

A Repeating Pattern based Query-by-Humming Fuzzy System for Polyphonic Melody Retrieval

Yo-Ping Huang¹, Shin-Liang Lai² and Frode Eika Sandnes^{3,4}

¹Department of Electrical Engineering
National Taipei University of Technology
Taipei, Taiwan 10608
e-mail: yphuang@ntut.edu.tw

²Department of Computer Science and Engineering
Tatung University
Taipei, Taiwan 10451
e-mail: sinla.lai@gmail.com

³Faculty of Technology, Art and Design
Oslo and Akershus University College of Applied Sciences
Oslo, Norway
Email: Frode-Eika.Sandnes@hioa.no

⁴Faculty of Technology
Westerdals School of Art, Communication and Technology
Oslo, Norway

Abstract—Query-by-Humming involves retrieving music with a melody that matches the hummed query. An improved Query-by-Humming system for extracting pitch contour information based on a fuzzy inference model is introduced. In addition, an improved content-based music repeating pattern extraction model is introduced. Our bar-indexing method can extract the melody, identify repeating patterns and handle polyphonic MIDI files. To verify the effectiveness of the system, 15 volunteers recorded queries that were fed as input to the system and the Longest Common Subsequence (LCS) was used to identify the most related top N matches. The system achieves 70% accuracy among the top 5 items retrieved.

Keywords: fuzzy inference system, content-based music information retrieval, query-by-humming, repeating pattern, pitch contour.

1. INTRODUCTION

The Internet is commonly used to search for information such as text, images, video, and music. The capacity of storage devices is growing, and so is the amount of information stored. Multimedia files have become commonplace due to enhanced bandwidth and fast transmission networks. Traditional record shops are replaced with Internet-based pay-per-download and streaming services. The objective of Music Information Retrieval (MIR) is to quickly and correctly find the right contents from a music database. Several music file-formats are commonly used, including MIDI, MP3, Wave, and Voice. MIDI (Musical Instrument Data Interface) is a popular format as it contains semantic music information and not the actual audio. Moreover, MIDI files require less storage space than audio data, they are faster to process and therefore result in faster and more accurate query results.

Monophonic melodies are more common than polyphonic melodies in music information retrieval systems since most people are only able to sing, hum and whistle monophonically. Furthermore, humans perceive and remember monophonic melodies even though they hear polyphonic music. Therefore, the first task is to extract the main monophonic melody from a polyphonic melody to simplify the query matching.

Previous studies have explored several innovative query models, including Query-by-Humming [1-7], Query-by-Tapping [8], Query-by-Example [9], Query-by-Tags [10], and Query-by-Descriptions [11]. Query-by-Humming in particular has received much attention. This is probably because humming is a simple, intuitive and direct way for individuals to express music. However, it is more difficult to retrieve music based on humming compared to other approaches.

Several approaches translate hummed melodies into MIDI [12] as it simplifies comparison. Several music matching methods have been studied such as Bayesian classifiers [13], hidden

Markov models [14, 15], string matching [16, 17], dynamic programming [18, 19], dynamic time warping (DTW) [20, 21], and tree based searching [22, 23]. However, most of the previous approaches are computationally expensive. Previous research has used repeating patterns as a search index, since a repeating pattern is usually the prominent and recognizable part of a music piece. In such systems, when a user hums a repeating pattern as a query, the query-by-humming system uses this repeating pattern as the search index when searching the music database.

This study explores an improved music retrieval method based on humming. First, a method for extracting the perceivable main melody of a complete polyphonic musical piece is described. The output from several existing main melody extraction algorithms are used as input to the system and the one with the highest average pitch is selected. The melodies are coded using more symbols than previous methods and hence more of the musical information is preserved. In addition, fuzzy inference is employed to make the method more robust to the errors that are introduced when the recorded humming audio is converted into semantic music data. The objective of the method is to achieve more accurate retrieval results with less computational effort compared to previous approaches.

The remainder of this paper is organized as follows. Section 2 introduces definitions and notations used throughout the paper and previous research is reviewed. Section 3 presents the details of our approach. Section 4 documents the test results. Finally, Section 5 concludes the paper and indicates areas of further study.

2. RELATED WORK

Uitdenbogerd and Zobel [24, 25] explored four melody extraction algorithms. Fig. 1 shows results obtained using the four methods. Fig. 1a shows the original polyphonic music piece spread across the G-clef and the F-clef. Their first algorithm, Skyline [26], combines all the notes of a MIDI file into single MIDI channel by taking the note with the highest pitch (see

Fig. 1b). Fig. 1c shows the perceivable melody extracted by an algorithm that only includes the notes with the highest pitch in the G-clef. Fig 1d shows the results of an algorithm that considers the notes with the highest pitch in the F-clef only. Finally, the results in Fig. 1e are generated by an algorithm that only includes the lowest pitch of the F-clef, namely the bass line.



Fig. 1. (a) Original music, (b)-(e) the monophonic melodies extracted using the four algorithms [24].

The Skyline algorithm was later improved by including note durations to obtain better recognition results [27]. This is achieved using a time overlap parameter. Another variation of this algorithm splits the process into three steps: (1) remove unwanted channels, (2) select the channel with the largest volume, and (3) keep the highest note when several notes are played simultaneously [28].

Methods for extracting the main melody from multiple MIDI channels of polyphonic music pieces either break up all notes from their original positions and then regenerate the main melody, or they identify the MIDI channel holding the main melody. However, these approaches may not fully capture the perceived melody. First, the result may be different

from the original melody when the notes are broken up. Second, composers may distribute the main melody across different channels.

Fig. 2. Illustrates how the main melody spans multiple MIDI channels in an arrangement of “Symphony No. 40 first Movement” by Wolfgang Amadeus Mozart. The main melody is indicated by using highlighted gray boxes. The five first bars of the melody start on channel 4, then move to channel 1 for six bars, and finally return to channel 4. Such channel changes occur frequently as composers often assign different instruments to the main melody and each MIDI channel can be assigned a different instrument. A bar is a segment of time corresponding to a specific number of beats and the boundaries of the bar are indicated by vertical bar lines (see Fig. 2).



Fig. 2. The main melody across multiple MIDI channels.

Shih et al. [29] indexed music using a dictionary to find repeating patterns based on the

classic work of Lempel and Ziv. Each bar is indexed and an adaptive dictionary method is used for identifying repeating patterns. Tseng [30] used a pitch profile encoding for queries and an n-node indexing method for approximate matching in sub-linear time. Experiments showed that the pitch profile encoding and a 3-node indexing were able to overcome the key mismatch problem and the random errors caused by pitch error, note deletion and node insertion. Chang et al. [31] believed melody contours could cover most of the repeating patterns in the melody string. Jonah et al. [32] used hidden Markov models (HMMs) to represent music. In their work, the query was treated as an observation sequence and a piece is considered to be similar to the query if its HMM has a high likelihood of generating the query. The top pieces were returned to the user in rank-order. Hsu et al. [33] developed a data structure called correlative matrix to extract repeating patterns and a string-join operation and a random projection tree to find repeating patterns. Medina et al. [34] used a self-similarity matrix to extract the themes of music and Barlow and Morgenstern's Dictionary of Musical Themes to verify the correctness of the themes.

Most studies translated each note from the music database to form indices. If a composition has many notes such as semiquavers or demisemiquavers, more computational effort is needed. Thus, the goal of our approach is to balance the size of the sampling time windows of the melody to achieve a meaningful and effective index.

Various indexing units have been used including notes, bars and phrases. The purpose of the index is to maintain the integrity of the original music. This study considers the bar and pitch interval of music compositions as index elements. The bars are the natural larger units of music, and each melody is constructed from bars.

Ghias et al. [35] processed MIDI files by replacing each note with a symbol representing its pitch relative to the previous note. In this way, three pitch directions were defined, namely up (U), down (D) and level (S), and each pitch is assigned one of three symbols, (U, D, S).

Several researchers have improved upon these, including Typke et al.'s Parsons Code [36]. They also used the pitch contour from humming and represented them with strings of U, D, and R (repeat) symbols. McNab et al. [37] combined a rhythm feature with the pitch contour. Li et al. [6], Sonoda et al. [38], and Mo et al. [39] considered duration using L (longer), R (repeat), and S (same) to represent changes. These studies are mostly based on Ghias's method using U, D, and S symbols. Although this approach is simple, it suffers from the drawback that different melodies may result in the same sequences. For example, the pitch contours of the two different strings "g5, e5, e5, f5, d5, d5" and "g5, c5, c5, a5, f5, f5" are identical, that is "D, S, U, D, S". In this study we overcome this problem by including seven symbols.

3. IMPROVED MUSIC RETRIEVAL

The approach described in this work first translates the original polyphonic music recording to a monophonic melody, and then uses a four phase algorithm to extract the main melody. Next, repeating patterns are extracted for each MIDI file and transformed to pitch contours. These pitch contours are stored in the pitch contour database. The hummed queries are fuzzified to make the approach more robust to acoustic audio processing errors. Finally, we compare the user's pitch contour with the entries in the pitch contour database using the LCS algorithm. Each of these phases is described in the following sections.

3.1 Main Melody Extraction

Four phases are used to identify the melody, namely the removal of the percussion channel, channel-based pitch extraction, average pitch computation, and main melody salience.

3.1.1 Percussion Channel Removal

Ghias et al. suggested that the percussion channel never contributes to a melody and should be discarded during melody selection [26, 36]. Consequently, the elimination of the

percussion channel enhances the accuracy of the search and reduces the computational load. Therefore, we removed the percussion channel.

3.1.2 Channel-based Pitch Extraction

In the Skyline algorithm, all notes of a musical composition are collected into a single channel. From these notes, the highest pitch line is determined and used to define the main melody. In order to preserve the original melody, the note durations are not modified.

3.1.3 Average Pitch

To determine the channel containing the main melody, the average pitch of each channel is calculated and the channel with highest average pitch is defined as the melody channel with the highest pitch. This study targets folk songs and most melodies in this genre may appear in the highest pitch melody channel. Note that this assumption may not hold for other music genres such as jazz, soul or funk where the bass lines often convey the perceived melody of a music piece.

Assume that M is a MIDI file with channels C such that $M = \{ C_1, C_2, \dots, C_i \}$ where $1 \leq i \leq 16$. Each channel, C_i , contains j bars. Mathematically, the bars of a channel can be expressed as $C_i = \{ B_{i,1}, B_{i,2}, \dots, B_{i,j} \}$. The average pitch of $B_{i,j}$ being $P_{i,j}$ is also defined and then the average pitch of each channel (P_i) can be calculated:

$$P_i = \frac{\sum_{k=1}^j P_{i,k}}{j} . \quad (1)$$

After calculating the average pitch within each of the 16 channels, these values are sorted to determine the channel having the highest pitch, labeled as the highest pitch melody channel.

3.1.4 Main melody salience













A function that determines the location of the main melody within a bar is referred to as the main melody salience. First, we use the main melody salience function to calculate the

difference between two adjacent bars in the same channel to detect whether the main melody has moved to another channel or not. Second, we use the main melody salience function to compare bars in different channels to determine to which channel the melody may have moved. For this, we consider pairs of bars, and sequentially check each pair from the first to the last bar. Next, we defined a threshold parameter that provides a decision cutoff for assigning the melody into a particular channel. Thus, if the main melody salience difference of an adjacent pair of bars is less than the threshold, the main melody may be moved to the other channels. In practice, the threshold was set to the values between 0.10 and 0.15 in our implementation. The main melody salience $MMS_{i,j}$ of $B_{i,j}$ is formally defined as follows:

$$MMS_{i,j} = \frac{1}{2} \left| \frac{P_{i,j}}{P_{i,j} + P_{i,j+1}} + w \left(\frac{D_{i,j}}{D_{i,j} + D_{i,j+1}} \right) - \frac{P_{i,j+1}}{P_{i,j} + P_{i,j+1}} - w \left(\frac{D_{i,j+1}}{D_{i,j} + D_{i,j+1}} \right) \right|, \quad (2)$$

where $P_{i,j}$ is introduced in the previous subsection and $D_{i,j}$ is the duration average of notes. Table I shows the contrast between note names and the defined values. The constant variable w is a weight. For the pitch and duration, pitch is considered more influential. In our experiment, w was set to 0.5 to reduce the influence of duration.

TABLE I
THE CONTRAST BETWEEN NOTES AND DEFINED VALUES

Note	Defined value	Note	Defined value
	1		1.5
	2		3
	4		6
	8		12
	16		24
	32		48
All rest notes	0		

When the main melody departs from the highest pitch melody channel, Eq. (3) can be used to determine its main melody salience at the same bar of the other channels, and the channel with the highest main melody salience is defined as the next location of the main melody.

$$MMS_{i,j} = \frac{1}{n} \left(\frac{P_{r,j}}{\sum_{k=1}^n P_{k,j}} + w \left(\frac{D_{r,j}}{\sum_{k=1}^n D_{k,j}} \right) \right), \quad (3)$$

where n represents the number of channels. Next, Eq. (4) is used to detect whether the main melody returns to the highest pitch melody or not.

$$MMS_{i,j} = \frac{1}{2} \left(\frac{P_{i,j}}{P_{i,j} + P_{i,j}} + w \left(\frac{D_{i,j}}{D_{i,j} + D_{i,j}} \right) - \frac{P_{i,j}}{P_{i,j} + P_{i,j}} - w \left(\frac{D_{i,j}}{D_{i,j} + D_{i,j}} \right) \right). \quad (4)$$

If $MMS_{i,j}$ is less than 0 the main melody returns to the first channel (highest pitch melody); otherwise, it remains in the current channel C_i .

Finally, the main melody is extracted from the MIDI file, and it is combined with the main channel (highest pitch melody) and a part of other channels. This algorithm is illustrated below.

1. Let $C_1 := MM$ (highest pitch melody)
2. Let $i = 1$
3. For each $E_{l,i}$ such that $i < B$
 - 3.1 If $(|E_{l,i} - E_{l,(i+1)}| > T)$ then
 - 3.1.1 Let $C_j = \text{Max}\{E_{l,(i+1)}, E_{2,(i+1)}, \dots, E_{15,(i+1)}\} := MM$
 - 3.1.2 If $(C_j \diamond C_1)$ then
 - Let $k = i + 2$
 - For each $E_{j,k}$ such that $k < B$
 - If $(E_{j,k} < E_{l,k})$ then
4. Let $C_1 := MM$, $i = k - 1$, and break loop

There are only 15 channels since the percussion channel is removed, and $P_1 > P_2 > \dots > P_{15}$, such that C_1, C_2, \dots, C_{15} is sorted. C_1 is the highest pitch melody. The number of bars, main melody, and threshold are presented as B , MM , and T , respectively.

Fig. 3 illustrates the process. The staff in Fig. 3(a) represents a piece from a MIDI file and after channel-based pitch extraction process it becomes the staff in Fig. 3(b). Next, the main

melody salience values are calculated and compared to determine the main melody represented by the staff in Fig. 3(c). Finally, the main melody is generated and shown as the staff in Fig. 3(d), that is, a monophonic melody.

Note that this method is not designed to handle complex musical structures such as musical counterpoints, which are commonly found in many Baroque pieces. When retrieving a particular piece of music one would expect the user to hum a continuous and recognizable part of the melody and not to hum the counterpoints.

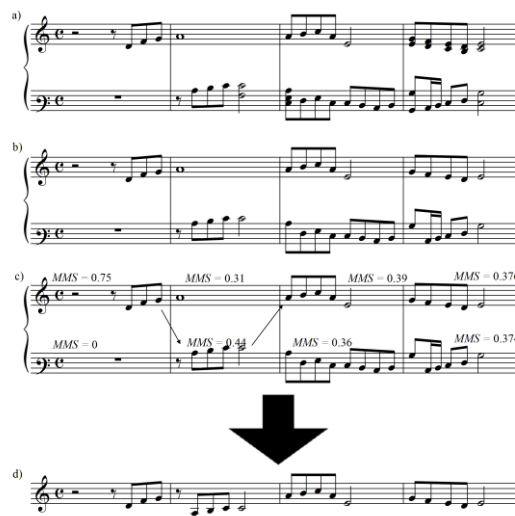


Fig. 3. Main melody extraction.

3.2 Extracting Repeating Patterns

The process of extracting of repeating patterns consists of indexing, extracting, and pruning.

3.2.1 Indexing

The pitch of the first and last notes from each bar are retained, and the pitch interval variation is obtained by subtracting the lowest pitch from the highest pitch in each bar. Each bar can then be represented simply by three elements, i.e., the first pitch, the last pitch and the pitch interval. The pitch interval is represented by the MIDI pitch value (from 0 to 127). For example, the four bars in Fig. 4 can be represented using $\{g5, e5, 3\}$, $\{f5, d5, 5\}$, $\{c5, f5, 5\}$, and $\{g5, g5, 0\}$, respectively.



Fig. 4. MIDI file example.

3.2.2 Extracting

The repeating pattern search algorithm is illustrated next:

1. Let S be a sequence of bar index set of length n
2. Let $i = 0$, and $k = 0$
3. For each S_i such that $i < n$
 - 3.1 If $i \leq k$, let $j = k + 1$
 - 3.2 For each S_j such that $j < n$

If $S_i = S_j$ then

put S_i to the pre-repeating list

set the value of j to k

and break loop

Step 3 sequentially compares adjacent index sets from the first to the last pitch. To reduce the number of comparisons, the variable k is used to locate the last repeating pattern found. When the repeating pattern is found, k contains the position, and in the next iteration, only position $i+1$ with $k+1$, not $i+1$ with $i+2$ will be compared. In this case, the number of comparison can be reduced to $(k+1)-(i+1)-1$. By inspecting our music database, consisting of mostly Western and Chinese folk songs, we found that the duplicate pieces have a regular

pattern. Therefore, we can skip several search steps, starting directly after the location of the repeated section. As an illustration of how the algorithm proceeds, Fig. 5 shows different iterations of the algorithm. In the first iteration, the first and sixth bars are the same, indicated by “A”. In the second iteration we compare the second and seventh bars and find the same “B”. In the third iteration we compare the third and eighth bars with “C” and “F”. In the fourth iteration we therefore compare the fourth and eighth bars since “C” is different from “F”, and so on. In this phase, all repeating patterns found are put into a pre-repeating list, and each pattern is separated by the symbol “,”. Duplicate patterns may exist in the pre-repeating list and a pruning algorithm is used to generate the final repeating patterns. For example, imagine that there are two repeating patterns “{g5, e5, 3}, {f5, d5, 5}” and “{g5, e5, 3}, {f5, d5, 5}, {a5, g5, 7}” (see Fig. 4), then the pre-repeating list is “{g5, e5, 3}, {f5, d5, 5}, {g5, e5, 3}, {f5, d5, 5}, {a5, g5, 7}”.

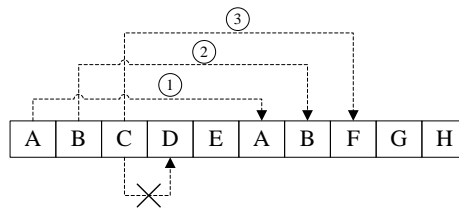


Fig. 5. Repeating pattern extraction.

3.2.3 Pruning

The algorithm for pruning the pre-repeating list is illustrated next:

1. Let P be a string set sequence of pre-repeating list that includes m patterns and each pattern is separated by “,”
2. Add a character “.” to the end of P
3. Let $i = 0$
4. For each P_i such that $i < m$, let $j = 0$

4.1 For each P_j such that $j < m$

4.1.1 If the first character of P is “.” then

break and

return final repeating pattern

Else

4.1.1.1 If $P_i \subseteq P_j$ then

delete P_i from P

Else

move P_i to the end of P

4.1.1.2 update string P

In Step 2 of the algorithm, the character “.” is added to the end of the pre-repeating list. When the pruning algorithm finds the end point in step 4.1.1, it indicates the completion of the pruning process. If pattern A is included within pattern B , then pattern A is pruned, and only pattern B is retained as given in step 4.1.1.1 above. The final step in this phase involves updating the pre-repeating list. For example, let pre-repeating pattern P contains $P_1, P_2, P_3,$ and P_4 (see Fig. 6). Since P_2 comprises P_1 then P_1 is removed; and P_2 is the same as P_4 and P_2 is then also removed. After the pruning process, the final repeating pattern is the set that contains P_3 and P_4 . In Fig. 4, two patterns “{g5, e5, 3}, {f5, d5, 5}” and “{g5, e5, 3}, {f5, d5, 5}, {a5, g5, 7}” were obtained from the pre-repeating list after the repeating pattern extraction phase. Because the latter pattern consists of the former pattern, the former pattern is pruned and only the repeating pattern “{g5, e5, 3}, {f5, d5, 5}, {a5, g5, 7}” is retained.

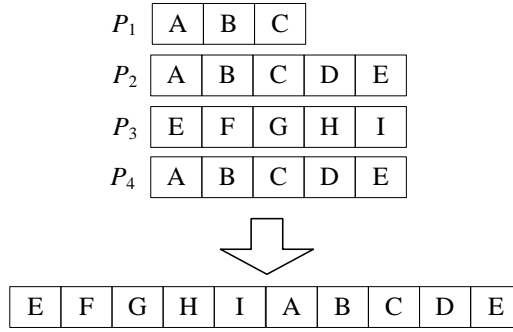


Fig. 6. Pruning.

3.3 Query-by-Humming

Based on the recommendations presented in Zhu et al. [40], we used Akoff Music Composer version 3.0 as a tool for transforming WAV to MIDI.

3.3.1 Pitch Contour

After translating WAV to MIDI, the pitch interval is analyzed. Pairs of neighboring pitch intervals are divided into seven symbols as shown in Table II.

TABLE II
PITCH CONTOUR

Symbol	MIDI Pitch Interval
H	> 12
R	7 ~ 12
U	1 ~ 6
S	0
D	-1 ~ -6
B	-7 ~ -12
L	< -12

3.3.2 Fuzzy Inference System (FIS)

We used a Fuzzy Inference System (FIS) to transform time intervals of pitch queries into symbolic representations [43-46]. Such a soft method is effective since the input humming queries may contain slight pitch errors, resulting in erroneous notes when translated to MIDI. By using a fuzzy inference, soft membership functions provide more robust classification of these input pitch contour errors introduced in the audio to MIDI transformation. Seven Gaussian membership functions are defined for the input query variable in this study (see

Table II). The seven terms are labeled mf1 to mf7 and correspond to the output membership functions L, B, D, S, U, R, and H, respectively. Seven fuzzy rules are defined accordingly.

3.3.3 Longest Common Subsequence

Longest Common Subsequences (LCS) are used for similarity measurements of music pitch contours because of the temporal characteristics of music. Given two string sequences $X(1, 2, \dots, m)$ and $Y(1, 2, \dots, n)$ of length m and n , respectively. X_i represents a subsequence of $X(1, 2, \dots, i)$ or the prefix of the sequence X of length i and x_i represents i th element of the subsequence. The LCS of X and Y is described in the following expressions:

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ (LCS(X_{i-1}, Y_{j-1}), x_i) & \text{if } x_i = y_j \\ \text{Max}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases} \quad (5)$$

Function $LCSLen(X[1, \dots, m], Y[1, \dots, n])$

Initialize $C[m][n]$ to zero

for $i = 0$ to m

$C[i][0] = 0$

for $j = 0$ to n

$C[0][j] = 0$

for $i = 1$ to m

for $j = 1$ to n

if $X[i] = Y[j]$

$C[i][j] = C[i-1][j-1] + 1$

else

$C[i][j] = \text{Max}(C[i][j-1], C[i-1][j])$

Return $C[m][n]$

For example, the two pitch contours “U, S, S, D, R, D, R, B” and “U, S, D, R, B” yields an LCS length of 4 as shown in Fig. 7. Thus, LCS length is used for sorting entries and selecting

the top ten as candidate sets when queries are compared with entries in the database.

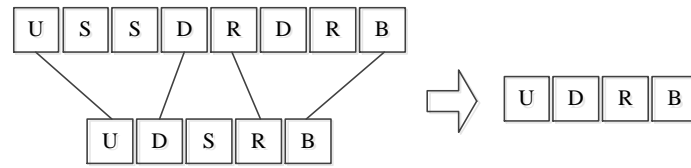


Fig. 7. LCS example.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

Experiments were conducted to investigate the effectiveness of our approach and the results are compared with other Query-by-Humming systems. However, it is challenging to compare different Query-by-Humming systems, as the retrieval performance is affected by the audio format used, music styles, recording conditions, database compositions and retrieval methods. The focus is therefore on previous studies with similar experiments to ours.

4.1 Experimental Setup

The database of 527 polyphonic MIDI files were collected from the Internet, mostly consisting of Chinese and western folk songs. The duration of the MIDI songs varied from approximately two to five minutes and the number of bars varied from 32 to 116. The total playback time of all the MIDI files was 31 hours and 45 minutes, and the total number of bars was 34,585. On average, each composition contained 66 bars, and the average duration was 3.6 minutes. The pitch contours associated with the repeating patterns for each music piece were stored in the database as indices.

A total of 15 volunteers were recruited including 9 females and 6 males who had never undertaken any professional music training. The musical abilities of the volunteers were not tested and no training was provided before recording the queries. A total of 60 hummed queries were recorded ranging in duration from 5 to 15 seconds. The queries were translated to MIDI using Akoff Music Composer and the FIS was used to transform the MIDI data into

pitch contours. After transforming the queries to pitch contours, the length of the pitch contour strings ranged from 20 to 34 symbols. The FIS was implemented in MATLAB.

4.2 Repeating Pattern Extraction

Repeating patterns were extracted for all the compositions in the database and for each query. Fig. 8 plots the repeating pattern extraction processing times as a function of the number of bars. The execution times ranged from 0.4 seconds with 32 bars to 1.7 seconds with 116 bars. The mean execution time was 0.94 seconds. Note that these results are achieved without optimizations. Related studies did not report execution times and comparisons to these studies could therefore not be made.

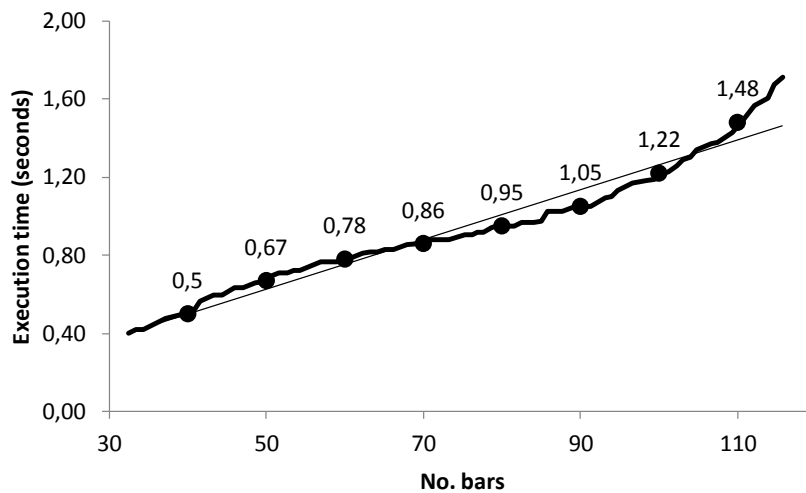


Fig. 8. Processing time as a function of number of bars.

Furthermore, the effect of the repeating patterns was evaluated using a hit rate measure. This hit rate is defined as the portion of retrievals that correctly matches the query. The hit rates achieved with and without repeating patterns are compared in Fig. 9. The length of the pitch contours representing the raw unprocessed queries ranged from 20 and 34 symbols. Comparatively, the length of the simplified pitch contours (repeating patterns) ranged from as little as 10 symbols to 32 symbols. Fig. 9 shows that the hit rate is larger for the simplified

queries as the best results occur for queries comprising 10 to 15 symbols with a hit rate of 66.7% for 10 symbol pitch contours. Even when users hum a melody repeatedly, or hum a lengthy melody, suitable key-indices are found without relying on full queries.

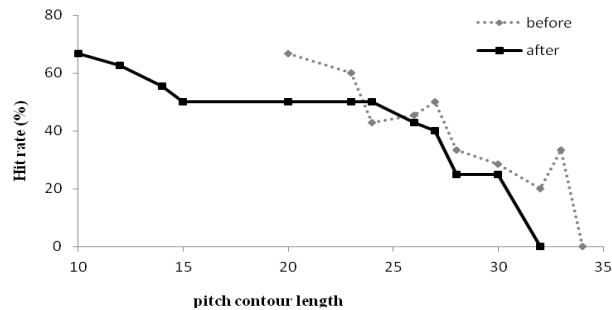


Fig. 9. Hit rates before and after extracting the repeating patterns.

4.3 Fuzzy Inference System

Fig. 10 demonstrates the effects of FIS compared to the results acquired without FIS. The horizontal axis represents the top N and the vertical axis represents the retrieval rate. The retrieval rates achieved with FIS are 5 to 19 percentage points higher than the retrieval rates achieved with previous methods. On average FIS achieves 10 percentage points above those of the other methods. That is, the observations obtained with FIS (the stippled line) are on average 10% points higher than those of previous methods (the solid line). In just one case the FIS improvement was only 5 percentage points.

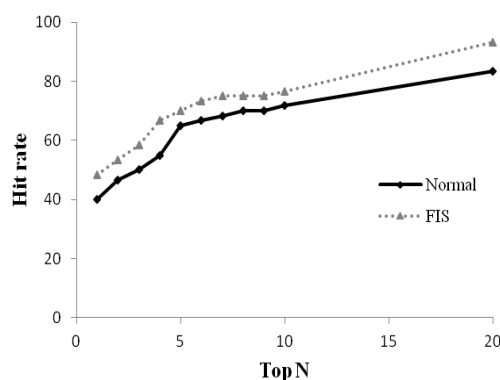


Fig. 10. Results with and without FIS.

4.4 Performance Comparisons

Finally, our method is compared with three similar Query-by-Humming systems, Guo et al. 2008 [41], Li et al. 2010 [6], and Jeon and Ma 2011 [42]. Unfortunately, there are no standard benchmark databases for query-by-humming research available and the systems described in the previous studies are not generally available. We were therefore forced to design our own test suite. The benefits of the studies selected for comparison is that they all used complete and polyphonic compositions. Note that all the compared methods are based on different test suites and there are therefore several sources of potential errors that could bias the results such as different music styles, quality of humming and the quality of the humming itself. The comparisons are valid because all the compared observations are made using the same unit of measure. Moreover, the mean performances of the approaches are compared, thus counterbalancing potential biases of individual music pieces.

The results are presented in Fig. 11 where the horizontal axis represents the mean percentage of top N retrieval and the vertical axis compares top 1, top 3, top 5, and top 10 because of the limitations of the data reported in the previous studies. Fig. 11 reveals that our system has the best performance, yielding 48.3% at top 1 while the other systems achieve 47.1%, 41.2%, and 38%, respectively. The results of Guo et al. are the most similar to ours. Finally, our system achieves 76.7% at top 10, while all the other systems achieve below 70%.

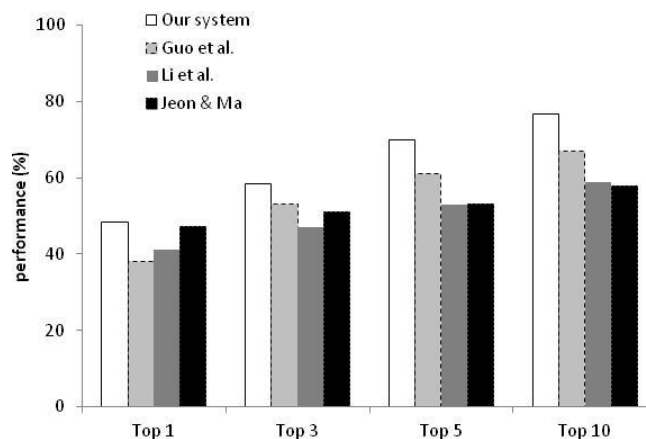


Fig. 11. Four Query-by-Humming systems contrasted.

Yet, there are unresolved issues such as errors introduced during the complex audio to

MIDI conversion. Moreover, problems occurred during the extraction of the repeating patterns. First, the ending bar of a composition might be incomplete. This mistake may occur when the system indexes each bar of source MIDI file. Second, the composition contained many grace notes that appeared exactly at the first or the last note of a bar. When the melody repeated again with these grace notes, our algorithm might misrecognize them as two different patterns. Third, if the grace notes or variations occurred in the highest or lowest note, it would generate different bar indices, and our algorithm might misrecognize them as two different patterns. Finally, errors may occur during the audio recording, for instance the background noise or the user humming with an incorrect pitch.

5. CONCLUSIONS AND FUTURE WORK

This study introduces an improved content-based music retrieval method that employs a bar-indexing method that reduces the processing time and provides an effective strategy to deal with MIDI files. In addition, the bar-indexing approach is robust to small variation in melody when extracting repeating patterns.

An improved pitch interval coding is introduced using seven symbols. We use a fuzzy inference system to transform time intervals of pitch queries into symbolic representations, since input humming queries may contain slight pitch errors, resulting in erroneous notes when translated to MIDI. By using a fuzzy inference, soft membership functions provide more robust classification of these input pitch contour errors. LCS is applied as approximate matching algorithm to locate the top N retrieval results. Experimental evaluation demonstrates that the FIS improves retrieval accuracy.

In summary, the advantages of our method are as follows: (1) The method can be applied to complete and polyphonic music, which is useful for the automatic establishment of music databases. (2) The use of repeating patterns as a search index requires less database storage space, require less computation and yield better retrieval accuracy. The repeating pattern of

each music composition is easy to remember and thus suitable as a query. (3) The FIS adds fault tolerance and improves retrieval accuracy.

Future work involves increasing the size of the database including other types of music to test system scalability and robustness of the method to other genres of music, improving the audio to MIDI conversion process with improved acoustical analysis algorithms utilizing recent advances in digital signal processing research, and improving the general retrieval accuracy by for instance introducing score-informed constraints and extending the model to include timbre in addition to temporal and chromatic information. Another possibility is to explore whether more musically inclined participants achieve higher retrieval accuracies than less musically inclined participants.

ACKNOWLEDGMENTS

This work was supported in part by Ministry of Science and Technology, Taiwan under Grants NSC102-2221-E-027-083- and NSC102-2218-E-002-009-MY2.

REFERENCES

- [1] N. H. Adams, M. A. Bartsch, and G. H. Wakefield, "Note segmentation and quantization for music information retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp.131-141, Jan. 2006.
- [2] B. D. Roger, P. B. William, P. Bryan, H. Ning, M. Colin, and T. George, "A comparative evaluation of search techniques for query-by-humming using the MUSART testbed," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 5, pp.687-701, March 2007.
- [3] E. Unal, E. Chew, P. G. Georgiou, and S. S. Narayanan, "Challenging uncertainty in query by humming systems: a fingerprinting approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp.359-371, Feb. 2008.

- [4] Y. Jinhee, P. Sanghyun, and K. Inbum, "An efficient frequent melody indexing method to improve the performance of query-by-humming systems," *Journal of Information Science*, vol. 34, no. 6, pp.777-798, Dec. 2008.
- [5] S. Rho, B. Han, E. Hwang, and M. Kim, "MUSEMBLE: a novel music retrieval system with automatic voice query transcription and reformulation," *Journal of Systems and Software*, vol. 81, no. 7, pp.1065-1080, July 2008.
- [6] P. Li, M. Zhou, X. Wang, X. Wang, N. Li, and L. Xie, "A novel MIR framework and application with automatic voice processing, database construction and fuzzy matching," in *Proc. of the 2nd International Conference on Computer and Automation Engineering*, Singapore, vol. 1, pp.20-24, Feb. 2010.
- [7] K. Kichul, R. P. Kang, S.-J. Park, S.-P. Lee, and Y. K. Moo, "Robust query-by-singing/humming system against background noise environments," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp.720-725, May 2011.
- [8] H. Pierre and R. Matthias, "Query by tapping system based on alignment algorithm," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Taipei, Taiwan, pp.1881-1884, April 2009.
- [9] F. Pereira, A. Vetro, and T. Sikora, "Multimedia retrieval and delivery: essential metadata challenges and standards," *Proceedings of the IEEE*, vol. 96, no. 4, pp.721-744, April 2008.
- [10] J.-C. Wang, M.-S. Wu, H.-M. Wang, and S.-K. Jeng, "A content-based music search system using query by multi-tags with multi-levels of preference," in *Proc. of IEEE International Conference on Multimedia and Expo*, Suntec City, Singapore, pp.1-6, July 2010.

- [11] B. Whitman and R. Rifkin, "Musical Query-by-Description as a multiclass learning problem multimedia signal processing," in *Proc. of IEEE Workshop on Multimedia Signal Processing*, St. Thomas, Virgin, USA, pp.153-156, Dec. 2002.
- [12] N. Hu and R. B. Dannenberg, "A comparison of melodic database retrieval techniques using sung queries," in *Proc. of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, Portland, Oregon, USA, pp.301-307, July 2002.
- [13] P.P. de Leon and J. Inesta, "Pattern recognition approach for music style identification using shallow statistical descriptors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, no. 2, pp.248-257, Mar. 2007.
- [14] J.-S. R. Jang, C.-L. Hsu, and H.-R. Lee, "Continuous HMM and its enhancement for singing/humming query retrieval," in *Proc. of the International Symposium on Music Information Retrieval*, London, UK, pp.546-551, Sep. 2005.
- [15] J.-S. Lee and C. H. Park, "Hybrid simulated annealing and its application to optimization of hidden Markov models for visual speech recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 4, pp.1188-1196, Aug. 2010.
- [16] K. Lemström, *String matching techniques for music retrieval*, Ph.D. thesis, Department of Computer Science, Faculty of Science, University of Helsinki, 2000.
- [17] C. Parker, "Towards intelligent string matching in Query-by-Humming systems," in *Proc. of IEEE International Conference on Multimedia and Expo*, vol. 2, Baltimore, Maryland, USA, pp.25-28, July 2003.
- [18] J.-S. R. Jang and H.-U. Lee, "A general framework of progressive filtering and its application to Query by Singing/Humming," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp.350-358, Feb. 2008.

- [19] T. Nishimura, H. Hashiguchi, J. Takita, J. X. Zhang, M. Goto, and R. Oka, "Music signal spotting retrieval by a humming query using start frame feature dependent continuous dynamic programming," in *Proc. of the International Symposium on Music Information Retrieval*, Bloomington, Indiana, USA, pp.211-218, Oct. 2001.
- [20] H.-M. Yu, W.-H. Tsai, and H.-M. Wang, "A query-by-singing system for retrieving karaoke music," *IEEE Transactions on Multimedia*, vol. 10, no. 8, pp.1626-1637, Dec. 2008.
- [21] A. N. Myna, V. Chaitra, and K. S. Smitha, "Melody information retrieval system using dynamic time warping," in *Proc. of 2009 WRI World Congress on Computer Science and Information Engineering*, vol. 5, Los Angeles, California, USA, pp.266-270, March-April 2009.
- [22] U. Bagci and E. Erzin, "Automatic classification of musical genres using inter-genre similarity," *IEEE Signal Processing Letters*, vol. 14, no. 8, pp.521-524, Aug. 2007.
- [23] J. Shen, D. Tao, and X. Li, "QUC-Tree: integrating query context information for efficient music retrieval," *IEEE Transactions on Multimedia*, vol. 11, no. 2, pp.313-323, Feb. 2009.
- [24] A. Uitdenbogerd and J. Zobel, "Manipulation of music for melody matching," in *Proc. of the ACM International Multimedia Conference*, Bristol, United Kingdom, pp.235-240, Sep. 1998.
- [25] A. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," in *Proc. of the Seventh ACM International Multimedia Conference*, Orlando, Florida, USA, pp.57-66, Oct.-Nov. 1999.
- [26] G. Ozcan, C. Isikhan, and A. Alpkock, "Melody extraction on MIDI music files," in *Proc. of the Seventh IEEE International Symposium on Multimedia (ISM'05)*, Irvine, California, USA, pp.414-422, Dec. 2005.

- [27] W. Chai, *Melody Retrieval on the web*, MS Thesis, Massachusetts Institute of Technology, Boston, Massachusetts, USA, 2000.
- [28] K. M. Shan and F. K. Fang, "Music style mining and classification by melody," *IEICE Transactions on Information and Systems*, vol. E86-D, no. 4, pp.97-100, 2003.
- [29] H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo, "Automatic main melody extraction from MIDI files with a modified Lempel-Ziv algorithm," in *Proc. of the 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, Hong Kong, pp.9-12, May 2001.
- [30] Y.-H. Tseng, "Content-based retrieval for music collections," in *Proc. of ACM SIGIR Proceedings: Research and Development in Information Retrieval*, Athens, Greece, pp.176-182, July 2000.
- [31] C. W. Chang and H. C. Jiau, "Extracting significant repeating figures in music by using quantized melody contour," in *Proc. of the 2003 IEEE International Symposium on Computers and Communication*, vol. 2, Antalya, Turkey, pp.1061-1066, June 2003.
- [32] S. Jonah, P. Bryan, M. Colin, and B. William, "Hmm-based music query retrieval," in *Proc. of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, Portland, Oregon, USA, pp.295-300, July 2002.
- [33] J. L. Hsu, C. C. Liu, and L. P. Chen, "Discovering nontrivial repeating patterns in music data," *IEEE Transactions on Multimedia*, vol. 3, no. 3, pp.311-325, Sept. 2001.
- [34] R. A. Medina, L. A. Smith, and D. R. Wagner, "Content-based indexing of musical scores," in *Proc. of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, Houston, Texas, USA, pp.18-26, June 2003.
- [35] A. Ghias, J. Logan, D. Chamberlin, and B.C. Smith, "Query by humming: musical information retrieval in an audio databases," in *Proc. of the 3rd ACM International Conference on Multimedia*, San Francisco, California, USA, pp.231-236, Jan. 1995.

- [36] R. Typke and L. Prechelt, "An interface for melody input," *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 2, pp.133-149, June 2001.
- [37] R. J. McNab, L. A. Smith, I. H. Witten, and C. L. Henderson, "Tune retrieval in the multimedia library," *Multimedia Tools Application*, vol. 10, no. 2-3, pp.113-132, April 2000.
- [38] T. Sonoda and Y. Muraoka, "A www-based melody retrieval system - an indexing method for a large melody database," in *Proc. of International Computer Music Conference*, Berlin, Germany, pp.170-173, Aug.-Sep. 2000.
- [39] J.-S. Mo, C. H. Han, and Y.-S. Kim, "A melody-based similarity computation algorithm for musical information," in *Proc. of Workshop on Knowledge and Data Engineering Exchange*, Illinois, Chicago, USA, pp.114-121, Nov. 1999.
- [40] Y. Zhu and D. Shasha, "Query by Humming: a time series database approach," in *Proc. of ACM Special Interest Group on Management of Data*, San Diego, California, USA, June 2003.
- [41] L. Guo, X. He, Y. Zhang, and Y. Lu, "Content-based retrieval of polyphonic music objects using pitch contour," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, Nevada, USA, pp.2205-2208, March-April 2008.
- [42] W. Jeon and C. Ma, "Efficient search of music pitch contours using wavelet transforms and segmented dynamic time warping," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech, pp.2304-2307, May 2011.
- [43] P.J. Giabbanelli, T. Torsney-Weir and V.K. Mago, "A fuzzy cognitive map of the psychosocial determinants of obesity," *Applied Soft Computing*, vol. 12, issue 12, pp.3711-3724, December 2012.

- [44] E.I. Papageorgiou and A. Kannappan, “Fuzzy cognitive map ensemble learning paradigm to solve classification problems: Application to autism identification,” *Applied Soft Computing*, vol. 12, issue 12, pp.3798-3809, December 2012.
- [45] M. Agarwal, K.K. Biswas and M. Hanmandlu, “Generalized intuitionistic fuzzy soft sets with applications in decision-making,” *Applied Soft Computing*, vol. 13, issue 8, pp.3552-3566, August 2013.
- [46] Y.-P. Huang and S.-L. Lai, “Novel query-by-humming/singing method with fuzzy inference system,” *Journal of Convergence*, vol. 3, no. 4, pp.1-7, Dec. 2012.