# SOUNDCOMPASS™

## *- A FAST QUERY-BY-HUMMING SYSTEM USING MULTI-DIMENSIONAL FEATURE VECTORS*

HIDENOBU NAGATA†, NAOKO KOSUGI†, RYOJI KATAOKA†, and TAKASHI HONISHI†
†*NTT Cyber Space Laboratories, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa, 239-0847, Japan*

**Abstract**: Retrieval using sound information as queries for a music database is intuitive and very useful. However, it is difficult to use hummed tunes as queries because they are often unclear. Thus, the SoundCompass™ music retrieval system employs a similarity retrieval technique to overcome this problem. The retrieval result is a ranked list of songs that are similar to the hummed tune. The most significant ways in which our system are superior to other query-by-humming systems are that 1) musical data is processed based on "beats" instead of "notes", and 2) the retrieval is done through the use of an index based on multi-dimensional feature vectors. These features allow the system to retrieve songs quickly and precisely even if erroneously hummed tunes are used as queries. The database currently holds over 10,000 songs, and the retrieval time is about one second. The system is able to recognize a song and rank it within the first five places on the retrieval list for about 70% of hummed tunes that are recognizable to human subjects as being a part of the song.

**Key words**: similarity retrieval, music database, query-by-humming

## 1. Introduction

The use of multimedia data has spread throughout the world with the availability of high-performance, low-cost personal computers, and this has led to a need for accurate, efficient retrieval methods for large multimedia databases. General retrieval systems accept only key words as queries. However, many users experience difficulty in formulating a query in words when they want to retrieve something from a multimedia database. Content-based retrieval is seen as a solution to this problem [1~3]. Retrieval using sound information as queries for a music database is intuitive and very useful. However, it is impossible to use hummed tunes as queries for an exact matching because hummed tunes may contain errors. Thus, similarity retrieval is useful when hummed tunes are used as queries.

Speed of query processing is an important issue for similarity retrieval, because such a retrieval needs a complex similarity calculation based on matching scores. The authors have developed the HyperMatch [4] engine, a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35660-0_65
R. Nakatsu et al. (eds.), *Entertainment Computing*

multi-dimensional feature vectors. Defining effective feature vectors that can reduce the influence of errors in hummed tunes is important when applying HyperMatch to SoundCompass™. In this paper, we describe feature vectors for the SoundCompass™ system using the HyperMatch engine and determine the most relevant parameters for feature vector extraction.

## 2. SoundCompass™

## 2.1 Database Construction

The SoundCompass™ database currently stores 10,069 songs* in MIDI format. The songs include many musical genres. Some are short, simple songs such as nursery songs and folk songs, and others are longer, more complex songs from such as pop genres and rock. Currently, only melodies are used for matching because most people remember a song by its melody. All the segments in the melody data in which multiple notes overlap are modified to have only a single note. The melody data is then chopped into melody pieces of constant length by using the sliding window method [5]. The pieces are defined as subdata. Each subdatum is given redundancy with respect to its predecessor and successor by letting the length of the slide be shorter than the length of the window (Fig. 1). Chopping the melody data into overlapping subdata allows a user to select whichever part of a song he or she wishes to hum. Feature information is extracted from each subdatum and the extracted features are converted into multi-dimensional feature vectors. The features are described in detail in Section 3. These feature vectors are loaded into the system's memory and indexed in the database's server.

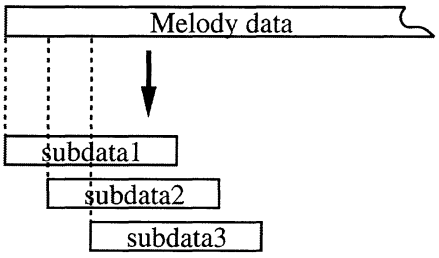


*Figure 1.*Melody data splitting with the sliding-window method.

*Presented by Daiichi Kosho Co., Ltd.

## 1.2 Hummed Tune Processing

The process of generating a query key from a hummed tune is similar to the database construction process described in Section 2.1. The hummed tune is recorded through a microphone and converted into MIDI format by commercial composition software [6]. The user must clearly hum the song notes using only the syllable "ta". This is done so that the hummed tune can be transcribed as accurately as possible. The user also must hum following the beats of a metronome. This is done to enable people to hum in constant tempo. Moreover, the server program must have the tempo information that the singer followed because it uses beat-based processing. Of course, the user may adjust the speed of the metronome to the desired tempo. Next, the hummed tune is chopped up into hummed pieces of the same window length and sliding length as those of the subdata. Finally, the same kind of features as that extracted from subdata is extracted from each hummed piece. These are also converted to feature vectors and used for queries.

## 1.3 Similarity Measurement, Similarity Retrieval

SoundCompass™ finds vectors that are close to the vectors generated by humming the tune. While more than one hummed piece is generated if the hummed tune is longer than the length of the window, only one hummed piece generated from the middle of the hummed tune is used as the query, because it is most stable. The City-Block distance is used for the distance measurement. The shorter the distance to the subdata is, the higher the ranking of the retrieval result becomes.

# 3. Feature Vectors

## 3.1 Pitch Transition Feature Vectors

To represent features of a melody, we use a feature vector that represents a time-wise pitch transition relative to a certain tone. Pitch difference is defined as the difference in pitch between two notes, and the minimum difference is a halftone. For example, a tone that is a half tone higher/lower than the other is described as +1/-1, respectively. The feature representing the time-wise pitch transition is called the pitch transition feature and the vector based on the feature is called the pitch transition feature vector [7]. The vector is represented as a sequence of tone numbers. Each tone number indicates the dominant pitch in each successive constant beat (we call it "beat resolution"). Thus, a short note that is incorrectly transcribed (due to variation in tone) does not affect the generation of the feature vectors. A tone is considered to be representative if it is the longest tone in the resolution

beat. Figure 2 shows how to make the pitch transition feature vector from a tune segment with the beat resolution of an eighth-note (8th-note). E4, F4, and G4 are MIDI codes, and the numbers to their right are MIDI note numbers. To solve the problem of key difference between melody data and hummed tune, each vector component is represented with the pitch difference from a certain tone (we call it a "base tone"). For example, if the most common tone in the tune in Fig. 2 is selected as the base tone (F4(65)) and let it be "0", a feature vector of eight dimensions (-1, -1, 0, 0, 2, 2, 0, 0) is generated.

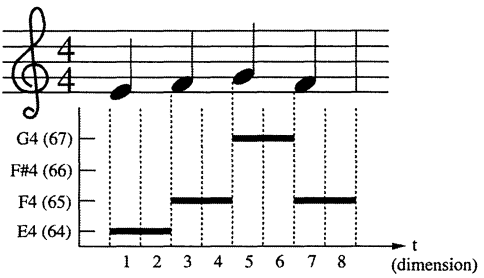


*Figure 2.*Generating a pitch transition feature vector.

## 3.2 Beat Resolution for Pitch Transition Feature Vector

To find an effective beat resolution of the pitch transition feature vector, we analyzed the distribution of note lengths in all 10,069 songs. Figure 3 shows the result. X-axis and y-axis represent tick time and number of notes, respectively. The length of a quarter-note is 480 tick times, thus, eighth-notes (240 tick times) are dominant in all the 10,069 songs. Based on this result, the most appropriate beat resolution can be evaluated quantitatively (Section 4.3).

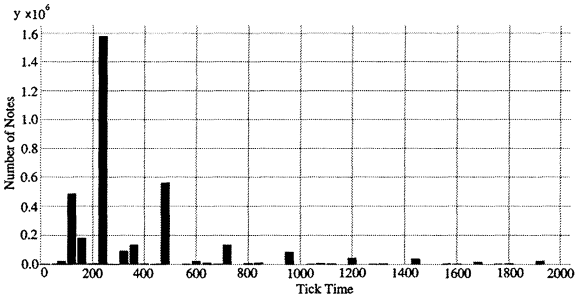


*Figure 3.*Note distribution of all songs.

# 4. Evaluations and Discussion

We optimised the window length of Section 2.1 and the beat resolution for the pitch transition feature vector based on retrieval precision. Retrieval speed was also evaluated.

## 4.1 System Parameters

The 10,069-song database was used in the experiments for the precision evaluation and the performance evaluation. A total of 258 tunes were hummed by 25 people (21 males and 4 females). Of the 258 tunes, we selected 186 tunes that were recognizable as being a part of a particular melody after they had been transcribed into MIDI format (these tunes were also parts of songs in the song database). A SUN Ultra80 with quad 450-MHz UltraSPARC-II's and 4-GB main memory was used for the evaluation. Figure 4 shows the set up of the experimental system. A database server and a client PC were connected through a network. The query-by-humming system server (SoundCompass Server), a master program that handles queries, and a database server program worked together in the server machine. The database server held an index of feature vectors. A microphone was connected to the client PC, which included a GUI and the transcription software. The hummed tune, input through the microphone, was transcribed and converted into MIDI format and sent to the SoundCompass Server. The server processed the hummed tune according to the method described in Section 2.2 and sent a query to the database server. Then, the database server returned the retrieval result to the SoundCompass Server, which sent the result to the client PC.

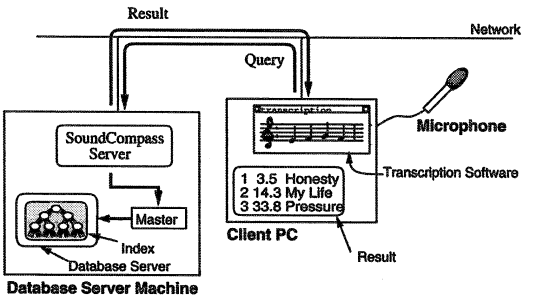


*Figure 4.*Experimental system setup.

## 4.2 Optimization Window Length

To investigate the effect of changing the window length on retrieval precision, we compared the precision of the retrieval results generated with

the window lengths of 8, 12, and 16 beats. The sliding length in this experiment was 4 beats and the pitch transition feature vectors had a beat resolution of an 8th-note. Figure 5 shows the percentage of times in which a correct song name appeared within the ranking. For example, in 66% of the hummed tunes, the correct answer was retrieved within the second rank when 16 beats was used as the window length. This figure reveals that the longer the window length is, the higher the precision becomes. However, the requirement humming for longer than 16 beats received unfavorable criticism from all trial subjects. Based on these results, all the music data in the database and hummed tunes were chopped, into 16-beat window lengths and 4-beat sliding lengths.

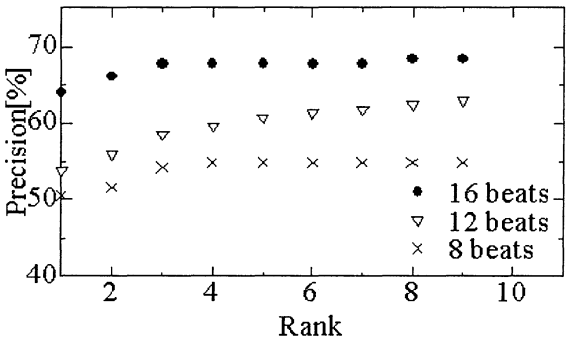


*Figure 5*.Precision for window length of 8,12,and 16 beats.

## 4.3 Optimization Beat Resolution

The beat resolution is equivalent to the unit of a pitch transition vector component. Setting a fine beat resolution easily obtains the correct answer, because vectors accurately represent short notes, but tolerates only a few errors in the hummed tune. Conversely, setting a coarse beat resolution gives a wider range of results but allows for more errors in the hummed tune. As shown in Section 3.2, the 8th-note is the most frequently occurring note. To determine the effective beat resolution for pitch transition feature vectors, the precisions of the retrieval results with the beat resolution of a quarter-note, 8th-note, and 16th-note were compared. Figure 6 shows the results. The figure reveals that the 8th-note is clearly better than a quarter-note as the beat resolution for the pitch transition feature vector. Also, the difference in precision between the 8th-note and 16th-note is relatively small, and in both cases, the correct answer was retrieved within the 5th rank with a precision of 70%. The finer the beat resolution is, the more the dimensions of the feature vectors become, and that means the database size become bigger and the similarity calculation becomes more complex. Thus, the 8th-note is the appropriate beat resolution for the pitch transition feature vectors.

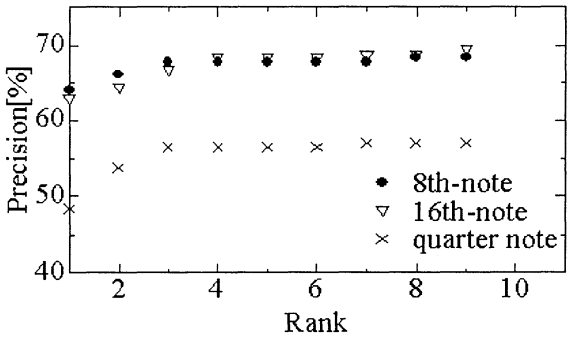


*Figure 6.*Precision where the beat-resolution for pitch transition feature vectors are aquarter-note, 8th-note, and16th-note.

## 4.4 Performance Evaluation

This section describes the effectiveness of using an index. Retrieval time was measured when an index is used and when brute force searching is used. Figure 7 shows the execution time. The solid line represents the retrieval time when an index is used, and the dotted line, brute force searching. The x-axis represents the number of songs stored in each database and y-axis represents retrieval time. Songs that made up the subsets of the database for this experiment were chosen randomly. Retrieval time was measured as the duration from when a query is sent to the database server to when the retrieval result is received. This figure reveals that the greater the size of the database is, the higher the efficiency of the index becomes. Moreover, the database server with an index can provide a retrieval result within one second for the database with over 10,000 songs.

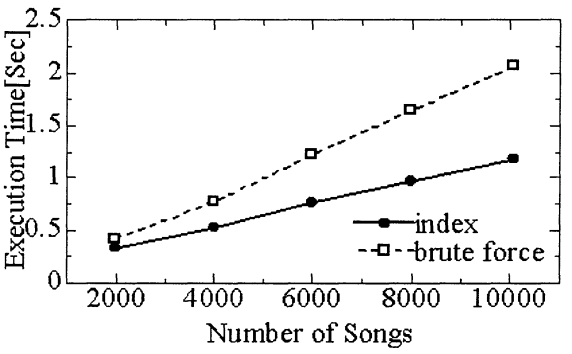


*Figure 7.*Retrieval time evaluation where an index is/is not used.

## 5. Conclusions

We studied both feature vectors and parameters for feature extraction with SoundCompass™. SoundCompass™ 's database currently holds 10,069 songs, and the database server is able to retrieve songs from the database within one second. SoundCompass™ allows a user to select whichever part of a song he or she wishes to hum. Moreover, it retrieves the correct song within the 5th rank for 70% of the recognizable hummed tunes. To achieve these results, we used beat-based processing for music data and a high-speed similarity retrieval technique with multi-dimensional feature vectors. The user can retrieve a song by humming whichever part of a song he/she wishes to hum because the melody data is chopped into smaller pieces by using the sliding window method and each piece of subdata is registered in the database. Based on the distribution of the note length for all the songs, we determined that the pitch transition feature vector generated by an 8th-note beat resolution was most effective. Future investigations will focus on the use of new feature vectors and the evaluation of various combinations and weightings of vectors in the similarity calculation.

## References

[1]Ma, W. Y., Manjunath, B. S., Luo, Y., Deng, Y. and Sun, X.: NETRA: A Content-Based Image Retrieval System.

[2]Smith, J. R. and Li, C. S.: Image classification and querying using composite region templates, Journal of Computer Vision and Image Understanding (1999).

[3]Ghias, A., Logan, J. and Chamberlin, D.: Query By Humming, Proc. ACM Multimedia 95, pp. 231-236 (1995).

[4]Curtis, K., Taniguchi, N., Nakagawa, J. and Yamamuro, M.: A comprehensive image similarity retrieval system that utilizes multiple feature vectors in high dimensional space, Proceedings of International Conference on Information, Communication and Signal Processing, pp. 180-184 (1997).

[5]Kosugi, N., Nishihara, Y., Kon'ya, S., Yamamuro, M. and Kushima, K.: Music Retrieval by Humming. Proceedings of PACRIM'99, pp. 404-407 (1999).

[6]Wildcat Canyon Software: AUTOSCORE, http://www.wildcat.com/Site/Homepage.htm.

[7]Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M. and Kushima, K.: A Practical Query-By-Humming System for a Large Music Database, Proc. of the 8th ACM International Conference on Multimedia, pp. 333-342 (2000).