Content-based Music Retrieval Using Linear Scaling and Branch-and-bound Tree Search[1]

Jyh-Shing Roger Jang, Hong-Ru Lee, Ming-Yang Kao

Multimedia Information Retrieval Laboratory
Computer Science Department, National Tsing Hua University, Taiwan
jang@cs.nthu.edu.tw

Abstract

This paper presents the use of linear scaling and tree search in a content-based music retrieval system that can take a user's acoustic input (8-second clip of singing or humming) via a microphone and then retrieve the intended song from over 3000 candidate songs in the database. The system, known as **Super MBox**, demonstrates the feasibility of real-time content-based music retrieval with a high recognition rate. Super MBox first takes the user's acoustic input from a microphone and converts it into a pitch vector. Then a fast comparison engine using linear scaling and tree search is employed to compute the similarity scores. We have tested Super MBox and found the top-20 recognition rate is about 73% with about 1000 clips of test inputs from people with mediocre singing skills.

1. Introduction

This paper presents a content-based music retrieval system, called Super MBox, which allows the user to retrieve an intended song via her/his acoustic input. That is, the system's intelligent comparison engine allows the user to retrieve songs based on a few notes sung or hummed naturally to the microphone. Therefore the traditional ways of song retrieval (particularly in the applications of karaoke or digital music library) based on the search of keywords of titles, singers or lyrics can be totally avoided.

The tasks of Super MBox can be categorized into three stages: preprocessing, on-line processing, and postprocessing. For the preprocessing stage, we need to read each candidate song in the database and put the pitch/beat information into several indexed files for quick access during comparison. Usually the songs in the database are in MIDI (Music Instrument Digital Interface) format, which contains all the music elements of a song and is equivalent to sheet music of the song. The pitch/beat information of a song is extracted from the MIDI's major (or vocal) track, which is defined as the track that, when played alone, can be identified immediately by a human who knows that song. The pitch/beat information of each candidate song is then converted into a pitch sequence that is suitable for comparison. (We can also obtain the pitch sequence of each candidate song from the solo singing of a person or from the playing of a single music instrument.)

During the on-line processing stage, the user can specify a query by singing or humming a piece of tune (that contains several notes for identifying a song) to a PC microphone directly. Autocorrelation [20] is used to find the most likely pitch, and some heuristics (such as continuity and human's pitch range) are employed to eliminate unwanted/false pitches which might result from either unvoiced segments of the acoustic input or the undesirable effect of pitch doubling/halving.

At the postprocessing stage, the computed pitch sequence together with related timing information are transformed into the same middle representation that was used in encoding the pitch/beat information of songs in the database. Before invoking similarity comparison, the middle representation must be transformed into a format that does not rely on the absolute values of the identified pitches. This is usually achieved by the difference operator, as reported in the literature [7][18][17][16][19]. However, we found that the difference operator amplifies noises and leads to poor performance. Thus we adopt a heuristic method to shift the input pitch sequence to have the same average value as that of each song's in the database. The input pitch sequence is then stretched/compressed to match to the template midi files in the database. The branch-and-bound tree search procedure [6] is applied to reduce the time for nearest neighbor search. The system then returns a ranked list according to similarity scores.

The preprocessing requires human's interaction to identify the major track for extracting the pitch/beat information from a song in MIDI format. Once the major track is specified, the pitch/beat transcription and the similarity score computation are totally automatic. The average response time of our system (with 3000 songs) is about 2 seconds on a Pentium III 800 when the user specifies a query from the start of a song. The performance is satisfactory, as long as the users can sing or hum the intended song with more or less correct pitches.

2. Related Work

Ghias et al. [7] published one of the early papers on query by singing. They applied autocorrelation to obtain the fundamental frequency, and the pitch vector is then cut into notes. To accommodate the problem of different starting key, the obtained notes are converted into ternary contour of three characters: U (up, meaning this note is higher than the previous one), R (repeat, meaning this note is the same as the previous one), D (down, meaning this note is lower than the previous one). The comparison engine, based on longest common subsequence, is then applied to the ternary contours to find the most likely song. However, due to the limited computing power at that time, their

[1] This is an on-going project supported by Cweb Inc. (http://www.4music.com.tw) and the Excellency Project sponsored by the National Science Council at Taiwan, where the authors express many thanks.

pitch tracking takes 20-45 seconds, and there were only 183 songs in the database.

R. J. McNab et al. [18][17][16][19], in collaboration with New Zealand Digital Library, have published several papers on their experiment of query by singing. They applied Gold-Rabiner algorithm [8] for pitch tracking, and the pitch vector was then cut into notes based on energy levels and transition amounts. There were about 9400 songs in their database and they are the first one to put their system on the web. Their system, though lack of performance evaluation in terms of recognition rate, is still considered an excellent example of content-based music retrieval for real-world applications.

Kosugi et al. [13] at NTT, Japan, have published several papers on their MIR system called SoundCompass. Their system has 11,132 song versions (or 10,069 songs) and the comparison engine takes several different feature vectors including tone transition, partial tone transition, and tone distribution. The final ranking is a "or'ed" result among all rankings from different features. The retrieval time of SoundCompass is about 1 second and the top-5 recognition rate is about 75%. However, their system requires users to hum “ta” and follow the beats of a metronome, which is a significant restriction and not always possible for other input channel (such as from a telephone.)

The author has also published several papers [3][14][10] on content-based music retrieval. The focus of the publications (in particular the last one) is on the use of dynamic programming techniques for elastic match in the comparison engine.

Other related work can be found at the first International Symposium on Music Information Retrieval held at Plymouth, Massachusetts. Most of the published papers at the conference can be found at the conference web site [9].

3. Operations of Super MBox

The operations of Super MBox can be divided into two steps: pitch tracking and comparison engine.

3.1 Pitch Tracking

The 8-second acoustic input is first put into a low-pass filter with cutoff frequency at 1047 Hz. Then the input signals are put into frames of 512 points, with 340 points of overlap; this corresponds to 1/64 second for each pitch frequency. Then every 4 points of the pitch sequence are averaged to merge into a single frequency, thus the final pitch vector has a time scale of 1/16 second.

There are plenty methods for pitch tracking in the literature. For our system, we have implemented pitch tracking using autocorrelation function [20].

After obtaining the pitch frequencies, we use the following formula to transform them into the representation of **semitone**:

$$semitone = 12 * \log_2\left(\frac{freq}{440}\right) + 69$$

The semitone representation here is equivalent to the one used by MIDI format, where 69 represents central LA (A440, 440 Hz). Subsequent smoothing and comparison operations are based on semitones.

To avoid pitch-doubling and other undesirable effects, we take the following steps to smooth the pitch contour:

1. If the energy level is lower than a threshold, then the corresponding pitch semitones are set to zero (or rest).
2. If the identified pitch semitones are higher than 84 (1046.5 Hz in frequency) or lower than 40 (82.4 Hz in frequency), they are also set to zero (or rest).
3. Use center clipping [20] after autocorrelation.
4. Use a median filter of order 5 to smooth the pitch sequence.

3.2 Comparison Engine Using Linear Scaling and Branch-and-bound Tree Search

Differences between the input pitch sequence and the intended song in the database comes from two origins:

1. Key transposition
2. Tempo variation

Key transposition can be simply handled by mean subtraction. For most users, tempo variation is attributed to linear (proportional) variation. Hence we can apply linear scaling to the input sequence before comparing it to the songs in the database. For simplicity, we assume the comparison always starts at the beginning of a song. Suppose that the identified pitch sequence corresponds to d seconds. Then we need to compress/stretch the sequence to get 10 variants with lengths equally distributed between $0.75d$ to $1.25d$. The distance between the pitch sequence and a song is then defined as the minimum of the ten Euclidean distances between the 10 compressed or stretched versions and the song. The system then lists the top-20 songs with the smallest distances.

The primary goal of Super MBox is to find the songs in the database that is closest to the query input in terms of the distance measure. This is a typical problem of nearest neighbor search (NNS) and has been widely studies in the pattern recognition literature. A more formal definition of NNS can be stated as follows. Given a set of n objects $S = \{o_1, o_2, \cdots, o_n)$ and a query input q , find the object o_k in S such that $dist(q, o_k)$ is the smallest among all $dist(q, o_i), i = 1, \ldots, n, i \neq k$, where $dist(\cdot,\cdot)$ is a distance function defined for any two objects. Most speedup mechanisms [2][21] for NNS assume that the objects in S can be expressed as vectors in a high dimensional space and the distance function is simply the Euclidean distance. To fulfill this constraint, we require the user to singing at least 8 seconds such that we can obtain a pitch sequence of 128 elements. Moreover, whenever there are rests in the pitch sequence, we replace it with the previous non-rest pitch. These little tricks make both q and o_i vectors in 128-dimensional space, so we can apply tree search methods, such as vantage-point tree search [23] and branch-and-bound tree search [6], to speed the process of nearest neighbor search.

4. Performance Evaluation

In this section we present the performance evaluation of Super Mbox based on linear scaling and branch-and-bound tree search.

We have a dataset of around 1000 wave files of singing or humming by about 20 persons (12 males, 8 females). Specs of the dataset are:

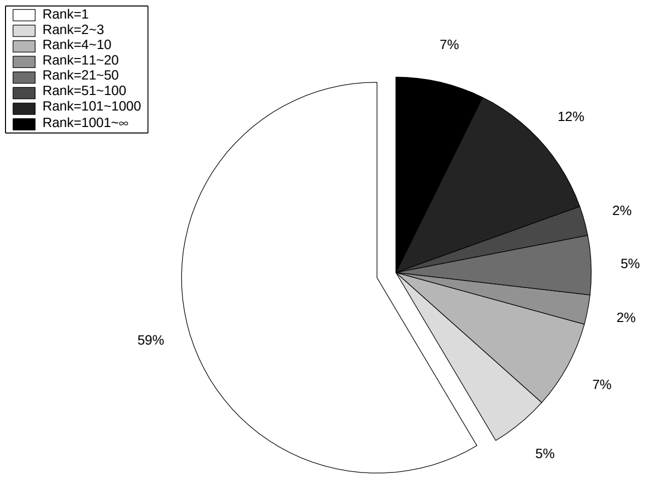
- No. of clips: 1000
- Duration: 8 seconds
- Sample rate: 11025 samples/second
- Resolution: 8 bits
- Type: Single channel (mono)
- Start position: Beginning of a song

All of the recordings start from the beginning of a song. The specs of Super MBox and our platform are listed next:

- CPU: Pentium-III 800MHz
- RAM: 256 MB
- No. of candidate songs: 3000
- Pitch interval: 1/16 seconds (or equivalently, 128 elements in each pitch sequence)
- No. of variants of the input sequence: 10 (whos length are equally spaced between $0.75d$ and $1.25d$.)

In our branch-and-bound tree search, we used k-means clustering method to construct a search tree with 4 levels and each node has 5 children, corresponding to a tree with 156 nodes. The speedup factor brought by the tree search is about 4.

With the use of tree search, the response time of the system is reduced from 8 to 2 seconds. The ranking statistics can be view from the following pie chart:



From the above plot, the top-20 recognition rate (percentage of recordings that Super MBox can find the correct intended songs in the top-20 ranking) is about 73%, the top-3 recognition rate is 64%, and the top-1 recognition rate is 59%.

The high Top-20 recognition rate demonstrate the feasibility of Super Mbox as an real-time content-based music retrieval system. Moreover, since there is no need to do note segmentation, the system does not require the user to sing “la” or “ta”. Hence the user can sing or hum in a way that is the most natural to the user. This can help the user to present the song more accurately.

We did carry error analysis on the low-ranked wave files and found that most of the mistakes comes from non-technical issues, such as:

1. The gain of the microphone is too low, leading to a low signal to noise ratio.
2. Some people started to talk or laugh during the 8-second recording.
3. Some of the recording was performed on a PC with a sound card installed with wrong drivers. This makes the recording quality very poor.

The recognition rate could be higher if we eliminated these disqualified wave files. On the other hand, via the error analysis, we did find certain drawback of our system. Most importantly, clips with low pitch are prone to error. This is a weakness of our pitch tracking algorithm and will be investigated in the near future.

Several implementation techniques also speed up the comparison engine up to a factor of three. These techniques are summarized as follows.

1. Partial distance computation: During the distance computation, if the accumulated distance is already greater than the maximum distance of the current top-20 ranked songs, quit distance computation immediately.
2. Comparison order rescheduling: Always put the most frequently selected songs at the beginning for comparison. This is to be used jointly with "partial distance computation".
3. Integer computation: Do pitch tracking and distance computation using the integer data type instead of floating or double precision data type. The truncated error can be partially offset by multiply integer variables (for instance, pitch in semitone) by a scaling factor, say 10.
4. Programming techniques: There are quite a few programming techniques in C to speed up execution. For instance, avoid using branching statements such as "if" and “switch”:

We have implemented the Super MBox for both standalone and network versions. To test drive the network version of Super MBox, you can download a client program from the author’s homepage at:

http://www.cs.nthu.edu.tw/~jang

The pitch tracking part is done at the client program to reduce server load.

The standalone version of Super MBox can be downloaded from the homepage of CWeb Technology Inc.:

http://www.4music.com.tw

The standalone version has much more personalization features, such as query by speech, and personalized recording for candidate songs, karaoke function, etc.

5. Conclusions and Future Work

In this paper, we have presented a content-based music retrieval system with a comparison engine based on linear scaling and branch-and-bound tree search. The system can take the user’s acoustic input (singing, humming, or music instrument playing) and return the intended song in about 2 seconds. We have performed extensive tests on Super MBox; the response time and recognition rate of the systems demonstrate the feasibility of the system’s usage as the query engine for music digital libraries. We have also implemented a network version so people all over the world can try it directly.

Though Super MBox is a viable project, there are many things that we can explore to improve the system. Immediate future work includes the following items:

1. Low pitch sometimes leads to error in our pitch determination algorithms; we need to explore other methods for a better, reliable pitch tracking. Since pitch tracking takes only a small amount of time, perhaps we can try some other sophisticated methods such as combination of classifiers.
2. To deal with nonlinear tempo variation, we can apply dynamic time warping [20] to serve as a suitable distance measure, as reported in [10]. However, dynamic time warping is computation intensive and we need to find a balance between performance and response time.

6. References

[1] Brown, J. and Zhang, B. “Musical frequency tracking using the methods of conventional and ‘narrowed’ autocorrelation”. *Journal of the Acoustical Society of America*, Volume 89, Number 5, pages 2346-2354, 1991.

[2] Chan, Chok-ki, and Ma, Chi-Kit, “A Fast Method of Designing Better Codebooks for Image Vector Quantization,” IEEE Transactions on Communications, Vol. 42, No. 2/3/4, PP. 237-242, February/March/April, 1994.

[3] Chen, B. and Jang, J.-S. Roger "Query by Singing", 11th IPPR Conference on Computer Vision, Graphics, and Image Processing, PP. 529-536, Taiwan, Aug 1998.

[4] Flickner, M. and Sawhney, H. S., Ashley, Huang, J., Q., Dom, Gorkani, B., Hafner, Lee, M., J., D., Petkovic, D., D. Steele, and Yanker, P. “Query by image and video content: the QBIC system,” IEEE Computers, Vol. 28, No. 9, pp.23-32, 1995.

[5] Foote, J. "An Overview of Audio Information Retrieval," In Multimedia Systems, vol. 7 no. 1, pp. 2-11, ACM Press/Springer-Verlag, January 1999.

[6] Fukunaga, Keinosuke and M. Narendra, Patrenahalli "A Branch and Bound Algorithm for Computing K-Nearest Neighbors", IEEE Transactions on Computers, July 1975.

[7] Ghias, A. J. and Logan, D. Chamberlain, B. C. Smith, “Query by humming-musical information retrieval in an audio database”, ACM Multimedia ’95 San Francisco, 1995. (http://www2.cs.cornell.edu/zeno/Papers/humming/humming.html)

[8] Gold, B. and Rabiner, L. "Parallel processing techniques for estimating pitch periods of speech in the time domain," J. Acoust. Soc. Am. 46 (2), pp 442-448, 1969.

[9] International Symposium on Music Information Retrieval (MUSIC IR 2000), Plymouth, Massachusetts, Oct. 23-25, 2000. (http://ciir.cs.umass.edu/music2000/)

[10] Jang, J.-S. Roger and Gao, Ming-Yang "A Query-by-Singing System based on Dynamic Programming", International Workshop on Intelligent Systms Resolutions (the 8th Bellman Continuum), PP. 85-89, Hsinchu, Taiwan, Dec 2000.

[11] Katsavounidis, Ioannis and Kuo, C.-C Jay and Zhang, Zhen, “Fast Tree-Structured Nearest Neighbor Encoding for Vector Quantization,” IEEE Transactions on Image Processing, Vol. 5, No. 2, PP. 398-404, Feb. 1996.

[12] Kosugi, N. Y., Kon'ya, Nishihara, S., Yamamura, M. and Kushima, K. "Music Retrieval by Humming – Using Similarity Retrieval over High Dimensional Feature Vector Space," pp 404-407, IEEE 1999.

[13] Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M., and Kushima, K., "A practical query-by-humming system for a large music database," In Proc. ACM Multimedia 2000, November 2000.

[14] Lee, I-Yang, Jang, J.-S. Roger and Hsu, Wen-Hao "Content-based Music Retrieval from Acoustic Input", 12th IPPR Conference on Computer Vision, Graphics, and Image Processing, PP. 325-330, Taiwan, August 1999.

[15] Liu, C. C. and Chen, A. L. P., “A Multimedia Database System Supporting Content-Based Retrieval”, *Journal of Information Science and Engineering*, 13, PP. 369-398,1997.

[16] McNab, R. J. and Smith, L. A. “Melody transcription for interactive applications” *Department of Computer Science University of Waikato*, New Zealand.

[17] McNab, R. J., Smith, L. A. and Witten, Jan H. “Signal Processing for Melody Transcription” *Proceedings of the 19th Australasian Computer Science Conference*, 1996.

[18] McNab, R. J., Smith, L. A. and Witten, Jan H. “Towards the Digital Music Library: Tune Retrieval from Acoustic Input” ACM, 1996.

[19] McNab, R. J., Smith, L. A., Witten, I. H. and Henderson, C. L. "Tune Retrieval in the Multimedia Library,"

[20] Proakis, J. R. J. G. and Hansen, J. H. L. "Discrete-time processing of speech signals," New York, Macmillan Pub. Co., 1993.

[21] Torres, L. and Huguet, J., “An Improvement on Codebook Search for Vector Quantization,” IEEE Transactions on Communications, Vol 42, No. 2/3/4, PP. 208-210, February/March/April, 1994.

[22] Uitdenbogerd, A. and Zobel, J. "Melodic Matching Techniques for Large Music Databases", (http://www.kom.e-technik.tu-darmstadt.de/acmmm99/ep/uitdenbogerd/)

[23] Yianilos, Peter N. “Data structures and algorithms for nearest neighbor search in general metric spaces,” In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311-321, Austin, Texas, 25-27 January 1993

[24] Yianilos, Peter N. “Excluded Middle Vantage Point Forests for Nearest Neighbor Search,” NEC Research Institute Technical Report, 1998