A Novel Approach Based on Fault Tolerance and Recursive Segmentation to Query by Humming

Xiaohong Yang, Qingcai Chen, Xiaolong Wang
Department of Computer Science and Technology
Key Laboratory of Network Oriented Intelligent Computation
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
{yxh2008, qingcai.chen}@gmail.com, wangxl@insun.hit.edu.cn

Abstract

With the explosive growth of digital music, content-based music information retrieval especially query by humming/singing have been attracting more and more attention and are becoming popular research topics over the past decade. Although query by humming/singing can provide natural and intuitive way to search music, retrieval system still confronts many issues such as key modulation, tempo change, note insertion, deletion or substitution which are caused by users and query transcription respectively. In this paper, we propose a novel approach based on fault tolerance and recursive segmentation to solve above problems. Music melodies in database are represented with specified manner and indexed using inverted index method. Query melody is segmented into phrases recursively with musical dictionary firstly. Then improved edit distance, pitch deviation and overall bias are employed to measure the similarity between phrases and indexed entries. Experimental results reveal that proposed approach can achieve high recall for music retrieval.

Keywords: *Query by humming, melody partition, edit distance, fault tolerance, recursive segmentation.*

1. Introduction

As is well known, at present the prevalent music search engines like Google [1], Yahoo [2] primarily take textual metadata such as song title, singer name, composer, album or lyrics as keywords to retrieve music, and then return a list of link address of related music to users. However, the current text-based music retrieval approaches may be not convenient to users when they forget the metadata associated with desired music. Therefore alternative music retrieval methods are necessary when people wish to search music and only remember some short parts of music melody.

Content-based music information retrieval (MIR) can be regarded as a good complementary music searching method which is more natural and intuitive to users. Hence content-based MIR has been attracting widespread attention and becoming an important research topic with the explosive expansion of digital music on the World Wide Web (WWW) over the past decades. Content-based MIR systems [3-4] usually utilize extracted features such as pitch, duration, rhythm, contour, chord etc. to represent music melody and retrieve music from large scale music database.

Query by humming (QBH) is one of the most natural content-based MIR methods, which takes a fragment of melody hummed, singed or whistled by users via microphone as query to search music [5-7]. QBH system mainly contains following components: i) automatic query transcription module which transforms the acoustic input into note sequence or melody

contour; ii) melody representation and indexing module for music in database; iii) searching and matching module that retrieves related melodies from database, carries out similarity measurement between query and indexed melodies, then return a ranked list of candidates to users. The framework of proposed QBH system is illustrated in Figure 1 where component i is belong to the front-end (FE) module and the back-end (BE) module includes components ii and iii.

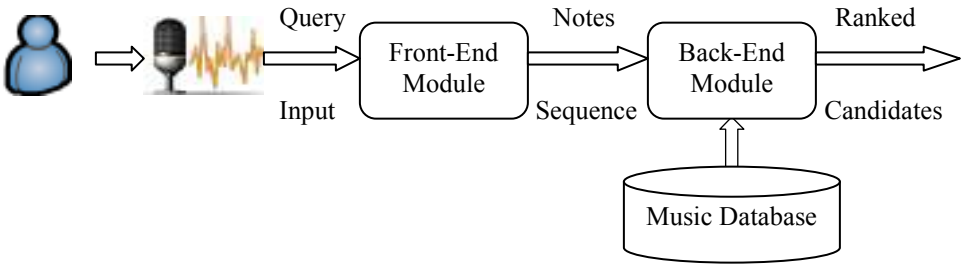

Figure 1. The framework of proposed QBH system

Although QBH method could provide an intuitive way to search music in terms of a short piece of melody remembered in users' mind, in practical implementation there are many issues that require better techniques to resolve. In front-end module the quality of queries and the accuracy of automatic query transcription affect the final performance of QBH system [8]. Due to the wide variety of musical backgrounds, regardless of musical training or no training, people always cannot sing a song with the same key and rhythm as the original music. Consequently users usually produce singing errors such as key modulation, variable rhythm, wrong notes, notes insertion or omission when singing or humming pieces of songs as input queries without accompaniment. Even with perfect queries, it is still difficult for automatic query transcription module to extract accurate pitches. Most of errors appear in pitch extraction are caused by capturing double or half pitches, which results in notes are split into several notes or unified as one note [9]. To sum up above argumentations, the searching and matching methods used in back-end module must be robust to deal with the singing errors and note segmentation errors caused by the front-end module.

This work proposes a novel approach based on fault tolerance and recursive segmentation to handle aforementioned problems. Query transcription module tracks the fundamental frequency F0 of input query and converts it into note sequence. We just use the pitch information, and represent the query and music melodies in database with melody strings. Like Chinese words segmentation the query is partitioned into several phrases in terms of a musical dictionary composed of repeating patterns which denote motives or music themes. Music melodies in database are segmented by the same way and indexed. When matching the query with indexing melodies, edit distance with fault tolerant mechanism is employed to measure similarity. Because of imperfect recall about desired music, pitches always deviate from true values. Therefore when calculating the edit distance of two phrases, if the absolute value of pitch difference between two notes is smaller than or equal to predefined threshold, the proposed approach deems the two notes to be identical. Since notes insertion or omission in query leads to inaccurate partition, we employ recursive segmentation to solve this problem and search music.

The remainder of the paper is organized as follows. Section 2 portrays melody representation and partition with musical dictionary. Section 3 introduces query transcription firstly, and then describes proposed approach based on fault tolerance and recursive segmentation. Experimental results are reported in section 4. The conclusions of this work are summarized in Section 5.

2. Index construction

So far, content-based MIR systems primarily employ note-based or symbol-based manner to represent melody. In former systems music melodies are directly extracted from digital music stored in MP3, MIDI, wav or scores formats, and transformed into notes sequences by extracting pitch and duration features. The latter utilize relative-value based melody representation to reduce the difference between hummed query and melodies in database [3]. Relative pitch can be obtained by subtracting previous pitch from current pitch for each note. Relative duration can be calculated by dividing the continuous note by duration.

With the explosive growth of music information, it is essential to construct index for large scale music collections to compact music storage space and support quick search. Inspired by text information retrieval, we consider music as documents and construct inverted index for music melodies with musical dictionary using repeating patterns as lexical items.

Table 1. An example of melody representation

SongId	Melody String
1029	r'r'ubr'uuteuttetuterBrr'r'ubr'uuteuttetuterBr
1030	FcccVAcVAGFCVAGAAcVAGFcccVAcVAGFCVAGAAcVAG
1031	CCB,A,G,G,G,A,B,CE,E,F,G,A,CCB,A,G,G,G,A,B,CE,E,F,G,A,

2.1. Melody representation

This work integrates the note-based and symbol-based methods together to represent melody, and employs melody strings to represent transcribed query and music melodies in database. When measuring similarity between query and music melodies, relative pitch is used to calculate edit distance. In addition, this work follows the format of ABC music notation to describe pitch name in different octave. In order to represent the twelve semitones in every octave better and easier, some letters such as R, S, T, U and V are introduced into pitch name mechanism.

Therefore according to the format of ABC music notation, semitones in different octave can be defined as follows. Twelve uppercase letters (C R D S E F T G U A V B) are adopted to denote the semitones in central octave, and one comma is appended on bottom-right corner of each letter for those below central octave. More commas are added for lower octave halftones by analogy. Similarly lowercase alphabets (c r d s e f t g u a v b) are used to represent the semitones above central octave, and more commas are appended on the top-right corner of each alphabet for higher octave semitones. Table 1 shows an example of melody representation for short pieces of some songs. Music melodies are extracted from digital music in database firstly, then parsed and presented with above format.

2.2. Musical dictionary

Most of music works are composed in terms of musical form which contains some rules such as repetition, comparison, transposition and extension for developing musical themes [10]. In particular the repetition rule means that existing specific notes sequences, known as motives, appear repeatedly in movements [11]. As showed in Table 1, notes sequences like "F-c-c-c-V-A" and "c-V-A-G" appear repeatedly in melody string of SongId 1030. Any sequence of notes appearing more than once in music melody is considered as repeating pattern. Therefore, repeating patterns usually stand for motives or musical theme, and can be used for theme mining.

Based on the ground-truth MIDI files used for MIREX 2008 Query-by-Singing/Humming evaluation task and the Essen Folk Song Database [12] which is a collection of more than 8400 European and Chinese folk songs written in ABC music notation, all melodies are parsed with specified format depicted in previous subsection firstly, and then the frequencies of distinct repeating patterns are calculated. If the statistical frequency of each distinct repeating pattern satisfies the predefined threshold, the repeating pattern is selected as candidates of lexical item.

In addition, statistical results about the frequencies of repeating patterns show that the frequency of occurrence decreases as the length of repeating patterns increases. Therefore it is not sensible to select too long repeating patterns as lexical items. The same is true for much shorter repeating patterns since their semantic meanings are ambiguous and like stopwords they appear too frequent among music melodies to be good discriminators for music works. If the length of candidate belongs to the preset range, it is chose as final words to comprise musical dictionary with hash table structure.

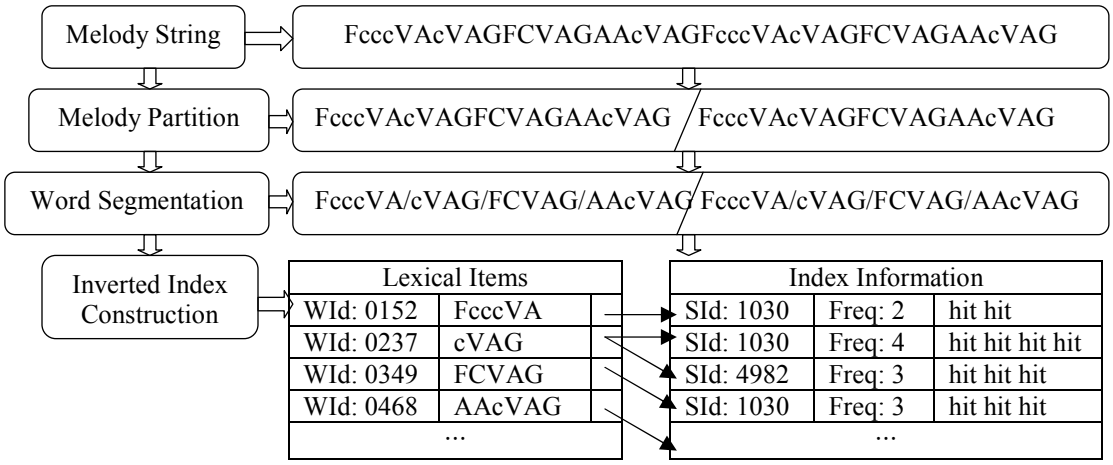

Figure 2. An example of melody partition and inverted index construction

2.3. Melody partition

Analogous to Chinese information retrieval, transcribed queries and music melodies without obvious separator should be partitioned into phrases with musical dictionary before retrieval. For each melody in database, optimal longest repeating string (OLRS) is extracted firstly. The melody is partitioned into several passages in terms of OLRS by recursion, which will not terminate until no OLRS or the length of OLRS is smaller than predefined condition.

Then word segmentation, in other words repeating pattern partition, for every passage is carried out with musical dictionary using backward maximum matching method.

An inverted index is constructed for all words in order to speed the music searching and matching. The inverted index structure consists of two elements: the lexicon that is a set of all different words appearing in music melodies and the hits list which corresponds to a list of occurrences of a word in a music melody including position, frequency and other necessary information. Figure 2 depicts an example of melody partition and inverted index construction for melody string of SongId 1030 in Table 1.

3. Proposed QBH approach

In this section we introduce query transcription, similarity measurement firstly, and then describe proposed approach based on fault tolerance and recursive segmentation to music retrieval.

3.1. Query transcription

Illustrated as Figure 1, the queries hummed or sung by users via microphone are acoustic signals. The front-end module of QBH systems should convert the queries into notes sequence at first [13]. For this task, the automatic transcription component in FE module adopts the YIN algorithm based on the well-known autocorrelation method to estimate the fundamental frequencies of query signals [14].

Firstly we estimate fundamental frequency for each frame using the YIN algorithm in 25ms analysis window with 5ms interval between successive frames. If the difference of fundamental frequencies between continuous frames is small enough and satisfies the preset threshold, these continuous frames are integrated and the average frequency is regarded as their fundamental frequency. Then according to MIDI standard format all frequencies are transformed into semitones using logarithmic function. One step of the frequency corresponds to one semitone that is calculated as following equation:

$$Semitone = 69 + 12 / \log 2 \times \log(f_0 / 440) \quad (1)$$

where f_0 indicates the fundamental frequency.

Even though the YIN algorithm has high precision to detect fundamental frequency, it is unavoidable for users or query transcription to bring about errors. Therefore post processing is necessary. After getting the average of all semitones in a query, if the difference between a semitone and the average is equal to or greater than one octave, the semitone is set to 0.

3.2. Similarity measurement

Edit distance (ED) is a common distance metric to measure the similarity between two symbol sequences. On one hand our QBH system employs melody strings to represent the input query and database entries, on the other hand in order to compensate user singing or humming errors, relative pitch can be better to portray the coarse melody contour. Therefore the ED method is suitable to calculate similarity between transcribed query and indexed melodies with relative pitch.

The edit distance between two compared sequences can be defined as the minimal transitional number of note insertion, deletion and substitution which are necessary operations

to transform source sequence into target sequence. By selecting appropriate cost function, the ED can take user errors into consideration. Insertion cost covers extra hummed notes, while the deletion cost accounts for skipped notes [9]. The substitution cost penalizes errors between wrong notes and the expected reference notes.

The ED approach implements dynamic programming principle to calculate melody distance $D(Q,P)$ between the transcribed query $Q = q_1q_2...q_M$ and indexed entry $P = p_1p_2...p_N$ by completing a $(M+1) \times (N+1)$ distance matrix $D_{(M+1) \times (N+1)}$. The cell $D_{i,j}$ denotes the minimal melody distance between the two prefix sequences $Q_1...i$ and $P_1...j$. Let $1 \le i \le M$ and $1 \le j \le N$, the value of each cell in matrix $D_{(M+1)\times(N+1)}$ can be calculated by using the following recursive formula:

$$D_{i,j} = \min\{D_{i-1,j} + 1, D_{i,j-1} + 1, D_{i-1,j-1} + Cost_{i,j}\} \tag{2}$$

$$Cost_{i,j} = \begin{cases} 0 & if\ Q_i = P_j \\ 1 & if\ Q_i \neq P_j \end{cases} \tag{3}$$

with initial conditions:

$$D_{0,0} = 0$$
$$D_{i=1:M,0} = 1,2,3,\cdots,M,$$
$$D_{0,j=1:N} = 1,2,3,\cdots,N.$$

The above recursive formula defines a constant penalty of 1 for note insertion and deletion, the cost for note substitution is 0 if the i-th note of Q is equal to the j-th note of P, otherwise 1. The final edit distance between two sequences Q and P is the value of last cell $D_{M,N}$ in distance matrix.

3.3. Fault tolerance mechanism

On account of imperfect memory or without professional knowledge about music, people are usually unable to grasp the music key and tempo when humming or singing. This means errors like key modulation and tempo change caused by users are inevitable. In order to alleviate the impact made by above faults and other flaws such as notes insertion, deletion and substitution on final retrieval performance, it is indispensable for QBH system to apply fault tolerant approaches to promote the robustness of retrieval system.

3.3.1. Pitch deviation: Even if music is hummed faster or slower than correct version or sung in higher or lower key, they also can be recognized easily by audiences. This phenomenon indicates that music is time-scalable and tone-shiftable [15]. Since relative pitch always employed to portray melody contour is insensitive to music key, it is better than absolute pitch to describe the music melody. In the same way relative duration in other words duration ratio is used for time comparison due to its insensitivity to music tempo. Consequently in this work relative pitch is used to calculate the distance between transcribed query and indexed entries.

After transcribed query and music melody represented by melody string are partitioned into phrases, the relative pitch of each phrase is calculated by subtracting the

previous pitch from current pitch. An example of relative pitches of phrases is showed in Table 2. According to the analysis about relative pitches of phrases, we find that the absolute value of difference between two relative values among two phrases which are deemed to be same is usually smaller than or equal to one halftone. Like phrases "AcVAU" and "AAcVAG" in Table 2, we can consider these two phrases are same even though the latter inserts a zero and substitutes -1 with -2, since the pitch difference is just 1. As known users always lead to pitch deviation, but the difference satisfying predefined threshold is permitted. In proposed approach the threshold of admitted pitch deviation is set to 1, namely one halftone. Therefore we adjust the cost function as below formula when calculate the edit distance between two phrases:

$$Cost_{i,j} = \begin{cases} 0 & if \ \left|Q_i - P_j\right| \le 1 \\ 1 & if \ \left|Q_i - P_j\right| > 1 \end{cases} \quad (4)$$

where Q_i indicates the i-th value of relative pitch sequence Q and P_j is the j-th value of relative pitch sequence P.

Table 2. Relative pitches of phrases

Phrases	AcVAU	AAcVAG
Absolute pitch	69 72 70 69 68	69 69 72 70 69 67
Relative pitch	+3 -2 -1 -1	0 +3 -2 -1 -2

Although pitch deviation with one semitone is admissible, the number of relative pitches which have one halftone error between each other is restrained. Otherwise, the two phrases can not be regarded as the same if there are too many relative pitches existing deviation. For example, the relative pitch sequences "+3 -2 -1 -1" and "+2 -2 0 -2" is not identical since there are three relative pitches with semitone error, and the two phrases consist of only five notes. Therefore proposed approach specifies the upper bound of the number of relative pitches with halftone deviation in terms of phrase length.

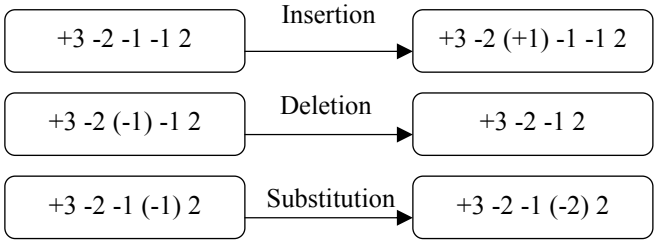

Figure 3. Insertion, deletion and substitution of relative pitch

3.3.2. Overall Bias: If a numerical value is inserted into or deleted from relative pitch sequence and the difference between it and its previous pitch or following pitch is large, we think that the inserted or deleted value alters the original melody contour. Similarly replacing one value for another number with much difference among relative pitch sequence will result in the same effect. These cases always take place when users input a query by humming or singing a short part of music. Unfamiliar with the music, users usually modify music melody

with errors like note insertion, deletion or substitution demonstrated in Figure 3. In order to tolerate these faults we allow preceding problems happening, but the difference between relative pitch inserted, deleted or substituted by users or query transcription module and its adjacent values must be in the range of [-1, 1]. This restriction ensures that the melody contour may not vary drastically.

While calculating the edit distance between two phrases using relative pitch sequences according to the equations (2) and (4), the overall bias caused by note insertion, deletion and substitution is counted up. Given relative pitch sequences $S = s_1 s_2 ... s_M$ and $T = t_1 t_2 ... t_N$ for query and indexed phrases respectively, edit distance matrix D and flag matrix F with M+1rows and N+1 columns are constructed firstly. In flag matrix symbols '←' and '↑' stands for insertion and deletion operations respectively, and '↖' denotes substitution if $|S_i - T_j| > 1$ or two relative pitches are deemed as same to each other if $|S_i - T_j| \leq 1$. Along with the recursive computation of edit distance between S and T, the flag matrix is filled step by step. Let $1 \leq i \leq M$ and $1 \leq j \leq N$, matrix F can be completed according to below formula:

$$F_{i,j} = \begin{cases} \leftarrow & if \;\; D_{i,j} = D_{i,j-1} + 1 \\ \uparrow & if \;\; D_{i,j} = D_{i-1,j} + 1 \\ \square & if \;\; D_{i,j} = D_{i-1,j-1} + Cost_{i,j} \end{cases} \quad (5)$$

with initial conditions:

$$D_{i=1:M,0} = \uparrow,$$
$$D_{0,j=1:N} = \leftarrow.$$

In particular when any two or all of $D_{i,j} = D_{i,j-1}+1$, $D_{i,j} = D_{i-1,j}+1$ and $D_{i,j} = D_{i-1,j-1}+Cost_{i,j}$ are equal in one step, we specify the priority for them: $D_{i,j} = D_{i,j-1}+1$ first, $D_{i,j} = D_{i-1,j}+1$ second and $D_{i,j} = D_{i-1,j-1}+Cost_{i,j}$ last.

	T_j	-1	-2	-2	2	5
S_i	0	←1	←2	←3	←4	←5
-4	↑1	↖1	←2	←3	←4	←5
-2	↑2	↖1	↖1	←2	←3	←4
-2	↑3	↑2	↖1	↖1	←2	←3
2	↑4	↑3	↑2	↑2	↖1	←2
-1	↑5	↑4	↑3	↖2	↑2	↖2
2	↑6	↑5	↑4	↑3	↖2	←3
2	↑7	↑6	↑5	↑4	↑3	↖3

Figure 4. Edit distance calculation and flag matrix

When recursion is over, the matrices of edit distance and flag are filled up. Starting from the last cell $F_{M,N}$ in flag matrix F, the optimal alignment path can be found along the direction of arrows. The operations of insertion, deletion and substitution can be

recognized in terms of the roles of arrow symbols on the optimal alignment path. Then overall bias can be summed up using aforementioned method. Providing source sequence "-4 -2 -2 2 -1 2 2" and target sequence "-1 -2 -2 2 5", Figure 4 illustrates an example for calculating edit distance and filling flag matrix as proposed approach, where edit distance is 3 and the shadow indicates the optimal alignment path.

3.3.3. Dynamic thresholds: Obviously as the length of query increases, the number of errors like note insertion, deletion and substitution will become large. In order to deal with this issue the maximal length of words in musical dictionary is limited to ten notes. Simultaneously three different thresholds of edit distance, the number of pitch deviation and overall bias vary with the note number of query phrase dynamically. The longer the length of query phrase is, the greater is the threshold, which can enhance the effect of fault tolerance.

3.4. Recursive segmentation

Inspired by Chinese information retrieval, documents are segmented into words and then keywords are used to search relative information. In our QBH system, analogous method is employed to process music search by humming or singing. Prior to performing retrieval, segmentation is carried out to partition melody string of query into phrases, which is rather different from Chinese word segmentation. The purpose of recursive segmentation is to tolerate faults caused by partition possibly, since it is difficult to judge whether the partition is right or not.

From the beginning of melody string, a phrase with maximal length namely ten notes is extracted and its relative pitch sequence is calculated. In terms of the length of this phrase, some indexed entries are selected as preliminary candidates. The edit distance, number of pitch deviation and overall bias described in previous subsection are calculated between this phrase and every preliminary candidate according to fault tolerant principle. If all of them satisfy the predefined dynamic thresholds, the music including preliminary candidate is regarded as a relative result. Meanwhile the end position of the phrase is added into a set of position. Otherwise the music is deemed as irrelevant to the query.

Subsequently one note is removed from the end of above phrase, which generates a new phrase. Same operations are implemented to the new phrase to search relative candidate music. This recursive segmentation is not over until the length of new phrase is shorter than or equal to specified threshold aforehand. Then position set is sorted in ascending order of its elements. The minimal element is selected as new position from which recursive segmentation is performed continuously. At the end of each recursion the minimal element is crossed out. Until the element of position set is equal to or greater than the end of query melody, the whole partition of input query terminates.

4. Experiments

4.1. Dataset

Our experiments take the corpus used for MIREX 2008 Query by Singing or Humming evaluation task to evaluate the proposed approach. The datasets consist of 154 monophonic ground-truth MIDI files (with MIDI 0 or 1 format) provided by Roger Jang and ThinkIT and 2000 non-target songs as music database. The non-target songs are randomly selected from the Essen Folk Song Database with ABC music notation format. Music melodies are

extracted from MIDI files or ABC files, and represented by melody strings with specified manner described in subsection 2.1. All melody strings are segmented into phrases on which inverted index are built.

The acoustic input of our QBH system comprises of 355 queries from ThinkIT corpus. There is no ``singing from beginning" guarantee for these queries. In other words these queries may be sung or hummed from any position of songs. Moreover the way to sing or hum is diverse: most of queries are sung with lyrics, a small part of them are hummed with syllables such as /Di/ and /Da/, and a very few are hummed nasal voice. The singing language contains Mandarin and Cantonese. All the queries are digitized at 8 kHz, 16 Bit, 128 kbps, PCM format.

4.2. Environment

The experiments are conducted on a personal computer with an Intel (R) Core (TM) 2 Quad 2.40GHz processor, 2 GB memory and running MS Windows XP operator system. All codes are written in ANSI C++ and complied with all optimization options.

4.3. Performance

In order to investigate the effect of proposed approach with three dynamic thresholds of edit distance (ED), pitch deviation (PD) and overall bias (OB), we carry out several experiments. The results are shown in Table 3 and the performance is measured by retrieval recall. In the table “ED 1-4: 0, 5-6: 1, 7-10: 2” means when the note number of query phrase is 1-4, 5-6 and 7-10, the threshold of ED is 0, 1 and 2 respectively. The specifications for other two thresholds of PD and OB are similar. Due to users’ imperfect memories and reproduction of melodies, there are more errors as the length of query increases. Consequently the admitted thresholds of ED, PD and OB increase gradually. All the values of parameters are obtained through repeated comparisons and experiments. From the results we can find that the retrieval recall of proposed approach is very high even up to 99.2%, which is directly proportional to the threshold of PD and inversely proportional to those of ED and OB.

Table 3. Experimental results-I (ED: edit distance, PD: pitch deviation, OB: overall bias)

No.	Dynamic Thresholds				Recall
1	ED	1-4: 0	5-6: 1	7-10: 2	97.2%
	PD	1-2: 0	3-6: 1	7-10: 2	
	OB	1-4: 0	5-6: 1	7-10: 2	
2	ED	1-4: 0	5-7: 1	8-10: 2	98.9%
	PD	1-2: 0	3-5: 1	6-10: 2	
	OB	1-4: 0	5-7: 1	8-10: 2	
3	ED	1-4: 0	5-7: 1	8-10: 2	99.2%
	PD	1-2: 0	3-4: 1	5-10: 2	
	OB	1-4: 0	5-7: 1	8-10: 2	

Without loss of generality, we repeatedly conduct the experiments by randomly selecting other 2000 songs from Essen Folk Song Database and build inverted index for new dataset.

Further, the experimental parameters are modified more concretely, especially the threshold of pitch deviation. The results of first and second experiments in Table 4 indicate that pitch deviation is very common in query and the recall can be improved to 99.4% by enlarging this parameter. However, only changing the variable PD usually produces more noise. Therefore it is necessary to decrease the other two thresholds to eliminate irrelevant data. Comparison between the second and third experimental results demonstrates that the performance can be advanced up to 99.7% by restraining the thresholds of ED and OB more strictly, and augmenting that of PD simultaneously.

Table 4. Experimental results-II (ED: edit distance, PD: pitch deviation, OB: overall bias)

No.	Dynamic Thresholds					Recall
1	ED	1-4: 0	5-7: 1	8-10: 2		99.2%
	PD	1-2: 0	3-4: 1	5-6: 2	7-10: 3	
	OB	1-4: 0	5-7: 1	8-10: 2		
2	ED	1-4: 0	5-7: 1	8-10: 2		99.4%
	PD	1-2: 0	3-4: 2	5-6: 3	7-10: 4	
	OB	1-4: 0	5-7: 1	8-10: 2		
3	ED	1-3: 0	4-7: 1	8-10: 2		99.7%
	PD	1-2: 0	3-4: 2	5-6: 3	7-10: 4	
	OB	1-3: 0	4-7: 1	8-10: 2		

5. Conclusions

Without singing experience or unfamiliar with desired music, users usually bring about many faults such as key alteration, tempo change, note insertion, deletion and substitution. Even query transcription with high precision may cause errors like note split or unification. In order to solve the above issues, this work proposes a novel approach based on fault tolerance and recursive segmentation to QBH system. Since users can not remember the melody inaccurately, they always produce pitch deviation when singing or humming. Therefore we permit the occurrence of pitch deviation within semitone when calculating edit distance between query phrase and indexed entries using relative pitch. However, the proposed approach restricts the number of note with pitch deviation, otherwise the melody contour change drastically. Moreover the overall bias caused by note insertion, deletion and substitution is not allowed to be large, which results in more errors. In order to enhance the robust of QBH system, the thresholds of ED, PD and OB vary with the length of query phrase dynamically. Recursive segmentation is employed to partition the query melody and search relevant music. Experimental results show that the performance of proposed approach is outstanding. Next step duration of notes will be added into calculation of edit distance to improve the retrieval precision.

6. Acknowledgments

The authors would like to thank the anonymous reviewers for helpful suggestions. This investigation is supported in part by National Natural Science Foundation of China (No. 60703015 and No. 60973076).

7. References

[1] Google, http://www.google.cn/music/homepage.

[2] Yahoo, http://music.cn.yahoo.com.

[3] S.P. Heo, M. Suzuki, A. Ito, and S. Makino, "An Effective Music Information Retrieval Method Using Three-Dimensional Continuous DP", IEEE Trans. on Multimedia, vol. 8, no. 3, 2006, pp. 633-639.

[4] I.S.H. Suyoto, A.L. Uitdenbogerd, and F. Scholer, "Searching Musical Audio Using Symbolic Queries", IEEE Trans. on Audio, Speech and Language, vol. 16, no. 2, 2008, pp. 372-381.

[5] S.Pauws, "CubyHum: A Fully Operational Query by Humming System", In Proceedings of International Society for Music Information Retrieval Conference (ISMIR), 2002.

[6] E. Unal, S. Narayanan, E. Chew, P.G. Georgiou, and N. Dahlin, "A Dictionary Based Approach for Robust and Syllable-Independent Audio Input Transcript for Query by Humming Systems", In Proceedings of International Conference of ACM Multimedia 2006, pp. 37-43.

[7] M.Ryynanen and A. Klapuri, "Query by Humming of MIDI and Audio Using Locality Sensitive Hashing", In Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008, pp. 2249-2252.

[8] T.D. Mulder, J.P. Martens, S. Pauws, F. Vignoli, and M. Lesaffre, "Factors Affecting Music Retrieval in Query-by-Melody", IEEE Trans. on Multimedia, vol. 8, no. 4, 2006, pp. 728-739.

[9] E. Unal, E. Chew, P.G. Georgiou, and S.S. Narayanan, "Challenging Uncertainty in Query by Humming Systems: A Fingerprinting Approach", IEEE Trans. on Audio, Speech and Language, vol. 16, no. 2, 2008, pp. 359-371.

[10] J.L. Hsu, C.C. Liu, and A.L.P. Chen, "Discovering Nontrivial Repeating Patterns in Music Data", IEEE Trans. on Multimedia, vol. 3, no. 3, 2001, pp. 311-325.

[11] N.-H. Liu, Y.-H. Wu, and A.L.P. Chen, "An effcient approach to extracting approximate repeating patterns in music databases", LNCS, vol. 3453, Springer-Verlag, Berlin Heidelberg, 2005, pp. 240-252.

[12] Essen Folk Song Database, http://www.esac-data.org.

[13] S. Rho, B.J. Han, E. Hwang, and M. Kim, "MUSEMBLE: A Novel Music Retrieval System with Automatic Voice Query Transcription and Reformulation", The Journal of Systems and Software, 2008, pp. 1065-1080.

[14] A.D. Cheveigne and H. Kawahara, "YIN, A Fundamental Frequency Estimatior for Speech and Music", Journal of the Acoustical Society of America, vol. 111, no. 4, 2002, pp. 1917-1930.

[15] N. Kosugi, Y. Sakurai, and M. Morimoto, "SoundCompass: A Practial Query-by-Humming System", In Proceedings of ACM SIGMOD, 2004, pp. 881-886.

Authors

Xiaohong Yang is a Ph.D. candidate in Department of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduage School. His main research interests include music retrieval, web information retrieval, natural language processing.

Qingcai Chen received the Ph.D. degree in computer application from Department of Computer Science and Technology, Harbin Institute of Technology in 2003. Since September 2004, he has been an associate professor in Department of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School. His main research interests include text/music

retrieval, natural language processing, speech signal processing, machine learning.

Xiaolong Wang received the Ph.D. degree in computer application from Harbin Institute of Technology in 1989. His is a professor and Ph.D. supervisor in Department of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School. His main research interests include web information retrieval, artificial intelligence, computational linguistics and Bioinformatics.