Music Retrieval by Humming

– Using Similarity Retrieval over High Dimensional Feature Vector Space –

Naoko Kosugi*, Yuichi Nishihara, Seiichi Kon'ya, Masashi Yamamuro, and Kazuhiko Kushima

NTT Laboratories, 1-1, Hikarinooka, Yokosuka-shi, Kanagawa, 239-0847, Japan

Abstract

This paper describes a retrieval system that enables a user to hum part of a melody as a query to obtain the name of a desired song from an audio database. The combined use of two musical features, *tone transition* and *tone distribution*, enhance the accuracy of the retrieval system. These features are represented as high dimensional vectors to facilitate the retrieval process. Similarity retrieval using indices of these feature vectors increases the speed of the system.

This paper presents the techniques utilized in the system. Experimental results are also reported based on hummed tunes of casual singers.

1 Introduction

The amount of multimedia data such as text, image, audio, and video from all over the world is increasing rapidly everyday. To deal with the overwhelming amount of multimedia data, a fast and efficient retrieval system that can satisfy three basic requirements must be established. This overabundance of data necessitates a database system that can support fast retrieval and accept queries oriented to each data type. A data-type oriented query is a query that is in the appropriate form for that database. For example, a drawing should be used as a query for an image database and a sung tune should be used as a query for an audio database.

We have studied and developed a data-type oriented retrieval system, the *ExSight* system, for an image database [11]. *ExSight* provides not only whole image retrieval, but also partial image retrieval. For example, it can search for images that include apple-like objects when an image of an apple is given as a search key.

Similarity retrieval is the third requirement for a multimedia retrieval system. This is because most people want to retrieve not just the same data as the search key itself, but entries similar to the search key. When searching for apple images using a photo of apples for example, the user often not only desires the same apple image as the search key but also all similar apple images in the image database.

We have also proposed the *HyperMatch* engine [1, 7, 9], a high speed similarity retrieval engine based on distance measurement employing multiple high dimensional vectors for the *ExSight* system. Each vector retains a feature information. Similarity is represented as a weighted combination of the search results from individual feature vector spaces. This actualizes an adaptable and fast retrieval system.

We are currently studying and developing a music retrieval system that accepts a hummed tune as a query to augment the data-type oriented retrieval system of a multimedia database[6]. The difficult aspect of the idea of "music retrieval by humming" is that the user generally wants one specific song from the music retrieval system, but the query key is vague because many people can not exactly reproduce the desired song to use as the key. We proposed to use the tone distribution information as well as the tone transition information for the music retrieval to adapt the uncertainty of a hummed tune as a searching key and to reveal more accurate retrieval. These are proved by the experimental results in the latter section.

The organization of this paper is as follows. Section 2 describes related work. The proposed music retrieval system and the experimental results are presented in Section 3 and Section 4 respectively. The final section presents the concluding remarks and future work.

2 Related Work

Music can be represented by a sequence of notes and this sequence can be converted into a string of letters. This implies that string matching can be applied to music retrieval [3, 5].

It is difficult for lay people to sing a song exactly, especially when singing from memory. Thus, string matching, which allows some errors in the input pattern, is applicable when retrieving music based on a hummed tune. Errors, for example, can be categorized into *insertion*, *deletion*, *replacement*, *fragmentation*, and *consolidation* of notes. The first three errors are common to approximate string matching, while the other two are particular to strings that are converted from hummed tunes [4].

Dynamic programming matching, DP matching, provides approximate string matching [5]. One of the well-known measures of similarity is the *edit distance*. *Edit distance* is the summation of the costs to correct the input pattern to a matched string. The cost can be defined based on what the error is and how the match is obtained. This implies that DP matching employing the edit distance is useful in music retrieval by humming because it allows for a vague search key, such as that required in this proposal, and can provide suitable calculation of the similarity. However, it is not applicable to a very large database, since the complexity of the DP matching is super-linear to the database size [3].

*nao@isl.ntt.co.jp

0-7803-5582-2/99/$10.00 ©1999 IEEE

On the other hand, *Wu et al.* proposed a fast approximate string matching algorithm[10]. It uses simple bit calculation for the matching. However, it cannot handle *fragmentation* and *consolidation* of notes. Moreover, there are restrictions in defining a score[1] for each error. Thus, it is difficult to apply this algorithm to music retrieval, since it cannot provide suitable calculation for match similarity.

3 Music Retrieval System

The music retrieval system by humming accepts a hummed tune as a query, searches for songs that have a section that is similar to the hummed tune, and ranks the songs according to the closeness of the match.

3.1 Database and Retrieval System

In our system, MIDI format songs are used in the database. MIDI music data can be regarded as musical indicator for each channel. Most *Karaoke* music stores the melody data in one channel. This allows the melody to be simply removed from the MIDI music data.

To construct a music database, first the melody data are extracted. Currently, only melodies are used for matching because most people recognize songs based on the melody. The data is then split into overlapped subdata, using a method called the *sliding window method*, to allow users to sing any part of a song when making a query, and to equalize the length of the matching data. Multiple features, defined in Section 3.2, are extracted from each subdatum. The tone distribution feature and the tone transition feature are converted into high dimensional vectors, which we call *feature vectors*. The vectors are indexed individually for each feature.

A hummed tune is recorded through a microphone and converted into the MIDI format using commercial composition software [8]. Then the hummed tune is processed almost the same way as when constructing the music database. The hummed tune is split into humming pieces[2] with the same window size and sliding amount as that of the subdata in the database. We conducted an experiment to find which humming piece should be used as a search key in Section 4.2. The same kinds of features defined in Section 3.2 are extracted from each humming piece. Finally, feature vectors are generated.

The similarity retrieval starts to find vectors that are close to the vectors generated by the hummed tune over each feature vector space. Similarity is calculated for each subdatum by using the weighted linear summation of the distance between the vectors. The shorter the distance of the subdata that a song has, the higher the ranking of the retrieval results. The final retrieval results are presented as a ranked list of songs. Thus, the user can find songs that have portions that are similar to the hummed tune.

Figure 1 simply shows all of the processes mentioned above.

3.2 Feature Vectors

We propose combined usage of multiple feature vectors that are generated from either of two forms of information, *transition* or *distribution*. As mentioned in Section 2, the conventional way of music retrieval by humming is based on matching sequences of tones. *Tone transition* is the best way to characterize a tune because music includes timing information. However, most lay people can not stay in tune and keep in tempo while singing. Therefore, we should eliminate the tone and tempo variations when using a hummed tune as a query. This leads us to recognize musical characteristics in another form, *tone distribution*.

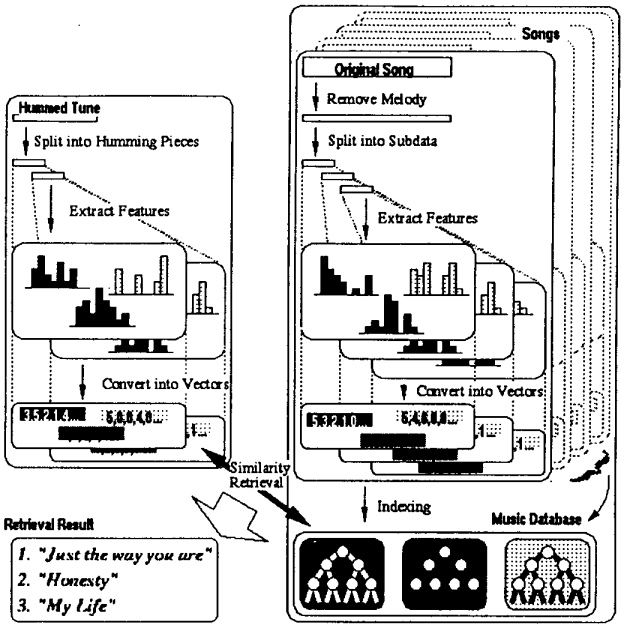

Figure 1: Constructing a music database, processing a hummed tune, and retrieving similar songs.

In the following, the features extracted from subdata and humming pieces, and the method of measuring these features are presented. Then, the method of converting these measurement results into feature vectors is explained.

Extracted features are as follows.

- *Tone transition*
  The time transition of tones.
- *Tone distribution*
  Tone distribution is measured based on three viewpoints. They all are recorded in the form of histograms.
  - *Absolute Tone Based* (*ATB*)
    Tone of each note.
  - *Relative Note Based* (*RTB*)
    Tone difference between successive notes.
  - *First Note Based* (*FTB*)
    Tone difference between a note and the first note of a subdatum and a humming piece.

The bin of histograms represents either the tones or tone differences. The number of bins for *ATB* is 128, ranging from 0 to 127. The number of bins for *RTB* and *FTB* is 255, raging from -127 to 127. This is because MIDI represents tones composing 128 integers.

We use two kinds of measurement units.

- *Time Measured*
  The duration of the tone or the difference in tone is measured.

[1] This corresponds to "cost" of the *edit distance*.
[2] For distinction from subdata.

- *Count Measured*
  The number of times that the tone or a tone difference appears is measured.

Then, we generate fuzzy histograms. As in Figure 2, extra accumulations is added to both sides of the true tone within n semitones. The amount of accumulation would depend on the distance from the true tone. This decreases the effect of variations of tone on a hummed tune.

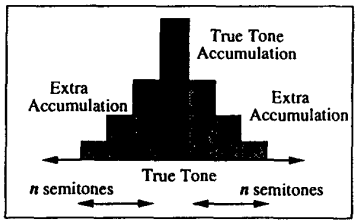

Figure 2: A fuzzy histogram allowing extra accumulation of n semitones on both sides of the true tone.

Converting a histogram into a vector is simple, since a n-bin histogram can be regarded as a n-dimensional vector. The number of bins is decreased in order to increase the matching speed using a shorter vector. Both Figure 3 and Figure 4 represent the shrinkage to a smaller range of "*left*+1+*right*". A mean note $\bar{N}$ is calculated for the center tone of the *ATB* feature vector. This can eliminate the effect of the key difference. The values outside the range are merged for *ATB* (Figure 3) and are neglected for *RTB* and *FTB* (Figure 4).

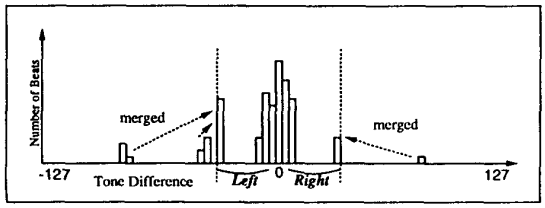

Figure 3: Shrinking the vector dimension. Determining the range of both sides by "*left*" and "*right*", the vector dimension is shrunk to "*left*+1+*right*". The values outside the range are merged.

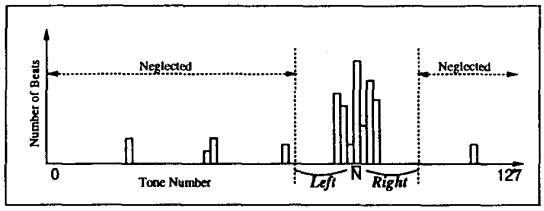

Figure 4: Shrinking the vector dimension. First, the mean note $\bar{N}$ is calculated. After determining the range of both sides by "*left*" and "*right*", the vector dimension is shrunk to "*left*+1+*right*". The values outside the range are neglected.

3.3 System Advantages

(1) Resolving Five Types of Errors

A histogram enables us to grasp roughly the tune characteristics. Thus, four of the five types of errors mentioned previously *insertion*, *deletion*, *fragmentation*, and *consolidation* of notes matter little because they affect only a part of a histogram. *Replacement* of notes can be managed by generating fuzzy histograms. The degree of flexibility of the note replacement can be adjusted by both the range of the extra accumulation and the percentage of each accumulation.

Both the difference and the variation in tempo in a hummed tune are also well managed in our system. All of the songs in the database are normalized using a fundamental tempo. The tempo of a hummed tune is converted into a fundamental tempo using the information of a metronome that accompanied the user. The variation in tempo is managed by the *Count Measured* histograms because the variation is influenced only slightly by the appearance of each tone.

(2) Fast Retrieval

Vectors can be indexed. The time complexity for retrieval with indices is expected to be less than linear for the database size. Thus, indexing is effective for retrieval from a large database.

(3) Flexible Similarity Retrieval

Similarity is a combination of the search results from individual feature vector spaces in our system. The search results are represented as the Euclidean distance between vectors. The combination is realized as a weighted linear summation as follows.

$$S = \sum_{i=1}^{n} w_i d_i \quad i : integer \tag{1}$$

where S represents the *similarity*. The parameters, n, w_i, and d_i, represent *the number of features*, the *weight* for the ith feature, and *distance* to the ith feature vector respectively. This provides flexible similarity retrieval, since the weights can be adjusted.

4 Experimental Results

This section describes two sets of experimental results of hummed tunes. One is from finding the best portion of a hummed tune as a query. The other is used to prove the advantage of the combined usage of the features, *tone transition* and *tone distribution*.

4.1 Parameters and Settings

There are 1,200 MIDI format songs in the database. Many genres are included in the database such as popular, and, Japanese traditional songs.

The window size and the sliding amount of subdata and humming pieces are 20 and 4 beats respectively.

Tone transition is measured based on the beat. The highest tone with the longest duration within a beat is recorded.

The range and the amount of the extra accumulation to generate a fuzzy histogram is set to 1 and 50%. This means that the half value of the true tone is accumulated on both sides of the true tone in the range of 1 semitone.

The *left* and *right* ranges for vector dimension shrinkage are 12, because there are 12 semitones in an octave.

Time Measured histograms are generated for *ATB*, *RTB*, and *FTB*. *Count Measured* histograms are generated for *RTB* and *FTB*.

A user is required to sing a melody through a microphone. A metronome is used in order to track the tempo when singing.

The music retrieving experiment was done with 18 people (male: 14, female: 4). They sang 77 songs. The average number of humming pieces was 3.12 per song. This means that they sang about 28.5 beats per song on average.

4.2 Best Portion of a Hummed Tune

As the default, only the middle piece of a hummed tune is used as the search key. Table 1 shows both the number of songs in which the top ranked song matched and those within the top three that matched under each of the conditions bellow.

Case1) Only the middle humming piece is used as the search key.

Case2) Multiple retrievals are done using each of the humming pieces. The result with the highest rank is selected.

Table 1: Difference in Retrieval Results Depends on Part of Hummed Tune Used as Query.

	Top rank matches	Matches within the top three
Case 1	32 (42%)	38 (49%)
Case 2	47 (61%)	53 (69%)

The results (32/47 matches) show that most of the people in the test group can sing more accurately in the middle of a test segment. However, it can be seen that some people sing more accurately at the beginning or ending of a test segment (15/47 matches). This means that the selection of the humming piece as the search key must be adjusted for each singer.

4.3 Combination of Tone Transition and Tone Distribution

The advantage of the combination use of the *tone transition* and the *tone distribution* is detailed in this subsection. Table 2 shows both the number of top ranked matches and the number of matches within the top three under each of the conditions bellow. In this experiment, the same as in Case2) above, the humming piece that had the highest match is selected as the search key.

Case3) Only the *tone transition* is considered in the retrieval.

Case4) Only the *tone distribution* is considered in the retrieval.

Case5) Both the *tone transition* and *tone distribution* are considered in the retrieval. Furthermore, the weights used in combining the search results of each feature are adjusted to obtain the best match.

Table 2: The advantage of the combination use of the *tone transition* and the *tone distribution*.

	Top rank matches	Matches within the top three
Case 3	37 (48%)	41 (53%)
Case 4	40 (52%)	46 (60%)
Case 2	47 (61%)	53 (69%)
Case 5	64 (83%)	66 (86%)

The results of Case 3 and Case 4 are lower than that of Case 2. This shows that the combined use of *tone transition* and *tone distribution* is effective for music retrieval with humming. Furthermore, Case 5 shows a significantly high percentage of matching. This means that a more accurate retrieval system can be actualized by sufficiently considering each feature.

5 Concluding Remarks and Future Work

This paper described a music retrieval system that accepts a hummed tune as a query. The proposed retrieval system splits the original music data into overlapped subdata to enable users to sing any part of a song when retrieving music. The system uses both *tone transition* and *tone distribution* to enhance the accuracy of the music retrieval. The tone distribution information is recorded in the form of histograms and copes with variation in tone and tempo of hummed tunes. Furthermore, the tone distribution incorporates flexiblity in the histograms by adding extra accumulation to both sides of the true tone to deal with variation in the tone of a hummed tune. The system is capable of searching over high dimensional feature indices to achieve efficient retrieval from a large database and is capable of flexible similarity retrieval by adjusting the weights when summing the search results from each feature vector space.

Further experiments are being conducted to verify the performance of the retrieval system. Among the various issues that we face are enlarging the music database, shortening the split subdata, investigating more features and adequate weights for feature vectors when similarity is calculated, and improving the resolution of the retrieval system.

References

[1] K. Curtis, N. Taniguchi, J. Nakagawa, and M. Yamamuro. A comprehensive image similarity retrieval system that utilizes multiple feature vectors in high demensional space. In *Proceedings of International Conference on Information, Communication and Signal Processing*, pages 180–184, September 1997.

[2] Jonathan Foote. An overview of audio information retrieval. In *Multimedia Systems 7*, pages 2–10. ACM, January 1999.

[3] Asif Ghias, Jonathan Logan, and David Chamberlin. Query By Humming. In *Proc. ACM Multimedia 95*, pages 231–236, November 1995.

[4] Rodger McNab. INTERACTIVE APPLICATIONS OF MUSIC TRANSCRIPTION. Master's thesis, Computer Science at the University of Waikato, 1996.

[5] Roger J. McNab, Lloyd A. Smith, David Bainbridge, and Ian H. Witten. The New Zealand Digital Library MELody inDEX. http://www.dlib.org/dlib/may97/meldex/05written.html, May 1997.

[6] Yuichi Nishihara, Naoko Kosugi, Seiichi Kon'ya, and Masashi Yamamuro. Humming Query System Using Normalized Time Scale. In *Proceedings of CODAS'99*, March 1999.

[7] Makoto Onizuka and Satoshi Okada. Query Model for Structured Objects. In Tharam S. Dillon, Robert Meersman, and Zahir Tari, editors, *Short Paper Proceedings*, pages 32–45. International Conference on Database Semantics, January 1999.

[8] WILDCAT CANYON SOFTWARE. AUTOSCORE. http://www.wildcat.com/Pages/AutoscoreMain.htm.

[9] Noburou Taniguchi and Masashi Yamamuro. Multiple Inverted Array Structure for Similar Image Retrieval. In *IEEE Multimedia '98*, pages 160–169, 1998.

[10] Sun Wu and Udi Manber. Fast Text Searching Allowing Errors. *Communications of the ACM*, 35(10):83–91, October 1996.

[11] M. Yamamuro, K. Kushima, H. Kimoto, H. Akama, S. Kon'ya, and J. Nakagawa. ExSight – Multimedia Information Retrieval System. In *20th Annual Pacific Telecommunications Conference, PTC'98 Proceedings*, pages 734–739, 1998.