A Query-by-singing Technique for Retrieving Polyphonic Objects of Popular Music

Hung-Ming Yu, Wei-Ho Tsai, and Hsin-Min Wang

Institute of Information Science, Academia Sinica, Taipei, Taiwan, Republic of China
{donny, wesley, whm}@iis.sinica.edu.tw

Abstract. *This paper investigates the problem of retrieving popular music by singing. In contrast to the retrieval of MIDI music, which is easy to acquire the main melody by the selection of the symbolic tracks, retrieving polyphonic objects in CD or MP3 format requires to extract the main melody directly from the accompanied singing signals, which proves difficult to handle well simply using the conventional pitch estimation. To reduce the interference of background accompaniments during the main melody extraction, methods are proposed to estimate the underlying sung notes in a music recording by taking into account the characteristic structure of popular song. In addition, to accommodate users' unprofessional or personal singing styles, methods are proposed to handle the inaccuracies of tempo, pause, transposition, or off-key, etc., inevitably existing in queries. The proposed system has been evaluated on a music database consisting of 2613 phrases extracted manually from 100 Mandarin pop songs. The experimental results indicate the feasibility of retrieving pop songs by singing.*

1 Introduction

Currently, the most prevalent way to music information retrieval is based on the so-called metadata search, which operates by manually annotating music data according to the title, lyrics, performer, composer, etc., so that users can retrieve their desired music in the same way as they retrieve the text information. However, since the concrete descriptions, such as title or lyrics, usually cannot reflect the abstract content of music directly, it is often the case that users know what the song they want sounds like, but just cannot recall or totally have no idea about its title or lyrics. As a result, formulating a text query explicitly sometimes could be a difficulty for users. To overcome this handicap, a new promising solution is the so-called query-by-humming or query-by-singing [1-9], which allows users to retrieve a song by simply humming or singing a fragment of that song. Since no textual input is needed, query-by-humming or query-by-singing could not only increase the usability of a music retrieval system, but also allow the access to the system with no keyboard supported, e.g., to retrieve music via mobile devices.

The format of digital music can be divided into two categories. One is the symbolic representation based on musical scores. It specifies some manner of

instructions about what, when, and how long a note should be played with which instruments. Examples of this category include MIDI (Musical Instrument Digital Interface) and Humdrum. Since no real acoustic signal is included, a MIDI or Humdrum file would have different sounds when it is played by different devices. The second category of digital music is concerned with the data containing the acoustic signals recorded from real performances. The most widespread formats are CD (.wav) and MP3 (MPEG-I Layer 3). This type of music is often *polyphonic*, in which many notes may be played simultaneously, in contrast to *monophonic* music, in which at most one note is played at any give time. From the perspective of music retrieval, searching for a MIDI object from a database is much easier than searching for an MP3 object, because extracting score information is easy from a symbolic file, but is rather difficult from a polyphonic music. Due to this difficulty, research on query-by-humming or query-by-singing [1-6] has almost focused on MIDI music. However, methods, specifically designed to retrieve CD or MP3 music [7-9] are still very scarce and needed to be explored.

Previous work on query-by-humming primarily concentrates on the similarity comparison between the symbolic sequences. Ghias *et al.* [1] proposed an approximate pattern matching approach, which converts a query or each of the MIDI documents as a sequence of symbols 'U' (the note is higher than its preceding one), 'D' (the note is lower than its preceding one), and 'S' (the note is the same as its preceding one). The similarity between a query's sequence and each of the MIDI documents' sequences are then computed by string matching. Since most users are not professional singers, a query's sequence inevitably contains transposition errors (e.g., UUSDD $\rightarrow$ UDSDD), dropout errors (e.g., UUSDD $\rightarrow$ USDD), and duplication errors (e.g., UUSDD $\rightarrow$ UUUSDD). To tolerate the above errors, several methods have been further proposed, with the dynamic time warping (DTW) [4][5] being the most popular. Moreover, it is obvious that the three symbols 'U', 'D', and 'S' are not sufficient to represent all kinds of melody patterns precisely. Thus, more sophisticated representations, such as MIDI note number representation and broken-edge graph representation [4], have been studied subsequently. In addition, related work in [2][3] further considered the tone distribution in a song, tone transition between two adjacent notes, and the difference with respect to the first note.

In contrast to the retrieval of MIDI music, this study presents our first investigation on retrieving polyphonic objects of popular music. To permit the comparison between monophonic queries and polyphonic documents, methods of main melody extraction and error correction are proposed, with the statistical analysis of the compositional structure of pop songs being taken into account. In addition, to accommodate users' unprofessional or personal singing styles, methods are proposed to handle the inaccuracies of tempo, pause, transposition, or off-key, etc., inevitably existing in queries.

The rest of this paper is organized as follows. The general characteristics of popular music are discussed in Section 2. The configuration of our music retrieval system is introduced in Section 3. Our approaches for melody extraction and melody comparison are presented in Sections 4 and 5, respectively. Finally, the experimental results are discussed in Section 6 and conclusions are drawn in Section 7.

2 General Characteristics of Popular Music

Characteristic analysis of the data to be processed is an essential step in designing a reliable information retrieval system. It is known that popular music is simple by the melody that is easy to sing and memorize, but is, however, also complicated by the melody that is difficult to extract automatically. This section briefs some characteristics of popular music, which could be exploited to benefit the realization of a popular music retrieval system.

In general, the structure of a popular song can be divided into five sections:

1. *intro*, which is usually the first 10-20 seconds of a song, and simply an instrumental statement of the subsequent sections;
2. *verse*, which typically comprises the main content of story represented in a song's lyrics;
3. *chorus*, which is often the heart of a song where the most recognizable melody is present and repeated;
4. *bridge*, which comes roughly two-thirds into a song, where a key change, tempo change or new lyric is usually introduced to create a sensation of something new coming next;
5. *outro*, which is often a fading version of chorus or an instrumental restatement of some earlier sections to bring the song to a conclusion.

Except for intro and outro, each of the sections may repeat several times with varying lyrics, melodies, etc. The most common structure of a popular song consists of "intro-verse-chorus-verse-chorus-bridge-outro" or "intro-verse-verse-chorus-chorus-bridge-outro". In essence, verse and chorus contain the vocals sung by the lead singer, while intro, bridge, and outro are often largely accompaniments. This makes it natural that verse and chorus are the favorites that people go away humming when they hear a good song, and hence are often the query that a user may hum or sing to a music retrieval system.

Depending on the song, the notes produced by a singer may vary from F2 (87.3 Hz) to B5 (987.8Hz), corresponding to a varying range of 43 *semitones*[1]. However, it is observed that the sung notes within a music recording usually vary less than this range, and the varying range of the sung notes within a verse or chorus section can be even narrower. Fig. 1 shows an example of a segment of a song performed with MIDI[2]. It is clear that the range of notes within the verse can be distinguished from that of the chorus, because the sung notes within a section do not spread over all the possible notes, but only distribute in their own narrower range. An informal survey using 50 pop songs shows that the range of sung notes within a whole song and within a verse or chorus section are around 25 and 22 semitones, respectively. Fig. 2 details our statistic results. This information is useful for those attempting to transcribe the sung notes, by discarding the virtually impossible notes.

---

[1] A semitone is one twelfth part of the interval (called *octave*) between two sounds one of which has twice the frequency of the other.

[2] We convert the sung notes into MIDI note numbers for ease of illustration.

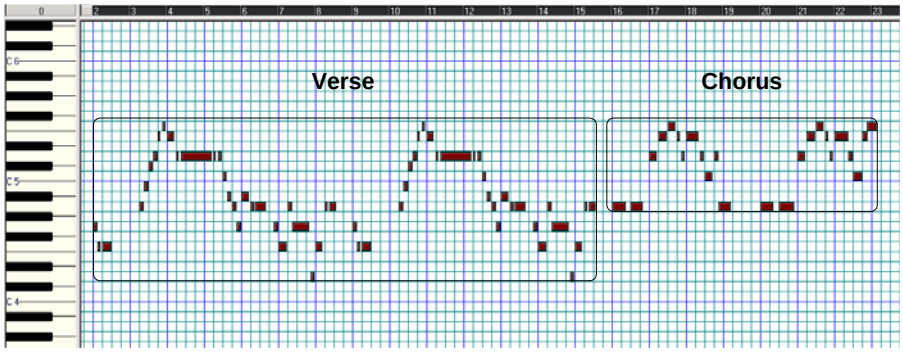
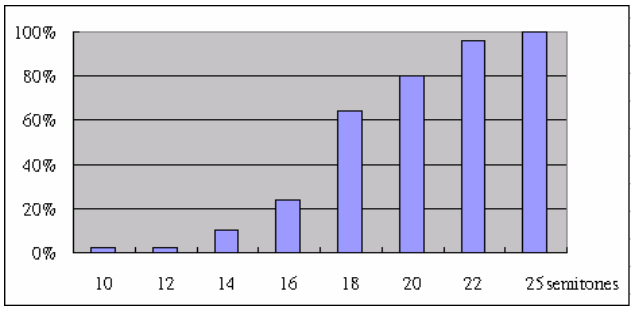
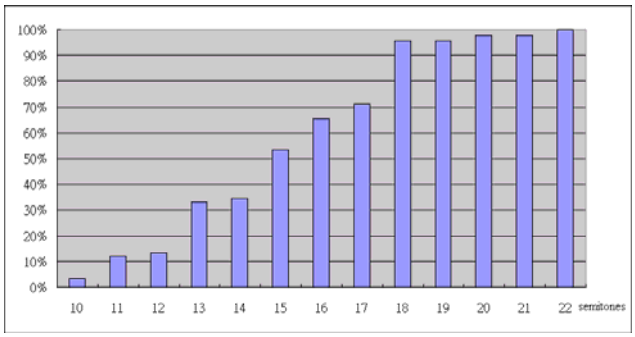


Fig. 1. A fragment of the pop song "Yesterday" by *The Beatles*, in which the singing is converted into a MIDI file and shown by software Cakewalk™ [11] for ease of illustration.



(a) The range of sung notes within a pop song



(b) The range of sung notes within a verse or chorus section

Fig. 2. Statistics of the range of sung notes in 50 pop songs.

In addition to the singing, a vast majority of popular music contain background accompaniment during most or all vocal passages. Various signals from different sources are mixed together into a single track in a CD. Even in stereo, signals in each of the channels are the accompanied voice, rather than the solo voice or accompaniment only. This makes it more difficult to design a system for retrieving CD music than to design a system for retrieving MIDI music, since the desired information, usually residing in the solo voice, is inextricably intertwined with the background signals. In addition, the background accompaniments often play notes several octaves above or below the singing, in order that the mix of music can sound harmonically. However, such harmonicity between singing voice and accompaniments further make the vocal melody notoriously difficult to extract. An example of song performed with MIDI is shown in Fig. 3. We can see from the notes indicated by arrows that a large proportion of sung notes are accompanied by the notes one or two octaves above them. Nevertheless, the harmonicity, viewed from another angle, may be exploited as a constraint in the determination of sung notes. A method based on this idea to improve the main melody extraction is discussed in a greater detail in Section 4.

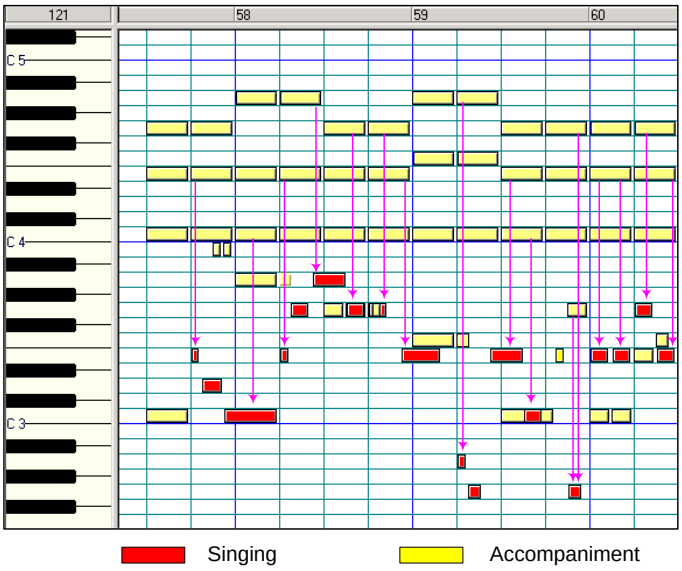


Fig. 3. A fragment of the pop song "Let It Be" by *The Beatles*, in which the tune is converted manually into a MIDI file.

3 System Configuration

Our popular music retrieval system is designed with such an aim to take as input an audio query sung by a user, and to produce as output the song containing the most similar melody to the sung query. Fig. 4 shows a block diagram of the retrieval system. It operates in two phases: indexing and searching.

The indexing phase is concerned with the generation of melody description for each of the songs in the collection. It starts with the segmentation of each song into *phrases* which reflect the expected patterns of query that users would like to sing to the system. In view of the fact that the length of a popular song is normally several minutes, it is virtually impossible that a user sings a whole song as a query to the system. Further, a user's singing tends to begin with the initial of a sentence of lyrics. For instance, a user may query the system by singing a piece of *The Beatle's* "Yesterday" like this, "Suddenly, I'm not half the man I used to be. There's a shadow hanging over me." By contrast, a sung query like "I used to be. There's a shadow" or "half the man I used to be." is believed almost impossible. Therefore, segmenting a song into semantically-meaningful phrases could not only match users' queries better, but also improve the efficiency of the system in the searching phase. Next, the second step of the indexing proceeds with the main melody extraction for each of the phrases. It converts an audio signal from the waveform samples into a sequence of musical note symbols. Accordingly, the database is composed of note-based sequences of phrase, referred to as documents' note sequences hereafter. During the initial design stage of this system, the phrase segmentation is performed manually.

In the searching phase, the system determines the song that a user looks for based on what he/she is singing. It is assumed that a user's sung query can be either a complete phrase or an incomplete phrase but always starts from the beginning of a phrase. The system commences with the end-point detection that records the singing voice and marks the salient pauses within the singing waveform. Next, the singing waveform is converted into a sequence of note symbols by using the main melody extraction module as in the indexing phase. Then, the retrieval task is narrowed down to a problem of comparing the similarity between the query's note sequence and each of the documents' note sequences. The song associated with the note sequence most similar to the query's note sequence is regarded as relevant and presented to the user.

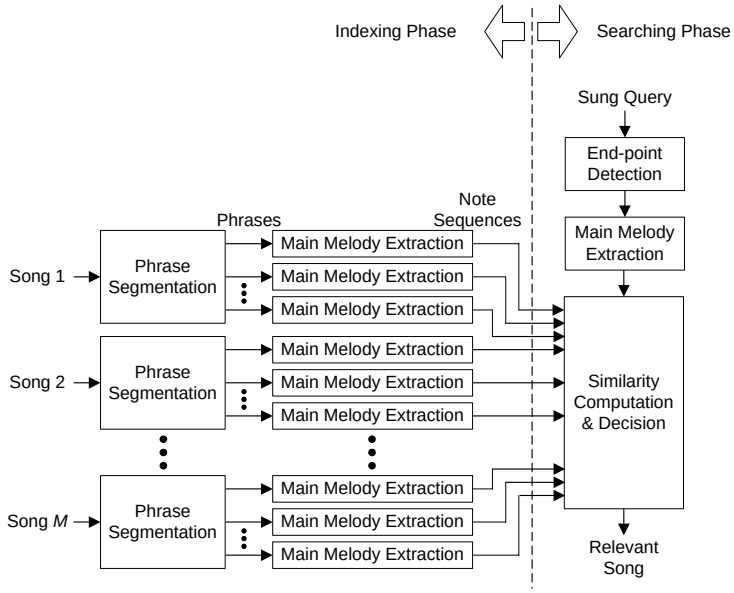


Fig. 4. The proposed popular song retrieval system.

4 Main Melody Extraction

Given a music recording, the aim of main melody extraction is to find the sequence of musical notes produced by the singing part of the recording. Let e_1, e_2,…, e_N be the inventory of possible notes performed by a singer. The task, therefore, is to determine which among N possible notes is most likely sung at each instant. To do this, the music signal is first divided into overlapping frames by using a fixed-length sliding Hamming window. Every frame then undergoes a fast Fourier transform (FFT) with size J. Since musical notes differ from each other by the fundamental frequencies (F0s) they present, we may determine if a certain note is sung in each frame by analyzing the spectral intensity in the frequency region where the F0 of the note is located.

Let $x_{t,j}$ denote the signal's energy with respect to FFT index j in frame t, where $1 \le j \le J$. If we use the MIDI note number to represent e_1, e_2,…, e_N, and map the FFT indices into MIDI note numbers according to the F0 of each note, the signal's energy on note e_n in frame t can be estimated by

$$y_{t,n} = \max_{\forall j, U(j)=e_n} x_{t,j}\,, \tag{1}$$

and

$$U(j) = \left\lfloor 12 \cdot \log_2\left(\frac{F(j)}{440}\right) + 69.5 \right\rfloor, \tag{2}$$

where $\lfloor\ \rfloor$ is a floor operator, $F(j)$ is the corresponding frequency of FFT index j, and $U(\cdot)$ represents a conversion between the FFT indices and the MIDI note numbers.

Ideally, if note e_n is sung in frame t, the resulting energy, $y_{t,n}$, should be the maximum among $y_{t,1}$, $y_{t,2}$,…, $y_{t,N}$. However, due to the existence of harmonics, the note numbers that are several octaves higher than the sung note can also receive a large proportion of the signal's energy. Sometimes the energy on a harmonic note number can be even larger than the energy on the true sung note number; hence, the note number receiving the largest energy is not necessarily what is sung. To determine the sung note more reliably, this study adapts Sub-Harmonic Summation (SHS) [10] to this problem.

The principle applied here is to compute a value for the "strength" of each possible note by summing up the signal's energy on a note and its harmonic note numbers. Specifically, the strength of note e_n in frame t is computed using

$$z_{t,n} = \sum_{c=0}^{C} h^c \, y_{t,\, n+12c} \,, \tag{3}$$

where C is the number of harmonics that are taken into account, and h is a positive value less than 1 to discount the contribution of higher harmonics. The result of this summation is that the note number corresponding to the signal's F0 will receive the largest amount of energy from its harmonic notes. Thus, the sung note in frame t could be determined by choosing the note number associated with the largest value of the strength, i.e.,

$$o_t = \arg\max_{1 \le n \le N} \; z_{t,n} \,. \tag{4}$$

However, since popular music usually contains background accompaniments during the vocal passages, the note number associated with the largest value of strength may not be produced by a singer, but the concurrent instruments instead. As a consequence, whenever the strength of the sung note is not the maximum, an error estimation of the sung note would happen. This problem may be alleviated by using the *tone chroma*, which maps all the notes into 12 tone classes (C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, and B) by ignoring the difference between octaves. As mentioned in Section 2, since the background accompaniments often play notes several octaves above or below the singing, a mis-estimated sung note could still map to a correct tone class. However, because of using 12 classes only, tone chroma cannot express a melody pattern with sufficient precision to distinguish from one another. Recognizing this, we focus on investigating the method to correct the error estimation of the sung notes, instead of using the tone chroma representation.

The method to correct the error estimation of the sung notes is based on a concept of rectification, which identifies the abnormal individuals in a note sequence and forces them back to the normal. The abnormality in a note sequence roughly arises from two types of errors: short-term error and long-term error. The short-term error is concerned with the rapid changes, e.g. jitters, between adjacent frames. This type of error could be amended by using the median filtering, which replaces each note of frame with the local median of its neighboring frames. On the other hand, the long-term error is concerned with a succession of the estimated notes not produced by a singer. These successive wrong notes are very likely several octaves above or below the true sung notes, which could result in the range of the estimated notes within a

sequence being wider than that of the true sung note sequence. As mentioned in Section 2, the sung notes within a verse or chorus section usually vary no more than 22 semitones. Therefore, we may adjust the suspect notes by shifting them several octaves up or down, so that the range of the notes within an adjusted sequence can conform to the normal range. Specifically, let $\mathbf{o} = \{o_1, o_2, \ldots, o_T\}$ denote a note sequence estimated using Eq. (4). An adjusted note sequence $\mathbf{o}' = \{o'_1, o'_2, \ldots, o'_T\}$ is obtained by

$$o'_t = \begin{cases} o_t & , \quad \text{if } |o_t - \bar{o}| \le (R/2) \\ o_t - 12 \times \left\lfloor \dfrac{o_t - \bar{o} + R/2}{12} \right\rfloor, & \text{if } o_t - \bar{o} > (R/2) \\ o_t - 12 \times \left\lfloor \dfrac{o_t - \bar{o} - R/2}{12} \right\rfloor, & \text{if } o_t - \bar{o} < (-R/2) \end{cases} , \tag{5}$$

where R is the normal varying range of the sung notes in a sequence, say 22, and $\bar{o}$ is the mean note computed by averaging all the notes in $\mathbf{o}$. In Eq. (5), a note o_t is considered as a wrong note and needs to be adjusted, if it is too far away from $\bar{o}$, i.e., $|o_t - \bar{o}| > R/2$. The adjustment is done by shifting the wrong note $\lfloor (o_t - \bar{o} + R/2)/12 \rfloor$ or $\lfloor (o_t - \bar{o} - R/2)/12 \rfloor$ octaves.

5 Melody Similarity Comparison

Given a user's query and a set of music documents, each of which is represented by a note sequence, our task here is to find a music document whose note sequence is most similar to the query's note sequence. Since users' singing may be significantly different from what they want to retrieve in terms of key, tempo, ornamentation, etc., it is impossible to find a document's sequence exactly match the query's sequence. Moreover, the main melody extraction is known to be frequently imperfect, which further introduces errors of substitution, deletion, and insertion into the note sequences. To perform a reliable melody similarity comparison, an approximate matching method tolerable to occasional note errors, is therefore needed.

Let $\mathbf{q} = \{q_1, q_2, \ldots, q_T\}$, and $\mathbf{u} = \{u_1, u_2, \ldots, u_L\}$ be the note sequences extracted from a user's query and a particular music document to be compared, respectively. The most apparent problem we face here is that the lengths of $\mathbf{q}$ and $\mathbf{u}$ are usually unequal. Thus, it is necessary to temporally align $\mathbf{q}$ and $\mathbf{u}$ before computing their similarity. For this reason, we apply Dynamic Time Warping (DTW) to find the mapping between each q_t and u_ℓ, $1 \le t \le T$, $1 \le \ell \le L$. DTW constructs a $T \times L$ distance matrix $\mathbf{D} = [D(t, \ell)]_{T \times L}$, where $D(t, \ell)$ is the distance between note sequences $\{q_1, q_2, \ldots, q_t\}$ and $\{u_1, u_2, \ldots, u_\ell\}$, computed using:

$$D(t, \ell) = \min \begin{cases} D(t-2, \ell-1) + 2 \times d(t, \ell) \\ D(t-1, \ell-1) + d(t, \ell) - \varepsilon \\ D(t-1, \ell-2) + d(t, \ell) \end{cases} , \tag{6}$$

and

$$d(t, \ell) = | q_t - u_\ell | , \tag{7}$$

where ε is a small constant that favors the mapping between note q_t and u_ℓ, given the distance between note sequences $\{q_1, q_2,\ldots,q_{t-1}\}$ and $\{u_1, u_2,\ldots, u_{\ell-1}\}$. The boundary conditions for the above recursion are defined by

$$\begin{cases} D(1,1) = d(1,1) \\ D(t,1) = \infty,\ 2 \le t \le T \\ D(1,\ell) = \infty,\ 2 \le \ell \le L \\ D(2,2) = d(1,1) + d(2,2) - \varepsilon \\ D(2,3) = d(1,1) + d(2,3) \\ D(3,2) = d(1,1) + 2 \times d(3,2) \\ D(t,2) = \infty,\ 4 \le t \le T \\ D(2,\ell) = \infty,\ 4 \le \ell \le L \end{cases}, \tag{8}$$

where we have assumed that a sung query always starts from the beginning of a document. After the distance matrix **D** is constructed, the similarity between **q** and **u** can be evaluated by

$$S(\mathbf{q}, \mathbf{u}) = \begin{cases} \max\limits_{T/2 \le \ell \le \min(2T, L)} \left[1/D(T, \ell)\right], & \text{if } L \ge T/2 \\ 0, & \text{if } L < T/2 \end{cases}, \tag{9}$$

where we assume that the end of a query's sequence should be aligned to a certain frame between $T/2$ and $\min(2T,L)$ of the document's sequence, and assume that a document whose length of sequence less than $T/2$ would not be a relevant document to the query.

Since a query may be sung in a different key or register than the target music document, i.e., the so-called *transposition*, the resulting note sequences of the query and the document could be rather different. To deal with this problem, the dynamic range of a query's note sequence needs to be adjusted to that of the document to be compared. This could be done by shifting the query's note sequence up or down several semitones, so that the mean of the shifted query's note sequence is equal to that of the document to be compared. Briefly, a query's note sequence is adjusted by

$$q_t \leftarrow q_t + (\bar{u} - \bar{q}), \tag{10}$$

where $\bar{q}$ and $\bar{u}$ are the means of the query's note sequence and the document's note sequence, respectively. However, our experiments find that the above adjustment can not fully overcome the transposition problem, since the value of $(\bar{q} - \bar{u})$ can only reflect a global difference of key between a query and document, but cannot characterize the partial transposition or key change over the course of a query. To handle this problem better, we further modify the DTW similarity comparison by considering the key shifts of a query's note sequence. Specifically, a query sequence **q** is shifted with ±1, ±2,..., ±*K* semitones to span a set of note sequences $\{\mathbf{q}^{(1)}, \mathbf{q}^{(-1)}, \mathbf{q}^{(2)}, \mathbf{q}^{(-2)},\ldots, \mathbf{q}^{(K)}, \mathbf{q}^{(-K)}\}$. For a document sequence **u**, the similarity $S(\mathbf{q}, \mathbf{u})$ is then

determined by choosing one among {$\mathbf{q}^{(0)}$, $\mathbf{q}^{(1)}$, $\mathbf{q}^{(-1)}$, $\mathbf{q}^{(2)}$, $\mathbf{q}^{(-2)}$,…, $\mathbf{q}^{(K)}$, $\mathbf{q}^{(-K)}$} that is most similar to **u**, i.e.,

$$S(\mathbf{q},\mathbf{u}) = \max_{-K\le k\le K} S(\mathbf{q}^{(k)},\mathbf{u}), \tag{11}$$

where $\mathbf{q}^{(0)} = \mathbf{q}$.

In addition to the difference of key and tempo existing between queries and documents, another problem needed to be addressed is the existence of voiceless regions in a sung query. The voiceless regions, which may arise from the rest, pause, etc., result in some notes being tagged with "0" in a query's note sequence. However, the corresponding non-vocal regions in the document are usually not tagged with "0", because there are accompaniments in those regions. This discrepancy may severely discount the similarity $S(\mathbf{q}, \mathbf{u})$ for any **q** and **u** having the same tune. Fig. 5 shows an example illustrating this problem. The regions in Fig. 5(b) marked in gray are those do not contain singing voice. Although the voiceless regions in a sung query can be detected by simply using the energy information, the accurate detection of non-vocal regions in a music document remains a very difficult problem. Therefore, to sidestep this problem, we further modify the computation of $d(t,\ell)$ in Eq. (7) by

$$d(t,\ell) = \begin{cases} |q_t - u_\ell|, & q_t \neq 0 \\ \varphi \quad , & q_t = 0 \end{cases}, \tag{12}$$

where φ is a small constant. Implicit in Eq. (12) is equivalent to bypassing the voiceless regions of a query.

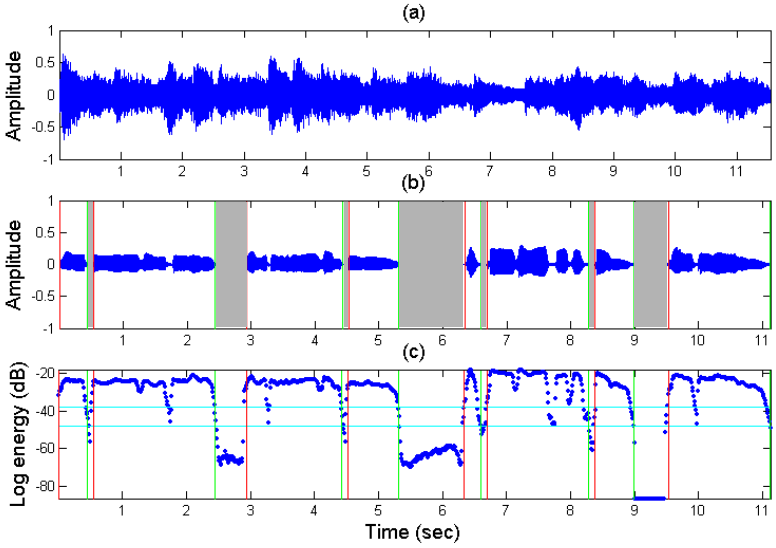


Fig. 5. (a) a phrase document, (b) a query sung according to this phrase, (c) the log-energy profile of this sung query.

6 Experiments

6.1 Music database

The music database used in this study consisted of 100 tracks[3] from Mandarin pop music CDs. Each of the tracks was segmented manually into several phrases, which gives a total of 2,613 phrase documents. The waveform signal of each phrase document was down-sampled from the CD sampling rate of 44.1 kHz to 22.05 kHz, to exclude the high frequency components that usually contain sparse vocal information. In addition, we collected 253 queries sung by 5 male and 2 female users. Each query is sung according to one of the 2,613 phrase documents, but can be an incomplete phrase.

Performance of the song retrieval was evaluated on the basis of phrase accuracy and song accuracy. The phrase accuracy is defined as the percentage of the queries that can receive their corresponding phrase documents, i.e.,

$$\text{Phrase accuracy}(\%) = \frac{\#\text{ queries receiving the corresponding phrase documents}}{\#\text{ queries}} \times 100\%.$$

In addition, considering a more user-friendly scenario that a list of phrase documents ranked according to the query-document similarity can be provided for users' choices, we also computed the Top-N phrase accuracy defined as the percentage of the queries whose corresponding phrase documents are among Top-N.

The song accuracy reflects the fact that some of the phrase documents belong to the same song, and what a user would like to retrieve is a song instead of a phrase. It is computed by

$$\text{Song accuracy}(\%) = \frac{\#\text{ queries receiving the corresponding songs}}{\#\text{ queries}} \times 100\%.$$

We also computed the Top-N song accuracy defined as the percentage of the queries whose corresponding songs are among Top-N.

6.2 Experimental results

Our first experiment was conducted to evaluate the performance of song retrieval with respect to the potential enhancement of the main melody extraction. Specifically, we compared the three methods to main melody extraction, namely, the note sequence generation by Eq. (4) along with the six-frame median filtering, the conversion of note sequences to tone chroma sequences, and the note sequence rectification by Eq. (5). The inventory of possible sung notes consisted of the MIDI numbers from 41 to 83, which corresponds to the frequency range of 87 to 987 Hz. The melody similarity comparison in this experiment was performed on the basis of Eqs. (9) and (10). Table 1 shows the retrieval results. We can see from Table 1 that

[3] The database did not contain the 50 pop songs used for analyzing the range of sung notes, described in Section 2.

the retrieval performance obtained with the method of using Eq. (4) and median filtering was the worst among the three methods compared, mainly because this method determines the sung notes based on the largest values of strength, which is vulnerable to the interference of background accompaniments. It is also shown in Table 1 that a slightly better performance can be achieved by converting note sequences into tone chroma sequences, which avoids the risk of mis-estimating a sung note as its octaves. However, due to the limited precision in melody representation, the tone chroma method has its inherent limit in distinguishing among songs, and so in the retrieval performance. By contrast, the note sequence rectification by Eq. (5) keeps the fine precision of using note numbers in melody representation and tries to correct the errors in a note sequence. We can see from Table 1 that the note sequence rectification noticeably improves the retrieval performance, and proves superior to the tone chroma method.

Table 1. Performance of the song retrieval for different main melody extraction methods.

Main melody extraction method		Phrase accuracy / Song accuracy (%)		
		Top 1	Top 3	Top 10
Note sequence generation by Eq. (4) and six-frame median filtering		32.0 / 37.9	41.1 / 49.8	50.6 / 62.9
Conversion of note sequences to tone chroma sequences		36.8 / 45.1	45.9 / 55.7	54.2 / 68.4
Note sequence rectification by Eq. (5)	$R = 16$	40.3 / 45.9	48.6 / 61.3	60.1 / 72.3
	$R = 18$	42.7 / 49.4	49.4 / 62.5	60.5 / 72.7
	$R = 20$	39.9 / 49.0	47.0 / 59.7	57.7 / 71.9
	$R = 22$	37.6 / 46.3	46.3 / 59.3	54.6 / 70.4

Next, we examined if the retrieval performance can be improved by further addressing the transposition problem. Specifically, we used the method of shifting a query's note sequence upward or downward several semitones together with Eq. (11) to perform the similarity comparison with each of the documents' sequences. Table 2 shows the experimental results. Here, $K = 0$ means that no shifting is performed, and its result corresponds to the best result (note sequence rectification with $R = 18$) shown in Table 1. We can see from Table 2 that the retrieval performance improves as the value of K increases, which indicates that the more the possible changes of key is taken into account, the greater the chance that a query's sequence matches the correct document's sequence. However, increasing the value of K heavily increases the computational cost, because the similarity comparison requires two extra DTW operations whenever the value of K is increased by one. An economic value of $K = 1$ was thus chosen in our subsequent experiments.

Table 2. Performance of the song retrieval obtained with and without upward/downward shifting a query's note sequence during the DTW similarity comparison.

Value of K in Eq. (11)	Phrase accuracy / Song accuracy (%)		
	Top 1	Top 3	Top 10
0	42.7 / 49.4	49.4 / 62.5	60.5 / 72.7
1	47.0 / 56.1	59.3 / 70.0	66.8 / 77.9
2	48.6 / 58.1	60.5 / 71.5	68.4 / 78.7

Finally, we compared the retrieval performance obtained with and without explicitly considering the singing pause of a query, that is, Eq. (7) vs. Eq. (12). The experimental results are shown in Table 3. It is clear that the retrieval performance can benefit greatly by detecting and excluding the non-singing segments of a query during the DTW similarity comparison. This indicates that the proposed system is capable of handling the inadequate pause, key-shifting, or tempo of a sung query. In summary, our experimental results show that whenever a user sings a query to search for one of the one hundred songs, the probability that the desired song can be found in a Top-10 list is around 0.8, in a Top-3 list is around 0.7, and in a Top-1 list is around 0.6. Although there is much room to further improve, our system shows the feasibility of retrieving polyphonic pop songs in a query-by-singing framework.

Table 3. Performance of the song retrieval obtained with and without explicitly considering the singing pause of a query

	Phrase accuracy / Song accuracy (%)		
	Top 1	Top 3	Top 10
DTW with Eq. (7)	47.0 / 56.1	59.3 / 70.0	66.8 / 77.9
DTW with Eq. (12)	52.6 / 60.5	62.5 / 71.9	72.7 / 80.6

7 Conclusions

This study has presented a popular song retrieval system that allows users to search for their desired songs by singing. Since in most pop songs, the singing voices and various concurrent accompaniments are mixed together into a single track, the melody extraction process can be seriously interfered by the accompaniments, leading to the inevitable errors. Drawn from the observations that the varying range of the sung notes within a verse and chorus section is usually less than 22 semitones and a large proportion of sung notes are accompanied by the notes several octaves above or below them, we have developed a feasible approach to melody extraction and error

correction. Meanwhile, we have also devised a similarity comparison method based on DTW to handle the discrepancy of tempo variation, pause, transposition between queries and documents.

With regard to practicability, more work is needed to extend our current system to handle a wider variety of queries and songs. Specifically, the current system assumes that a query can be either a complete phrase or an incomplete phrase of a song, and a query must start from the beginning of a phrase. It is necessary to further address the case when a query contains multiple phrases of a song or when a query does not start from the beginning of a phrase. In addition, methods for automatic segmentation of songs into phrases are needed in order to automate the whole indexing process. Furthermore, our future work will incorporate some sophisticated methods in the general document-retrieval field, such as relevance feedback, to improve the current system.

8 Acknowledgement

This work was supported in part by the Nation Science Council, Taiwan, under Grants NSC92-2422-H-001-093 and NSC93-2422-H-001-0004.

Reference

1. Ghias, A., H. Logan, D. Chamberlin, and B. C. Smith, "Query by Humming: Musical Information Retrieval in an Audio Database," *Proc. ACM International Conference on Multimedia*, pp. 231-236, 1995.
2. Kosugi, N., Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima, "Music Retrieval by Humming," *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 404-407, 1999.
3. Kosugi, N., Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima, "A Practical Query-By-Huming System for a Large Music Database," *Proc. ACM International Conference on Multimedia*, 2000.
4. Mo, J. S., C. H. Han, and Y. S. Kim, "A Melody-Based Similarity Computation Algorithm for Musical Information," *Proc. Workshop on Knowledge and Data Engineering Exchange*, pp. 114-121, 1999.
5. Jang, J. S. Roger, and H. R. Lee, "Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input," *Proc. ACM International Conference on Multimedia,* pp. 401-410, 2001.
6. Liu, C. C., A. J. L. Hsu, and A. L. P. Chen, "An Approximate String Matching Algorithm for Content-Based Music Data Retrieval," *Proc. IEEE International Conference on Multimedia Computing and Systems*, 1999.
7. Nishimura, T., 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," *Proc. International Symposium on Music Information Retrieval*, 2001.
8. Doraisamy, S., and S. M. Ruger, "An Approach Towards a Polyphonic Music Retrieval System," *Proc. International Symposium on Music Information Retrieval*, 2001.

9. Song, J., S. Y. Bae, K. Yoon, “Mid-Level Music Melody Representation of Polyphonic Audio for Query-by-Humming System,” *Proc. International Conference on Music Information Retrieval*, 2002.
10. Piszczalski, M., and B. A. Galler, “Predicting musical pitch from component frequency ratios,” *Journal of the Acoustical Society of America*, 66(3), pp. 710–720, 1979.
11. Cakewalk, Inc., http://www.cakewalk.com/