A NEW APPROACH TO QUERY BY HUMMING IN MUSIC RETRIEVAL

Lie Lu, Hong You, Hong-Jiang Zhang

Microsoft Research, China
5F, Beijing Sigma Center
49 Zhichun Road, Beijing 100080, China
{i-lielu, hjzhang}@microsoft.com

ABSTRACT

In this paper, we present a method for querying desired songs from music database by humming a tune. Since errors are inevitable in humming, tolerance should be considered. In order to suit or adapt to people's humming habit, a new melody representation and new hierarchical matching method are proposed in this paper. Query processing and database processing algorithm are also described in detail. The proposed approach achieves high performance in experimental evaluation. For 88% queries, the correct song can be retrieved among the top 10 matches.

1. INTRODUCTION

With the rapid growth of digital audio data, rapid access of such data is strongly desired, especially for large music databases. Content-based method is an appropriate choice, at least complementary to textural description based methods. Some early works on content-based audio or music retrieval are totally based on signal processing and using acoustic similarity, such as in [4]**.** Query by humming(QBH) is another popular content-based retrieval method for large music database. QBH system aims at searching a desired piece of music by singing or whistling its tune, which is very useful when you want to find a song from music library but forget its title or artist.

Most of recent works on QBH is focused on melody representations, similarity measures and matching processing. In some works, such as [1][3], only pitch contour is used to represent melody. Music melody is transformed to a stream of U, D, R, which stands for a note is higher than, lower than, or equal to the previous note, respectively. But it simplifies the melody so much that it cannot discriminate music very well, especially when the music database is large. In order to represent the melody more accurately and discriminatively, new feature sets have been proposed. In [2], pitch interval and rhythm are considered as well as pitch contour. In [7], relative interval slope is used in music information retrieval. And [6] introduces four basic segment types(A,B,C,D) to model music contour.

When rhythm and pitch interval is considered, more complex similarity measure and matching algorithm should be used. [6] uses two-dimensional augmented suffix tree to search the desired song, rather than approximate string matching algorithm used in [1][3]. In [5], a new distance metrics between query and songs is proposed. But its computation is very time-consuming because it need adjust many parameters step by step to find the minimum distance.

In fact, not only should melody representation maintain melody information as much as possible, but also it should suit the habits of people humming. Considering this request, a new melody representation is proposed in this paper. Correspondingly, a new hierarchical matching algorithm is also proposed.

Our proposed QBH system contains three main components, which are query processing module, melody database and matching engine, as shown in Figure 1. All MIDI files are processed first. Melody features are extracted from them and stored. When a search is initiated, humming query is converted into melody representation, which is in turn used to match against the melody in the feature database, according to their similarities and produce a rank list of songs.

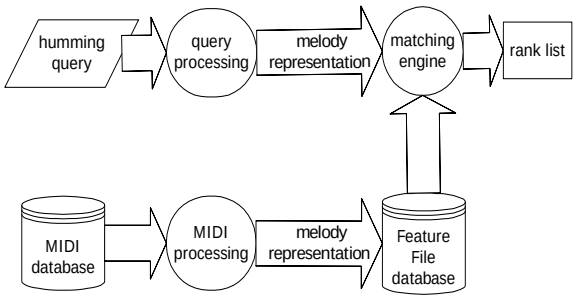

Figure 1. QBH System flowchart

The rest of this paper is organized as follows. A new representation of melody is proposed in Section 2. The algorithms on how to extract the melody representation feature from a MIDI file and humming are also discussed. Then, a hierarchical matching algorithm is presented in Section 3. In section 4, experiments and evaluations of the proposed algorithm are given.

2. MELODY REPRESENTATION

As mentioned above, a good melody representation should capture all melody information and suit people's humming habits. Considering this principle, we have proposed a triplet melody representation.

First, melody contour is used. Unlike [1], which uses UDR sequences, we do not use R(Repeat), because it is very difficult if not impossible to distinguish each repeated note from continuous humming correctly. Repeated notes have the same pitch, so we can discriminate them only in time domain according to its energy information. However, when one hums by lyric, one always hums continuously. Therefore, there may be no obvious energy break when a note is repeated. Thus, it is impossible to extract all R information from humming. Even if we meet repeated note and detect it in database or humming, we just merge them as one note.

Second, it is much easier for human to discriminate or reproduce the relative pitch variations than absolute pitch from perception [9]. So we select pitch interval rather than absolute pitch as one component of melody.

At last, rhythm is also important melody information, obviously. In our melody representation, we use each note's duration as a feature. When a note is repeated, its duration is added to the previous one.

We use these three components to represent melody information. That is, we have a triplet:

(*pitch contour, pitch interval, duration*)

We use UD string to represent pitch contour, where U or D indicates a note is higher or lower than the previous one, respectively. Pitch interval stands for the amplitude of note variation, which is the difference between the frequencies of two consecutive notes. Duration represents how long a note is played or hummed. That is, it represents rhythm. The feature file of a song is composed of a series of these triplets.

For example, a segment of scores from a song is illustrated in Figure 2.

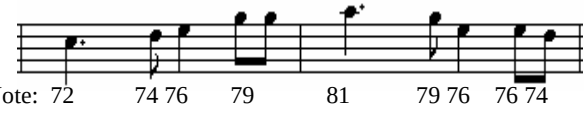

Figure 2. Part of the score of a song

Using our triplet melody representation, this segment is represented as:

(*, *, *)(U, 64.1, 1) (U, 71.9, 2) (U, 124.7, 2) (U, 96.0, 3) (D, -96.0, 1) (D, -124.7, 3)(D, -64.1,1)

Because the first note has no reference, so it is represented by a star, which will not be used in match process.

2.1. MIDI database processing

We construct our music database directly from web collected MIDI files. In addition to melody information, MIDI file also contains other information that will become noise at the matching process and decrease retrieval performance drastically. In general, there are three possible ways in processing multi-track MIDI files:

1. If MIDI file denotes the melody track, then, select the main melody track and omit all other tracks.
2. Keep all tracks in MIDI file.
3. Combine all tracks into one. When meet simultaneous notes or chord, only select the highest pitch note the same way as [8].

Method 1 is the best way, because human always hums the main melody of a song. Unfortunately, not every MIDI file denotes which track is melody track. So it is difficult to select melody track from each MIDI file automatically. Method 2 keeps all tracks, so that the performance will decrease because other tracks have negative influence on retrieval. Method 3 combines all tracks. We experimented this approach by combining all the tracks and playing it back. The melody of original music is changed a little by this way. So it is uncomfortable to use in retrieval.

In order to overcome the above shortcomings, we proposed a heuristic method compromising above methods. If MIDI file denotes the melody track, we select it and discard other tracks. If we have no information on which is the melody track, then, we use each possible metadata to exclude all impossible tracks. For example, channel 10 is usually fixed to percussion, so it is always excluded. Also, if the length of one track is very short, it will be discarded too. Although it is impossible to discard all non-melody tracks, this process removes noise as much as possible.

After the candidate track is obtained, we select the highest note if there exists chord in one track. Finally, we convert the track into our triplet melody representation, and store them into feature files.

2.2. Query processing

Few papers discuss how to process humming signal. However, it really has an important influence on system performance. If we can extract melody from humming correctly, the desired song can be retrieved more accurately. Figure 3 illustrates our query processing algorithm in detail.

In our algorithm, energy and zero-crossing rate are used to discriminate silence and noise from useful humming signal, because pitch detection module always obtain a false pitch value from these segments. After removing these false pitches, we use a median filter to smooth pitch contour and remove atypical points.

An adaptive method is then used to get the melody representation. It is divided into three steps.

First, energy contour is used to divide the humming into segments. The transition from one segment to another represents a note change. Following rule is applied in this process: when energy drops into a very low level and then goes up rapidly, or after a long silence, a new note may begin. In fact, this method is very coarse; thus, it often misses note changes. It is because when one hums a song with lyric, one usually hums it so continuously that no

obvious energy drop exists when some notes change. Nevertheless, the result of this process can be used as a reference. Different methods are used for intra and inter segment processing.

Then, it is detected if there are note changes in one segment (intra-segment) according to the pitch contour. Simple clustering and merging for pitch is processed first. If there exists a large pitch variance, e.g., larger than a threshold *th1*, it is likely a note change occurs. At the same time, U or D information and pitch interval of the segment is obtained. Thus, if a note is repeated in one segment, it will not be detected. That is, R(repeat) information cannot always be extracted. For this reason we do not use R information in our melody representation.

Finally, we extract UD information and pitch interval from inter-segments, a different threshold *th2* is used at this step. Generally, we set *th1* > *th2*, because of human humming habit. People can't keep the pitch relativity well after a pause.

Integrating the above three steps, we can get the melody representation from human vocal input.

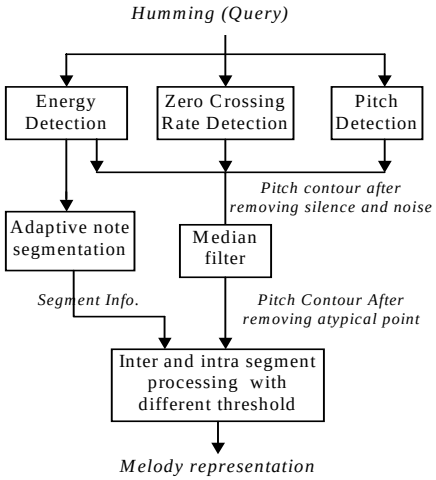

Figure 3. Query processing algorithm

3. HIERARCHICAL MATCHING

For those works which use distance or area metrics in matching, the algorithm is usually very time-consuming. This is because the absolute value of rhythm and pitch interval of humming are different from those of original song, and both scales on time and frequency need to be adjusted step by step to compute the similarity score [5]. For those works only using pitch contour, the match process can be very fast but the accuracy is low. A promising way to match humming against database should be both fast and accurate. Hierarchical matching method is proposed to satisfy this request and the algorithm is illustrated in Figure 4.

The algorithm is divided into three steps:

1. UD contour alignment. Approximate string matching and dynamic programming are used to align the UD contours between query and candidate music segments. If the distance N_{mis} is larger than a threshold, we discard this unqualified piece. Otherwise, the matched path between the pairs will be recorded for further processing. Here N_{mis} is the count of deletion, insertion and substitution.

2. According to the matched path, compute the similarity of pitch interval and rhythm between the pairs. Suppose the aligned pitch interval list is $\{q_i\}$ and $\{d_i\}$, and the aligned rhythm array is $\{x_i\}$ and $\{y_i\}$, where $i = 1,...,N$, and N is the length of the aligned sequences. The similarity is measured using the following equations.

$$rank\_i = \frac{\sum_{i=0}^{N-1}(q_i - \bar{q}) * (d_i - \bar{d})}{\sqrt{\sum_{i=0}^{N-1}(q_i - \bar{q})^2}\sqrt{\sum_{i=0}^{N-1}(d_i - \bar{d})^2}} \quad (1)$$

$$rank\_r = \prod_{i=0}^{N-1}(1 - \beta \cdot \frac{|x_i - y_i|}{\max(x_i, y_i)}) \quad (2)$$

where β is a weight, and *rank_i* and *rank_r* are pitch interval similarity and rhythm similarity, respectively; $\bar{q}$ and $\bar{d}$ are averages of $\{q_i\}$ and $\{d_i\}$, that is:

$$\bar{q} = \frac{1}{N}\sum_{i=0}^{N-1} q_i, \qquad \bar{d} = \frac{1}{N}\sum_{i=0}^{N-1} d_i \quad (3)$$

3. Combine the result of alignment and similarities, the final rank is obtained as:

$$Rank = (\alpha \cdot rank\_i + (1-\alpha) \cdot rank\_r) \cdot {}^{N_{mis}}\!/_{N} \quad (4)$$

where α is a weight.

In our experiment, we have observed that people hum the pitch variations more correctly than rhythm. That means errors are more likely to involve rhythm than pitch interval. Therefore, we should give a larger weight to *rank_i*, namely, $\alpha > 0.5$. In our system, we set $\alpha = 0.7$.

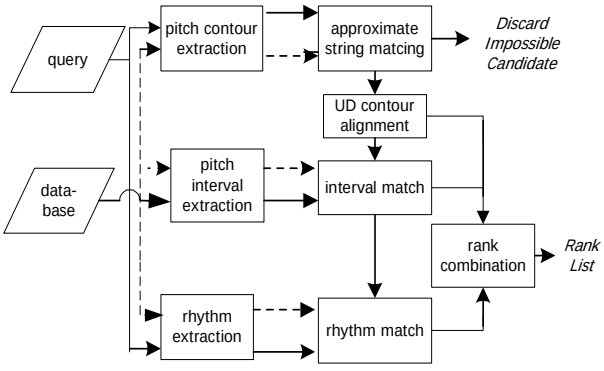

Figure 4. Hierarchical matching algorithm

4. EXPERIMENTS AND DISCUSSION

Our database consists of about one thousand MIDI songs downloaded from the Internet, in which most of them are pop music and the rest are classical and folk songs. Melody information is extracted from each song using the method presented in Section 2, then, divided into several overlapped candidate music segment, and stored together with metadata of these songs, including titles as the test corpus. Each query is performed against each candidate segment of a song and the maximum similarity is regarded as the match score between the query and a song.

42 queries are matched against the test corpus. The queries are hummed by several males and females who have no music background. The hummed segments can be any part of a song. All queries are about 10-second long, 8kHz sampled and mono-channel. For each query, we extract about 8-16 Us or Ds from its pitch contour. It is sufficient in our matching algorithm.

Experiments show most of targeted songs can be retrieved corrected from the databases. More detail results are listed in Figure 5. It indicates how many queries can retrieve the correct song at a given ranking level. From Figure 5, it is shown that, for about 74% queries, the correct song can be listed among the first three matches; and among the first 10 matches for 88% queries. It can be also seen that, for 25 out of 42 queries, their corresponding correct song can be retrieved as the first match.

Our system is also capable of identifying different versions of a song, which have possibly different tempos, slightly modified tunes, and/or different singers. In our experiments, almost all different versions of a song can be retrieved by each query.

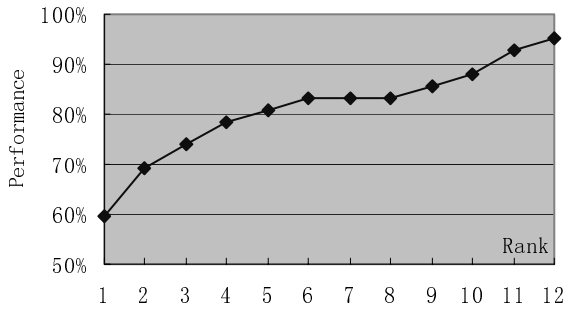

Figure 5. Experiment results on match rank

There are also some shortcomings of the proposed approach, as we observed from the experiments. For example, we use correlation function (1) to measure the similarity between the aligned pitch interval sequences. Because the correlation functions always give a relative higher score than distance or area metrics, it makes the score of some false songs are also high. Thus, the score difference between correct matches and incorrect matches are sometimes too small. Therefore, to find a similarity metrics which can make the matching algorithm both fast and discriminative is a challenging task. Moreover, if we can find a more effective method to correctly segment each note in query, the retrieval performance can be improved significantly. However, as mentioned in Section 2, it is difficult, unless we request the subject to hum 'DaDaDa' or 'LaLaLa' instead of the lyric, or to insert a pause between each note.

5. CONCLUSION

Content-based music retrieval is a very promising method for large music library, yet it is a very challenging task. In this paper, a new triplet melody representation and new hierarchical matching algorithm have been presented. The method to process query and database are also described in detail. The system works effectively and efficiently. For 74% queries, the correct song can be retrieved among the first three matches, and for 88% queries, it is among the first 10 matches.

6. REFERENCES

[1] A. Ghias, *et al*, "Query By Humming—Musical Information Retrieval in an Audio Database". *Proc.s of ACM Multimedia95*, pp231-236, 1995.

[2] R. J. McNab, et al, "Towards the Digital Music Library: Tune Retrieval from Acoustic Input". *Proc. of Digital Libraries*, pp 11-18, 1996.

[3] S. Blackburn and D. DeRoure, "A Tool for Content Based Navigation of Music", *Proc. ACM Multimedia98*, pp 361-368, 1998.

[4] J. T. Foote, "Content-Based Retrieval of Music and Audio." In C.-C. J. Kuo et al., editor, *Multimedia Storage and Archiving System II, Proc. of SPIE*, volume 3229, pp.138-147,1997

[5] C. Francu and C. G. Nevill-Manning. "Distance Metrics and Indexing Strategies for a Digital Library of Popular Music". In *Proc. of IEEE International Conference on Multimedia and Expo.* 2000.

[6] A. L.P. Chen, M. Chang and J. Chen. "Query by Music Segments: An Efficient Approach for Song Retrieval". In *Proc. of IEEE International Conference on Multimedia and Expo.*, 2000

[7] K. Lemstrom, P. Laine and S. Perttu. "Using Relative Interval Slope in Music Information. Retrieval". In *Proc. of International Computer Music Conference 1999(ICMC '99)*, pp. 317-320, 1999.

[8] A. Uitdenbogerd and J. Zobel. "Melodic Matching Techniques for Large Music Database". *Proc. of ACM Multimedia99*, pp57-66, 1999.

[9] D. J. Levitin. "Absolute Memory for Music Pitch: Evidence from the Production of Learned Melodies". In *Perception & Psychophysics, 56(4)*, pp414-423, 1994.