency bins, and assign a weight for each part. These weights (eight in total) are then finetuned by Nelder–Mead Simplex search [43] to achieve the best performance for onset detection. The goal of this improved onset detection is to find different weighting among frequency bins in both sources in order to approximate human's perception (labeled as the groundtruth for hit-timing) for both the percussive source (such as human's voices and percussive instruments) and the harmonic source (mostly harmonic instruments). The final hit timing is obtained by combining the onsets from both sources and eliminating hit timings that are too close in time. Moreover, based on their density in time, the hit timings are grouped into various sets corresponding to different levels of difficulty in the rhythm game.

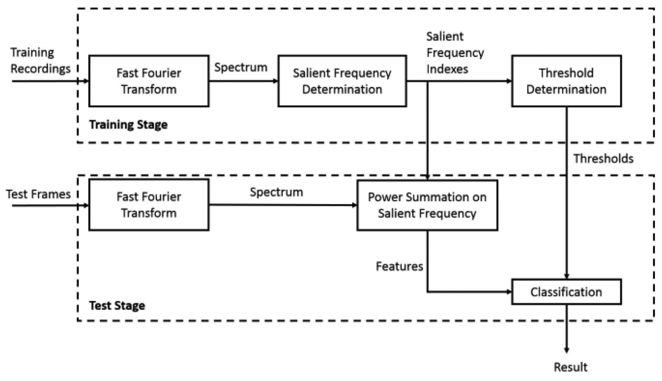


Fig. 4. Flowchart of the proposed method for UPI.

The output is a hit-timing file shown in Fig. 3, where the first column is the hit timings in seconds, the second column is the channel ID of the falling icons, and the last column is the levels of difficulty. The channel assignment is based on the result of *k*-means clustering for Mel-Frequency Cepstral Coefficients at the hit timings, such that different channels are likely to correspond to onsets from different instruments.[3]

B. *User Percussion Identification*

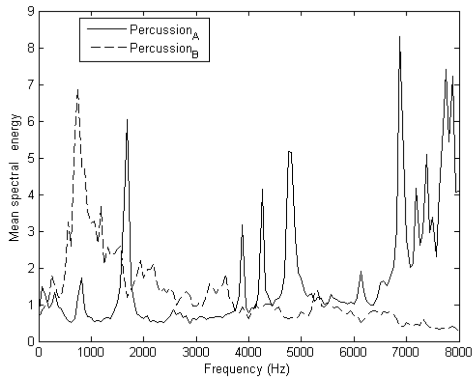
Due to the very nature of gameplay, our algorithm for UPI must fulfill the following requirements.

1) Before gameplay: We need to perform fast pre-game training based on the user's percussion inputs from the left and right hands.
2) During gameplay: We need to perform real-time identification for the timing of the user's percussions, and classify it into two types, either from the left or the right hand.

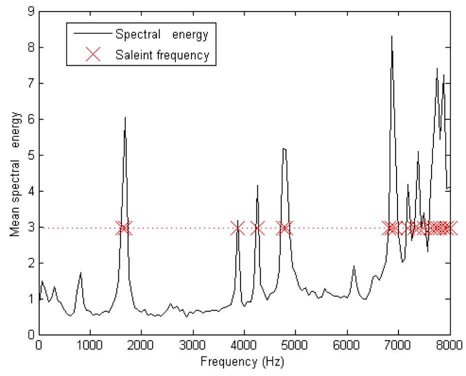
To fulfill these requirements, we propose a simple yet effective method which classifies an incoming frame according to spectrum-based features, as shown in Fig. 4. The features and the classification algorithm are described in detail in Sections IV-B1 and IV-B2.

1) Features: After observing the magnitude spectra of percussive sounds from different objects against different surfaces, we found that different percussive sounds are likely to have different spectral patterns with distinctive peaks at different frequencies, as shown in Fig. 5(a). Based on this observation, we can define the salient frequencies as those frequencies with energy higher than α percentile of all energy for a given type of percussion, as shown in Fig. 5(b). These salient frequencies are thus specific to a percussion sound and they are unlikely to be interfered by the music being played. Before users start to play the game, they are asked to record two types of percussive sounds to be used for the game. Each recording lasts for 5 s, and energy-based endpoint detection is applied to locate the percussions. After obtaining the frames containing the percussive sounds, the first three frames of each percussion sound are used to calculate the salient frequencies because the timbre of a percussion sound may change over time and we only care about the frames at percussion onset.

[3]For the example code of hit-timing generation, refer to: https://zenodo.org/record/3368941#.XVUqJegzaF4.



(a)



(b)

Fig. 5. (a) Mean spectral energy of two different kinds of percussive sounds. (b) Salient frequencies of percussion$_A$ in Fig. 5(a). The information of both percussions are listed in Table II.

In this study, we consider only two types of percussions to be used in the game, resulting in two sets of salient frequency bins which are identified before each game. We use the total energy within these salient frequency bins as the frame-based feature for classifying a frame into percussion$_A$, percussion$_B$, or none of the above. The determination of α will be described in Section V-C.

2) Proposed Algorithm: As mentioned above, the system needs to generate the prediction almost immediately, so here we propose an efficient and reliable method for prediction within a single frame time, as follows. After obtaining the salient frequencies of each percussion, the average energy at each salient frequency of three frames is calculated. Then, the average energy on salient frequencies are summed up and multiplied by a given ratio β to obtain the threshold pair, which respectively produce the thresholds of two types of percussion, θ_a and θ_b.

Let $\mathrm{SF}_{(i,A)}$ and $\mathrm{SF}_{(i,B)}$ respectively denote the sum of magnitude on salient frequencies of frame$_i$ of percussion$_A$ and percussion$_B$. Our method needs to determine if there is a percussion by the user or not within a single frame time, as explained in the following three rules.

1) If both $\mathrm{SF}_{(i,A)}$ and $\mathrm{SF}_{(i,B)}$ are larger than their respective thresholds θ_a and θ_b, then the bigger one makes the call.

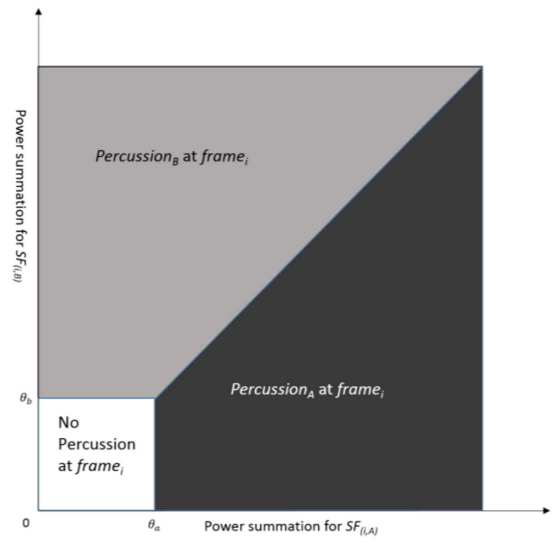


Fig. 6. Decision boundary for 3-class (percussion$_A$, percussion$_B$, and none) classification based on energy of salient frequency pair (SF$_{(i,A)}$, SF$_{(i,B)}$).

Algorithm I: Pseudo Code for Classifying a Frame.

```
if $SF_{(i,A)} > \theta_a$ and $SF_{(i,B)} > \theta_b$
    if $SF_{(i,A)} > SF_{(i,B)}$)
        '$percussion_A$ at $frame_i$.'
    else
        '$percussion_B$ at $frame_i$.'
    end
else if $SF_{(i,A)} > \theta_a$
    '$percussion_A$ at $frame_i$.'
else if $SF_{(i,B)} > \theta_b$
    '$percussion_B$ at $frame_i$.'
else
    'No percussion at $frame_i$.'
End
```

2) If either one of SF$_{(i,A)}$ or SF$_{(i,B)}$ is larger than its corresponding threshold, then the surpassing one makes the call.
3) Otherwise, there is no percussion at all.

Since the proposed method does not utilize any long-term feature, the prediction of a frame can be returned almost immediately. The pseudo code for the proposed method is shown in Algorithm 1. The corresponding decision boundary of the above method is also shown in Fig. 6 for easy visualization. The optimization of β will also be detailed in Section V.

Since a percussion sound usually lasts for more than one frame, some postprocessing is necessary to prevent a sound from being identified as two events. In fact, in the 1980s Takahashi Meijin set the world record for triggering 16 times per second in gameplay with a joystick. Therefore, it is unlikely for a player to make two consecutive percussive sounds within 62.5 ms (1/16 = 0.0625). In this study, each frame lasts for 16 ms (256/16 000 = 0.016), so we set a rule that if frame$_i$ is identified as a percussion sound, then frame$_{i+1}$ to frame$_{i+3}$ are considered as nonpercussion. Furthermore, the energy of a percussion sound decreases with time, so another rule states that the percussion counts only when SF$_{(i,X)}$ > SF$_{(i-1,X)}$.

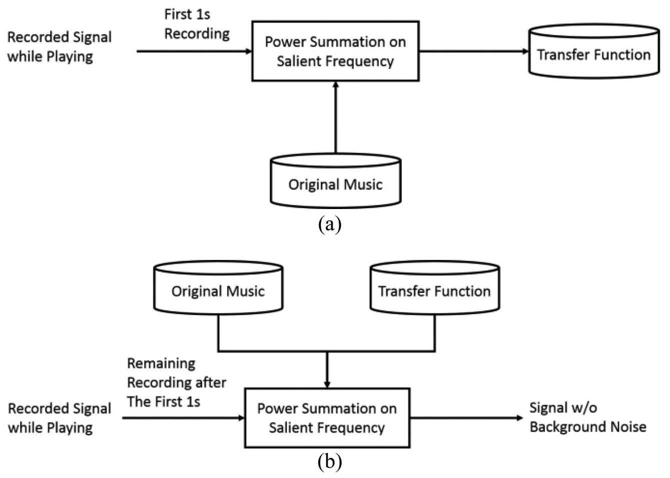

Recorded Signal while Playing
First 1s Recording
Power Summation on Salient Frequency
Transfer Function
Original Music
(a)
Original Music
Transfer Function
Remaining Recording after The First 1s
Recorded Signal while Playing
Power Summation on Salient Frequency
Signal w/o Background Noise
(b)

Fig. 7. Flowchart of the proposed method for BMR. (a) Flowchart of transfer function modeling. (b) Flowchart of background noise subtraction.

C. Background Music Reduction

Results in the UPI experiments indicate that more undesirable false positives were generated, which was not found in our previous work due to the data diversity [41]. After examination, these additional false positives are most likely induced by the background music while users perform percussions. Thus, it becomes critical to reduce the background music from the mixed recordings in order to improve the overall UPI.

Fig. 7 shows the flowchart of the proposed method for BMR. The goal of the method is to reduce the interference from the background music such that the extracted percussive sounds, as input by the user, are cleaner and thus easily identified by the classifier proposed previously.

Here we can apply the concept of ANC in this scenario, where the noise source is the original music (available to us) and we have the mixture recording which combines users' percussion and playback of the background music. In particular, the playback of the background music is a transformed version of the original music, so if we can find a filter (or a transfer function) that takes the original music and generate the playback music, then we can subtract the playback music from the mixture recording to obtain a clean version of the users' percussion. In the framework of ANC, the users' percussion can be viewed as a random noise while we are performing SID based on the original music (the input part to SID) and the mixture recording (the ground-truth of SID) directly to derive the transfer function (that accounts for the transformation from the original music to the playback music). In other words, by using the concept of ANC, we are able to find the transfer function of the recording environment which converts the original music to the playback music, and the percussive sounds act as a random component that should be ignored during the identification process. In practice, this SID process should be performed immediately before each game session since environmental conditions and device positioning (e.g., lying on top of a table, or leaning against a wall) will significantly affect the recording, and thus the transfer function. We do not perform online SID during gameplay since it is time consuming and the percussions in the recording (which acts as a random component within the output part of SID) can sometimes degrade the identified transfer function.

One way to make SID a transparent part of gameplay is to incorporate a prompt stage in which the user hears an audio prompt of “3, 2, 1, go” right before the game starts. We can simply take the first 1 s of the audio prompt to perform SID to obtain the transfer function of the recording environment. The identified transfer function can thus be used to estimate the music component in the mixture recording based on the original music, such that the user’s percussion inputs can be reliably extracted for better classification.

According to [44], several models, including linear and nonlinear ones, have been proposed in the literature for SID. Since efficient model construction is absolutely necessary for our real-time application of gameplay, we adopted a linear regression model of moving average (MA, which is a special case of ARX model in [44]) of order n for SID, where the target signal $Y_{i,t}$ at time t can be expressed as a linear combination of the input signals at various instants before t

$$Y_{i,t} = \gamma_0 + \sum_{j=1}^{n} \gamma_i X_{i,t-j} + \varepsilon_i. \tag{1}$$

Equation (1) can be rewritten as a matrix form

$$Y = X\gamma + \varepsilon \tag{2}$$

where Y is the target signal, X is the original signal, ε is the white noise, and γ is the desired MA coefficient, which can be obtained by the least-squares method. The corresponding experiments for this part will be described in detail in Section V-C.

Fig. 8 shows a typical example of BMR based on ANC, where (a) and (b), respectively, demonstrate the cases without and with BMR. Specifically, Fig. 8(a) shows the recorded signals (upper) and the corresponding onset strength curve when BMR is not invoked.

On the other hand, Fig. 8(b) shows a similar example when the BMR is turned on. It is obvious the recording with BMR exhibits distinct percussions, which leads to a better feature set for UPI. Moreover, the “purified” percussions in the recordings can also facilitate better classification for the percussions.

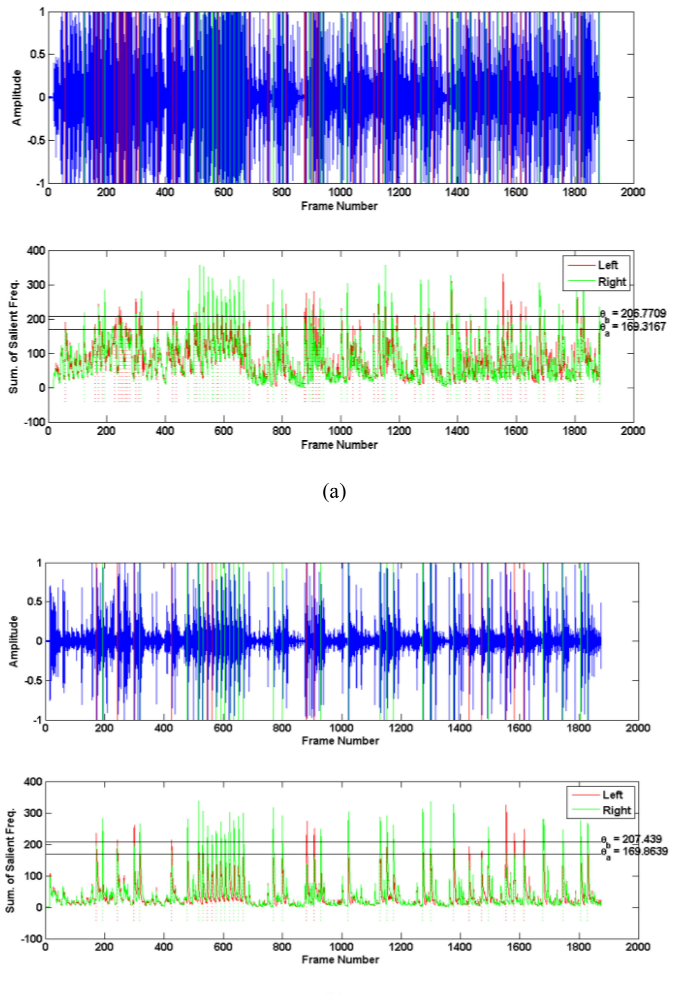


Fig. 8. Comparison between signals without and with BMR. (a) A typical example of recorded signals without using BMR. (b) The same signals with BMR. Note that common to both (a) and (b), the upper plot is the signals (without and with BMR) and the lower plot is the corresponding features described in Section IV-B1. Apparently with BMR, it becomes easier for us to identify percussions.

V. Experiments

A. Experiment on Hit-Timing List Generation

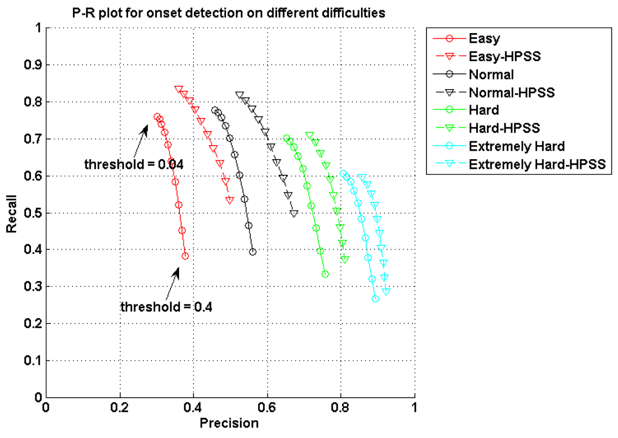
The data set for the experiment of hit-timing generation consists of music in MP3 format and corresponding hit-timing lists. All the music used in the experiment are collected from the web site [45]. The hit-timing lists of the data set in [45] are labelled by music game players into four difficulties of “easy,” “normal,” “hard,” and “extremely hard,” depending on the density of the hit timing. Since each music piece may have more than one hit-timing lists by different players, there are 977 human-labelled hit-timing lists in total based on a collection of 607 music pieces of six genres. The sample rate and the resolution of the music clips are 44.1 kHz and 16 b, respectively. The breakdown table of the hit-timing lists into different genres and difficulty levels is shown in Table I.

The goal of the experiment is to evaluate the performance of the proposed method when compared with the hit timing labeled by human. We use the plots of precision-recall and F-measures to evaluate the performance, with a tolerance of 0.05 s. The onset peak-picking threshold is set to a ratio the max value of the onset strength curve, with the ratio being 0.04 to 0.4, with a step of 0.04. The precision-recall and F-measure plots are illustrated in Fig. 9(a) and (b), respectively.

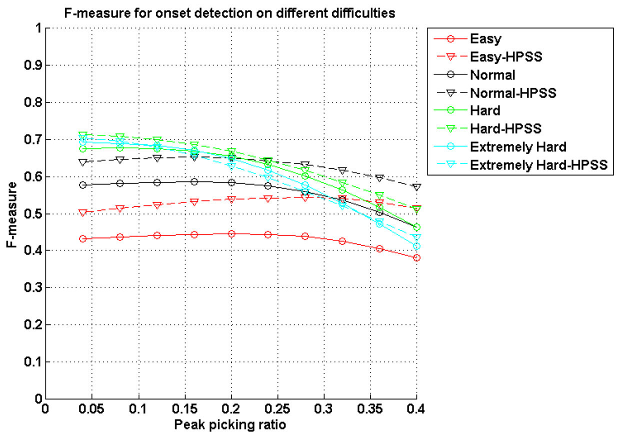
As shown in Fig. 9(a), most of the precision-recall curves with HPSS are better than those without HPSS, indicating the necessity of using HPSS for hit-timing generation. The corresponding F-measure curves shown in Fig. 9(b) also demonstrate

TABLE I
NUMBERS OF HIT-TIMING LISTS FOR VARIOUS GENRES AND DIFFICULTY LEVELS IN OUR DATA SET FOR THE EXPERIMENT OF HIT-TIMING GENERATION

Difficulty→ / Genre (no. of music clips) ↓	Easy	Normal	Hard	Extremely Hard	#Total
Original (91)	31	31	32	60	154
Child (23)	8	9	11	21	49
Classical (56)	6	8	10	36	60
Game (84)	19	23	26	61	129
Pop (338)	84	105	129	238	556
Movie (15)	3	4	8	14	29
#Total (607)	151	180	216	430	977



(a)



(b)

Fig. 9. (a) Precision-recall plot of hit-timing generation, with each point in a curve corresponding to a value of peak-picking ratio. (b) F-measure plot of hit-timing generation by different peak-picking ratios. Both plots indicate that the use of HPSS can generally lead to better performance. The definitions of the F-measure, precision and recall are shown in (3)–(5).

the effectiveness of HPSS, which enable us to identify onsets based on different criteria for harmonic and percussive sources. We have also performed Wilcoxon signed rank test [46] on the pairwise results w/o HPSS. The test result shows a p-value of 0.002 at the significance level $\alpha_p = 0.05$, which indicates the significance of using HPSS. However, although the onset detection with HPSS does improve the performance, it does make more mistakes when dealing with harmonic instruments and human vocals. One way to tackle this problem is to use different weights for different frequency bins when computing spectral flux. This will be a major direction of our future work.

B. *Experiment on User Percussion Identification*

In order to have a more objective result for UPI, we have constructed a new data set which is bigger than the one used in our previous work [41]. This new data set consists of 100 recordings obtained from ten users during their gameplay. More specifically, each user randomly selected ten audio clips from the GTZAN data set [47], with one clip from each genre. The percussive instrument pairs were also determined by the users without specific restriction. The hit timing for each music piece is generated by the proposed method, while the ground-truth (locations and types of percussions by the user) are transcribed by human. All of the recordings were obtained with an Android pad with Android OS 4.4.2, with a single channel, a sample rate of 44.1 kHz, and a bit resolution of 16 b. There are four possible results after the classification of a frame, namely, correct, insertion, deletion, and confusion. Assuming that there are two types of percussion denoted as percussion$_X$ and percussion$_Y$, we can define the following quantities.

1) *Correct$_X$*: The number of percussion$_X$ frames which are classified correctly (a percussion sound is considered to be classified correctly when the deviation from the groundtruth is within 2 frames or 32 ms.)
2) *Insertion$_X$*: The number of no-percussion frames which are classified as percussion$_X$.
3) *Deletion$_X$*: The number of percussion$_X$ frames which are classified as background music.
4) *Confusion$_{(X,Y)}$*: The number percussion$_X$ frames which are classified as percussion$_Y$.

For comparison, we also applied several well-known classifiers, including support vector machine (SVM), Gaussian mixture models (GMM), K-nearest neighbor classifier (KNN), Naïve Bayes classifier (NB), and quadratic classifier (QC), with the proposed features of energy in salient frequency bins to classify each frame into three categories of no-percussion, percussion$_X$, and percussion$_Y$. For SVM, the RBF kernel is used in this experiment since the performance is better than linear and polynomial kernels for the validation test. For KNN, we set $k = 1$ as the number of nearest neighbors to be picked since the performance is slightly better other values of k which was tested from 1 to 10. For GMM, the number of mixtures is equal to 2, and the type of the covariance matrix is diagonal. About 100 frames are randomly selected from the no-percussion potions of the recordings to be our training data of the no-percussion frames. The training data for percussions is obtained from the percussion recordings recorded by the user right before the gameplay.

The overall performance index is represented by F-measure, with the F-measure of percussion$_X$ being defined as follows:

$$F_x = 2 \times \frac{\text{precision}_x \times \text{recall}_x}{\text{precision}_x + \text{recall}_x} \tag{3}$$

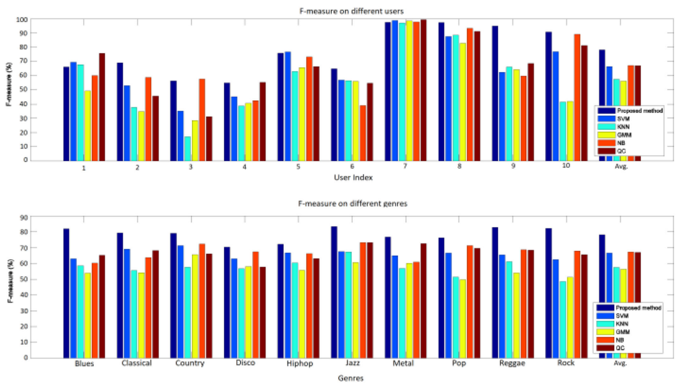


Fig. 10. User-specific and genre-specific F-measures for the proposed method and various different classifiers.

$$\begin{aligned}\text{precision}_x &= \text{true positive}/\text{test outcome positive}\\ &= \text{Correct}_x/\left(\text{Correct}_x + \text{Insertion}_x + \text{Confusion}_{(Y,X)}\right)\end{aligned} \tag{4}$$

$$\begin{aligned}\text{recall}_x &= \text{true positive}/\text{condition positive}\\ &= \text{Correct}_x/\left(\text{Correct}_x + \text{Deletion}_X + \text{Confusion}_{(X,Y)}\right)\end{aligned} \tag{5}$$

The user-specific and genre-specific F-measures are shown in Fig. 10, respectively, indicating the proposed method outperforms other classifiers in almost all recordings, with an average F-measure of 78.22%, which compares favorably with NB's 67.21%, QC's 67.02%, and SVM's 66.58%. Additionally, we performed Wilcoxon signed rank test on the results. For pairwise results of the proposed method and other classifiers, the p-values are all below 0.001 at the significance level $\alpha_p = 0.05$, indicating the significance of the improvement. The Bonferroni-correction p-values are also below 0.005 (five comparisons) at the significance level $\alpha_p = 0.05$.

In this experiment, we think there is at least one compelling reason why all the other classifiers do not perform as well as the proposed method. Note that these classifiers were constructed from the training data obtained from the background music which varies considerably from frame to frame. In other words, the training set of the background music does not have stable and consistent features since the background music is not consistent through all recordings during gameplay. As a result, a general good model of background music is thus difficult to obtained, leading to undesirable performance.

It should be noted that SVM's performance is highly influenced by its parameters [48], and the optimum parameters vary significantly for different percussive sounds. For example, the best parameters for classifying percussion instruments of user index 1 with BGM-1 are gamma = 0.0001, cost = 1, while the best parameters for classifying percussion instruments of user index 2 with BGM-2 are gamma = 0.0004, cost = 1. Therefore, the parameters which maximize the overall F-measure may not be the best parameters for each user's recording.

TABLE II
DETAILED INFORMATION OF EACH TYPE OF USER-INPUT PERCUSSIONS FOR BMR

Name of the user-input percussions	Object against which the steel stick is tapped
percussion$_A$	A smaller ceramic cup
percussion$_B$	A larger ceramic cup
percussion$_C$	A plastic box

C. Experiment on Background Music Reduction

For the experiment of BMR, we have another set of 30 recordings with user-input percussions, which were created during gameplay and transcribed by human to label the ground-truth (locations and types of percussions). To create the recordings, we first selected ten background music clips randomly from the GTZAN data set, with one for each genre. There are three different user-input percussions, as shown in Table II. For each piece of background music, we generated three recordings based on pairwise combinations of these three types of percussions, leading to a total of 30 recordings. For each recording, we picked the first second without user-input percussions for identifying the transfer function of the device from the source file to playback recording. All the other part of the recording is used to test the performance of the identified transfer function for percussion classification.

The goal of this experiment is to verify if the proposed method of BMR can improve the recalls and precisions of the previous percussive sounds identification. More importantly, we also want to know if the proposed method can be used in real-world scenarios. As mentioned in Section V-A, for this experiment, we need to separate each music clip in the data set into training and test segments. The training segment consists of the first 1 s of each recording with no user-input percussions, while the remaining part of each music clip is used for testing the performance. In other words, the training segment is used to construct the transfer function of the recording environment [see Fig. 7(a)], which is then used in the test segment to reduce the background music [see Fig. 7(b)]. Here we choose to use an MA (moving average) model to predict the transformed music after recording. Before identifying the MA model, we need to determine its order to achieve the best performance. This is accomplished by leave-one-out cross validation on the training segment, where the "one" (that was left for validation) refers to each sample point in the training segment. A typical result of this order-determination procedure for a given clip is shown in Fig. 11, where the best MA order of 380 is obtained at the lowest validation root mean squared error (RMSE).

For the proposed method, the optimal values of α (percentile for determining salient frequencies) and β (ratio for determining θ_a and θ_b) have to be identified in advance. Since the energy distribution may change during BMR, the optimal values of α and β may change before and after BMR. To determine the parameter pair (α, β) of original clips and (α', β') of clips after BMR, we employ the Nelder–Mead simplex method [48] for gradient-free optimization. Through leave-one-file-out cross

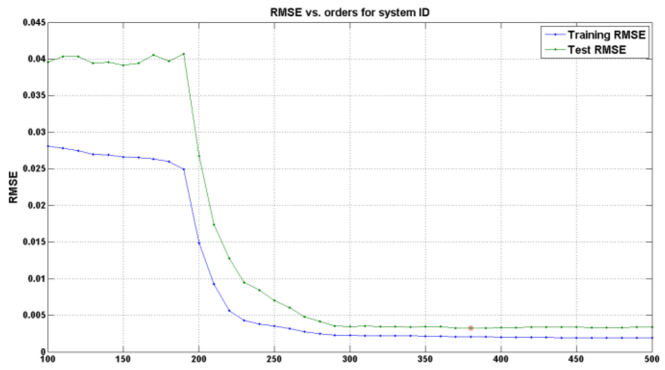


Fig. 11. Result of leave-one-out validation test for the best order of AR model.

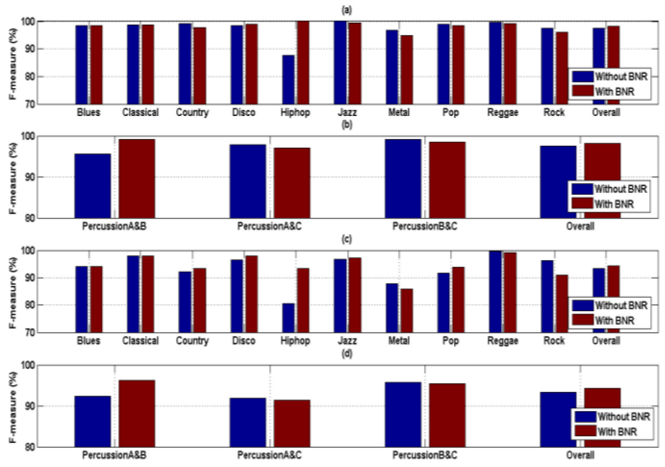
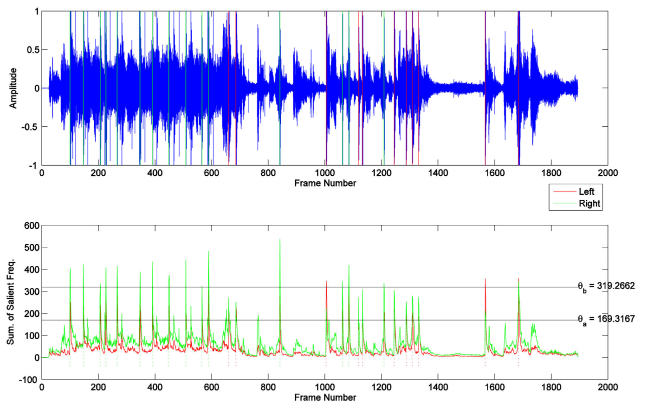
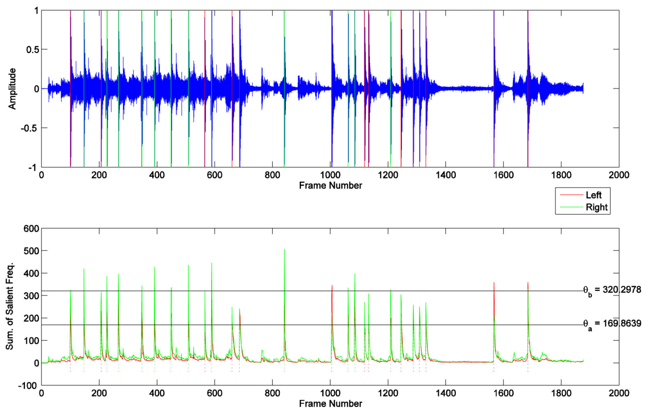


Fig. 12. F-measures with/without BMR using ANC for (a), (c) different genres of background music and (b), (d) different percussion pairs.



(a)



(b)

Fig. 13. (a) Typical misclassified example after BMR. The first and third estimated percussions in (b) are misclassified to left-class after BMR due to the energy change.

validation, the ratio pairs (α, β) and (α', β') are set to (24.10%, 89.41%) and (24.80%, 89.70%), respectively.

By using the above order-determination procedure, we can identify the best AR model order and perform model identification for each recording based on its training segment. The results of the experiment are shown in Fig. 12. By applying the proposed method for BMR and optimizing the ratio pair, the percussion-only F-measure is improved from 97.49% to 98.18%, and the overall F-measure is improved from 93.30% to 94.34%. A breakdown analysis, shown in Fig. 12, indicates that the F-measures for most music clips are improved, especially for Hiphop (for improvements from 87.66% to 100% in percussion-only F-measure and from 80.47% to 93.33% in overall F-measure).

Another breakdown analysis demonstrates that the average F-measure improves, as shown in Fig. 12(b). Specifically, we can see that the percussion-only F-measure of percussion$_A$ and percussion$_B$ improves from 95.59% to 96.24%, while that of the pair percussion$_A$ and percussion$_C$ slightly degrades from 97.77% to 96.96% and the pair percussion$_B$ and percussion$_C$ slightly degrades from 99.11% to 98.43%. The overall F-measure has the same trend that percussion$_A$ and percussion$_B$ improves from 92.27% to 96.24%, while that of the pair percussion$_A$ and percussion$_C$ slightly degrades from 91.82% to 91.39% and the pair percussion$_B$ and percussion$_C$ slightly degrades from 95.80% to 95.40%. Moreover, we performed Wilcoxon signed rank test on the overall results. The test result shows p-value is equal to 0.0428 at the significance level $\alpha_p = 0.05$, indicating the significance of the proposed ANC method.

As shown in Fig. 12(a)–(c), the improvement in the genre Hiphop is the most obvious since in the original music, the user-input percussions are highly interfered by the background music, and the background music is also louder than music of other genres. In particular, a typical example of Hiphop is shown in Fig. 8(a), where the volume of the background music is much louder than the user percussion inputs. By applying the proposed method, the volume of the background music as shown in Fig. 8(b), is much less than the one in Fig. 8(a), leading to a better accuracy. However, several cases show that the reduction of the background music energy may also interfere the energy of the salient frequency bins, resulting misclassification. A typical example is shown in Fig. 13. In Fig. 13(b), several percussions are misclassified in the first phrase (frame indices of 1 to 700 or so) of the clip due to the energy ratio changed in both

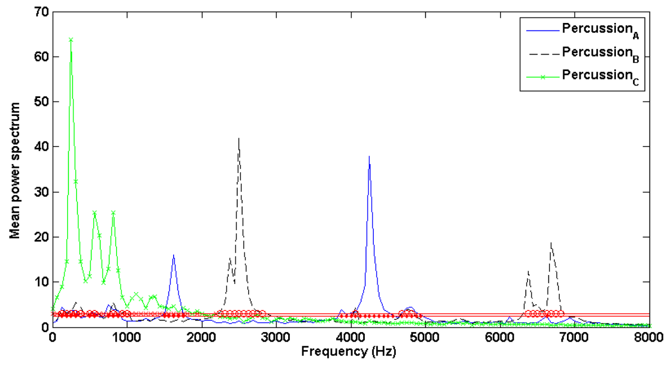


Fig. 14. Distribution of the salient frequency for each percussion, respectively.

salient frequency bins. For the different percussion pairs, there is a slightly degrade in both percussion$_A$ and percussion$_C$ and percussion$_B$ and percussion$_C$ due to the interference of the energy in the salient frequency by the background noise reduction. A possible reason for degradation is the similarity of the salient frequency between percussion$_C$ and the ones in the percussion sound of the reference music, leading to the misclassification between the percussion pair or the silence. For more details, Fig. 14 shows the distribution of the salient frequencies for each percussion. The salient frequencies of percussion$_C$ are mostly around 0–1500 Hz. Half of them are overlapped with the salient frequencies of percussion$_A$ and percussion$_B$. From the other observation, the energy of the lower frequency (below 1000 Hz) is interfered by the BMR the most, leading to the degrade of the classification of percussion$_A$ and percussion$_B$ and percussion$_A$ and percussion$_C$.

VI. Conclusion and Future Work

In this article, we improved the previously proposed music rhythm game *AutoRhythm* [41], with the introduction of the newly proposed method for BMR, together with extensive experiments with larger data sets for reliable and convincing findings. Moreover, we have provided comprehensive comparisons with existing rhythm games to pinpoint and distinguish our game in the game spectrum. More specifically, *AutoRhythm* has two unique and innovative features that make it stand out when compared with similar games in the literature.

1) Our system can automatically locate the hit timing of user-provided music and assign notes to different tracks in a "structural" way instead of random assignment. Such a meaningful assignment can minimize the discrepancy from manually generated game contents, and thus greatly increase user experience.
2) The user can tap 2 physical objects (e.g., a pen, a chopstick, etc.) against any other object (e.g., a table top, a cup rim, etc.) to produce clear and distinct percussive sounds to engage in the gameplay rather than touching or tapping the screen. The process of UPI is further improved by the proposed method of BMR based on the concept of ANC. Experimental results show that the proposed method for UPI can achieve an *F*-measure of 78.22%, which is better than other classifiers and quite satisfactory for gameplay purposes.

Future work will be focused on two directions. First, a more advanced method is called for to produce better hit-timing, not only to approximate human-labeled ground-truth, but also to increase playability and fun. To achieve this short-term goal, a reliable vocal and melody onset detection in polyphonic music should be designed to perform vocal detection and melody pitch extraction in polyphonic audio music, and note segmentation for the onset of each note in the extracted pitch vector. The second direction is a long-term goal that aims to render the proposed game content generation compatible to other advanced multimodal music games with various interaction and gameplay modes, such as music-based dancing or shooting games. To achieve this goal, a more general method for game content generation is needed for closer interaction, with supervised or semi-supervised learning to automate the whole process, thus creating a better user experience for music game players.

References

[1] Tapulous, Tap Tap Revenge, 2008. [Online]. Available: https://en.wikipedia.org/wiki/Tap_Tap_Revenge
[2] Rayark, Cytus, 2011. [Online]. Available: https://www.rayark.com/
[3] Rayark, Deemo, 2013. [Online]. Available: https://www.rayark.com/
[4] Konami, Jubeat Plus, 2012. [Online]. Available: http://www.konami.jp/jubeatplus/index.php5
[5] Cheetar Technology Co. Ltd., Piano Tiles, 2014. [Online]. Available: https://itunes.apple.com/uz/app/piano-tiles/id848160327?mt=8
[6] Tap Lab, Dream Piano, 2017. [Online]. Available: https://play.google.com/store/apps/details?id=com.eyu.piano&hl=en_US
[7] Amanotes, Tiles Hop, 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.amanotes.beathopper&hl=en
[8] CreApptive, BeatMp3, 2013. [Online]. Available: https://play.google.com/store/apps/details?id=com.studio7775.BeatMP3&hl=en
[9] SmartPlayland, TapTube, 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.joylol.taptube&hl=en
[10] Float32, Melobeat, 2017. [Online]. Available: https://play.google.com/store/apps/details?id=com.float32.themelobeat&hl=en
[11] Pocket Games, Musiverse, 2015. [Online]. Available: https://play.google.com/store/apps/details?id=com.pocketgames.musiverse
[12] Konami, Dance Dance Revolution, 1998. [Online]. Available: https://p.eagate.573.jp/game/ddr/ddra20/p/?___REDIRECT=0
[13] Sega, Samba de Amigo, 1999. [Online]. Available: https://en.wikipedia.org/wiki/Samba_de_Amigo
[14] Konami, DrumMania, 1999. [Online]. Available: https://p.eagate.573.jp/game/gfdm/gitadora_matixx/p/index.html?___REDIRECT=0
[15] Activision, Guitar Hero, 2006. [Online]. Available: https://www.guitarhero.com/
[16] Beat Games, BeatSaber, 2018. [Online]. Available: https://store.steampowered.com/app/620980/Beat_Saber/
[17] Audio Surf LLC, AudioSurf, 2016. [Online]. Available: https://store.steampowered.com/app/412740/Audioshield/
[18] Cold Beam Games, Beat Hazard, 2009. [Online]. Available: https://store.steampowered.com/app/49600/Beat_Hazard/
[19] Empty Clip Studios, Symphony, 2012. [Online]. Available: https://store.steampowered.com/app/207750/Symphony/
[20] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. New York, NY, USA: Springer, 2016
[21] A. Jordan *et al.*, "Beat: Music-based procedural content generation in a mobile game," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 320–327.
[22] S. Dixon, "Simple spectrum-based onset detection," in *Proc. Int. Conf. Music Inf. Retrieval*, 2006.
[23] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in musical signals," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 1035–1047, Sep. 2005.

[24] J. P. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Process. Lett.*, vol. 11, no. 6, pp. 553–556, Jun. 2004.

[25] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods," in *Proc. 14th Int. Conf. Music Inf. Retrieval*, Porto, Portugal, 2012, pp. 49–54.

[26] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2014, pp. 6979–6983.

[27] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Comput. Music J.*, vol. 26, no. 3, pp. 11–22, 2002.

[28] P. Herrera, A. Yeterian, and F. Gouyon, "Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques," in *Proc. Int. Conf. Music Artif. Intell.*, 2002, pp. 69–80.

[29] W. A. Schloss, "On the automatic transcription of percussive music - From acoustic signal to high-level analysis," Ph.D. thesis, Dept. Music, Stanford Univ., Stanford, CA, USA, 1985.

[30] F. Gouyon, F. Pachet, and O Delerue, "On the use of Zero-Crossing rate for and application of classification of percussive sounds," in *Proc. COST G-6 Conf. Digital Audio Effect*, 2000.

[31] K. Yoshii, M. Goto, and H. G. Okuno, "Automatic drum sound description for Real-world music using template adaptation and matching methods," in *Proc. Int. Conf. Music Inf. Retrieval*, 2004, pp. 184–191.

[32] K. Yoshii, M. Goto, and H. Okuno, "AdaMast: A drum sound recognizer based on adaptation and matching of spectrogram templates," in *Proc. Int. Conf. Music Inf. Retrieval*, 2005.

[33] K. Yoshii, M. Goto, and H. Okuno, "Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 15, no. 1, pp. 333–345, Jan. 2007.

[34] C. Dittmar and D. Gartner, "Real-time transcription and separation of drum recordings based on NMF decomposition," in *Proc. 17th Int. Conf. Digital Audio Effects*, 2014.

[35] P. Lueg, "Process of silencing sound oscillations," U.S. Patent No. 2043416, Jun. 9, 1936.

[36] L. J. Fogel, "Method of improving intelligence under random noise interference," U.S. Patent No. 2866848, Dec. 30, 1958.

[37] B. Widrow and M. A. Lehr, "Noise canceling and channel equalization," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA, USA: MIT Press, 1995, pp. 648–650.

[38] A. V. Oppenheim, E. Weinstein, K. Zangi, M. Feder, and D. Gauger, "Single-sensor active noise cancellation," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 2, pp. 285–290, Apr. 1994.

[39] B. Benoit, C. Camastra, M. Kenny, K. Li, R. Romanowski, and S. Kevin, *Engineering Silence: Active Noise Cancellation*. Raleigh, NC, USA: North Carolina State Univ., 2012.

[40] S. Liebich, C. Anemüller, P. Vary, P. Jax, D. Rüschen, and S. Leonhardt, "Active noise cancellation in headphones by digital robust feedback control," in *Proc. 24th Eur. Signal Process. Conf.*, 2016, pp. 1843–1847.

[41] P.-P. Chen, T.-C. Yeh, and J.-S. R. Jang, "*AutoRhythm*: A music game with automatic Hit-Timing generation and percussion identification," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2015, pp. 1–6.

[42] N. Ono, K. Miyamoto, J. L. Roux, H. Kameoka, and S. Sagayama, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," in *Proc. 16th Eur. Signal Process. Conf.*, 2008, pp. 240–244.

[43] J. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J*, vol. 7, pp. 308–313, 1965. doi: 10.1093/comjnl/7.4.308.

[44] M. Galrinho, "Least squares methods for system identification of structured models," Licentiate thesis, KTH School Elect. Eng., Stockholm, Sweden, 2016.

[45] Taiko Jiro's Dataset. 2013. [Online]. Available: http://tieba.baidu.com/p/1736272776

[46] I. C. A. Oyeka, *An Introduction to Applied Statistical Methods*, 8th ed. Enugu, Nigeria: Nobern Avocation Publishing Company, 2009, pp. 496–533.

[47] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Audio Speech Process.*, vol. 10, no. 5, pp. 293–302, Jul. 2002. [Online]. Available: http://opihi.cs.uvic.ca/sound/genres.tar.gz

[48] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm

Tzu-Chun Yeh was born in Taipei, Taiwan, in 1985. He received the B.S. degree from the Department of Computer Sciences, National Tsing Hua University, Hsinchu, Taiwan, in 2008, where he is currently working toward the Ph.D degree in computer science.

His research interests include query by singing/humming, audio melody extraction, onset detection, automatically music-game-content generation, and percussion identification.

Jyh-Shing Roger Jang (M'93) received the Ph.D. degree from the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA, USA, in 1992.

He studied fuzzy logic and artificial neural networks with Prof. Lotfi Zadeh, the father of fuzzy logic. As of August 2019, Google Scholar shows over 15 000 citations for his seminal paper on adaptive neuro-fuzzy inference systems (ANFIS), published in 1993. After his Ph.D., he joined MathWorks to coauthor the Fuzzy Logic Toolbox (for MATLAB). He has since cultivated a keen interest in implementing industrial software for pattern recognition and computational intelligence. He was a Professor with the Computer Science Department, National Tsing Hua University, Taiwan, from 1995 to 2012. Since August 2012, he has been a Professor with the Department of Computer Science and Information Engineering, National Taiwan University (NTU), Taipei, Taiwan. He has published one book titled *Neuro-Fuzzy and Soft Computing* (Prentice-Hall). He has also maintained toolboxes for machine learning and speech/audio processing, and online tutorials on data clustering and pattern recognition and audio signal processing and recognition.

Dr. Jang was the General Chair of the International Society for Music Information Retrieval (ISMIR) Conference, Taipei, 2014 and was a General Co-Chair of ISMIR Conference, Suzhou, 2017. He is currently serving as the Director for FinTech Center at NTU. His research interests include machine learning and pattern recognition, with applications to speech recognition/assessment/synthesis, music analysis/retrieval, image classification, medical/healthcare data analytics, and FinTech.