CONTENT-BASED MUSIC RETRIEVAL BY ACOUSTIC QUERY

ZHU YONGWEI
(M.Eng., NTU)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2004

ACKNOWLEDGEMENTS

I really feel lucky that Prof Mohan Kankanhalli has been my PhD supervisor for the 4 plus years. Without his inspiration, guidance, assistance and patience, I cannot imagine how my part-time PhD program could be completed. I would like to thank Prof Kankanhalli for allowing me to choose the research topic, music retrieval, which is of my interest. I would also thank him for giving me the valuable advice and help on how to do research and complete a PhD thesis.

I would also thank my boss and colleague, Dr Sun Qibin, for the support, urge and encouragement of the PhD work. I would like to acknowledge the help offered by many people: Prof Wu Jiankang, Prof Tian Qi, Dr. Xu Changsheng, Li Li, Li Zhi and many others.

This thesis is dedicated to my wife, Shu Yin, for the understanding and encouragement during the years. Her music knowledge has let me put much less effort in acquiring the music background knowledge of this thesis. This thesis is also dedicated to my parents Zhu Deyao and Liu Yuhuang, for their encouragement and inspiration for pursuing the PhD.

TABLE OF CONTENTS

SUMMARY

This thesis deals with content-based music retrieval using humming melody queries. The problems which have been tackled in this work are representation, extraction and matching of melodies of either music files or acoustic queries, considering inevitable errors or variations in the humming signals. The objectives of the work have been to develop a robust, effective and efficient methodology for content-based music retrieval by humming queries. The major contributions of the thesis are as follows.

A novel melody representation, called pitch line, has been proposed. Pitch line has been shown to be sufficient, efficient and robust for representing melody from humming based inputs.

A pitch extraction method has been proposed to reliably convert a humming signal into the proposed melody representation. The method has been shown to be robust against variations in note articulation and vocal conditions of different users.

A general melody matching approach for pitch line has been proposed. The melody matching approach consists of three distinct processes: key transposition, local melody alignment and similarity measure. This general approach has been shown to be more robust and efficient than the existing methods.

Melody similarity measures using pitch line, defined for particular key transposition and melody alignment, have been proposed. The geometrical similarity measures have been designed with considerations to minimize the effect of pitch inaccuracies in the humming melody input.

A melody alignment and transposition using global melody feature, called melody slope, has been proposed. Efficient melody alignment and key transposition has been achieved by using

melody slope sequence matching. This process also acts as a filtering step that can efficiently reject wrong candidates in the matching of melodies.

A melody matching technique using melody skeleton, an abstract of the melody representation, has been proposed to address the inconsistent tempo variation and occasional large pitch errors in the hummed melody. A novel point sequence matching using dynamic programming has been proposed for melody skeleton matching. This technique has been shown to be very robust against tempo variations and large pitch errors.

A melody scale and root estimation method has been proposed to assist melody matching. The scale root is estimated by using a scale modelling approach. The scale root of melody has been used for key transposition without local alignment in melody comparison, which has greatly reduced the computation requirement of melody matching, and improved the efficiency of music retrieval.

Extensive experiments have been conducted to evaluate the performance of the proposed techniques. The evaluation comprised pitch extraction, melody matching by melody slopes, melody matching by melody skeleton, and melody matching using scale root estimation. A comparison of the proposed methods with an existing method has also been presented.

A research prototype system based on the proposed methods has been developed, which has facilitated evaluation of the proposed techniques and demonstrated the feasibility of commercial applications on music retrieval by humming.

LIST OF FIGURES

LIST OF TABLES

Chapter 1. INTRODUCTION

1.1. Motivation

There has been a trend of growing availability of music in digital form due to the advancement of digital technology in the last 2 decades. In 1982, Philips and Sony developed the Compact Disc Digital Audio (CD-DA) standard, and for the first time, digital music reached consumers. In 1992, MPEG-1 Audio Layer III (MP3) for digital audio compression became an international standard. From about 1995, MP3-encoded music swept the world along with the spread of the Internet. Today, plenty of commercial websites are offering music downloading services. People can now easily create music collections from CDs or from the Internet and conveniently playback the music with numerous hardware or software players. In recent years, there have been efforts in some organizations to build digital libraries of music, in which large volumes of music are available for people to utilize for education and research purposes.

Nevertheless, the enormous and growing availability of music data has created difficulties for people to manage such data. A traditional way to organize the music data is to use text information of music, such as the music title, the composer or performer's name, the album title, and etc. However, the text information alone is inadequate for effective music retrieval (or search). For example, the text labels can be absent or incomplete in the corpus of music data, or in a query users may fail to recall the text labels of the music items that they are searching for.

Very often, people need to search music by the musical content, such as the tune, which is more intuitive and convenient than by using the text labels. A musicologist or a musician may want to use a few bars of music score to find any similar music pieces in the database or to

find out whether a composition is original. A layperson may just hum a tune and let the system identify the song by searching a melody database. This new approach is called content-based music retrieval.

Content-based music retrieval involves many aspects of music content, and a user can produce a musical query in many different ways, such as writing out the music scores, tapping a rhythm, playing a keyboard instrument, or singing out the tune. Melody has a key role in content-based music retrieval, since melody is very characteristic for music pieces and it is very convenient for a person to produce a tune by humming. Furthermore, a large amount of melody information is available in standard data encoding formats, like MIDI.

Content-based music retrieval imposes many technical challenges, and it has attracted much interest of researchers in the recent years. Some of the challenges are: how to represent the music content that can be extracted from music data and queries; how to perform matching of the query with the music data and how to cope with the errors in the queries; how to handle the variations in melody, such as key transposition and tempo change; and etc. These challenges have motivated the research work of this thesis.

1.2. Background

Music is the art and science of organizing sounds produced by instruments or human voice. A musical sound usually has some identifiable regularity, such as the tone/pitch. A music piece comprises a succession of musical sounds, which have relations and structures, such as rhythm, chord and melody. The study of the perceptual properties of musical sounds and their relations in the history has resulted in music theories. Understanding music theory is essential to the analysis of music content.

Since we are dealing with digital music, the data encoding format of digital music is also important in content-based music retrieval.

This section introduces the background knowledge on the relevant music terminology and the data encoding formats of digital music.

1.2.1. Music Terminology

Musical sound is the sound with an identifiable tone or pitch, as opposed to noise that has no pitch. ***Pitch*** characterizes the human perception of physical vibration in the sound, and can be associated with frequency (measured in Hertz). Human ears discern the difference between two tones by the ratio of the two frequencies (F_1 / F_2), instead of the absolute frequency difference ($|F_1 - F_2|$) [Croc_Hand]. And this is why octaves and semitones are used to measure and specify pitches. One ***octave*** corresponds to a frequency ratio of 2, and the audible frequency range (20Hz – 20kHz) can be divided into about 10 octaves. One octave is subdivided into 12 ***semitones***, and the frequency ratio for one semitone is $\sqrt[12]{2} \approx 1.059463$. One semitone can be further divided to 100 ***cent***, which is the smallest unit of pitch difference. The difference of any 2 pitches is called the ***pitch interval***.

Each musical sound in a music composition is called a ***note***. A music note has a few properties: pitch, duration, loudness, and timbre. The pitch value of a note corresponds to a note name. The note names are conventionally called: A, A#, B, C, C#, D, D#, E, F, F#, G, G#. The note names are defined in one octave and repeated in all octaves. The pitch interval of any two adjacent notes is one semitone. The "A" note in a middle octave has the pitch of 440Hz, and so the pitches of all the notes in any octaves are exactly defined. The duration of a note is measured in ***beats*** or fractions of a beat. The exact time of one beat, however, depends on the ***tempo***, which is measured in beats per minute (***BPM***). The loudness of a note is how loud the note is sounded. The ***timbre*** is the acoustic quality characterised by the energy distribution in the power spectrum, which depends on the instrument that plays out the note. In one piece of

music, there is usually a repeated pattern in the loudness of the notes, e.g. “strong, weak, sub-strong, weak”, which is called ***rhythm***.

A music piece is built up by a certain combinations of music notes and silence. If a music piece contains notes played simultaneously, it is called ***polyphonic***. Otherwise if each note is played once at a time, then it is called ***monophonic***.

A music ***scale*** is a sequence of notes in ascending or descending order. The diatonic scale is a seven note scale and is most familiar as the major and minor scale. A ***key*** defines the diatonic scale which a piece of music uses. A combination of simultaneous notes in the scale, which exhibit auditory coherence, is called ***chord***. A succession of monophonic notes in the scale, which contain change of some kind and be perceived as a single entity, is called a ***melody***. A popular song usually consists of a combination of chords and melody.

Although all pieces are composed using a specific key, changing the pitch level of the key will not change the structure or meaning of the music. For example, a tune played by a violin and a cello will have different keys, however the tunes will be perceived as the same.

The tempo of a music piece is usually not exact. The tempo can be different between different performances or be inconsistent during one performance. During a live performance, the tempo is controlled by the concert director, or the drum player.

1.2.2. Digital Music Formats

Computerized processing of music, such as retrieval, can be done only after the music is in a digital format. There are mainly 2 types of digital music format: digitalized acoustic signals of the music audio (digitised audio), and the Musical Instrument Digital Interface (MIDI).

Digitised audio is produced by capturing the acoustic signals of the music through analogue-to-digital conversion (ADC). The original music sound can be reproduced by digital-to-

analogue conversion (DAC). The digitised audio format has a sampling frequency and a number of bits to represent each quantized sample. The higher the sampling frequency and the more the bits of quantization, the higher fidelity the digital audio can reproduce the original musical sounds. There are many data formats for digitized audio. The waveform (.wav format) is one of the commonly used formats of uncompressed signal data. And popular lossy audio compression format is MPEG 1 Audio Layer 3 (MP3).

MIDI is a standard for communication between electrical musical instruments, which generate sounds by synthesising. Rather than encoding the acoustic signals of the music, MIDI encodes the symbols of events. The events are generated by the actions of the performer of the music, such as what note is pressed at what time using what intensity (speed and pressure). During the performance, these action events are sent as messages to the synthesiser, which can sound as piano, a guitar or any other types of instruments. The MIDI messages can be saved as data files on computer, which can be played back with a synthesiser on a sound card. Apart from playing an electrical instrument, one can also compose music in MIDI using software on a computer.

Digitised audio has the advantages in preserving the fidelity of the original sounds of the performance, which is suitable for live concert and studio recording. However, digitised audio contains very little information about the musical structures or notions, such as notes, harmonics or melody. And it usually takes large storage space, which is also true for compressed audio (like MP3) compared with MIDI.

On the contrary, MIDI music contains music symbols, which can be parsed and understood by computer. And a MIDI file is very compact compared with digital waveform. A disadvantage of MIDI is that the synthesiser may not be able to reproduce exactly the original sounds. Furthermore, MIDI cannot accurately represent the performance of mechanical instruments and particularly human singing voice.

In this thesis, we refer digitised audio as acoustic music and MIDI as symbolic music.

1.3. Scope

A content-based music retrieval system has a structure as illustrated in figure 1-1. The system extracts the music features of any incoming digital music files, and stores the features together with the music files into the database. When a user issues a query, the system extracts music features from the query using a similar feature extraction process. The music feature of query is then compared to those features stored in the database by feature matching. The music features that have high matching scores are returned as the retrieval results.

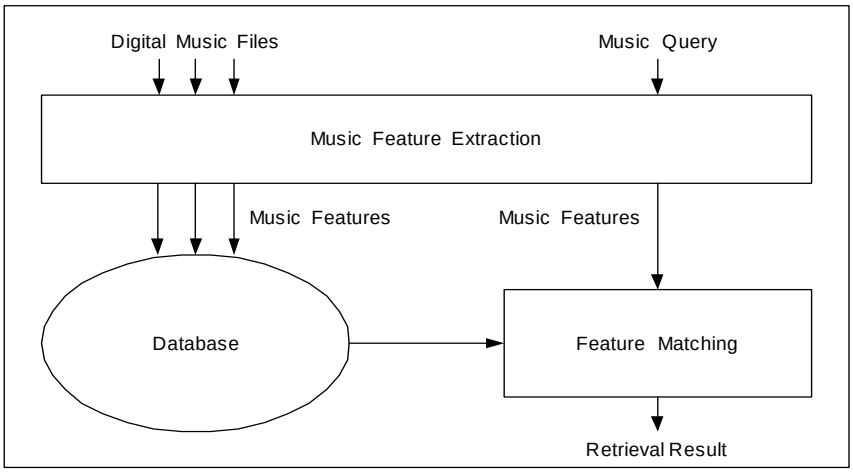

Figure 1-1: Block diagram of a content-based music retrieval system

As introduced in section 1.2, music data can be in acoustic format or symbolic format and music content can be polyphonic or monophonic. According to the data formats and content types of music data or queries, content-based music retrieval system can be classified to sixteen different categories, as shown in Table 1.1.

Table 1-1: Classification of content-based music retrieval systems

Music Data \ Query		Symbolic		Acoustic	
		Monophonic	Polyphonic	Monophonic	Polyphonic
Symbolic	Monophonic	**MR1**	*MR2*	**MR3**	*MR4*
	Polyphonic	*MR5*	*MR6*	*MR7*	*MR8*
Acoustic	Monophonic	**MR9**	*MR10*	**MR11**	*MR12*
	Polyphonic	*MR13*	*MR14*	*MR15*	*MR16*

MR1 stands for a music retrieval system, which uses monophonic MIDI for both the music data in the database and the queries.

MR3 is a monophonic MIDI retrieval system that can accept monophonic acoustic input as queries, such as humming a tune. MR3 is more attractive than MR1 for a lay user, since the requirement for music skills is minimal.

MR9 and MR11 are systems that can search for monophonic acoustic music, such as pure vocal singing music.

All the other types of retrieval system involve polyphonic music. If polyphonic MIDI music can be converted to monophonic MIDI by doing a melody extraction, some types of music retrieval systems can be simplified: MR2, MR5 and MR6 can be reduced to MR1; MR7 can be reduced to MR3; MR10 can be reduced to MR9.

However, music retrieval systems MR4, MR8, MR12, MR13, MR14, MR15 and MR16 involve polyphonic acoustic music, which will need an acoustic analysis technique that can

identify music notes [Bell_ISMIR00][Brow_JAS91]. The state-of-the-art in computer audition or auditory scene analysis is still far from being able to have a reliable melody or note extraction technique. Therefore, retrieval of polyphonic acoustic music is not covered in this thesis.

The scope of this thesis is on monophonic music retrieval: MR1, MR3, MR9 and MR11. The focus will be on MR1 and MR3, since there are much more symbolic music data (MIDI files) available to us than acoustic music data. Nevertheless, the techniques can be easily extended to MR9 and MR11. In another words, we focus on content-based music retrieval using melody.

Melody-based music retrieval is useful, because melody is characteristic and specific for each music piece, especially songs. And people can easily produce melody queries by humming. Query-by-humming (MR3) is particularly addressed in this thesis.

1.4. Objective

As mentioned previously, this thesis focuses on melody-based music retrieval.

Music retrieval using melody involves several issues: (1) acoustic analysis: how to extract melody from acoustic signals; (2) melody representation: how to represent the melody appropriately in the computer; (3) melody similarity: how to compare the similarity of melody for effective retrieval; (4) melody matching: how to achieve accurate and efficient matching when the query contains variations and the database contains a large music corpus.

People with music knowledge or skills may be able to transcribe the melody in their minds into a sequence of music notes, and even play it out on a keyboard instrument. Most people, however, do not possess such skills but could only sing or hum out the tune, although they might not aware of the name of the notes in their voice. Query-by-humming will require a signal processing technique to transcribe the singing voice into discrete music notes or other melody representation (pitch line proposed in this thesis).

Before comparing the melody of a query with those of music corpus, the melodies need to be converted into an appropriate numerical representation. The representation should be appropriate for similarity computation and matching, such that similar music pieces can be effectively retrieved. The melody representation can be a string of characters, or a sequence of numbers [Dann_ISMIR01], or some statistics of the pitches or notes. The representation will largely rely on the method of the query.

The queries for music retrieval are often inaccurate due to poor singing skills, so a similarity metric for melody is important for effective music retrieval.

The variation in a melody query could be large. Different people would use different keys while producing the same melody. And often, the tempo in a humming query can vary or be inconsistent. These variations pose a challenge in melody matching.

We address the above-mentioned issues in this thesis and target the development of a melody querying solution with high retrieval accuracy, robustness and efficiency.

1.5. Contribution of the Thesis

This thesis describes an approach for content-based music retrieval by melody queries, especially acoustic melody query (humming), although it can also be applied to symbolic melody queries. The contributions of the thesis are listed as follows:

1) A melody representation based on time sequence, called pitch line, has been proposed. This representation is insensitive to note boundaries in melody, thus it is robust against the inevitable note segmentation errors from the acoustic melody input. This representation can also provide a high pitch resolution, which is important for acoustic melody queries.

2) A pitch extraction method has been proposed to construct pitch time sequence from humming inputs. Pitch value is detected from each audio frame to provide high time resolution. Harmonic structure of singing voice has been exploited to achieve high detection rate. The extraction is robust against variations in volume and vocal features.

3) A melody similarity measurement for pitch line has been proposed for effective melody retrieval. The similarity metric is based on pitch shifting and alignment of the 2 pitch line sequences, so key transposition and tempo variation would not affect the similarity measurement.

4) Melody skeleton, a novel compact melody abstraction has been proposed to solve the key transposition and alignment problem in melody matching. Melody skeleton also serves for eliminating unrelated candidates thus increasing the matching speed. A value-run domain melody skeleton matching method has been proposed to tolerate possible errors in melody skeleton.

5) A music scale modelling technique has been proposed to estimate the scale type and scale root of a melody. The estimated scale root can be used for key transposition in melody matching. With the root, melody matching can be much more efficient. The technique for Major/Minor scale estimation has been shown to be very effective for song retrieval, since most popular songs are composed using Major or Minor scales.

6) Extensive experimentation has been conducted to evaluate the techniques proposed in this thesis, including:

   - the evaluation of pitch extraction for humming input,
   - the melody slope matching for key transposition and sequence alignment,
   - the melody skeleton based technique for key transposition and sequence alignment,

- the music scale estimation for both symbolic input and acoustic input,
- the melody metric and retrieval performance,
- a comparison of the retrieval performance with a previous retrieval method.

7) A prototype system has been implemented for music retrieval by humming. The system can automatically detect humming input from the microphone and trigger melody search. Melodies with high matching scores are presented to the user, and the user can select the melody to playback at the matching position. The prototype system has been shown to be a very convenient tool to evaluate the performance of the proposed music retrieval approach.

1.6. Organization

The thesis is organized as follows:

Chapter 2 presents the related works of content-based music retrieval using melody; Chapter 3 discusses the topic of melody representation and presents a method to extract the pitch from hummed melody queries; Chapter 4 presents a melody matching method based on pitch line using a melody similarity metric; Chapter 5 presents a novel melody alignment and pitch shifting approach using melody skeleton; Chapter 6 presents a music scale modelling technique for scale root estimation; Chapter 7 presents the experimental results and the implementation of a query-by-humming prototype system; Chapter 8 presents the conclusion and future works.

Chapter 2. RELATED WORKS

This chapter presents the related works on content-based music retrieval using melodic queries. The related works can be categorized based on their functions in a melody-based music retrieval system, as illustrated in figure 2-1. The categories are melody representation, melody similarity matching, melody extraction from polyphonic MIDI files, and pitch extraction for acoustic input.

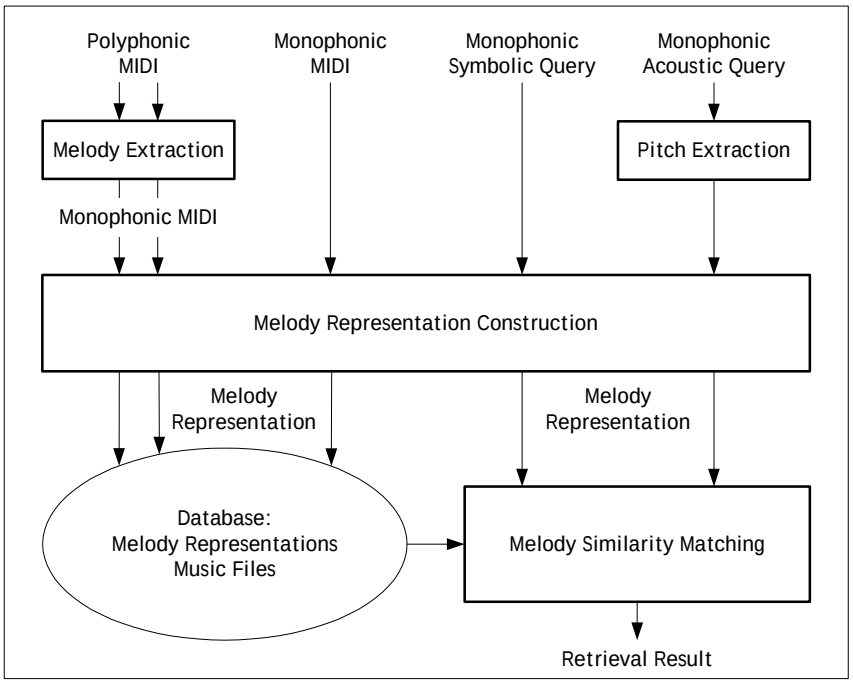

Figure 2-1: System structure of melody-based music retrieval systems

Melody extraction is about identifying melody tracks and notes from a polyphonic MIDI input, so that the system can support retrieval of polyphonic music using melody. Pitch extraction is for extracting pitch information from acoustic input, like humming, which is important for the system to support query-by-humming. Melody representation is about the numerical representation of melody, which can be constructed from both symbolic melody and acoustic

melody, and then be stored and searched. Melody similarity matching is about how to compute the similarity between melodies based on the representation. Melody representation and melody matching are closely related, and are the key components in the music retrieval system.

This chapter reviews firstly the works on melody representation and matching, secondly on melody extraction for polyphonic MIDI files, and finally on pitch extraction from acoustic input.

2.1. Melody Representation and Matching

Melody representation and matching is about how to represent melody using numbers in a computational device, and how to measure the difference or closeness of 2 melodies. This is the first topic addressed by researchers in music information retrieval.

Melody is naturally treated as a string of music notes, which is intuitive from the paper representation of music, the written score (figure 2-2). String matching techniques are conveniently adopted for melody matching [Mong_CH90]. In such approaches, the melody query is in symbolic format.

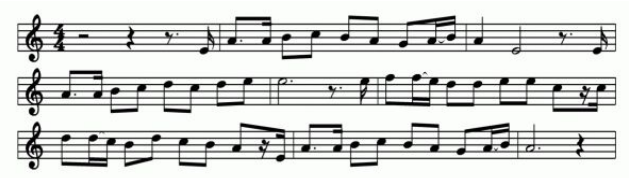

Figure 2-2: Music notes in music scores

The idea of music retrieval using acoustic input, such as humming, was proposed in [Kage_ICMC93, Ghia_MM95], when personal computers were capable of doing signal processing tasks, like detecting the pitch of a sound.

Although closely related, symbolic melody query and acoustic melody query have different conditions and requirements. The major difference is that symbolic melody queries are usually discrete and exact, while acoustic melody queries are inexact and erroneous. The related works on these 2 approaches are presented respectively in the following 2 subsections.

2.1.1. Symbolic Melody Query Approach

In symbolic melody query approach, a melody is seen as a succession of discrete monophonic notes. Either the pitch or time duration of the notes are used in the melody representation: a string of pitches or a string of durations.

Absolute pitch of the notes, such as the note name (C, C#,…) is avoided to represent melody. This is because a melody remains the same after transposition across the keys, which changes the absolute pitch of the notes. Instead, relative pitch is used for melody representation [Dowl_CH78, Mong_CH90, Gias_MM95, Lind_MIT96, Blac_MM98, Korn_CM98]. A relative pitch is the pitch difference of a note from its previous note, which is obviously invariant to key transpositions.

Another advantage of using relative pitch is that string matching method can be utilized for melody comparison or retrieval. Mongeau and Sankoff [Mong_CH90] presented an editing distance measurement for approximate string matching. The editing distance between 2 strings is the minimal cost to transform a string to the other. The transformation cost is calculated by summing the cost of local transformations. The local transformations usually considered are: insertion, deletion, and replacement. The typical cost metrics are: 1 for insertion, 1 for deletion, and 2 for replacement. The minimal total transformation cost (editing distance) is obtained using Dynamic Programming (DP) algorithms.

Different precisions of relative pitch are used in melody representation: pitch direction, rough pitch interval and exact pitch interval. The simplest and coarsest relative pitch is pitch

direction [Hand_MIT89], which uses 3 characters “U”, “D” and “S” to represent the pitch difference: “U” stands for “pitch going up”; “D” stands for “pitch going down”; and “S” stands for “pitch keeping the same”. So a melody is represented by a string of 3 characters, which is also called melodic contour. Many music retrieval methods are based on these approaches [Ghia_MM95, McNa_DL96, Korn_CM98, Roll_MM99, Chai_MMCN02].

[Blac_MM98] shows that melodic contour representation is not sufficient for retrieval in a large database. Rough pitch interval is introduced to encode the relative pitch using 5 levels: up, up a lot, repeat, down, and down a lot. Thus a melody is represented by strings of 5 characters. [Kim_ISMIR00] also suggests a 5-level contour derived by quantisation using 0, 1, and 3 semitones.

Exact pitch interval [Uitd_MM99] measures relative pitch in exact semitones, which gives the highest precision, but at the cost of lower recall [Pick_SIGIR01]. Rough pitch interval is then a trade-off between precision and recall.

[Uitd_ACSC02] compared various combinations of pitch representation (pitch contour and exact pitch interval) and editing distance (longest common contiguous sub-string/n-grams, longest common subsequence, local alignment using dynamic programming) and showed that exact pitch intervals and local alignment are most effective.

Compared to the pitch, the duration of notes has been used much less for melody representation. Absolute note duration is avoided, since the tempo can be changed without changing the melody. To achieve invariance to tempo change, the duration ratio between consecutive notes is adopted [Byrd_DL01, Jang_PCM01, Lems_ICMC99]. Melody retrieval is similarly done by string matching techniques. The results showed that duration-based melody representation and matching is much less discriminative than the pitch-based representation.

The techniques proposed for symbolic melody queries have shown some effectiveness. The limitation for such an approach, however, is that it can hardly be used for lay users, who

cannot transcribe a tune into discrete notes. A layperson would prefer to use acoustic queries, such as humming, to search for music.

2.1.2. Acoustic Melody Query Approach

Music retrieval using acoustic query is an attractive music retrieval strategy, since humming a tune through a microphone is much easier than keying in the music notes for most people. There are generally 3 classes of techniques for music retrieval by acoustic query: symbolic melody matching, time-based melody feature matching, and time sequence matching.

2.1.2.1. Symbolic Melody Matching

Conventional melody matching (as presented in section 2.1.1) can be used for acoustic melody query, if the humming signals can be automatically transcribed into music notes (with specific pitch and/or duration) [Kage_ICMC93, Ghia_MM95, McNa_DL96].

In [Ghia_MM95] the humming input is transcribed into notes by a traditional pitch detection method using autocorrelation. Melody contour and the approximate string matching are employed for retrieval. In the experiments, a user is required to hum each note literally, to ensure a high note detection rate, and the evaluation database contains only 183 music items.

[McNa_DL96] presents a music transcription module for humming input, which can adapt to the user's musical tuning. The melody matching is based on melody contour. However, it can only support matching at the beginning of each tune in the database.

There is a major disadvantage of query-by-humming using symbolic melody matching approach: it is very sensitive to errors on note detection from the acoustic input. Strict conditions were imposed to alleviate the problem: each hummed note is required to be separated by silence or by specific consonants (like, "d"). Such requirements can be very difficult to meet, since the users could have very poor singing skill and errors on note detection

could be inevitable. As a result, the techniques can only be used for small melody corpus and matching only the beginning of target melodies.

2.1.2.2. Time-Based Feature Matching

Errors on note detection can be less sensitive for melody retrieval, if the acoustic melody can be segmented into time-based units, like measures or beats. Kosugi et al. [Kosu_MM00] proposed a time-based approach for query-by-humming. Melodies are segmented into beats, and features vectors for each 4 beats (tone transition, tone distribution) are then used to represent the melody. Melody similarity matching is based on Euclidean distance between the feature vectors. The results have showed that note fragmentation has no effect on the retrieval performance. However the drawback of this method is that it requires the user to hum by following a metronome to control the beat segmentation, which is rather restrictive. Moreover, many people are not very discriminating when it comes to their perception of the beat of a melody. Different meters (e.g. duple, triple, quadruple meters) of the music can also contribute to the difficulties.

2.1.2.3. Time Sequence Matching

Since note detection error poses a major problem in melody retrieval by acoustic input, an alternative approach is to avoid doing note detection and represent a melody by a time sequence of absolute pitch values or continuous pitch contour [Fran_ICME00, Jang_ISR00, Nish_ISMIR01, Zhu_ICME01]. The time sequence of pitch values can be derived by detecting the pitch value for each short audio frame (say 40ms), and notes in the melody is not explicitly identified. Melody retrieval is done by measuring the distance between 2 time sequences using certain metrics.

[Fran_ICME00] presents a similarity metric for continuous pitch contour using the crossing area between 2 pitch time sequences. However, how to achieve an optimal matching of 2 time sequences considering key transposition and tempo change is not addressed.

[Jang_ISR00, Jang_MM01] proposed a method to compare the continuous pitch contour of a hummed query of with the melodies in the database. To tolerate tempo variations, this method uses dynamic time warping distance for the comparison. To deal with key shifting, it uses a heuristic to estimate the key transposition by doing multiple dynamic time warping computations. It reported a result of 85% success rate in the top-20 rank list. A shortcoming of this technique is the heavy computation requirement. The size of the table in dynamic time warping is 128x179 for 8 second query input. And the key transposition estimation by multiple trials is very inefficient. As a result, this method considered matching only at the beginning of a song in retrieval.

Nishimura [Nish_ISMIR01] proposed a continuous dynamic programming method to compute the accumulated distance between a query time sequence and a target time sequence. This method handles the key transposition by assuming a correct start frame in the time sequence and distance measurement is based on that start frame. The authors, however, did not suggest how to choose the start frame.

Time sequence of pitch value indicates a robustness property in melody representation for acoustic input. However, the major issue on how to effective and efficient compute the similarity between 2 melodies considering key transposition and tempo variation was not sufficiently solved. A solution on this issue is one of the major contributions of this thesis.

2.1.3. Summary

The related works on melody representation and matching are summarized in table 2-1.

Table 2-1: Comparison of melody representation and matching approaches

Approaches / Comparison	Symbolic Query [Uitd_MM99]	Acoustic Query		
		Symbolic melody [McNa_DL96]	Time-based feature [Kosu_MM00]	Time sequence [Nish_ISMIR01]
Note detection	N. A.	Yes	Yes	No
Sensitive to note detection error	N. A.	Yes	No	No
Time unit (beat) segmentation	N. A.	No	Yes	No
Pitch representation	Relative pitch	Relative pitch	Absolute pitch	Absolute pitch
Restrictions for acoustic input	N. A.	Literal articulation	Following metronome	No
Key transposition requirement	No	No	Yes	Yes
Tempo variation requirement	No	No	No	Yes
Matching approach	String editing distance	String editing distance	Euclidean feature vector distance with key shifting	Time sequence comparison with key shifting and time warping

Time sequence matching approach imposes no special requirements/restrictions on the acoustic input, and thus provides larger flexibility and is suitable for music retrieval by layperson users. The difficulty of this approach lie in how to effectively and efficiently compute the similarities between time sequences under 2 dimensions of variation: key transposition and time warping. The current melody retrieval methods by time sequence matching are summarized and compared using table 2-2. The last column in the table corresponds to the work of this thesis.

Table 2-2: Summary of works using time sequence matching

Methods	[Fran_ICME00]	[Jang_ISR00]	[Nish_ISMIR01]	This thesis
Key transposition method	Exhaustive search	Exhaustive search with heuristics	Using start frame	Utilizing global sequence features: melody slope / melody skeleton / scale estimation
Time warping method	Exhaustive alignment	Dynamic time warping of local pitch values	Dynamic time warping of local pitch values	Sequence alignment using the global sequence features
Distance metric	Euclidean distance	Cumulated time warping distance	Cumulated time warping distance	Non-linear accumulated distance
Time sequence compactness	Low	Low	Low	High

The exhaustive search approach for key transposition will lead to heavy computation requirements. Exhaustive search with heuristics sacrifices accuracy while still has high

computational complexity. Key transposition using start frame has the risk of wrong pitch shifting by using a frame with pitch error. This thesis proposes a new key transposition method by finding global shape features of the time sequence, which is both robust and efficient.

For time warping in sequence matching, exhaustive alignment is practically useless due to the computational complexity. Dynamic time warping of local pitch values treats each pitch value in the sequence uniformly, which make it prone to misalignment using the limited (practically tiny) leeway in dynamic programming. In this thesis, time sequence alignment is achieved by using the global sequence feature, in which some pitch values in the sequence are more important than the others and less computation is needed for proper alignment (warping) of the time sequences.

In distance metric for melody comparison, Euclidean distance or cumulated warping distance ignore the nature of pitch inaccuracy in the acoustic input. This thesis has proposed a metric that compensates the normal pitch inaccuracy presented in acoustic input.

Less compactness of a time sequence will indicate high storage requirement and matching complexity. In this thesis, the time sequence representing acoustic melody is very compact, thus is more efficient for storage and retrieval.

2.2. Melody Extraction

To retrieve polyphonic MIDI music, the monophonic melody needs to be extracted from the MIDI file. The extracted melody should agree to human perception of the original polyphonic music. However, human perception of polyphonic music is very complex. To identify which note is part of the melody is quite difficult.

Uitdenbogerd [Uitd_MM98] proposed 4 algorithms to extract monophonic notes (melody) from a polyphonic MIDI music. The MIDI file could consists of multiple parts (instruments, or channels), and each part can have polyphonic notes. The proposed algorithms are based on a

study of property of music, music perceptions and music database users. The 4 algorithms normally produce different results for one piece of music. User's feedbacks are used to evaluate each algorithm. The experimental results show that the melody of a polyphonic MIDI was best represented by choosing the highest pitched note at any instant among all the music parts. The results also show that none of the algorithms can perfectly extract the melody that represents human perception.

Melody extraction is relatively easier for songs (music with a vocal part). The part corresponding to vocal is usually monophonic. So melody extraction is simply to identify the vocal part in the channels of the MIDI file. [Blac_OHSW00] proposed classification methods to classify a MIDI channel to accompaniment, bass, drum, and lead. The K-Nearest Neighbour classification method is shown to perform the best. The lead channel identified can be used in melody-based retrieval.

In our work, we use MIDI music of songs (Karaoke) for music retrieval. In a Karaoke MIDI file, most of the time there is a particular track for the melody/tune, which the singer can follow while singing. Thus to identify and extract the melody track is sufficient for melody extraction, and the analysis of polyphonic notes is not necessary. We have proposed an effective method to compute the likeliness of any track in the MIDI file to be the melody track. The most likely tracks can then be prompted for verification by a human.

2.3. Pitch Extraction

For a music retrieval system to accept an acoustic query, a pitch extraction technique is needed to transcribe the acoustic signals into a form that can be used for melody matching.

There are basically two types of pitch extraction for humming: transcription to MIDI note, and continuous pitch sequence extraction.

2.3.1. MIDI Note Transcription

To transcribe the humming voice into a sequence of note, there are 2 major tasks: to locate the note boundaries (start and end); to determine the pitch value for the note.

Many systems [McNa_ICME00, Poll_ICME02] use the signal power (root mean square) to locate the note boundaries, which assuming that each note articulated is prefixed by a period of silence with very low signal power. [Haus_ISMIR01] uses consonant/vow classification to detection the boundaries of notes.

To determine the pitch value of a segmented note, McNab in [McNa_ICME00] used Gold-Rabiner algorithm for pitch detection. Pollastri in [Poll_ICME02] detect the pitch in frequency domain by locating the prominent peaks in the spectrum. The final pitch value of a note is assigned as the standard frequency of music notes (such as "C") that is closest to the detected pitch values. [Haus_ISMIR01] proposed a pitch refinement technique by estimating a relative scale.

The basic steps of MIDI note transcription techniques are: (1) note segmentation; (2) pitch tracking for each note. The note segmentation in these techniques would fail when the articulation is not very literal, especially for tie and slur notes in a melody.

2.3.2. Continuous Pitch Sequence Extraction

For melody representation using time sequence of pitch values (discussed in section 2.1.2.3), discrete note transcription is not required. Instead, a sequence of pitch values can be detected for each individual frame without knowing boundaries of notes. The pitch value in general has higher precision than that using note detection, where the pitch of a note is obtained by taking an average or a median pass filtering among all the frames of the note.

Although note segmentation error is not a problem for this approach, accurate pitch determination for each frame can be difficult. Errors, like missing pitch, double/triple pitch, could be presented in the pitch extraction, especially when the acoustic signal has large variation in terms of volume and vocal conditions.

Jang in [Jang_MM01] uses basic time-domain speech processing technique to detect the pitch: autocorrelation function and average magnitude difference function [Dell_NYMP93]. Pitch extraction performance has not been evaluated in this work.

A method for robust pitch contour extraction from acoustic input has been proposed in this thesis, which is one of the contributions of the work. The method is based on detecting reliable pitch corresponding to vowels and then tracking the pitch based on the detected reliable pitch.

Chapter 3. MELODY REPRESENTATION

In a melody-based music retrieval system, it is critical to use an appropriate melody representation (or the data type used to represent melody). The melody representation should be robustly constructed from the various melody inputs and can be effectively and efficiently compared or matched for similarity measure. Figure 3-1 illustrates a melody-based music retrieval system, which can accept both symbolic and acoustic melody as input. The melodies in both acoustic form and symbolic form are converted to a common melody representation, which is then either inserted into the database or searched in the database. The melody representation shall be derived from the acoustic and symbolic melody forms and be suitable for computing a similarity measure.

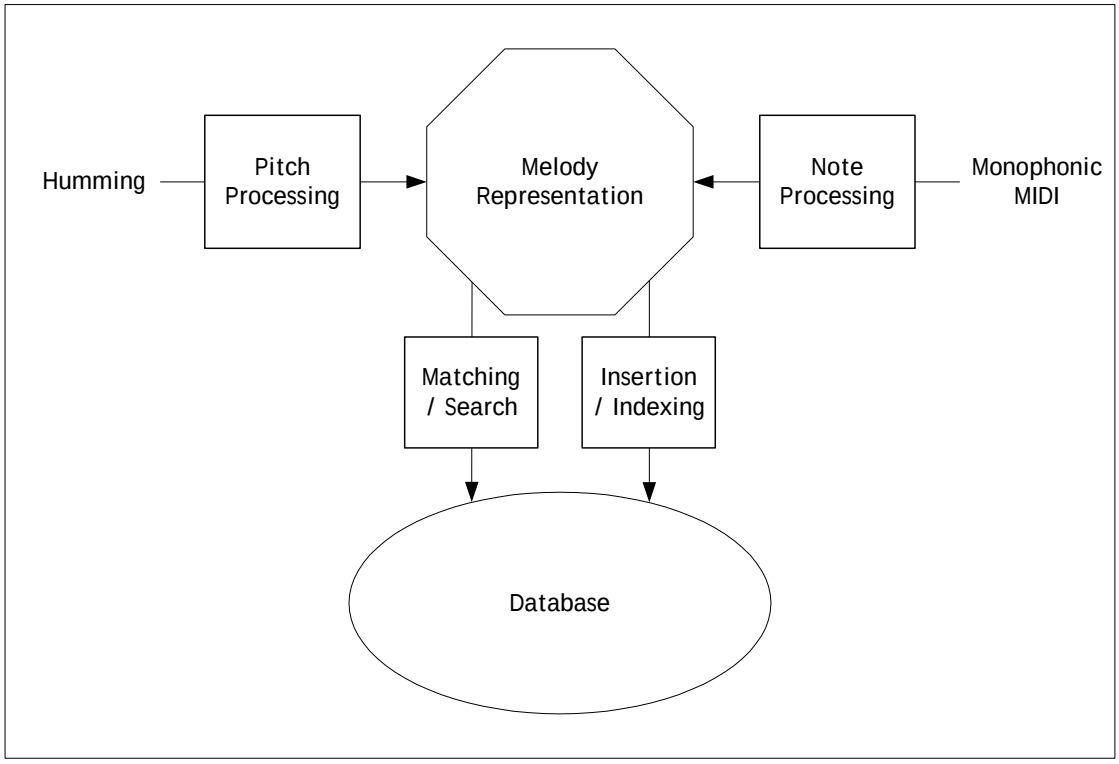

Figure 3-1: The system structure of a melody-based music retrieval system

This chapter develops a melody representation, called *pitch line*, which is designed to be appropriate for both acoustic and symbolic melody. Section 3.1 presents the design of pitch line. The construction of pitch line from acoustic and symbolic melody is presented in section 3.2. Section 3.3 presents the construction of pitch line from symbolic melody.

3.1. Design of Pitch Line

This section presents the design of "pitch line" for melody representation.

3.1.1. Note Sequence and Pitch Curve

The native form of symbolic melody is a sequence of music notes, where each note has a pitch value and time duration value. Silence or rest in a melody can be seen as a special note, which has no pitch value but a time duration value. The characteristics of note sequence are: 1) each note is a discrete entity without ambiguity; 2) the pitch value is exact in semitones; 3) the time duration is usually standardized in multiple of a small time unit, such as ½ quarter note; 4) the duration of a quarter note (beat) is determined by the tempo, which is usually fixed in one performance. Figure 3-2 (d) illustrates a melody using note sequence.

For acoustic melody, such as humming, note sequence is, however, unsuitable for melody representation. This is because: 1) it is difficult to accurately segment each note in the signals; 2) the pitch value is often inexact and unstable; 3) the time duration of the segmented note can vary and the tempo may also change during a performance. The only or most reliable feature of acoustic melody is the pitch values at every instant in the humming, which can form a time sequence of pitch (or a curve of pitch in time domain), which is called *pitch curve*. Figure 3-2 (a) illustrates a melody representation using pitch curve. The characteristics of pitch curve are: 1) there is no music note identity; 2) the pitch value is continuous and inexact; 3) there is no

standard speed unit, such as quarter note; 4) tempo may vary and be inconsistent during one performance.

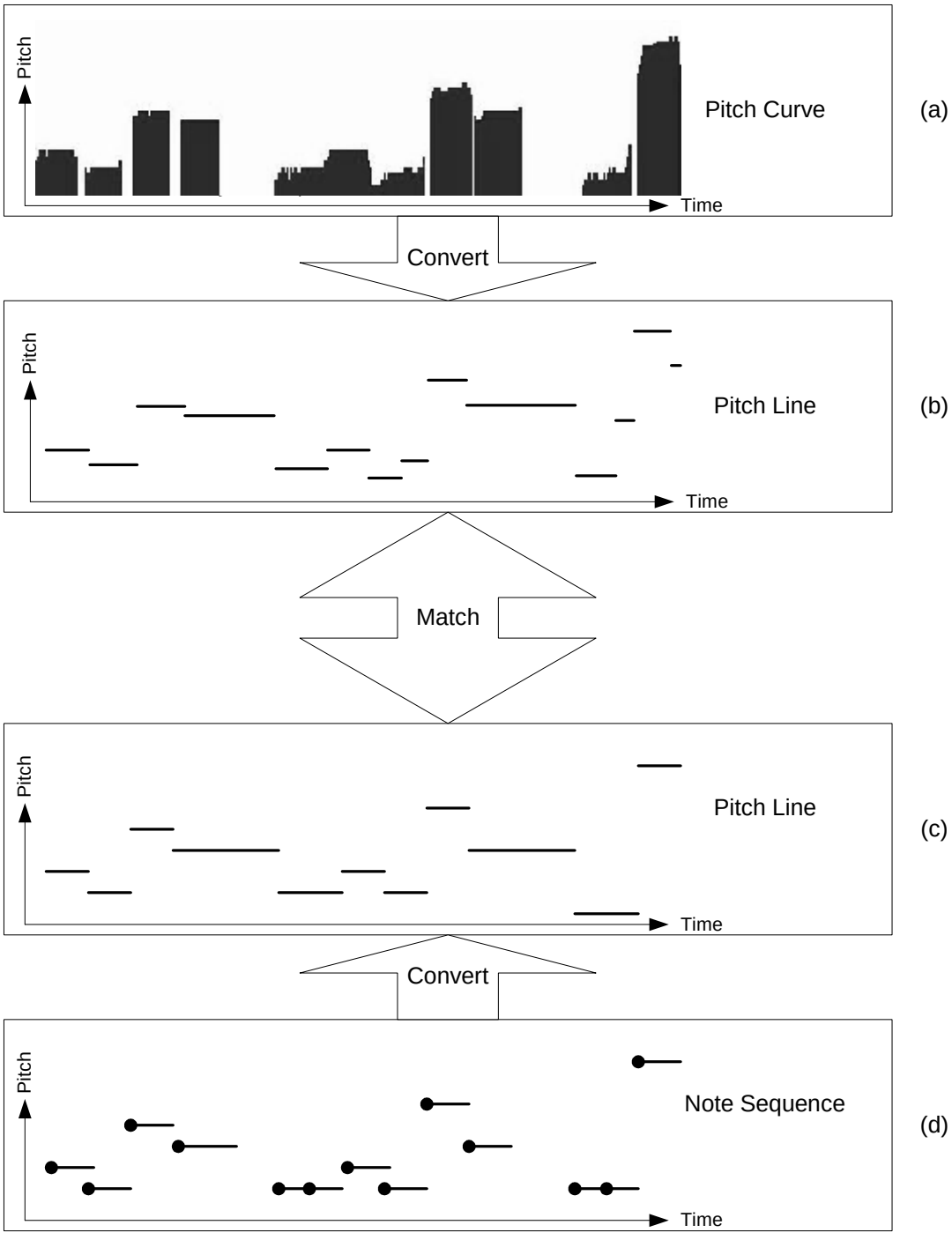

Figure 3-2: Pitch curve, note sequence and pitch line for melody representation

3.1.2. Pitch Line

Although note sequence and pitch curve are quite different, it can be seen from figure 3-2 (a) and (d) that the pitch versus time plots for note sequence and pitch curve are similar and computationally comparable. We thus propose pitch line as a common melody representation, which can be derived from both the note sequence as well as the pitch curve.

Pitch line is a sequence of horizontal line segments in the pitch versus time plot. Pitch line is similar to pitch curve in that there is no note identity, pitch value can take any real number rather than exact semitones, and tempo is subject to variations.

Figure 3-2 (b) illustrates the pitch line corresponding to the pitch curve in figure 3-2 (a). Pitch line can be seen as a product of dimension reduction on pitch curve, by which pitch line preserves the main features of the pitch curve with a more compact representation. The construction of pitch line for acoustic melody input is presented in section 3.2.

Figure 3-2 (c) illustrates the pitch line corresponding to the note sequence in figure 3-2 (d). It can be seen that note sequence can be converted to pitch line simply by combining any consecutive notes with same pitch. In fact, after the note combination processing human can still identify the same melody.

Pitch line derived from both acoustic and symbolic input can then be matched to compute the similarity between melodies. Key transposition and tempo variation concern the similarity matching of pitch lines, which is presented in chapter 4.

The advantages of pitch line for melody representation are: 1) errors in note identification in humming, which is common, does not affect the representation; 2) higher resolution of pitch values can be reserved in this representation, which can provide a better numerical precision of the melodic information than the music note, which uses a semi-tone or larger pitch interval as the measurement unit.

3.1.3. Temporal Levels of Pitch Line

Pitch line contains all the pitch information of a melody. The time information, however, can also be useful and sometimes important for a particular melody to be distinguished from others. The time information of a melody corresponds to the temporal segmentation of the melody into small units, such as notes or group of notes.

We propose three temporal levels for pitch line to encode the time information. They are sentence level, phrase level and word level.

- Sentence level: In sentence level, the pitch line for one melody is grouped together without any segmentation. This is illustrated in figure 3-3 (a).

- Phrase level: In phrase level, the pitch line in is segmented into phrases. Each phrase may correspond to a sequence of notes performed in one breath. Usually the boundary between 2 phrases is a period of silence, which is caused by breath inhaling of the performer. Thus, phrases are demarcated by the breath onset in the singing. Figure 3-3 (b) illustrates 2 phrases of pitch line.

- Word level: This is the lowest temporal level for pitch line. A word is the smallest unit, which corresponds to note. A word may not be exactly one note; it may be also 2 or 3 tie or slur notes. Vowel onsets in the singing demarcate the word boundaries. Figure 3-3 (c) illustrates pitch line in word level.

Segmentation of pitch line into phrase level and word level for acoustic melody is presented in section 3.2.3.

Section 3.3 presents how to claim the boundary of word and phrase for symbolic melody.

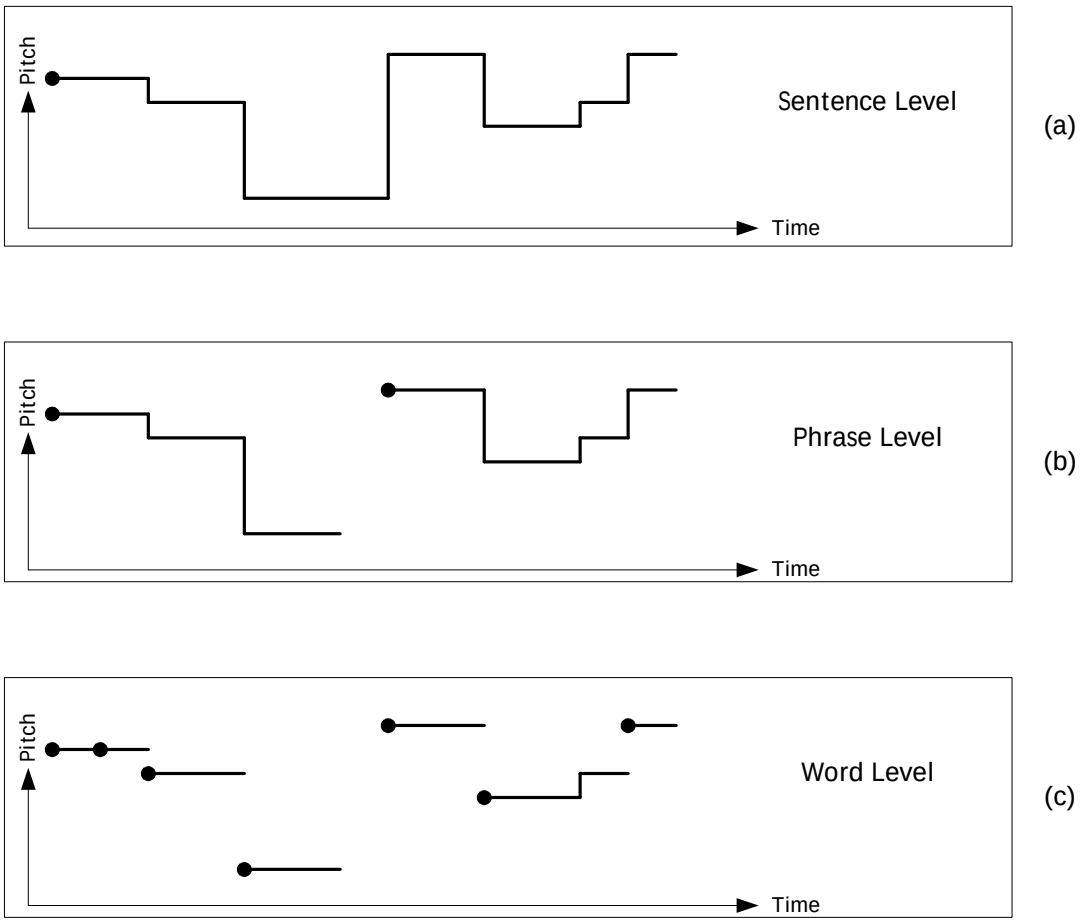

Figure 3-3: Temporal levels of pitch line: (a) Sentence level; (b) Phrase level; (c) Word level

3.2. Construction of Pitch Line from Acoustic Input

This section presents a novel approach for constructing pitch line from acoustic melody inputs. Although acoustic melody input can be produced by many types of musical instruments, we focus on the human voice, since humming is much easier than playing an instrument for most people.

The method involves extracting the pitch information from the voice signal and converting the extracted pitch contour into pitch lines. Extracting pitch contour with reliability is of key concern and a major contribution of the proposed method.

The following subsections present the problems in pitch detection from humming voice, the proposed general approach, a voice segmentation technique, the pitch detection and tracking method, and finally the pitch curve aggregation algorithm.

3.2.1. The Problems in Pitch Detection from Humming Voice

Although there have been many methods proposed in speech analysis area for pitch detection from speech signal, such as time-domain methods (Autocorrelation method, Average Magnitude Difference Function, and etc) and frequency-domain methods (spectral, cepstral method), reliable pitch extraction from humming voice is still a nontrivial problem. The challenges mainly come from the dynamics and variations in the ways the users articulating music notes or pitches using voice.

3.2.1.1. Humming Voice Production

Humming, like speech, is generated with a source-filter process. The source signal force is the vibration of the vocal fold and/or turbulence caused by the constriction of the airflow in vocal tract. The vocal tract works as a filter, which modifies the source signal and produces the humming/speech sound.

The basic units in the speech and humming sound are phonemes, which can be classified into different categories based on the source conditions and vocal tract configurations. According to whether the vocal fold vibrates, a phoneme can be either voiced or unvoiced. For voiced phonemes, the turbulence may also present. For unvoiced phonemes, turbulence is the only driving source signal. According to whether there are constrictions of airflow in the vocal tract, a phoneme can be either a vowel or a consonant. Vowels are produced with no constriction and are voiced. Consonants are produced with certain constrictions and can be either voiced or unvoiced. Table 3-1 lists the types of phonemes of English.

Table 3-1: Type of phonemes

Types of phonemes		Voiced	Unvoiced
Consonants	Plosives	[b], [d], [g]	[p], [t], [k]
	Fricatives	[v], [th]	[f], [th]
	Sibilants	[z], [j/zh], [kh]	[s], [sh], [h]
	Liquids	[l], [r]	
	Nasals	[m], [n], [ng]	
	Semi-vowels	[w], [y]	
Vowels		[a], [e], [i], [o], [u]	

The vibration of vocal folds is caused by repetitive opening and closing of the approximated vocal folds driven by the breath pressure force (opens the vocal folds) and Bernoulli force (sucks back the vocal folds). Depending on the tension of the vocal folds, the vibration will produce sound of different pitches.

However, vocal folds still have different conditions of vibration: modal, creaky, breathy, and falsetto. These conditions are also called vocal registers [Titze_PH94, Bloo_ES99]. Modal is the normal vocal condition in speaking and singing. Creaky voice is produced by tightly approximated vocal folds when the sound has a low pitch. Breathy voice is produced when the vocal folds are loosely approximated and the sound has a low loudness. Falsetto condition is for making very high-pitched sound, when the vocal folds have a very high tension and only part of the vocal folds vibrate.

For trained singers, only modal and falsetto vocal registers are used. Creaky and breathy vocal registers are usually undesirable, however, are often presented in lay users' singing voice. Pitch detection for creaky and breathy voices is much more erroneous than modal or falsetto voices.

When people hum, a vowel is typically used for a music note and a consonant is used to indicate a transition between notes. Thus in normal situation, a humming consists of a sequence of syllables, which is a consonant followed by a vowel, like "la la la" or "da da da". And often voiced consonants are used in humming, which might be attributed to the reason that tensions of vocal folds can be controlled more easily than using unvoiced consonants. The voiced consonants with turbulence [v] [th] [z] [j/zh] [kh] are seldom used while humming, since the noisy turbulence sound is interfering and irrelevant. [w] and [y] are semi-vowels, which are relatively less used. The most frequently used voiced consonants are: [b] [d] [l] [m] [n] [r], which are articulated with the easier movements of lips or the frontal portion of the tongue. Among them, [b] and [d] start with complete obstruction of airflow, while [l], [m] and [n] [r] do not start with complete airflow obstruction.

3.2.1.2. The Difficulties of Pitch Detection

A humming voice signal could contain the sounds of vowels, consonants, silence and turbulence. Thus pitch does not exist for certain time frame (short window) of the signal. Pitch can only be defined or perceived from voiced sound (vowels or voiced consonants). For voiced sound, the vocal conditions (registers) can affect the pitch quality and sometimes make pitch detection erroneous. A very common error is the doubling and halving of the correct pitch frequency, which is unacceptable for melody representation. Experiments have shown that the errors usually present in creaky and breathy vocal conditions. Such vocal conditions are more likely to occur in the articulation of consonants than vowel.

Given that the phonemes used by a user's humming are unknown and the various vocal conditions can be presented, the challenges of pitch detection are:

(1) how to decide whether a pitch is presented or reliable for each time frame (the time frames of vowels are of interests);

(2) how to accurately compute the pitch value of a time frame with pitch presented;

3.2.1.3. The Existing Pitch Detection Methods

The existing pitch detection methods detect the pitch of a humming voice by segmenting the voice into individual syllables. And the pitch is detected for each individual syllable.

The signal energy is commonly used for voice segmentation. Such segmentation method can work well for certain syllables, such as those starting with [b] or [d], which has a complete vocal tract obstruction and leads to a period of silence in the signal. But for consonants [m], [n] and [l], there is no silence period in the articulation, which makes the segmentation method fail.

Furthermore, the existing methods do not consider the vocal conditions in pitch detection.

3.2.2. The Proposed Approach

A novel approach for pitch extraction from humming voice has been proposed, which addresses the reliability issue of pitch detection. In this approach, converting the humming voice signal to the melody representation (pitch line) consists of a few steps of signal processing: voice segmentation, reliable pitch detection, pitch tracking and pitch curve aggregation, which are illustrated in figure 3-4.

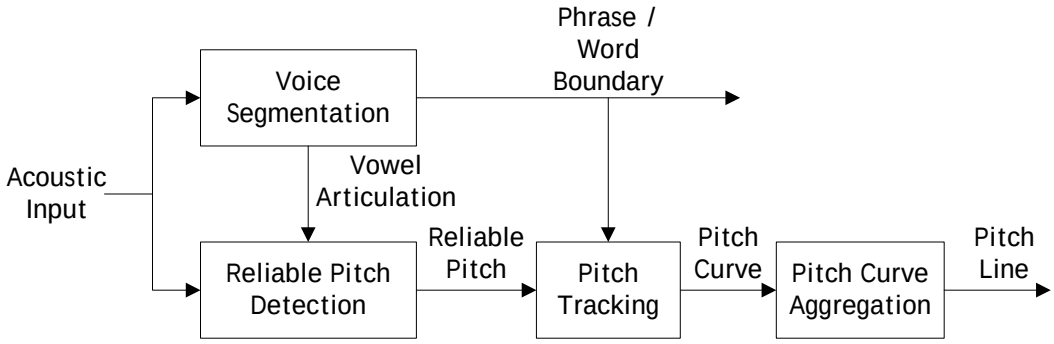

Figure 3-4: Pitch processing for acoustic melody input

In the voice segmentation step, the positions of vowels are estimated using formant information of vocal tract and as well as signal energy. The time frames corresponding to onsets of vowels are identified for reliable pitch detection in the next step. The voice segmentation process can also identify the phrase boundary of the melody input by detecting the breath inspiration.

Reliable pitch detection works on the time frames for vowel onsets, where the pitch is considered prominent. The pitch is detected using cepstrum analysis considering irregular vocal conditions (creaky or breathy).

The pitch tracking process is based on the result of reliable pitch detection. The pitches of the other time frames are tracked from the known and reliable pitch, where continuous pitch changes within each vowel articulation are assumed. The output of pitch tracking step is the continuous pitch curve (pitch contour).

The pitch curve aggregation step will produce the pitch line for final melody representation.

The subsequent sections will present each of these steps in detail.

3.2.3. Voice Segmentation

The major target of voice segmentation of humming is to locate the position of vowel onsets, where the pitch is more prominent and also more representative for user's intention. The segmentation is based on analysis of transitions from consonants to vowels in the voice. Formant features, which characterise vocal tract configurations (like vocal tract openness, tongue positions), are used for the transitions analysis.

The detected vowel onsets were then used for reliable pitch detection. The transition also indicates boundaries of words (or notes).

Based on the formant features, the breath inspiration can also be detected in the humming voice, which corresponds to boundaries between phrases.

3.2.3.1. Cepstral Formant Feature Analysis

The major distinction between consonants and vowels is the openness of the vocal tract. For vowels, the vocal tract is more open, while it is more constrictive for consonants. It is well known from speech analysis, such vocal tract configurations can be indicated by the formant features [Rabi_PH93].

The formant features of singing/speech voice represent the filter modal of the vocal track. The separation of the filter modal (vocal tract) and the source excitation (vibration or turbulence) can be done using a cepstral domain analysis.

Figure 3-5 shows an example of a humming signal using "la", where (a) shows the waveform and (b) shows the energy of the signal. The signal has a 44100 Hz sampling frequency and 16 bit sample size. The window size of each frame is 2048 samples, i.e. each frame corresponds to about 46 milliseconds, and consecutive windows have an overlap of 1024 samples.

Although from the energy contour, the boundaries between syllables can be roughly identified, however, the variations of energy can be quite large, thus make segmentation based on energy erroneous. Furthermore, it will be not known whether the boundaries correspond to onset of vowels or onset of voiced consonants.

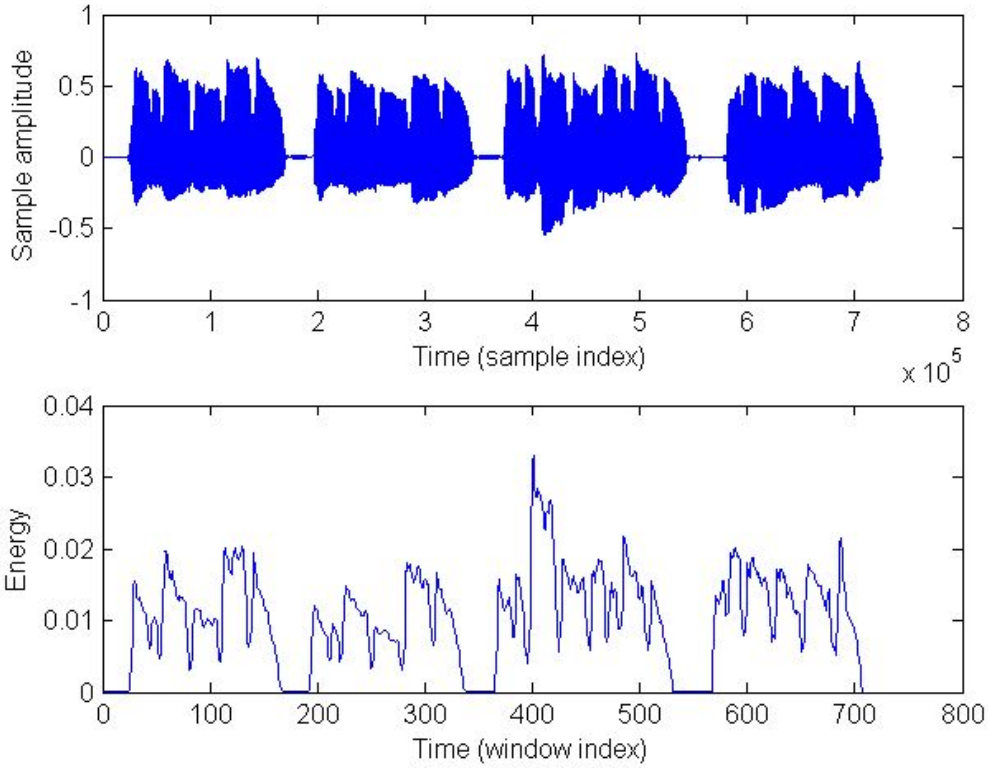

Figure 3-5: Waveform and energy of a humming signal

Figure 3-6 illustrates the spectrogram of the humming signal. In (a), the strips along the time direction indicate the presence of voice pitch. In (b) the spectrogram indicate the configuration of the vocal tract, which is obtained by removing the excitation component from the signal.

From 3-6 (b), it can be seen that transitions between consonants and vowels can be clearly identified. Thus we derived cepstral formant features for vowel onset detection.

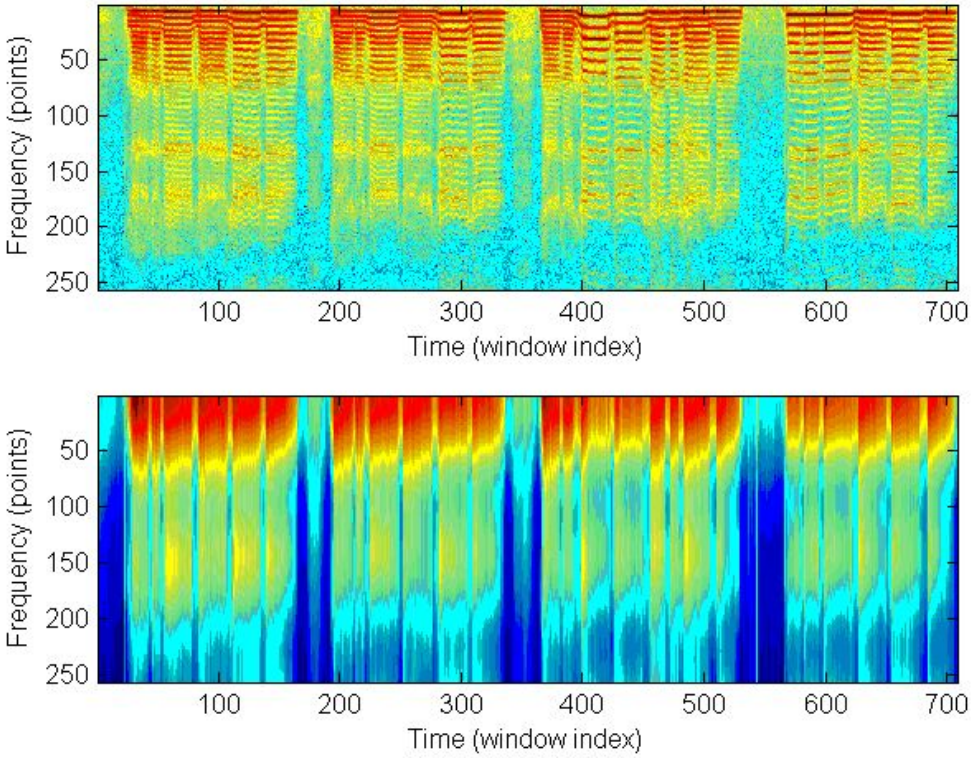

Figure 3-6: Spectrogram of humming signal and vocal tract formant

Cepstral formant feature analysis involves the following steps:

(1) Windowing of the signal: The input signal is divided into frames with a short period of time, such as 2048 samples using 44100 Hz sampling frequency. Hamming window is imposed for each frame, where Hamming window is defined as the following: $w[k+1] = 0.54 - 0.46\cos\left(2\pi \frac{k}{n-1}\right), k = 0,..., n-1$

(2) Power spectrum is then obtained by conducting Discrete Fourier Transform (DFT) on the frame of signal. $X = DFT(x)$. DFT is defined by $X(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k) e^{-jk2\pi n/N}$ for $n = 0,.., N-1$, where $x(k)$ is the sample and $X(n)$ is the Fourier coefficients. The power spectrum is thus $|X(n)|^2$

(3) Computing log power spectrum: $LOG|X(n)|^2$

(4) Computing the cepstrum by Inverse Discrete Fourier Transform (IDFT) of the log power spectrum $C(q) = IDFT(LOG|X(n)|^2$. IDFT is defined by $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{jk2\pi n/N}$ for $n = 0,.., N-1$.

Figure 3-7 shows the log power spectrum for window frame 35, 65, 80, and 180 in (a)-(d). Among them, (a) and (b) corresponds to vowels, (c) corresponds to voiced consonants, and (d) corresponds to breath inspiration.

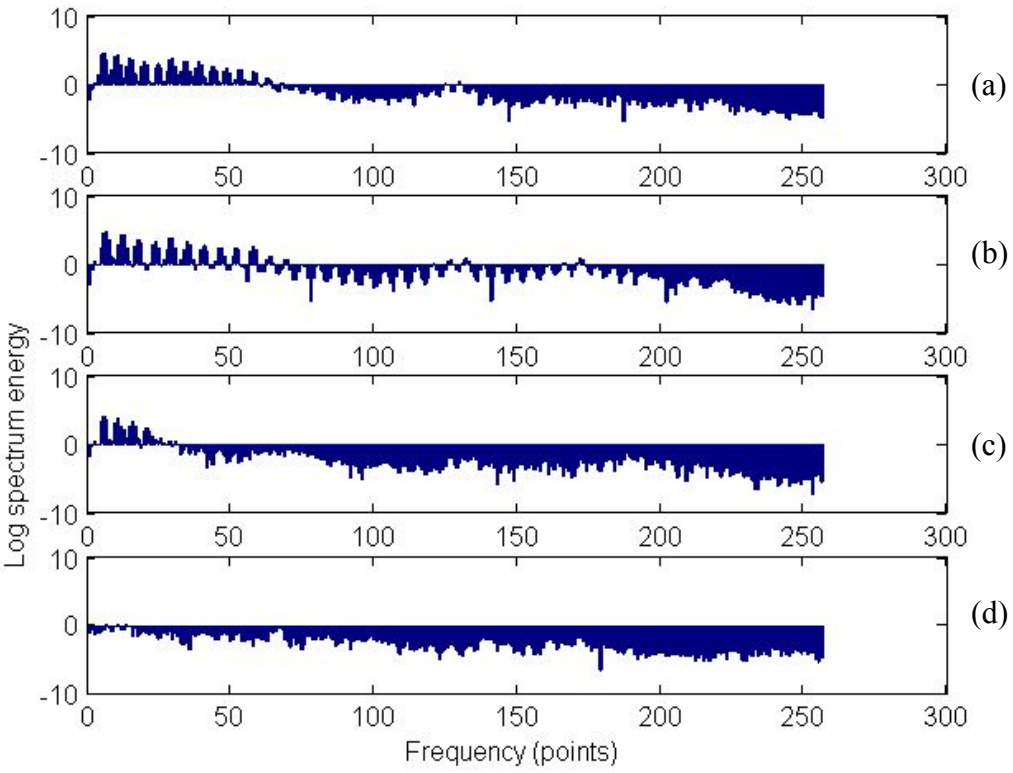

Figure 3-7: Log power spectra of the humming signal

Figure 3-8 shows the low range cepstral coefficients (from 1 to 50). And Figure 3-9 shows the high range cepstral coefficients (31 to 1024). It is known that the low range coefficients characterize the vocal tract feature (formant feature) of the signal, while peaks in high range

coefficients indicate the signal excitation of vocal fold vibration (figure 3-9 (a)-(c)) [Rabi_PH93].

In fact, figure 3-6 (b) is obtained by a DFT on the low range coefficients of the cepstrum. A technique proposed for vowel onset detection based on low range cepstral coefficients will be presented in the next section.

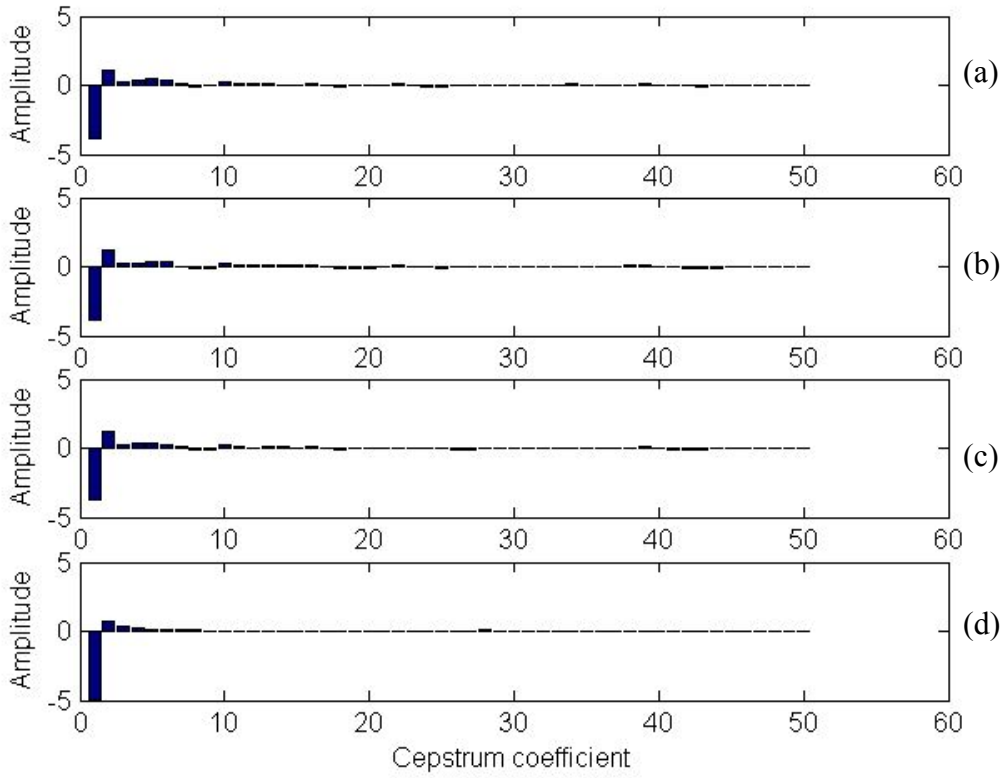

Figure 3-8: Cepstral coefficients and formant

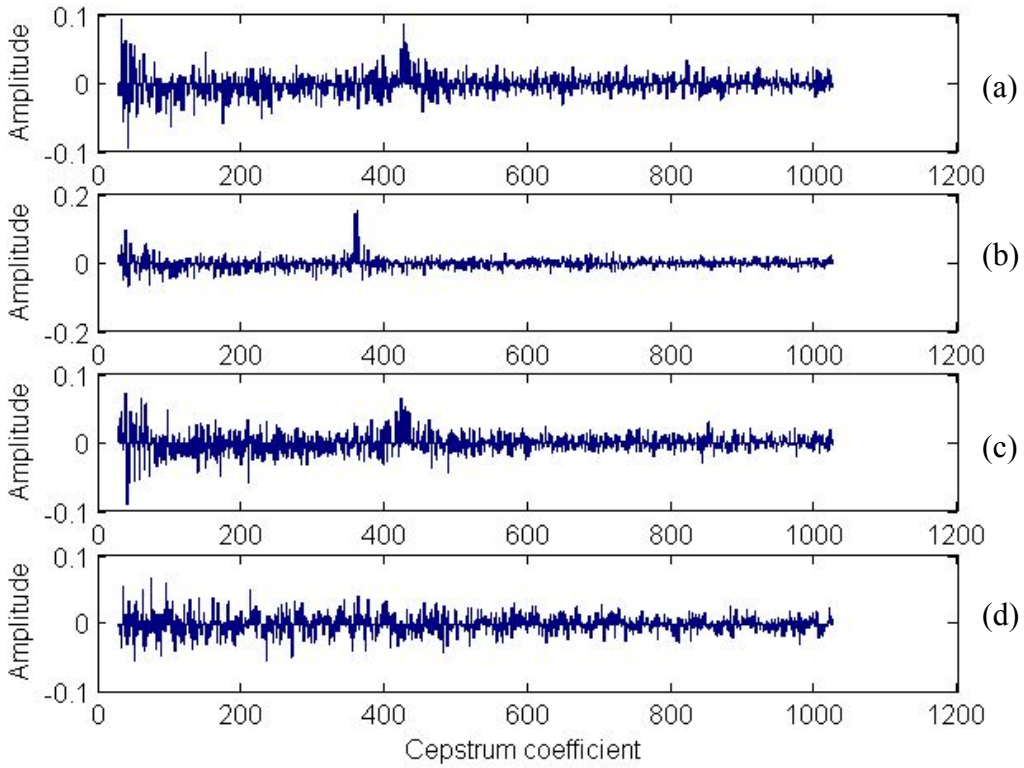

Figure 3-9: Cepstral coefficients and pitch excitation

3.2.3.2. Vocal Tract Opening Detection

As mentioned before, the openness of the vocal tract while articulating vowels and consonants is reflected by the low range cepstral coefficients. This section presents a technique to detect vowel onset based on cepstral coefficients. It is assumed that same consonants and vowels are used in each humming, thus the formant features would be similar for all the consonants or all the vowels. Since we are only interested in whether there is a vowel or a consonant in the signal, rather than which vowel (among "[a], [e], [i], [o], [u]") or which consonant (among "[b], [d], [m], [n], [l], … "), not all or too many of the ceptral coefficients are needed in the detection.

We first investigated the variance of the cepstral coefficients in the complete example humming signal to choose the appropriate coefficients for the vowel detection. The variance is

defined by $s_N^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2$, where N is the number frames, $\bar{x}$ is the coefficient value mean.

Figure 3-10 (a) and (b) show the variance and mean of the cepstral coefficients among all the frames of the signal. It can be seen that the first 2 coefficients have much larger variance than the remaining coefficients.

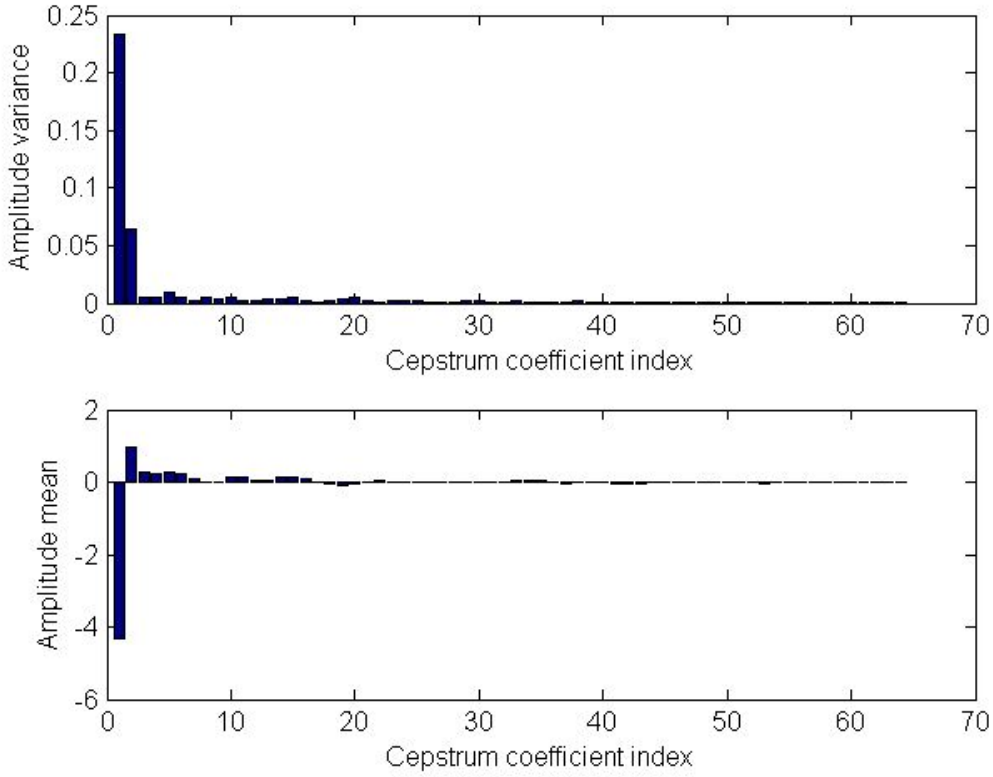

Figure 3-10: Variance and mean of the cepstral coefficients among all the frames of the signal

Figure 3-11 illustrates the absolute values of the first 5 cepstral coefficients of the signal. It can be seen from the contour plot that the first 2 coefficients prominently reveal the format features or vocal tract configurations.

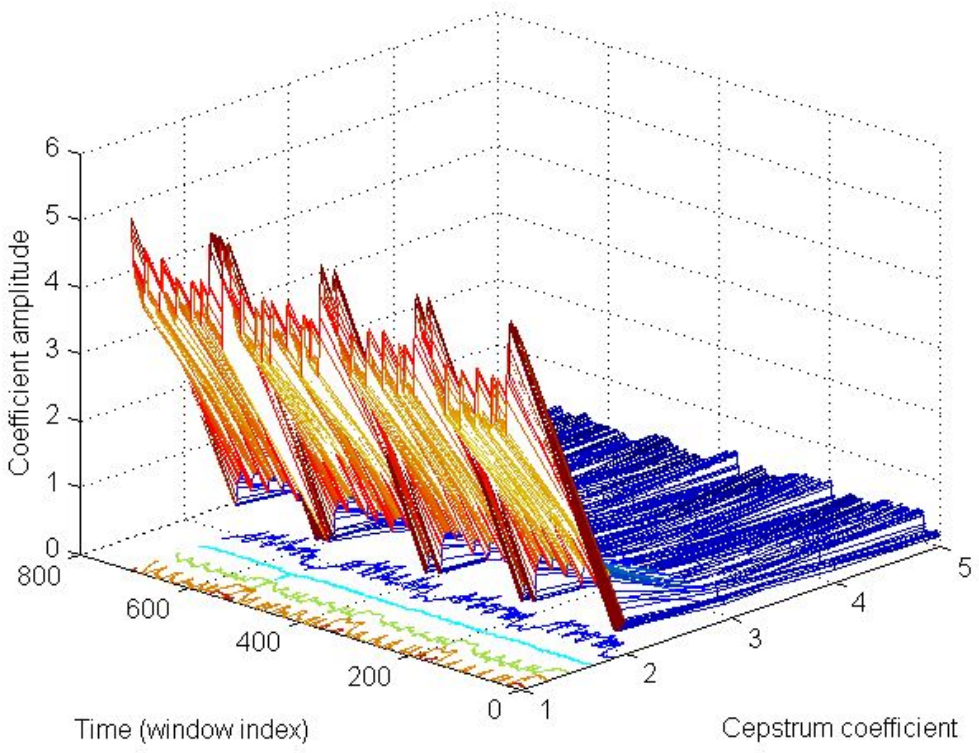

Figure 3-11: Absolute values of the first 5 cepstral coefficients in the signal

Denoting the first 2 cepstral coefficient by C_1 and C_2, the variances of C_1 and C_2 are 0.2339 and 0.0644. Since we only need to distinguish one feature of the vocal tract (openness), it is reasonable that one single value can be used for the classification. Thus we wanted to combine the 2 coefficients into a single value. An easy and straightforward way is to investigate the variance of different combinations of the 2 coefficients. Since C_1 is mostly negative and C_2 is mostly positive, there are 2 simple combinations of the 2 coefficients: $|C_1 + C_2|$ and $|C_1 - C_2|$. The variances of the combinations are 0.0614 and 0.5351 respectively. We thus finally choose $F = |C_1 - C_2|$ to characterize the formant features of the signal.

Figure 3-12 (a) shows the formant feature F of the signal along time. A higher F value indicates the closure (less openness) of vocal tract, and a lower F value indicates high

openness of the vocal tract. A dip of the value indicates the opening of the vocal tract, which happens during the transitions from consonants to vowels. The dips of the value at frames 180, 350, and 540 however do not corresponds to a vowel, but breath inspiration, which is also accompanied by opening of vocal tract.

A vocal tract opening change function is defined to detect the opening of vocal tract:

$D(t) = \sum_{k=t-W}^{t-1} F(t) - \sum_{k=t+1}^{t+W} F(t)$, where t is the time (window index) of the frame, W is the delta window size to compute the difference of $F(t)$. In the example, $W = 3$. Figure 3-12 (b) shows the values of $D(t)$. Each value peak corresponds to the articulation of a vocal tract opening. The detection of the vocal tract opening is done by comparing $D(t)$ with a threshold T_D. If $D(t_a) > T_D$, then a vocal tract opening is detected at time t_a. In practice, $T_D = 1$.

Figure 3-12 (c) illustrates the detection result, where each vertical bar indicates one vocal tract opening.

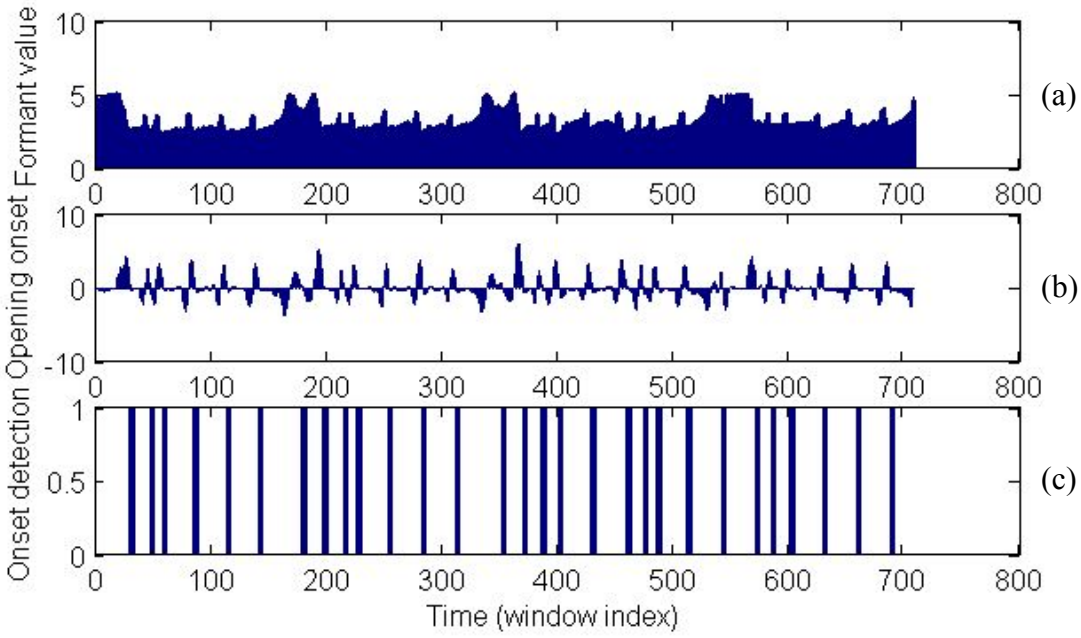

Figure 3-12: Formant feature value and onset of vocal tract opening

3.2.3.3. Vowel Onset and Breath Inspiration Detection

The vocal tract opening detected can be either onset of a vowel or onset of a breath inspiration. The distinction of these 2 cases can be done based on the energy of the signal for the corresponding frames. An energy threshold T_e can be used this purpose. If the energy of the detected onset frame is larger than T_e, then a vowel onset is claimed, otherwise a breath inspiration onset is claimed. T_e can be automatically learned from the onsets detected from a single humming signal. The energy difference between vowel and inspiration is usually very significant.

Figure 3-13 illustrates the vowel onset and inspiration onset detection result.

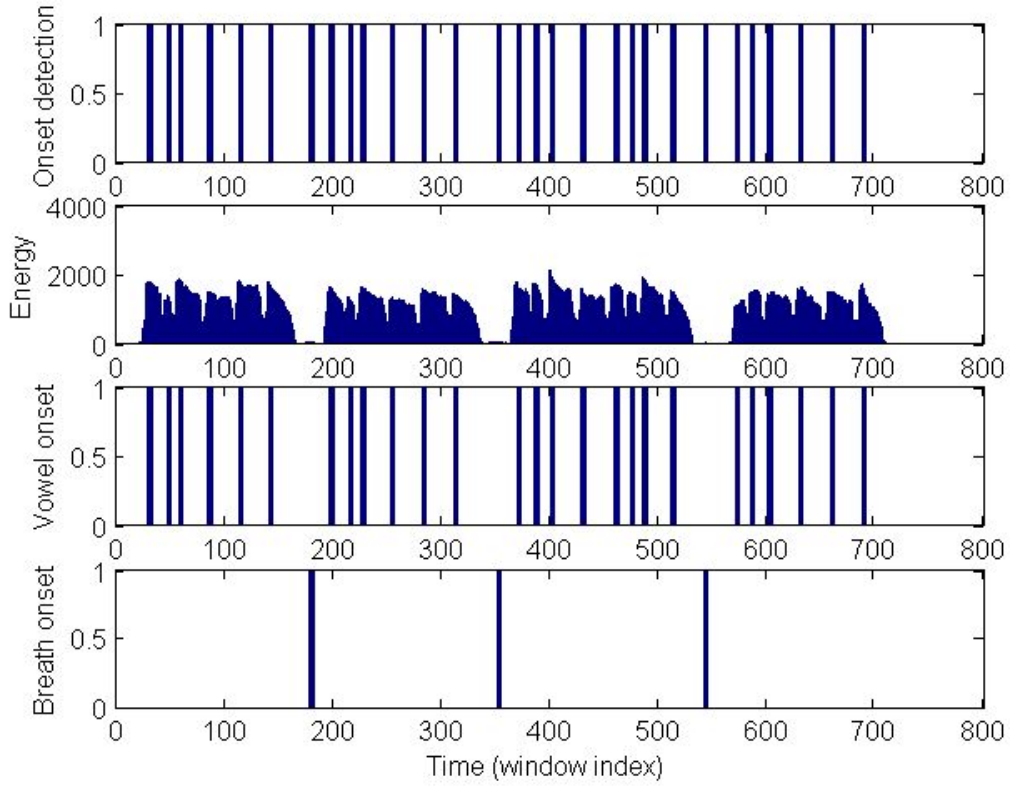

Figure 3-13: Detection of vowel onset and inspiration onset

3.2.4. Reliable Pitch Detection and Tracking

In the proposed approach, pitch detection is conducted by locating the peaks in the cepstrum, as shown in figure 3-9 (a) and (b). The index value of the peak in the cepstrum corresponds to the period (inverse of frequency) of the pitch.

Reliable pitch detection is done for the frames of vowel onset, where the pitch should be prominent and reflective for user's intention of pitch. Pitch tracking is done for the frame following the vowel onset frames based on the detected reliable pitches.

3.2.4.1. Reliable Pitch Detection

Pitch detection for the vowel onset frame is done by locating the high peak in the high range cepstral coefficients. The range corresponding to the frequency range 100-880Hz are

considered in peak finding, since this range covers Baritone (male middle range) to Mezzo-Soprano (female middle range).

Denoting the index of the peak by T_1 and the magnitude of the peak by M_1. The frequency for this peak is denoted by P_1. As shown in figure 3-14, occasionally due to abnormal vocal condition, half pitches can appear, and the peak with highest magnitude may not be the right pitch, but the peak on $T_1/2$ corresponds to the right pitch.

To handle those cases and promote the robustness of the pitch detection, the pitch corresponding to double of the frequency $2P_1$ is checked by locating the local peak around $T_1/2$, if $2P_1$ is still within the considered pitch range. The magnitude of the local peak is denoted by M_2. A third peak is then located in the range $T_1/2+\Delta$ to $T_1-\Delta$, where Δ is a margin (e.g. $T_1/4$). The magnitude of the third peak is denoted by M_3.

If $M_2/M_1 > T_{M1}$ and $M_3/M_1 < T_{M2}$, then $T_1/2$ or $2P_1$ is claimed as the detected pitch. In practice, $T_{M1}=0.5$ and $T_{M2}=0.5$. The first condition stands for that M_2 is large enough to be considered a right peak, while the second condition means there is no other peaks between the 2 peaks.

And if $M_2/M_1 < T_{M1}$ or $M_3/M_1 > T_{M2}$, then P_1 is claimed as the detected pitch.

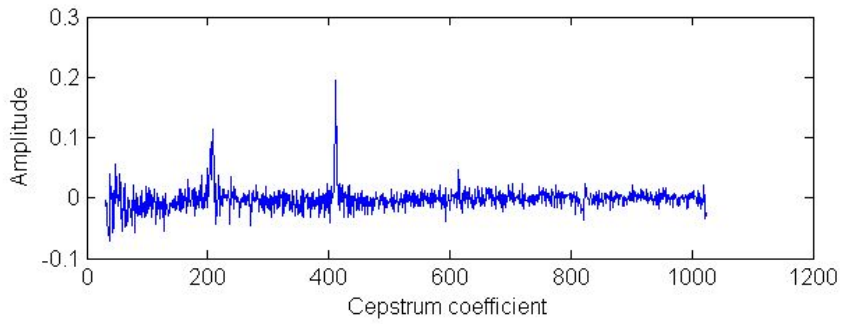

Figure 3-14: Half pitch in the cepstrum (the peak at 205 is for the right pitch)

The pitch detected for the vowel onset for the above example is shown in figure 3-15.

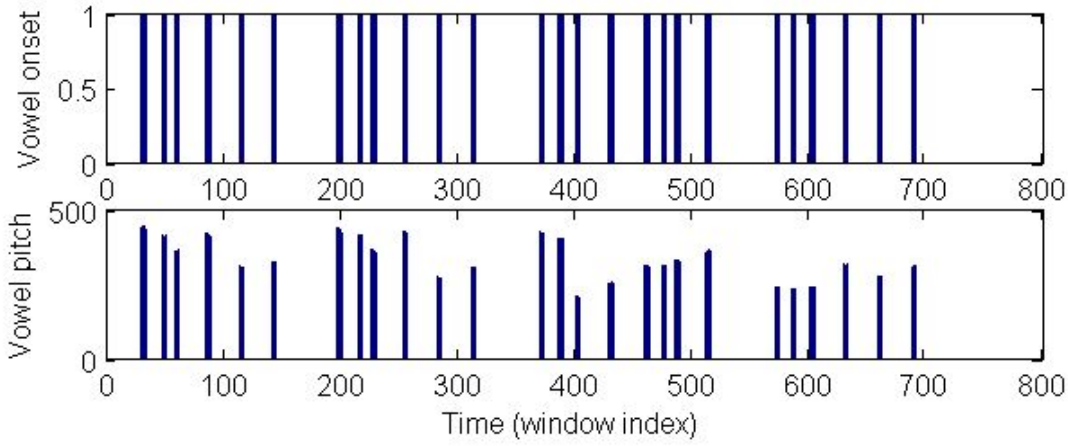

Figure 3-15: Reliable pitch on vowel onset

3.2.4.2. Pitch Tracking

Pitch tracking is the process following the reliable pitch detection, which tracks the pitch of the frames following the vowel onset based on the detected pitch. The difference between pitch tracking and pitch detection is that double pitch or half pitch errors in pitch detection would not be presented in pitch tracking. This is achieved by the assumption that pitch changes continuously and smoothly in a single articulation of vowel. The tracking is done by locating peaks in cepstrum that are nearby to the right peak of the previous frame. The pitch tracking for each vowel stops at the frames, where the vowel stops, which can be detected by verifying $D(t) < T_{D2}$. In practice, T_{D2} can be set to –1.

Figure 3-16, illustrates the final pitch tracking result. Figure 3-16 (a) shows the pitch in period, and (b) shows the pitch in frequency. Figure 3-16 (c) illustrate the complete time frames for breath inspiration.

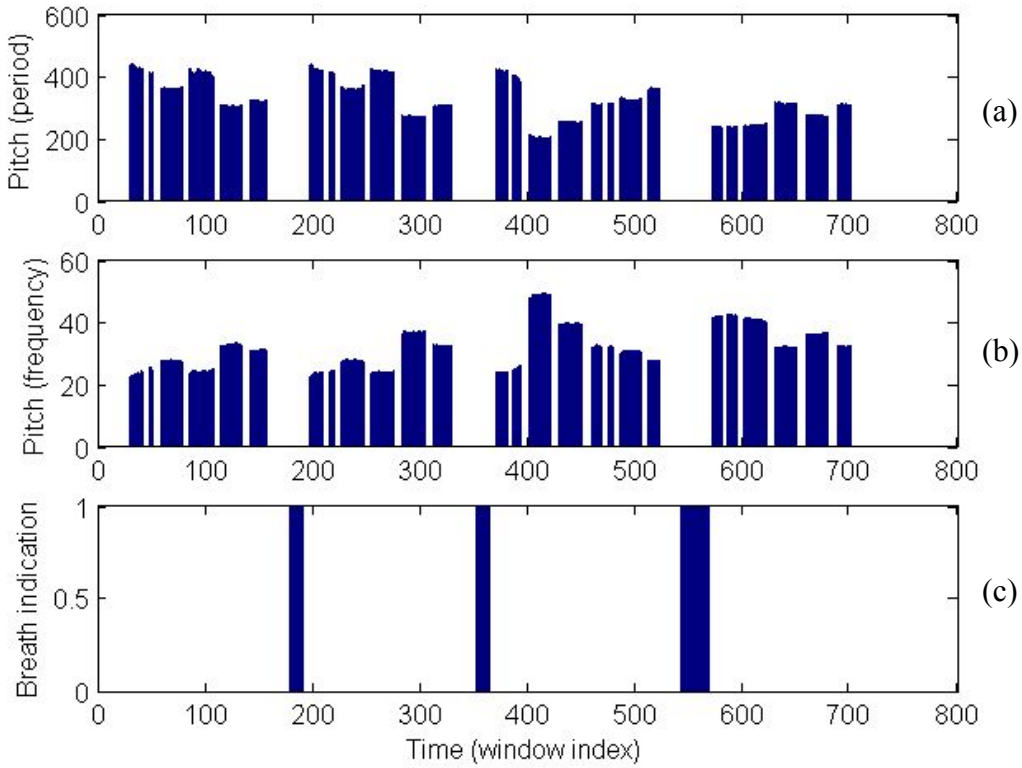

Figure 3-16: Final pitch detection and tracking result

3.2.5. Pitch Curve Aggregation

The result of pitch detection and tracking is the pitch contour (pitch curve) of the humming signal, which is then be converted into pitch lines using the following proposed algorithm.

As shown in figure 3-17(a), the pitch curve for a hum contains many value perturbances. We have developed a piece-wise line segment approximation method to approximate the pitch curve by a sequence of horizontal line segments, which is the pitch line. The method is described as follows:

0. Preprocessing: Converting the pitch levels in frequency to note scale values by a logarithmic scaling. All the pitch levels are floating point numbers larger than zero, and zero values stand for non-pitch in the frame, which represent the silent pauses in the humming.

1. Remove the zero values in the pitch curve, so that all the numbers in the pitch curve are values corresponding to pitch levels. Denote the new pitch curve by *qv[i], (1<= i <=n),* where *i* is the index and *n* is the length of the new pitch curve. Represent each value *qv[i]* as a horizontal line segment *(sv[j],sl[j]),* where *sv[j]* is the pitch value of the j^{th} line segment and *sl[j*] is its length. Initially, there are *n* horizontal line segments. Merge all temporally consecutive line segments with a same pitch value into one line segment, and update the length of the line segment. This sequences of horizontal line segments are denoted by *(sv[j],sl[j]) (1<= j <=m),* where *m* is the number of line segments.

2. For each two consecutive line segments *(sv[j],sl[j]), (sv[j+1],sl[j+1])*, compute the total approximation error *e[j]* for a new line segment which is obtained by merging the two original line segments, where

$$e[j] = \sum_{k=0}^{L-1} |qv[s+k] - V| \qquad (3.1)$$

$$L = sl[j] + sl[j+1] \qquad (3.2)$$

$$V = \left(sv[j] \times sl[j] + sv[j+1] \times sl[j+1]\right) / L \qquad (3.3)$$

s is the index of the value item in *qv[i]* which corresponding to the beginning of *sv[j]*.

3. Find *e[x]* which is minimum in *e[j] (1<=j<=m-1),* i.e. $x = \arg\min_j e[j]$. Merge the two line segments *(sv[j],sl[j])* and *(sv[j+1],sl[j+1])* into a new line segment *(sv'[j],sl'[j]),* where *sv'[j]=V*, *sl'[j]=L*, *V* and *L* are computed using the same equation as in the previous step.

4. Replace *(sv[j],sl[j])* by *(sv'[j],sl'[j])*, remove *(svl[j+1],sl[j+1])*, and shift the line segments behind one step ahead. The number line segments is reduced by *1: m←m-1*.

5. Find the minimum value difference in the line segment sequence: $d_{\min} = \min_j |sv[j] - sv[j+1]|$. If $d_{min} > Thres_d$, then go to step 6; otherwise go to step 2. $Thres_d$ is a threshold value, which takes value of one semitone in our experiments.

6. Pad the non-pitch frames in the original pitch curve into the line segment sequence according to the position of the frames. The pitch values of the line segments are not affected. Only the lengths of the line segments need to be updated.

This approximation method is a recursive line segment merging process. The result of the process is a sequence of horizontal line segments, which has the minimum global approximation error with the original time series while satisfying the following requirement: the pitch value distance between any two contiguous line segments is greater than a threshold (e.g. 1 semitone). This method is invariant to the sampling frequency of the time series. Figure 3-17(b) shows the line segments approximating the time series shown in figure 3-17(a). In figure 3-17(b) the gaps (non-pitch frames) in 3.17(a) are padded.

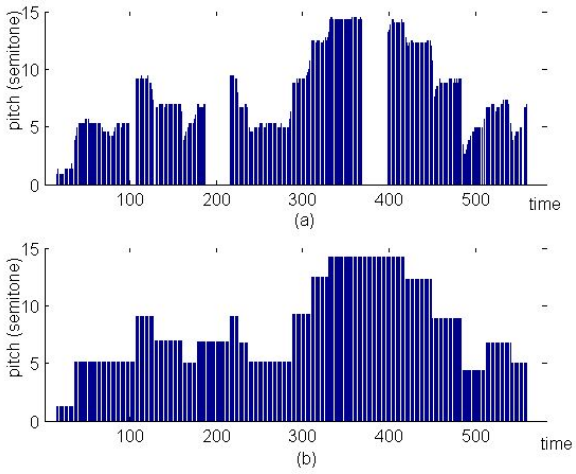

Figure 3-17: (a) The pitch curve for a hummed melody; (b) The pitch line for the pitch curve

The pitch line is a compact and optimal (in the sense of the global approximation error) representation of the original time series data while preserving the main structure. The pitch

values of the line segments are real numbers, which still preserves the fidelity of the original query data. A very important advantage of this approximation method is that the peak segments and valley segments can be quite reliably identified in hummed queries. These peak and valley segments are the anchor points that will be used for sequence alignment.

The pitch line produced by curve aggregation is at the sentence level. It is broken down to the phrase level by inserting the boundaries caused by the breath onsets, and further down to the word level by inserting the boundaries caused by the vowel onsets.

3.3. Construction of Pitch Line from Symbolic Input

To construct the pitch line for the melody of a MIDI file, the monophonic music notes are extracted from the melody track in a MIDI file, and each note is converted to a horizontal line segments. The height of the line segments corresponds to the note value (absolute pitch) and the length of the line segment corresponds to note duration. Two or more consecutive music notes with the same note value will correspond to one line segment and its length will correspond to the total duration of all the notes. When there is silence or rest in the melody, the duration of the silence is added to the duration of the previous non-silence note.

The phrase boundary can be claimed, when the silence or rest is longer than 400ms or a single note is longer than 800ms. And each note can be claimed as a word in the pitch line representation. Figure 3-18 illustrates the word, phrase and sentence segments for the melody of “Happy Birthday To You” using the piano roll of the song.

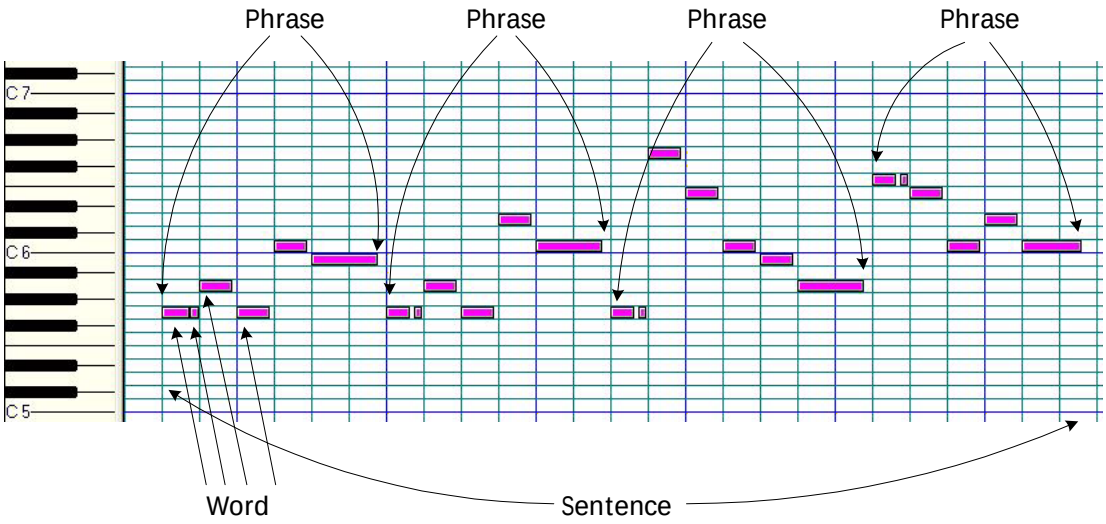

Figure 3-18: Word, phrase, and sentence of the melody of "Happy Birthday To You"

Since the music files we deal with are Karaoke MIDI files, the melody tracks in the MIDI files are mostly monophonic. So, as long as the melody track is identified, the melody can be reliably extracted and inserted into the system. A tool to assist automatically identifying the melody track in the MIDI file has been developed and implemented for building the melody database. The tool will be described in chapter 7.

3.4. Summary

This chapter has presented a novel melody representation, called pitch line. Unlike previous discrete melody representation based on strings of relative pitch, pitch line is based on continuous absolute pitch. The major advantage of pitch line is that it will not be affected by errors in note segmentation either in signal level (caused by algorithm) or music level (caused by user). Another advantage of pitch line is the compactness compared with raw pitch contour in terms of storage. As will be shown in the following chapters, global features of pitch line can be extracted and utilized for robust and efficient melody matching.

This chapter has also presented a pitch extraction method for converting humming signals into pitch line melody representations. A voice segmentation method based on vocal features

(vowel and consonants) has been proposed to locate time frames in the signal that possess reliable and intended pitch, which corresponds to onset of vowels. The onset of vowel can also be seen as boundaries of word level of pitch line. The segmentation method can also identify breath inspirations in the signal, which corresponds to phrase level boundaries of pitch line.

A technique for pitch detection and tracking at cepstrum domain has also been proposed. The double or half pitch errors caused by various vocal conditions have been considered in the method. The pitches for vowel onsets are firstly detected, and pitches for the following time frames are tracked based on the detected reliable pitch.

The pitch line representation is produced with a pitch curve aggregation algorithm, which in fact works as a feature dimension reduction process.

Finally, a method for deriving pitch line from Karaoke MIDI files has been discussed.

After the description of melody representation, the remaining parts of the thesis will mainly focus on melody matching methods based on the pitch line representation. Matching robustness, accuracy and efficiency are the important issues in melody matching. In this thesis, the matching of pitch line is conducted at sentence level, because we assume that the user may fail to consistently produce correct phrase boundaries or word boundaries. However, it is believed that pitch line matching at phrase level and word level shall have better retrieval accuracy, which is not covered by the thesis.

Chapter 4.
MELODY SIMILARITY MATCHING

Music retrieval by melody is based on computing the similarities or distances between a query melody and the corpus melodies. The corpus melody items with high similarity scores or low distance values are returned as the retrieval result. There are a few concerns in similarity matching of melodies. Firstly, the melody corpus can be potentially very large and a query melody could be inaccurate, thus an effective similarity/distance metric for melody comparison is very important. Secondly, melody could be freely transposed across keys, so the query melody has to be pitch-shifted to a correct key before it can be compared to a corpus melody. Furthermore, the query melody is usually incomplete (an arbitrary portion of the complete tune) and could contain tempo variations, so time warping or alignment of melodies is necessary in melody similarity matching.

This chapter describes a novel technique of melody similarity matching based on pitch line. The focuses are melody similarity metric, key transposition and melody alignment. Section 4.1 discusses relevant issues in melody similarity matching. Section 4.2 presents a melody similarity metric based on pitch lines. Section 4.3 presents a pitch shifting and melody alignment method by using a structural feature of pitch lines.

4.1. Issues in Melody Similarity Matching

There are a few issues to be considered for melody similarity matching based on pitch lines.

4.1.1. Key Transposition

A music piece is created (composed) by musician usually with a specific key (or reference pitch), and each note in the melody has a fixed absolute pitch. However, when the music is performed, the key can be transposed to other pitches if needed, such as to fit the pitch range of a particular music instrument. Key transposition changes the pitch of every note of a melody by a same quantity, and does not change the melody identity. When humming a tune, a female person usually uses a higher key than a male person. In doing melody matching by pitch lines, the key of the query melody need to be transposed to the key of the original melody for similarity measurements. Figure 4-1 illustrates a query melody is pitch-shifted to a corpus melody, as a result the pitch lines in the query melody can find correspondence in the corpus melody and a distance can be computed.

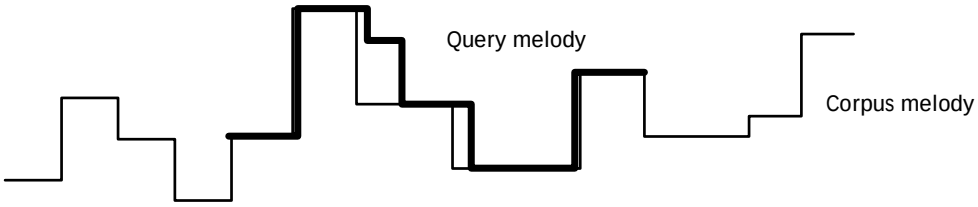

Figure 4-1: Key transposition for melody matching

4.1.2. Tempo Variation

Tempo characterizes the speed of progression of a music piece, which is measured in beats per minute (BPM). Although the tempo of a music piece could be specified in the composition, it can vary (to be faster or slower) between different performances. It is also possible that the tempo changes during a single performance, especially in casual humming. In order to avoid undesirable mismatching of the melodies due to tempo variation, a time scaling or warping of the pitch lines of the query melody is necessary in doing melody similarity matching. Figure 4-2 illustrates a query melody with slower tempo needs to be warped in matching with a corpus melody.

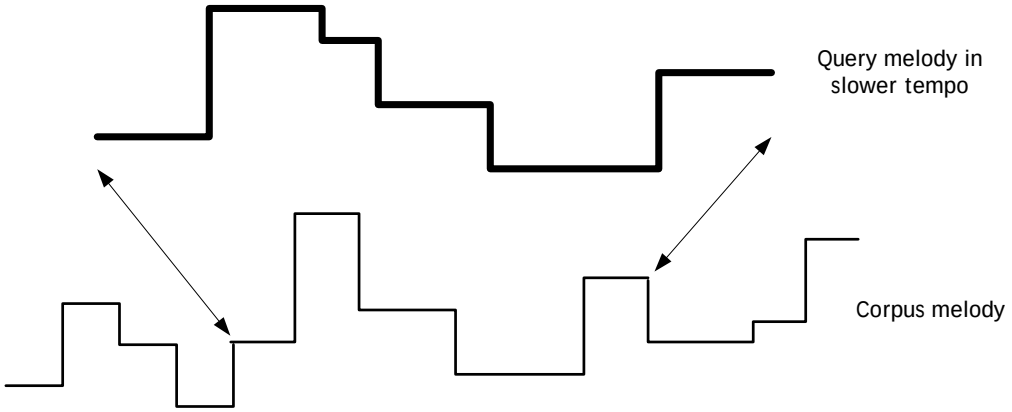

Figure 4-2: Tempo variation in melody matching

4.1.3. Subsequence Matching

A melody is represented by a sequence of pitch line segments. The complete melody of a music piece potentially is a long sequence. But a query melody is usually just a small portion of the whole melody, and might begin anywhere in the middle of the original melody. Thus similarity matching of pitch lines is a subsequence matching problem, i.e. sliding the query melody along the candidate melody and computing the similarity at each sliding position. Figure 4-3 illustrates the subsequence matching process.

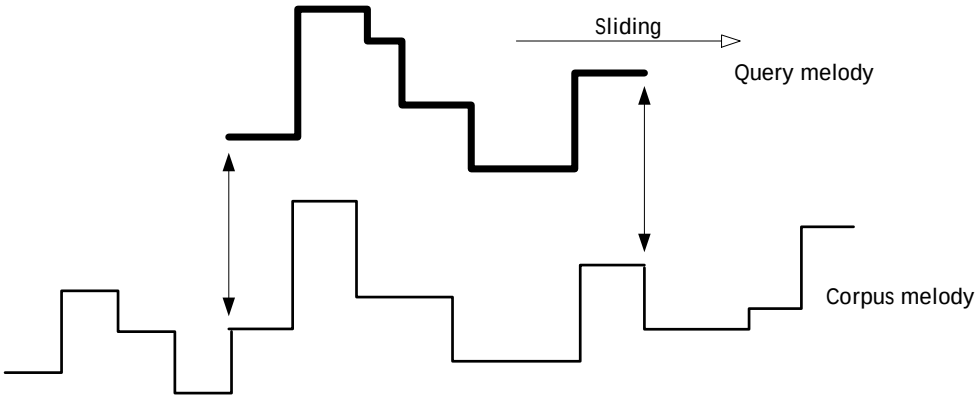

Figure 4-3: Subsequence matching issue in melody matching

If the melody can be segmented into predefined units, such as bars (measures) or beats, an alternative matching approach [Kosu_MM00] can be used, such as deriving a feature vector

from a fixed number of units, and matching melodies by computing the distance of the feature vectors. However, it is very hard, if at all possible, to do beat analysis or rhythm analysis based on melody information, especially the short and varying query melodies. This is because people could indicate mostly the correct pitch but hardly the right strength in humming.

In this thesis, melody similarity matching is based on the subsequence matching approach.

4.1.4. Pitch Inaccuracy

Most musical instruments, such as clarinet, piano and guitar, are tuned to accurate pitch levels, so that each note played has an exact and accurate pitch. Some string bowed instruments, like violin, cannot be tuned for each note. Then the pitch of each played note has to be judged by the ears of the performers. Similarly human voice is also free of tuning, and only trained singers can produce accurate (tuned) pitch. On the contrary, laypersons often sing "off keys" and sometimes cannot keep a stable pitch for one note, which causes pitch inaccuracy in the hummed queries. These cases of pitch inaccuracy are intricately different from the cases of "out of tune". In the "off key" cases, the pitch error is due to the poor vocal skill. In the "out of tune" cases, wrong notes are used such that the identity of the melody is lost. This is usually due to failing to recall the melody from memory.

The pitch inaccuracy due to poor singing skill is considered in the similarity metric of pitch lines.

4.1.5. Procedures for Similarity Matching of Pitch Lines

In this thesis work, melody similarity matching is based on matching of pitch lines. The general idea in pitch line matching is to transpose and piece-wise align the pitch line segments of a query melody to the target corpus melody and compute a similarity score for a particular

transposition and alignment. The procedure of melody similarity matching is illustrated in figure 4-4.

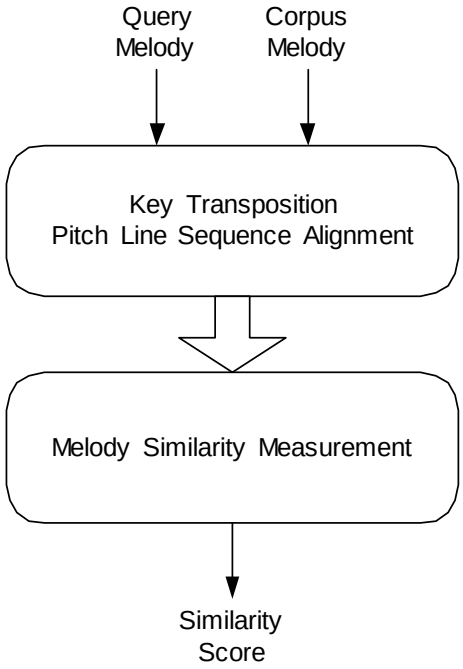

Figure 4-4: Procedures for melody similarity matching

The key transposition step determines the amount of pitch shifting between the 2 melodies. The melody alignment step targets to solve the tempo variation and subsequence matching issues. The result of melody alignment is an establishment of correspondences (or mapping) between the pitch line segments of the 2 melodies under comparison.

Melody similarity measurement is to compute the similarity score for a particular alignment based on the pitch difference of corresponding pitch line segments.

A melody similarity metric for pitch line is firstly presented in section 4.2. A key transposition and pitch line sequence alignment technique is presented in section 4.3.

4.2. Melody Similarity Metric

This section presents a melody similarity metric, based on which a similarity score for 2 pitch-shifted and aligned melodies can be computed. The mathematical definitions of key transposition and pitch line sequence alignment are given below.

Definition: Key Transposition

Denoting the pitch line segment sequence for a query melody by $[q_i]$ $(1 \le i \le m)$, where m is the number of pitch line segments and q_i is the pitch of the i^{th} segment. q_i is relative to a reference pitch q_0, so the absolute pitch of the i^{th} segment is $q_0 + q_i$.

Similarly denoting a corpus melody by $[p_j]$ $(1 \le j \le n)$, where n is the number of pitch line segments and p_j is the relative pitch of the j^{th} segment. The reference pitch is p_0, and the absolute pitch of each pitch segment is $p_0 + p_j$.

Key transposition of the 2 melodies is defined as a pitch shifting value T, such that the pitch difference between any q_i and any p_j can be evaluated as $\left|T + (q_0 + q_i) - (p_0 + p_j)\right|$.

■

Definition: Pitch Line Sequence Alignment

Denoting the pitch line segment sequence for a query melody by $[q_i]$ $(1 \le i \le m)$, where m is the number of pitch line segments. And denoting a corpus melody by $[p_j]$ $(1 \le j \le n)$, where n is the number of pitch line segments. An alignment of $[q_i]$ with $[p_j]$ can be defined as a sequence of correspondences $[A_k = (i_k, j_k)]$ $(1 \le k \le K)$, where K is the number of correspondences in the alignment, i_k is the suffix of q_i, and j_k is the suffix of p_j. A valid alignment A_k is required to meet the following conditions:

(1) $i_1 = 1$ and $i_K = m$

(2) $1 \le i_k \le m$ and $1 \le j_k \le n$ (for any k)

(3) $i_k \le i_{k+1}$ and $j_k \le j_{k+1}$ (for any k)

(4) if $i_k = i_{k+1}$ then $j_k < j_{k+1}$ and $j_{k+1} < j_{k+2}$ (for any k)

(5) if $j_k = j_{k+1}$ then $i_k < i_{k+1}$ and $i_{k+1} < i_{k+2}$ (for any k)

■

In the above pitch line sequence alignment definition, condition (1) specifies that an alignment starts with the first segment of the query melody and ends with the last segment of the query melody. Conditions (2) and (3) specify the range and sequential order of segments for the alignment. Conditions (4) and (5) specify that for any single pitch line segment there could be 1-to-1 mapping or 1-to-many mapping or many-to-1 mapping, but NO many-to-many mapping. In another words, there should be no alignment like […, $(i, j-1)$ (i, j) $(i+1, j)$, …]

An alignment of pitch lines can also be illustrated using a table as shown in figure 4-5. The alignment of q1, …, q8 with p3, …, p10 can be seen as a path (A1 to A10) in the table.

	q1	q2	q3	q4	q5	q6	q7	q8
p1								
p2								
p3	A1							
p4		A2	A3					
p5				A4				
p6					A5			
p7					A6			
p8						A7		
p9						A8		
p10							A9	A10
p11								

Figure 4-5: Melody alignment illustrated as a path through a table

The alignment shown in figure 4-6 is invalid, since $i_2 = i_3 = 1$, but $j_3 = j_4 = 3$, which violates the 4th condition in the definition of pitch line sequence alignment.

	q1	q2	q3
p1	A1		
p2	A2		
p3	A3	A4	A5

Figure 4-6: An invalid alignment

With key transposition T and melody alignment [A_k] specified, a local pitch distance d_k can be defined for each A_k, where $d_k = \left| T + \left(q_0 + q_{i_k} \right) - \left(p_0 + p_{j_k} \right) \right|$. The distance between the query melody with the corpus melody for the specified key transposition and alignment can then be calculated.

We propose the following melody similarity metrics:

(1) $D_1 = \sum_{k=1}^{K} d_k$. (4.1)

Metric D_1 measures the distance of 2 melodies by the accumulated pitch distance. This type of similarity measure is also traditionally used in sequence comparison.

(2) $D_2 = \sum_{k=1}^{K} d_k w_k$, where w_k is a weighting factor. (4.2)

In metric D_2, the local pitch distance is weighted before summing up. The weighting factor w_k can be set to the number of voiced frames in the pitch line segment of the query melody. The voice frames are those frames with pitch detected and not those frame with pitch not detected (e.g. silence). So a pitch line segment of query melody with more voiced frames will have higher weight.

D_2 is the modified version of D_1, where different pitch line segment has different weight in the distance computation. The weighting is reasonable, since the pitch line segment that contains longer voicing duration should be more important that those with shorter voicing durations.

(3) $D_3 = \sum_{k=1}^{K} F(d_k)$, where $F(\ )$ is a function of pitch difference. (4.3)

In metric D_3, we consider the pitch inaccuracy normally presented in the hummed queries. We set a margin for pitch inaccuracy, within which the pitch error is ignored. This is achieved by the margin function $F(\ )$:

$F(d) = 0$ if $d \le Th_d$

$F(d) = d$ if $d > Th_d$

where Th_d is a threshold value.

Function $F(\ )$ is illustrated in figure 4-7.

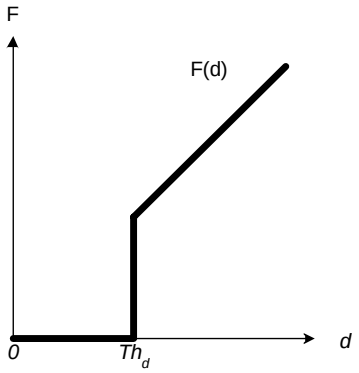

Figure 4-7: The margin function F(d)

D_3 targets compensating the pitch inaccuracy in the humming by a nonlinear mapping of the pitch difference. The intuitive idea is that a small pitch error is more acceptable, thus should not be accumulated in the distance measure. In practice, the value for Th_d could be determined by the level of pitch accuracy of the user. For a trained singer the value should be small, e.g. 0.2 semitones. For a layperson user, the value around 0.5 semitones is more appropriate. In this thesis, the threshold value is 0.5 semitones, which is determined from experiments.

(4) $D_4 = \sum_{k=1}^{K} F(d_k) w_k$. (4.4)

Metric D_4 includes both segment weighting and margin function. Metric D_4 is a superset of D_3, D_2 and D_1. When $w_k = 1$ (for all k) and $Th_d = 0$ for $F(\ )$, D_4 is same as D_1.

The setting of threshold Th_d and performance of the similarity metrics in music retrieval is discussed in chapter 7 (experimentation). How to get key transposition T and the sequence

alignment [A_k] is the main subject of this work and is addressed in the remaining parts of this thesis.

D_4 is a combination of D_2 and D_3, where both pitch inaccuracy and weighting are considered.

Table 4-1 summarizes the proposed melody similarity measures.

Table 4-1: Summary of the similarity metrics

Similarity Metrics	D_1	D_2	D_3	D_4
Definition	Accumulated pitch distance	Local temporal weighting	Pitch inaccuracy compensation	Temporal weighting and pitch inaccuracy compensation
Usage	Traditional sequence matching	Sequence items having different importance, e.g. pitch line segment with longer voicing time has higher weight	Sequence items having value variations (errors) within a certain small range, e.g. 0.5 semitone errors should be ignored.	Combination of D_2 and D_3.

4.3. Key Transposition and Melody Alignment by Melody Slope

In melody representation by pitch lines, the pitch line segments corresponding to local maximum or local minimum, can be easily identified, and we call them peaks and valleys. We define the portion of a pitch line, starting from the beginning of a peak or a valley and finishing at the beginning of the next valley or peak, a *melody slope*. So a melody slope is a small portion of a melody, in which the pitch changes only in one direction, either increasing or decreasing. A melody slope has a *pitch range value*, which is the relative pitch difference between the corresponding peak and valley, and a *duration value*, which is the length of the melody slope. Figure 3-10 shows the melody slopes for a melody in pitch lines.

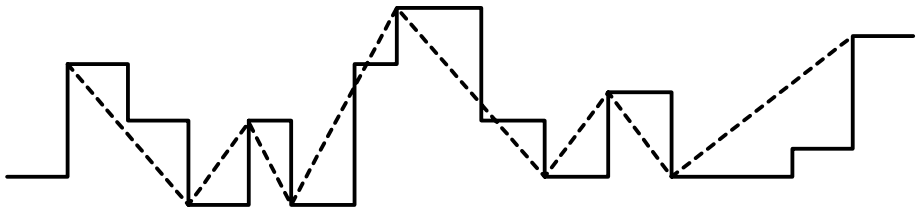

Figure 4-8: Melody slopes of pitch lines

Key transposition and melody alignment stands for temporally scaling and pitch-wise shifting the pitch line sequence of a query melody, such that it is positioned "closely" with the target corpus melody and a similarity can be computed. We propose a pitch line transposition and alignment method that utilizes melody slopes. This method targets two objectives in melody matching: 1) robust transposition and alignment of pitch lines, 2) the method also serves as a preliminary filtering step to reject, as many as possible, unlikely melody candidates without rejecting the desired candidate.

The basic idea in the method is to locate the correspondents of a sequence of melody slopes of the query melody in the melody slope sequence of the corpus melody. The pitch range values of melody slopes are used to determine whether two melody slope sequences potentially match. The duration values of the 2 melody slope sequences can also be used to in the

matching. A correlation of the duration of the 2 slope sequences is computed. The computation of pitch difference and duration correlation is as following equations.

$$dist_{slope} = \left(\sum_{i=1}^{n} \left| (I_1(i) - I_2(i)) \right| \right) / n \qquad (4.5)$$

$$corr_{slope} = \sqrt{\frac{\sum_{i=1}^{n} (L_1(i) \times L_2(i))}{\left(\sum_{i=1}^{n} (L_1(i))^2 \right) \left(\sum_{i=1}^{n} (L_2(i))^2 \right)}} \qquad (4.6)$$

where $I_1(i)$ stands for the pitch range of the i^{th} slope in query melody and $I_2(i)$ is for the corpus melody. n is the number of slope in a sequence for matching. L_1 and L_2 are the duration value of the slopes in respective melody. A match is claimed when $dist_{slope}$ is smaller than a threshold and $corr_{slope}$ is bigger than a threshold.

When there is a slope sequence match, the pitch lines of a pair of corresponding melody slopes are mapped to each other by the time overlapping, i.e. a pitch line segment can map to any pitch line segment of the other melody that has time overlapping with it. If a many-to-many mapping occurs, the mapping that with shorter time overlapping is sequentially removed until there is no many-to-many mapping between the 2 pitch line sequences. Figure 4-8 illustrates the pitch line alignment using melody slopes.

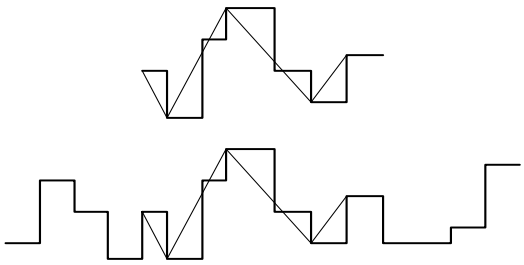

Figure 4-9: Pitch line alignment of 2 melody by melody slopes

After pitch line sequence alignment, pitch shifting is conducted. The basic idea is to compute the centroid (mean pitch) of two pitch line sequences, and use the centroid to vertically shift the two contours and finally measure the difference of the two contours segment by segment. The process is presented as follows:

The melody contour of a query melody is denoted as $\{A(i), B(i)\}, i = 1 \cdots N_1$, and the melody contour of a reference melody is denoted as $\{C(j), D(j)\}, j = 1 \cdots N_2$. where A(i) is the pitch value of the i[th] horizontal line segment in the query contour, and B(i) is the length of the line segment. C(j) and D(j) are those values for the reference melody contour. N_1 and N_2 are the number of line segments in the melody contours respectively. We assuming the pitch value of the first line segment in the contour is 0. The centroid of the contours are computed respectively as

$$G_1 = \frac{\sum_{i=1}^{N_1} A(i)B(i)}{\sum_{i=1}^{N_1} B(i)} \tag{4.7}$$

and

$$G_2 = \frac{\sum_{j=1}^{N_2} C(j)D(j)}{\sum_{j=1}^{N_2} D(j)} \tag{4.8}$$

With the centroid of the contour, the two melody contours can be put together and distance between the line segments of the two contours can be calculated. This is illustrated in figure 4-10.

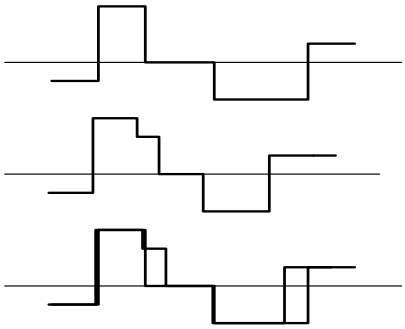

Figure 4-10: Pitch shifting of pitch lines

4.4. Summary

This chapter has presented the issues in melody matching using pitch line melody representation: key transposition, tempo variation, subsequence matching and pitch inaccuracy. A general melody matching approach has been proposed: key transposition and pitch line alignment followed by melody similarity measure (figure 4-4).

Several melody similarity measures have been proposed. The similarity measures are based on particular melody key transposition and alignment. Weighting in the similarity measures have been proposed to cater for various local importance of pitch line as well as pitch inaccuracy.

A key transposition and pitch line alignment technique using melody slope has been then proposed. Melody slope sequence matching can achieve efficient alignment of pitch lines. The key transposition is done based on the mean pitch of the 2 aligned pitch line sequences.

The melody matching approach presented in this chapter is the first approach for matching pitch lines. This is also the first approach that separates the melody transposition and alignment with similarity measurement. In fact, the melody alignment and transposition is based on a global feature of the melody: melody slope sequence. The global feature is both robust and efficient for melody alignment than local features.

Past methods only work on raw pitch contours, while sequence alignment using local features is done as same time as similarity is computed. Such methods are both intensive in computation and sensitive to errors.

Chapter 5. MELODY SKELETON

5.1. Melody Skeleton

For a melody representation, such as pitch line, pitch information is more important than time information. This is because time can vary significantly while still reserving the identity of the melody. But if the pitch changes, the melody is usually perceived different. To deal with the tempo variation and inconsistency, we map the line segment sequence (represented in the time domain) to a sequence of points in a newly proposed *value-run* domain.

Tempo is a malleable parameter in music performance. Any tune can be performed at different tempos. Variation of the tempo will lead to variation in the lengths of the line segments in the previously discussed melody representation. This could complicate the matching between two sequences of line segments. This is because for subsequence search, the shorter sequence needs to be stretched or squeezed "properly" before scanning through the long sequence and doing the comparison at any possible position. That means the matching is done for every modified version of the query. And the situation is even worse if the tempo is inconsistent, so the time warping is non-linear. Such problem would be computationally intractable. To the best of our knowledge, there has been no elegant solution for this problem.

To deal with the tempo variation and inconsistency, we map the line segment sequence (represented in the time domain) to a sequence of points in a newly proposed *value-run* domain.

Definition 1: Given a real valued data sequence v[i], where i is the sequence index, the *value-run* from the a^{th} value to the b^{th} value *R[a,b]* is defined as follows:

$$R[a,b] = \sum_{i=a+1}^{b} \left| v[i] - v[i-1] \right|, \qquad \text{if b > a} \qquad (5.1)$$

$$R[a,b] = 0, \qquad \text{if a = b} \qquad (5.2)$$

■

Definition 2: Mapping of a line segment sequence to points in the value-run domain: Denote a line segment sequence by *(sv[i], sl[i]),* where *i* is the sequence index ($1 \le i \le N$), *sv[i]* is the value of the ith line segment and *sl[i]* is the length of the i^{th} line segment, and *N* is the number of line segments in the sequence. Each line segment *(sv[i], sl[i])* is mapped to a point *(v[i], R[1,i])* in the value run domain, where *v[i]* is still the sequence value *sv[i]* and *R[1,i]* is the value-run of *sv[i]* from the first line segment to the i^{th} line segment.

■

Figure 5-1(a) shows a line segment sequence [(3,20), (5,40), (8,30), (4,10), (6,50)] in time domain, 5-1(b) shows the corresponding data sequence in value-run domain [(3, 0), (5,2), (8,5), (4,9), (6,11)], and 5-1(c) shows the points connected by dotted straight lines.

After the mapping, a line segment becomes a point and the time information of the line segment is discarded. Thus it is a *lossy* mapping. However, it is a very beneficial loss because now the data sequence in the value-run domain is invariant to linear or non-liner warping of the original time domain.

It is also interesting to observe that all the straight lines connecting the points have and angle of 45° or 135° to the value-run axis.

Theorem 1: A straight line connecting two consecutive points in the value-run domain has an angle of 45° or 135° to the value run axis.

Proof: Denote the two points by (v[i],R[1,i]) and (v[i+1],R[1,i+1]). The slope of the line is:

$$Slope = \frac{v[i+1]-v[i]}{R[1,i+1]-R[1,i]} = \frac{v[i+1]-v[i]}{|v[i+1]-v[i]|} = \pm 1 \qquad (5.3)$$

■

In figure 5-1(b) and 5-1(c) the solid square points (A, C, D, and E) correspond to local maximum (peak) and minimum (valley) line segment in figure 5-1(a), and the empty circle point (B) corresponds to the non-extreme line segment, the line segment B in figure 5-1(a).

We call a local maximum or minimum point /line segment as an extreme point/line segment. And we call the other points/segments as non-extreme points/line segments.

It can be easily derived from Theorem 1 that a non-extreme point always resides on the straight line connecting the extreme points. Thus the pitch level variation of the non-extreme points does not affect the structure of the extreme points. The variation just induces the non-extreme point to slide along the straight line.

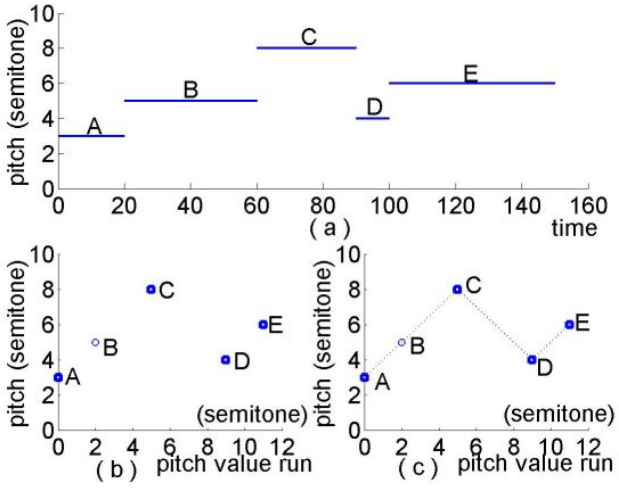

Figure 5-1: (a) A line segment sequence; (b) The corresponding points in value run domain; (c) The points connected by dotted straight lines

The melody is now represented by a sequence of data points in value-run domain. This novel value-run representation plays a key role in our retrieval by humming method, and is a major contribution of this thesis.

This representation has the following merits: (1) the data sequence only relies on the pitch changes in the melody while ignoring the how fast the pitch changes take place; (2) variation of the non-extreme point does not affect the shape structure of the data sequence (a non-extreme point would only shift along the straight line connecting the two extreme points); (3) this representation can support subsequence search, since the relative value difference and value run difference of the points in the data sequence does not depend on the starting line segment or point.

We use the extreme points in the data sequence as the *skeleton* of a melody. This skeleton is in general very robust against the variations in humming, either the tempo variation or the pitch level variation. Our melody matching principle is that any two similar melodies must have a similar skeleton as well. So the melody matching strategy is to find/locate the similar melody skeletons first and then do a detailed similarity measure between the two melodies.

The melody skeleton matching is presented in section 4, and detailed melody similarity measure is presented in section 5.

5.2. Melody Skeleton Matching

The melody skeleton matching serves two roles: (1) locates only the likely candidates who have a skeleton similar to that of the query melody; (2) provides a proper alignment between the query data sequence and the candidate data subsequence. The first function is to filter out the wrong candidates using a less computation. The second function is to help conduct a detailed similarity measure match in the next step.

5.2.1. A Subsequence Alignment Problem

The melody skeleton matching is basically a mapping or subsequence alignment problem, for which we utilize the following principle: a peak point is always mapped to a peak point, and a valley point is always mapped to a valley point. To tolerate pitch level inaccuracy in the hummed query, which may causes wrong or absent extreme points in the melody skeleton, some extreme points can be mapped to none of the extreme points in the other data sequence. Figure 5-2 (a) and (b) illustrate two sequences. A good mapping between the two sequences is [(A1,B1), (A2,B2), (A3,B5), (A4,B6)]. In this mapping B3 and B4 are not mapped to any points, to accommodate the errors.

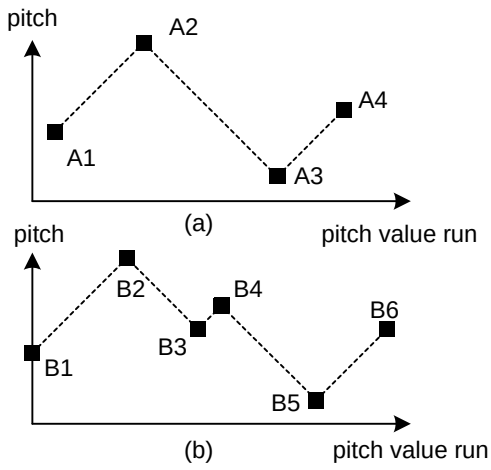

Figure 5-2: An alignment of 2 sequences: 2 points in (b) are not mapped/aligned with any points in (a)

The most probable case of errors in matching extreme points is shown in figure 5-3. In both (a) and (b), the point E1 and E2 should be skipped in the matching. These two points are either incorrectly introduced or wrongly omitted by a user in a query. Usually the two points E1 and E2 have a small pitch difference, since only small pitch level perturbance is likely to be introduced or omitted by users.

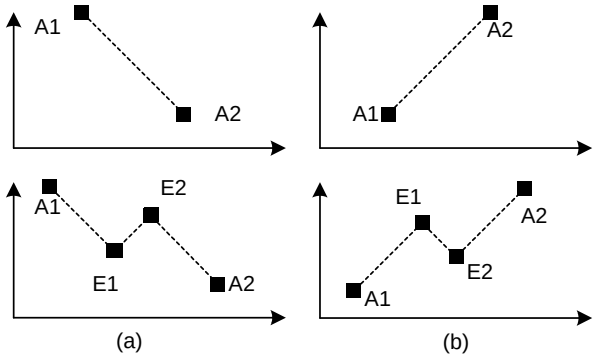

Figure 5-3: The most possible case of errors of extreme points E1 and E2. (a) The case of pitch level going down; (b) The case of pitch level going up.

Some less likely cases of errors of extreme points are illustrated in figure 5-4. In these cases (a)(b)(c)(d), four points E1, E2, E3, and E4 are skipped from mapping. The cause of the errors is the same as the previous cases.

Note that all the extreme points of errors should be presented in pairs, such as (E1, E2) and (E3, E4).

Other cases of errors of the extreme points can be considered when necessary, but from our experiments such cases are very rare.

So our mapping procedure take the following into account: (1) it is a subsequence search problem; (2) it should be pitch value shifting invariant; (3) it should tolerate errors of wrong or absent points, as illustrated in the previous examples.

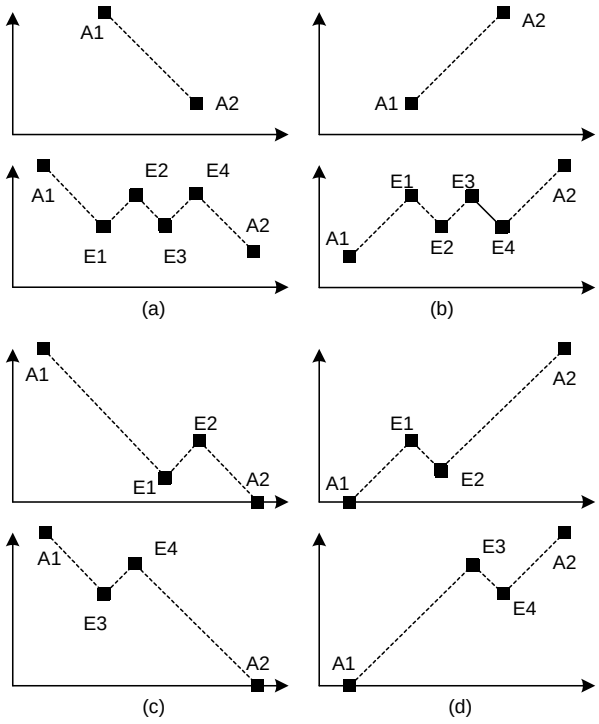

Figure 5-4: Another 4 cases of errors in extreme points

5.2.2. A Novel Dynamic Programming Technique for Subsequence Alignment

We present a novel dynamic programming technique to solve the melody skeleton subsequence mapping or alignment problem.

	t1	t2	t3	t4	t5	t6	t7	t8
q1	$D_{1,1}$		$D_{1,3}$		$D_{1,5}$		$D_{1,7}$	
q2		$D_{2,2}$		$D_{2,4}$		$D_{2,6}$		$D_{2,8}$
q3	$D_{3,1}$		$D_{3,3}$		$D_{3,5}$		$D_{3,7}$	
q4		$D_{4,2}$		$D_{4,4}$		$D_{4,6}$		$D_{4,8}$
q5	$D_{5,1}$		$D_{5,3}$		$D_{5,5}$		$D_{5,7}$	
q6		$D_{6,2}$		$D_{6,4}$		$D_{6,6}$		$D_{6,8}$

Figure 5-5: The table for computing the distance between two sequences q[i] and t[i]

A query data sequence is denoted as *q[i],* where $1 \le i \le m$, *i* is the index of the sequence, *m* is the number of points in the sequence. The pitch value and value run of *q[i]* are denoted as *qv[i]* and *qr[i]*.

A target data sequence is denoted as *t[i]*, where $1 \le i \le n$, *i* is the index of the sequence, *n* is the number of points in the sequence. The pitch value and value run of *t[i]* are denoted as *tv[i]* and *tr[i]*.

For the simplicity of presentation, we assume $n > m$, and *q[1]* and *t[1]* are both peak points or both valley points.

A table for calculating the distance between two sequences starting from *q[1]* and *t[1]* is illustrated in figure 5-5.

A value for a cell in the table $D_{i,j}$ stands for the minimum accumulated distance from cell *(q[1], t[1])* to cell *(t[j], q[i])*. It should be noted that the distance values of the shaded cells in the table are never computed, since such cells correspond to matching between a peak point in a sequence to a valley point in the sequence.

In this dynamic programming formulation, there are two issues of concern: (1) computing the distance value in a cell; (2) tracing the path of an alignment that has the minimum distance.

In our method, we use accumulated distance for each cell *(i,j),* which means $D_{i,j}$ equals a local distance added by the distance value $D_{x,y}$ of a "previous" cell *(x,y)*. Depending on the possible cases of point skipping discussed in section 4.1, the possible "previous" cells of *(i,j)* are illustrated in figure 5-6.

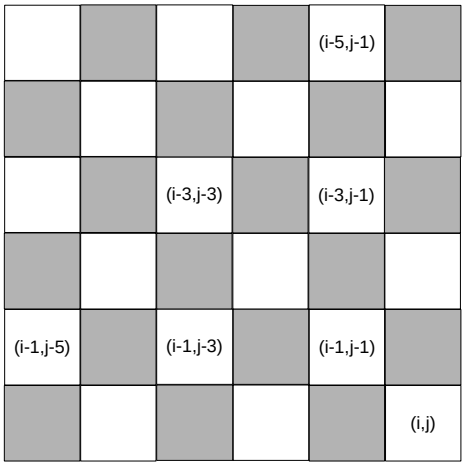

Figure 5-6: The possible previous cells for (i,j)

If the cell *(i-1,j-1)* is the previous cell, then it means there is no point skipping for $D_{i,j}$. If *(i-1,j-3)* or *(i-3,j-1)* is the previous cell, then there is a 2-point-skipping as in the case shown in figure 5-3. If *(i-1,j-5)* or *(i-5,j-1)* is the previous cell, then there is a 4-point-skipping as in the case shown in figure 5-4 (a) and 5-4 (b). If *(i-3,j-3)* is the previous cell, then there is a 4-point-skipping as in the case shown in figure 5-4 (c) and (d).

Other possibilities of previous point for *(i,j)* are not considered in our algorithm, since they are very unlikely to be present.

With the possible previous cells for *(i,j)* given, the distance value for $D_{i,j}$ can then be computed as follows:

$$D_{i,j} = d_{base}(i,j) + \min\begin{cases} D_{i-1,j-1} \\ D_{i-1,j-3} + P(i,-1,j,-3) \\ D_{i-3,j-1} + P(i,-3,j,-1) \\ D_{i-1,j-5} + P(i,-1,j,-5) \\ D_{i-5,j-1} + P(i,-5,j,-1) \\ D_{i-3,j-3} + P(i,-3,j,-3) \end{cases} \qquad (5.4)$$

where i>3 or i>5 or j>3 or j>5 are required for the respective case to be considered.

$$d_{base}(i,j) = |qv(i) - tv(j) - \lambda| \quad (5.5)$$

$$\lambda = qv(1) - tv(1) \quad (5.6)$$

$$P(i,-k,j,-l) = P_Q(i,k) + P_T(j,l) \quad (5.7)$$

$$P_Q(i,k) = 0 \text{ , if k = 1} \quad (5.8)$$

$$P_Q(i,k) = \eta \sum_{x=1}^{(k-1)/2} |qv(i-2x+1) - qv(i-2x)| \quad (5.9)$$

$$P_T(j,l) = 0 \text{ , if l = 1} \quad (5.10)$$

$$P_T(j,l) = \eta \sum_{x=1}^{(l-1)/2} |tv(j-2x+1) - tv(j-2x)| \quad (5.11)$$

$d_{base}(i,j)$ is the local distance between *q[i]* and *t[j]*, and λ is the shifting between *q[1]* and *t[1]*. $P(i,-k,j,-l)$ is the penalty imposed for point skipping, in which $P_Q(i,k)$ is the penalty for skipping points in query, and P_T is the penalty for skipping points in target. The penalty is based on the sum of the value differences of the pairs of points that are skipped. η is a weight for the penalties, which takes a value of 1 in our algorithm.

The previous cell, which gives *(i,j)* the minimum distance value, is chosen and recorded. Another table, which looks like the table shown in figure 5-5, is used for this. The cells of the table store the pointers to (or the index of) the respective chosen previous cells.

The border cells are initialized as:

$D_{1,1} = 0$;

$D_{1,j} = \infty$; *for* $j > 1$

$D_{i,1} = \infty$; *for* $i > 1$

since the alignment starts with *q[1]* and *t[1]*.

The order of computation of distance values for other cells is from top to bottom and from left to right. Since the possible previous cells and the border initialization are known, not all the cells in the table need to be computed. This is because distance values of some cells are determined to be ∞. Furthermore, the value-run can also be used to constrain the number of cells to be computed. Because for an alignment, the mapped points from query sequence and target sequence should not have large difference in their value run after shifting the run difference between *q[1]* and *t[1]*.

After the computation of distance value of the cells, the best alignment is obtained by locating the $D_{m,x} = \min_j D_{m,j}$, which means *(q[1],...,q[m])* has the minimum accumulated distance with *(t[1],...,t[x]),* and $D_{m,x}$ is the distance value.

The mapped path is obtained by tracing back from the cell *(m,x)* in the path table. The tracing is stopped when the pointer points to cell *(1,1)*.

The above mentioned dynamic programming technique will find the best subsequence of target sequence starting from *t[1],* which can be aligned with the query sequence *(q[1],...,q[m])*. For the other subsequence in the targeting sequence starting from *t[1+2x] (x>0),* the dynamic programming computation can be done in a same way by replacing *t[1]* by *t[1+2x]*.

Thus finally, for each starting position *(2x-1) (0 <x < n/2+1)* in the target sequence, the best alignment with the query sequence is found and the corresponding accumulated distance $D_m(x)$ is obtained. In these *n/2* alignments, the alignments at the following position are selected as matches with the query sequence based on $D_m(x)$*:*

$D_m(x)$ is a local minimum;

$D_m(x) < D_{thres}$.

The local minimum of $D_m(x)$ is selected, because the best alignment should always have a smaller distance than the alignment at adjacent positions. D_{thres} is a threshold, which is to ensure that the aligned target subsequence is close enough to the query sequence. In our algorithm, we use $D_{thres} = m-1$, which means one semitone error tolerance is given to every (excepting the first) extreme point in the query. The selected target subsequences are likely candidates, on which an accurate final melody similarity will be computed.

This subsequence alignment method requires the first extreme point in the query data to be reliable. The experiments show that extreme points are robust against variations in the humming, and are reliable for the alignment. To make it even more reliable, an extreme point that has the largest value differences with its predecessor and successor extreme points can be used.

5.3. Final Alignment of Data Points

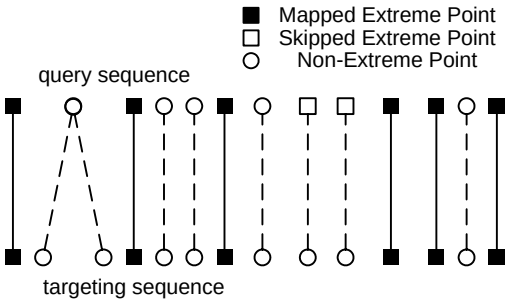

Figure 5-7: Mapping of data points

The alignment of all the data points in two data sequences is based on the alignment of the extreme points of the two sequences, which has been presented in the previous section. We only align the non-extreme points and skipped extreme points between two not skipped

extreme points in a sequence with the non-extreme points or skipped extreme points between corresponding mapped extreme points in the other sequence. This is illustrated in figure 5-7. The empty round points represent non-extreme points, empty square points denote extreme points that are skipped in the extreme point alignment, and the solid square points are the extreme points that are mapped in the extreme point alignment process. The solid line stands for mapping of extreme points (discussed in the previous section), and the dashed line stands for the mapping of non-extreme points or skipped extreme points, which is discussed in this section. We call the mapped extreme points as the *skeleton points* and call the non-extreme points and not mapped extreme points the *non-skeleton points*.

The mapping of non-skeleton points, requires the following steps: (1) shifting of the value of the two sequence based on the aligned skeleton points; (2) mapping of the non-skeleton points.

In the alignment of skeleton points, the value shifting of two sequences is based on the first point of the respective sequence. This shifting value might be too much biased towards the beginning points. So the shifting value is recalculated based on all the skeleton points. Denote the pitch values of the skeleton points in the query sequence and target subsequence by $qv_{sk}(i)$ and $tv_{sk}(i)$, $0 < i <= L$. The new shifting value is given by:

$$\lambda = \left(\sum_{i=1}^{L} qv_{sk}(i) - tv_{sk}(i) \right) / L \qquad (5.12)$$

This new shifting value will be used in the mapping of the non-skeleton points.

Suppose a skeleton point q(a) in the query sequence is mapped with the skeleton point t(b) in the target subsequence. The pair of skeleton points following these two points are q(a+x) and t(b+y) respectively. So the points q(a+1),…,q(a+x-1) are the non-skeleton points in the query sequence, and points t(b+1),…,t(b+y-1) are the non-skeleton points in targeting sequence.

The alignment of non-skeleton points is done based on value distance of the points and a dynamic programming process. The dynamic programming setup is illustrated in figure 5-8.

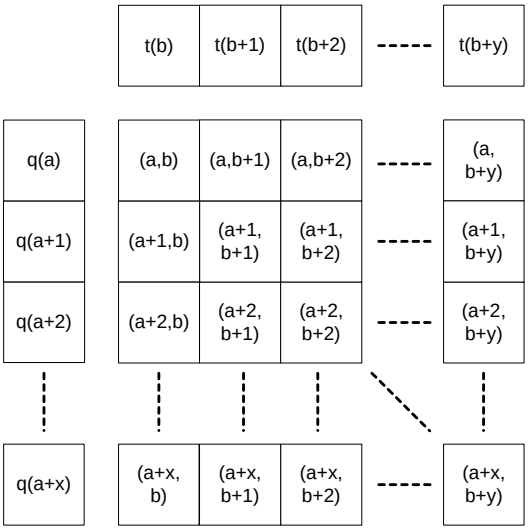

Figure 5-8: Dynamic programming table for aligning non-skeleton points

For each cell *(i,j)* in the table, a local distance value *d(i,j)* is calculated using the following equations:

$$d(i,j) = \left| qv(i) - tv(j) - \lambda \right| \qquad (5.13)$$

where λ is given by equation 12.

The mapping of the non-skeleton points is obtained by tracing a path in the table from *(a,b)* to *(a+x,b+y),* which has the minimum accumulated distance.

In this way, any non-skeleton point can be aligned by using its leading skeleton point and its following skeleton point. Finally, all points in the query sequence are mapped to the points in the target sequence. And the similarity measure between the two sequences can now be computed as follows:

For each point *q(i)* in the query sequence, there is a distance value $d_{qt}(i)$, which is the distance between *q(i)* and its mapped point in the target sequence. If *q(i)* has more than one mapped

point, then the maximum distance is taken by $d_{qt}(i)$. But if $q(i)$ is a skeleton point, then $d_{qt}(i)$ always takes the distance between it and the mapped skeleton point in the target sequence.

Then for each point $t(j)$ in the target subsequence, a distance value $d_{tq}(j)$ can be computed in a manner similar to that of $d_{qt}(i)$.

All the distance values of $d_{qt}(i)$ and $d_{tq}(j)$ can be obtained in the tables of non-skeleton points mapping computation.

5.4. Summary

This chapter has presented a novel melody matching method using point sequence representation, which is derived from the pitch line. The difference between point sequence and pitch line is basically the disregarding of time information, which was intended for coping with nonlinear variations of tempo. Melody skeleton consisting of the local extreme points captures the key global feature of a melody.

A melody skeleton matching technique that can allow skipping of points has been proposed for robust melody key transposition and alignment. The technique is based on a dynamic programming (DP) paradigm, where points in sequences are matched in pairs. The point skipping is implemented by defining the previous cells in the DP table, which corresponds to the various cases of missing or additional point pairs in the melody representation.

Melody skeleton matching achieves melody transposition and alignment, following which the final melody similarity is computed using all the points in the sequences.

The major advantage of the proposed melody matching method is the robustness of melody matching considering tempo variation and large pitch errors (wrong local maximum pitches). The melody skeleton matching technique is also the first approach for matching of sequences of paired points.

Chapter 6. MUSIC SCALE ANALYSIS

6.1. Introduction

All the previous methods represent melody using a sequence of local melody features, such as the relative pitch difference or the absolute pitch value. The global information of the melody has not been considered for melody matching. However, from the music theory point of view, the pitches/notes in a melody do have a global structure, which is the *scale*. A music scale is a particular set of notes that are defined by musicians as being appropriate for a song [Wiki_Scale]. Most music pieces are created (composed) using particular music scales, and the notes only or mainly from the scale are used in the composition. The Major scale and Minor scale are the widely used scales in popular music.

We have found that most music pieces are consistent in the scale and usually only a small portion of a melody can reveal its music scale and the root note. This has motivated us to use music scale information to help solve the key transposition problem in melody matching.

In this thesis, we propose a scale modelling technique to estimate the scale root (the pitch of "do" in diatonic scale) of a melody. The scale root of the melody is then used for proper key transposition in melody matching using the time series method. The computation for key transposition in melody matching is drastically reduced (compared to the previous approaches), which makes the time series matching approach more practical for a music retrieval system. This chapter is organized as follows: section 2 presents background knowledge on music scale and our proposed scale model for Major scale and Minor scale; section 3 presents how to estimate the scale root of a melody in either symbolic form or

acoustic form; section 4 presents a melody matching method by using scale root; section 5 presents the summary.

6.2. Music Scale Modelling

6.2.1. Music Scales

Equal-tempered tuning is the de facto standard for music note tuning in modern and western music [Wiki_Pitch]. In equal-tempered tuning system, each octave has 12 notes, and the pitch difference between contiguous notes is uniformly one semitone. The 12 notes are conventionally named as: A, A#, B, C, C#, D, D#, E, F, F#, G, G#. The pitch difference between any 2 notes, measured in semitones, is called the pitch interval.

A music scale is a series of notes defined by musicians as appropriate for a music piece. A particular scale is specified by the pitch intervals between the series of notes in one octave (and repeated in all octaves). The Major scale and the Minor scale are the most widely used music scales. Both of these 2 scales have 7 notes in one octave. The pitch intervals for a Major scale are: 2 2 1 2 2 2 1. And the pitch intervals for a Minor scale are: 2 1 2 2 1 2 2. A scale is usually referenced to a root note (e.g. C). In equal-tempered tuning any note can be the root note of a scale, thus key transposition is possible.

It should be noted that the minor scale mentioned before is called Natural Minor scale, and it has two variations that are named as Harmonic Minor scale and Melodic Minor scale. The scope of this thesis is restricted to only the Natural Minor Scale.

For example, Major C scale (Major scale using C as the root note) consists note C D E F G A B, Major D scale consists note D E F# G A B C#, and Minor A scale consists note A B C D E F G.

Figure 6-1 illustrates the Major scale and the Minor scale. It can be seen that a Major scale coincides with a Minor scale with its 6th note as the root note. And a Minor scale coincides with a Major scale with its 3rd note as the root note.

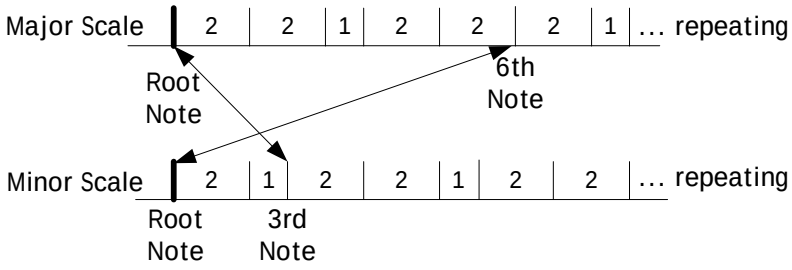

Figure 6-1: Pitch intervals in Major scale and Minor scale

Typically a musician composes a music piece with the scale determined *a priori*. The notes of the music are mostly from the notes in the scale, although notes outside of the scale may occasionally be used. A performer will also need to have the scale to perform the music. This is even true for a hummer, who is not a musician or a trained singer. Although the hummer may not know whether it's a Major C or a Minor G, the scale (the root and the series of notes can be used) is decided (maybe subconsciously) when the tune is being hummed.

6.2.2. Music Scale Model

The music scale information is usually not encoded in the music data file, either in MIDI or in the acoustic waveform formats. But with the knowledge of the scale type and the assumption that notes of a melody are mostly from the scale, it is possible to determine the scale and its root note from the melody.

We propose a music scale model for major and minor scales for music scale estimation. As illustrated in figure 6-2, the music scale model is a circle of 12 equally spaced notes. The highlighted notes labelled by (0, 2, 4, 5, 7, 9, 11) correspond to the note in the music scale (Major or Minor), and we call them a *scale note*. The notes labelled by (1, 3, 6, 8, 10)

correspond to the notes outside of the scale, and we call each of them a *non-scale note*. A major scale is represented by the clockwise enumeration of the scale notes starting with note 0. And a minor scale is the list starting with note 9. In other words, this model represents both major scale and minor scale.

If a melody can fit into the model (all notes can match with the scale notes in the model), the note in the melody corresponding to note 0 of the model is then the root note of a major scale. Moreover, the note in the melody corresponding to note 9 of the model is also the root note of a minor scale.

Fortunately the ambiguity between major and minor scales for this model will not affect the melody matching task. We would only need to estimate the root note of one scale (say the major scale), and use this root note to do key transposition in melody matching.

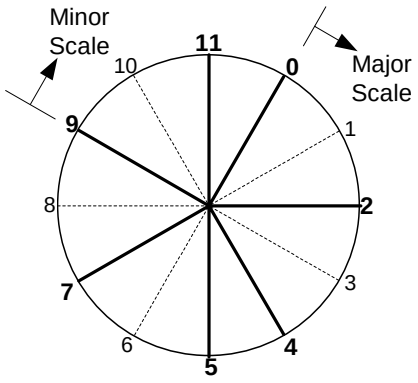

Figure 6-2: A music scale model for Major and Minor scale

Estimating the root note of Major scale for a melody implies finding which note in the melody corresponds to note 0 in the scale model. This can be done by fitting the notes of the melody into the model and computing a fitting error. A small fitting error will indicate that the quality of the scale estimate is good. The details of the major scale estimation procedure are presented in the next section.

6.3. Music Scale Estimation

This section presents how to estimate the major scale root of a melody using the scale model. There are 2 melody formats in music retrieval: symbolic and acoustic. Scale estimation for these 2 formats is presented in section 6.3.1 and 6.3.2 respectively.

6.3.1. Symbolic Melody

MIDI is a popular music encoding format, in which the music notes are represented by numbers (from 0 to 127). A melody is just a sequence of such numbers. The pitch range of a melody is usually within 2 octaves (24 semitones).

Estimating the scale of a symbolic melody consists of 2 steps: (a) construction of a note histogram for the melody; (b) comparing of the note histogram with the scale model and locating the root note.

(a) Since music scale is cyclic in the octaves, the notes of a melody are mapped into a single octave by constructing a note histogram with 12 bins numbered from 0 to 11. The notes in the melody are assigned to the bins by using the following equation:

$$B = MOD(P,12), \qquad (6.1)$$

where P is the note number (from 0 to 127), B is the bin number (from 0 to 11), and MOD is the modulus operation.

(b) The note histogram is compared with the scale model by doing a clockwise alignment of the histogram bins with the notes of the scale model. There are 12 possible alignments when the note 0 of the model is matched to each different bin of the histogram. For each alignment, a model fitting error is computed by summing the histogram bins matched with non-scale notes in the model. If the fitting error is under a threshold for an alignment, good scale estimation is claimed.

For example, the melody of “Auld Lang Syne” is composed by the notes: 58, 63, 62, 63, 67, 65, 63, 65, 67, 65, 63, 67, … And the note histogram is [23] [0] [3] [50] [0] [37] [0] [30] [0] [0] [27] [0]. The scale model fitting results is shown in Table 1. It can be seen that the fitting error is 0 when bin 2 is matched to note 0. This means that the note in bin 2 is estimated to be the root note for major scale.

Table 6-1: Scale model fitting for "Auld Lang Syne"

Start Bin Number	0	1	2	3	4	5	6	7	8	9	10	11
Fitting Error	77	93	0	167	3	114	56	50	120	0	137	33

It can be seen that the fitting errors are also small for bin 4 and bin 9. So the notes in bin 4 and 9 can also be claimed to be the root of major scale. Multiple root notes are claimed for this melody because not all 7 Major scale notes are used in the melody. In fact, this melody is mainly composed with 5 scale notes (refer to the note histogram). The issue of ambiguity of root note will be discussed in section 6.4 on melody matching.

6.3.2. Acoustic Melody

Humming is an important way of providing melody input, which can be used by laypersons without any formal music knowledge or skills. The melody from acoustic input can be transcribed to a time series of pitch values by pitch tracking (section 2.3.2), where each pitch value corresponds to a frame (about 50ms) of the acoustic input. The pitch values are continuous and are measured in semitones and cents. One cent is 1/100 semitone, and one octave pitch range has 1200 cents. The lowest pitch considered is 55Hz, which is corresponds

to pitch value 0.00. And the highest pitch considered is 880Hz, which has the pitch value 48.00 (4 octaves higher than 0.00).

Scale root estimation for acoustic melody is conducted at the precision of a cent. The estimation also involves 2 steps: (a) construction of a pitch value histogram for the pitch time series; (b) comparison the pitch histogram with the scale model in order to locate the root pitch.

(a) The pitch histogram has 1200 bins numbered from 0 to 1199. The pitch values are assigned to the histogram using the following equation:

$$B = MOD(P \times 100,1200) \tag{6.2}$$

where P is the pitch value (from 0.00 to 48.00, 4 octaves), B is the bin number (from 0 to 1199), and MOD is the modulus operation. Frames with no pitch are not assigned to the pitch histogram.

(b) The pitch histogram is compared with the scale model by aligning the histogram bins with the notes of the scale models. Only 7 bins can be exactly aligned with a note of the scale model, since there are 1200 bins but only 12 scale model notes. Some bins can be 100 cents away from the scale model note. Starting by the scale model note 0, there are 1200 possible alignments. For each alignment, a model fitting error is computed by summing the distance of each bin to the model by using the following equation:

$$D = \sum_{i=0}^{1199} N_i \times d_i \tag{6.3}$$

where N_i is the number of counts in the ith bin and d_i is the distance of the ith bin to a closest scale model note for the alignment. d_i is measured in cents and takes the value from 0 to 100.

If the model fitting error is under a threshold, a good scale estimate can be claimed.

Figure 6-3 shows a short portion of Auld Lang Syne produced by humming. The scale model fitting error is illustrated in figure 6-4. It can be clearly seen that the local minimums at 77 and 577 with fitting error of 25 correspond to the root of the major scale. The local minimum at 1074 of fitting error of 30 could also be regarded as a root of the major scale. The actual pitch of the root note in the melody can be easily derived from the pitch histogram bin number.

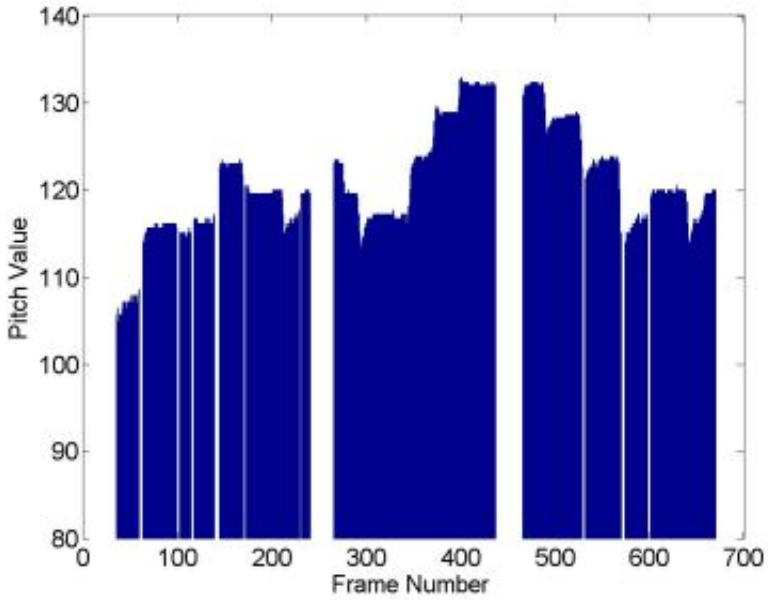

Figure 6-3: A humming input of Auld Lang Syne

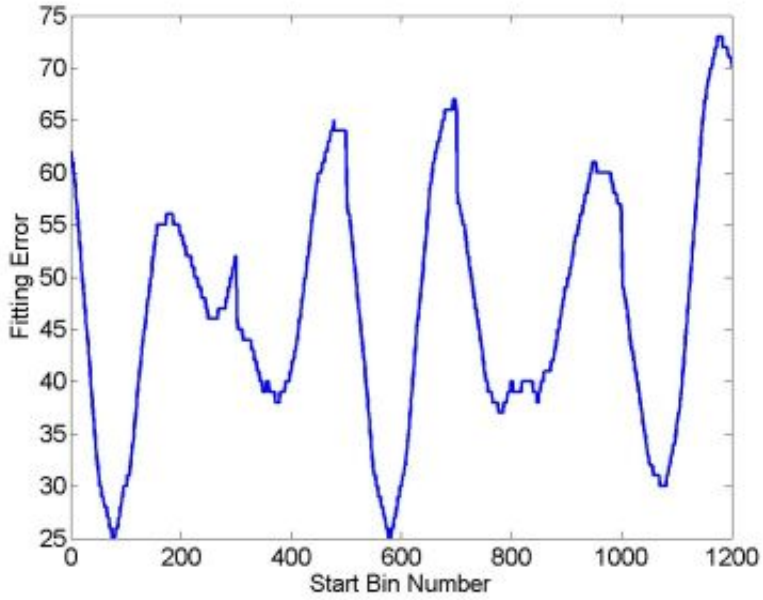

Figure 6-4: Scale model fitting error for the hummed melody

6.4. Melody Matching

Music retrieval can be done by similarity matching of melodies. This section presents a melody matching method by utilizing the estimated scale root. The matching consists of 2 processes: first, transposing the query melody by matching the keys (root); and second, computing the similarity by using dynamic time warping distance.

6.4.1. Key Transposition

Usually a melody query is a short excerpt of the target melody. Since the query contains a subset of the notes of the original melody, the estimated scale root of the query is then a superset of the estimated scale root of the original melody. This is because, the less the number of notes used, the smaller are the constraints in scale model fitting, and there are more possible estimated scale roots. As a result, we only need to use one root per octave for the original melody in the database, and all the roots estimated for the query are used for melody transposition. It is guaranteed that no valid key transposition will be missed.

Key transposition for melody matching is straightforward with the scale roots available, which is demonstrated using the example shown in figure 6-5. The root note for the original melody is R1. R2 is the same root one octave higher than R1. The query melody has 3 roots estimated: Ra, Rb and Rc. Rd is the same root one octave higher than Ra. The key transposition can be done by matching any root in the original melody with any root in the query melody. So there could be multiple valid key transpositions. However, only (R1 to Ra) and (R2 to Rd) transposition are valid in this example. This is because the pitch range of the query has to be within the pitch range of the original melody after transposition. Furthermore, since the 2 valid transpositions are redundant, there is only one key transposition for this melody matching (R1 to Ra).

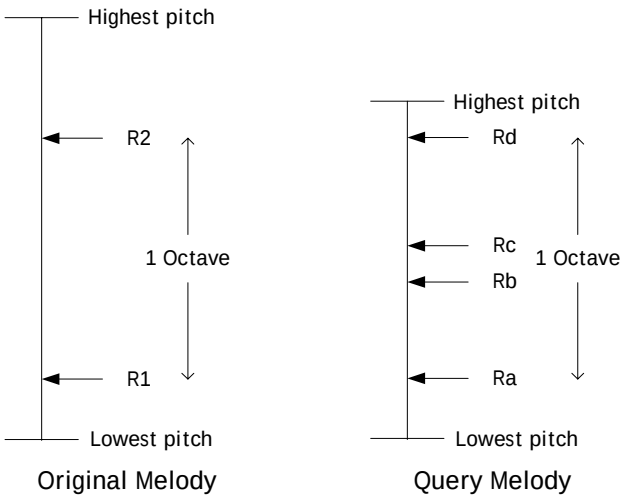

Figure 6-5: Key transposition for melody matching

Without the scale roots, any transposition that allows query pitch range to be within the original melody pitch range can be a potential valid transposition. It can thus be clearly seen that key transposition using scale root has drastically reduced the computation in key transposition for melody matching. For example, if the pitch range margin is 6 semitones, and pitch precision is in 10 cents, then the computation for transposition is reduced to 1/60 by utilizing scale roots.

6.4.2. Melody Similarity Matching

With the melodies properly transposed, the melody similarity matching can be done by followed melody local alignment and similarity measurement. The melody alignment is much easier than the previous proposed approaches, since absolute pitch values are used. The melody skeleton technique is used for robust melody alignment. Since melody alignment is done only in the time domain, a local error bound, 1 semitone, is imposed to efficiently skip the wrong candidates. Finally, melody similarity is computed using the proposed similarity measures, after the melody alignment is achieved.

6.5. Summary

This chapter has presented a novel technique to estimate the scale root of a melody. The root can be used for key transposition in melody matching. The scale (Major or Minor) of a melody is estimated by fitting the notes/pitches into a scale model, which is based on the pitch intervals of scale notes in an octave. A good model fitting stands for detection of a scale root.

In melody matching, multiple estimated scale roots of the hummed melody are aligned with (transposed to) the scale root of a MIDI melody. The pitch range of the 2 melodies can be utilized to reduce the number of key alignment.

The proposed method is the first approach for scale root estimation of a melody. The method totally separates the key transposition from melody alignment and similarity measure, and has greatly reduced the computation used for key transposition search in melody matching.

Chapter 7. EXPERIMENTATION

This chapter presents the experimentation on the proposed music retrieval techniques. The experimentation includes: evaluation of the pitch extraction method (chapter 3); evaluation of the slope-based melody alignment method and retrieval performance (chapter 4); evaluation of the melody skeleton method and retrieval performance (chapter 5); evaluation of the music scale estimation and retrieval performance (chapter 6); evaluation of the melody similarity metrics; performance comparison with existing techniques; and finally the implementation of a query-by-humming music retrieval system.

7.1. Experimental Setting

The experiments have been conducted on a PC system with P4 (1.8GHz), 512MB RAM, Window 2000. Intel performance library for signal processing is used for Discrete Fourier Transform.

7.1.1. Music Data

The music data used in the experimentation are MIDI files downloaded from the Internet [Kar_Download]. Most of the MIDI files are in Karaoke format, which is standard MIDI format with an explicit melody track and a lyrics track. The Karaoke file players [Vanbasco] can playback the music while prompting the lyrics, so a user can sing along to enjoy Karaoke. The MIDI files consist of English, Chinese and Japanese pop songs, as listed in Table 7-1.

Table 7-1: Song types of the music data

Song Types	English	Chinese	Japanese
Number of Songs	1850	550	200
Total Duration	138 hours	40 hours	9 hours

For the MIDI files to be searched by melody, the melody tracks are firstly extracted from the MIDI files. This is done by manually identifying the melody track and saving it as a separate MIDI file. A MIDI file manipulation tool has been developed for this purpose. With this tool, each individual track in a MIDI file can be played back separately and the track name encoded in the MIDI file is also displayed. The melody tracks usually have a track name of "melody" or "vocal". To further facilitate the identification of the melody track, the tracks in a MIDI file are ordered by computing a score. The higher the score, the more likely the track is the melody track. The score for each track is computed by summing the product of strength, time and pitch of each note in the track:

$$S = \sum_{n=1}^{N} W_n \times D_n \times P_n , \qquad (7.1)$$

where N is the number notes in the track, W_n is the strength of a note, D_n is the duration of a note, and P_n is the pitch of a note.

This score has been shown quite effective for identifying the melody track. The final decision of the melody track is verified by playing back the particular track.

The melody track identification tool is shown in figure 7-1. The Karaoke file for "American Pie" is loaded into the system. The evaluation result shows that track 0 has highest score, which is in fact the melody track. By clicking on button "Trk Cfm", the selected track is extracted and stored as a separated MIDI file.

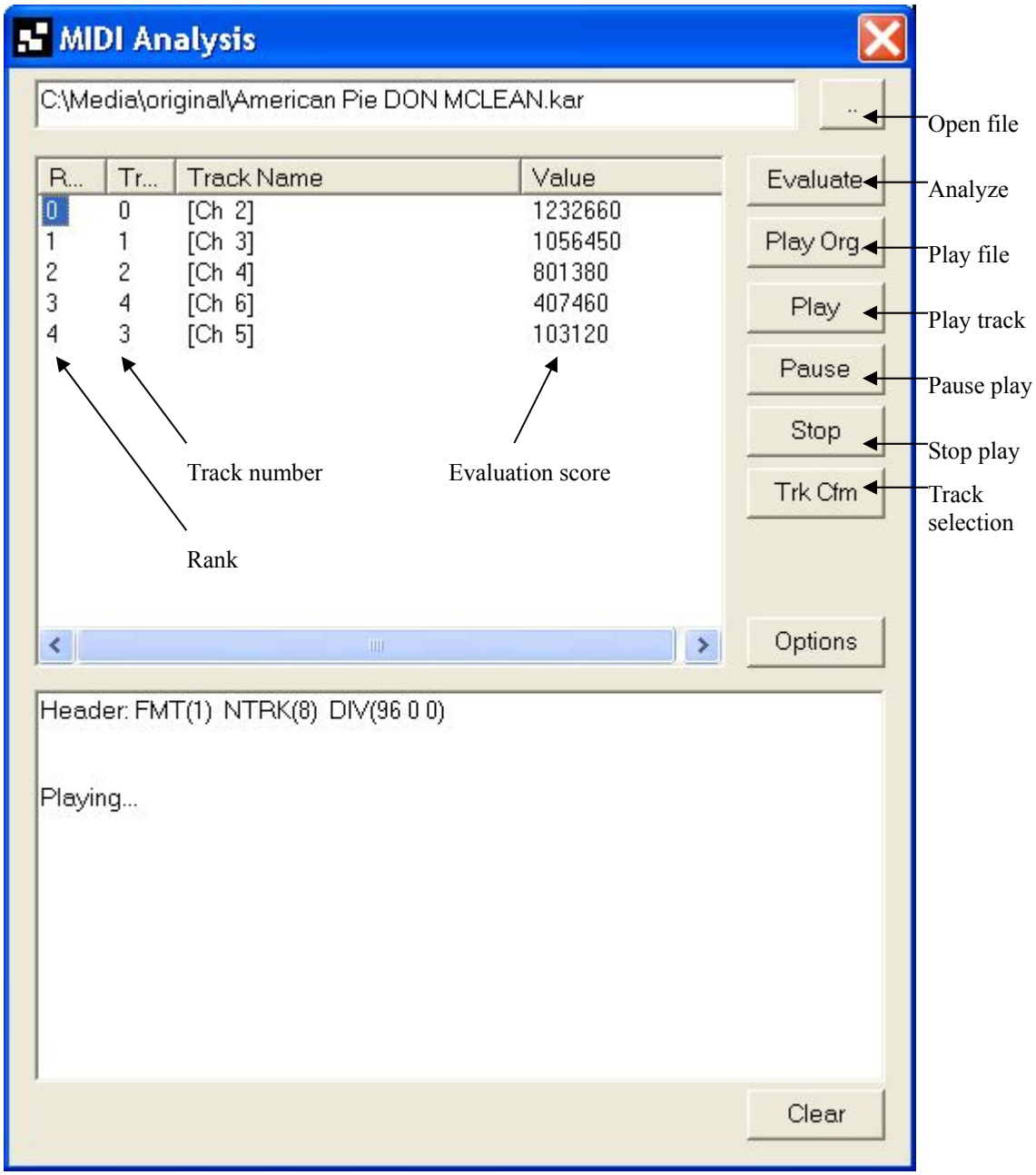

Figure 7-1: A tool for melody track identification from Karaoke MIDI files

The extracted melody track is then used in melody feature extraction and indexing. The melody slope, melody skeleton and music scale of the melody are constructed or estimated, and saved in the ASCII text file format. The correspondence between the melody feature files and the original MIDI file is established by using a same file header name.

In doing music retrieval by melody matching, the feature of a query melody is matched with feature files of each MIDI melody. In the experimentation, the retrieval methods based on melody slope, melody skeleton, and music scale are used separately. When doing querying, a user would specify which method to use for melody matching.

7.1.2. Acoustic Query Input

The experimentation is based on humming query input. The humming voices have been recorded through microphone using 44100 Hz 16 bit waveform (PCM) format. The recordings have been conducted in a normal computer laboratory, and the microphone is less than 10 centimetres away from the singer's mouth.

Five subjects including 3 male and 2 female have been involved in the experimentation. None of them are professional singers. Ten popular songs, which all the subjects are familiar with, are chosen as the target melodies. The 10 songs are listed in table 7-2. Each humming query starts at the beginning of the target melody.

Table 7-2: Target melodies in the experimentation

Number	Song titles	Original language	English meaning
1	Akatombo	Japanese	Red dragonfly
2	Sakura	Japanese	Oriental cherry
3	Spring of the north nation	Japanese	
4	Tian Mi Mi	Chinese	Honeyed sweet
5	Qing Wang	Chinese	Web of love
6	Wen Bie	Chinese	Kiss goodbye

7	Happy birthday to you	English	
8	Love me tender	English	
9	Silent night	English	
10	Yesterday once more	English	

In collecting the humming query data, variations are deliberately introduced. Each subject has hummed 4 times for each target melody: once in moderate tempo, once in fast tempo, once in slow tempo, and once in combination of fast and slow tempo. Thus there are 4 sets of humming queries, as listed in table 7-3. The experiments presented in this chapter have been based on some or all of the query data sets.

Table 7-3: Query data sets

Query set	Q1	Q2	Q3	Q4
Tempo	Moderate	Fast	Slow	Fast and slow
Number of queries	50	50	50	50

7.2. Pitch Extraction Evaluation

Pitch extraction is the first step in doing music retrieval using hummed melody. The proposed pitch extraction method is evaluated by inspecting the pitch errors in the extracted pitch curve.

Queries based on melody 2, 4, 5, 7, 9 (table 7-2) in data set Q1 and Q2 have been used in this experiment. Each humming input contains more than 10 notes of the melody. And only the portion corresponding to the first 10 notes are used in the evaluation.

The pitch extraction result for a melody query is a sequence of pitch values, where each value corresponds to a frame of size 2048 samples (46 millisecond). The evaluation is based on

errors on each note. The pitch extraction for a note is considered wrong, if any one frame for the note has a wrong pitch extracted or more than half of the frames having no pitch extracted.

The error inspection is done by plotting out the energy and pitch for each frame. From the energy, the portion of silence can be easily identified, so frames in that portion is not considered in evaluation for any note. The errors of wrong pitch can usually be easily identified from the pitch plot, since the wrong pitch are mostly the double or triple of the correct pitch. Slightly off key in the humming is not considered wrong pitch.

As for the silence portions in all the 50 humming inputs, the decisions of no pitch is 100% accurate, i.e. no pitch is wrongly claimed for the silence portions. The result of pitch extraction for the notes is shown in Table 7-2.

Table 7-4: Pitch extraction result

	Melody 2	Melody 4	Melody 5	Melody 7	Melody 9	Average
Subject 1	100%	100%	95%	95%	100%	98%
Subject 2	95%	100%	100%	85%	100%	97%
Subject 3	100%	100%	100%	95%	95%	98%
Subject 4	90%	90%	85%	85%	95%	89%
Subject 5	95%	100%	95%	100%	100%	98%
Average	96%	98%	96%	92%	98%	96%

The table shows the correct rate for each subject and each melody, and the average correct rates. Since each humming query has 10 notes and each subject hummed each melody twice (in Q1 and Q2), the correct rate is based on 20 notes. The average pitch extraction is 96%. The results also show that pitch extraction errors is slightly dependant on the subjects. The subject

4 has more errors than the other subjects. The pitch extraction error is quite independent of the query melody.

7.3. Melody Matching by Using Melody Slope

This section presents the results of melody matching method using melody slope.

From the 2600 melodies in the database, totally 240500 melody slopes are extracted. The number of pitch line segments in a melody slope is illustrated in table 7-3. It can be seen that mostly melody slope has less or equal to 3 pitch line segments, and more than 40% melody slopes have one pitch line segments. And no melody slope contains more than 5 pitch line segments.

Table 7-5: The number of pitch line segments in a melody slope

Notes per Slope	1	2	3	4	5
Ratio	42.8%	22.2%	27.0%	4.8%	3.2%

Melody matching using melody slopes has 2 major steps: melody alignment and pitch shifting (key transposition) using melody slopes; and melody similarity measurement.

Thus in the experiment we first evaluated the performance of melody alignment and pitch shifting using melody slopes, and second evaluated the retrieval performance.

7.3.1. Melody Slope Matching Evaluation

The melody alignment and pitch shifting is conducted by doing melody slope sequence matching. There are 2 issues regarding melody slope sequence matching: robustness and efficiency. Robustness of melody slope stands for the correct presence of melody slope in the query melody. If the melody slope is not robust to the variations or errors in the query melody,

the target melody might be missed in the first stage and correct melody retrieval is then impossible. The efficiency stands for how many wrong candidates can be rejected in the melody slope matching stage, so computation would not be wasted on measuring similarities of the query with wrong candidates.

(1) Robustness evaluation

To evaluate the robustness of melody slope, query data set Q1 and Q2 have been used. The length of the hummed melody is in average 4 bars. The time duration is about 16 seconds for moderate tempo and about 10 seconds for fast tempo. For the 10 target melodies, there are totally 82 melody slopes.

We manually verify each identity of melody slope in the query melodies. For the moderate tempo, in average 94% of the melody slope are correctly presented in the humming. For the fast tempo, the rate is 87%. The missed melody slopes are usually of small pitch span values, such as 1.0 semitone.

We then conducted melody slope sequence search using our proposed algorithm (chapter 4 section 4.2). We used 4 melody slopes in the slope sequence matching. In the testing, the threshold for $dist_{slope}$ and $corr_{slope}$ are respectively 2.0 and 0.1. (See chapter 4 equation (4.5) and (4.6)). That means if difference on pitch range of the melody slopes is above 2 semitones, the candidate is skipped.

We used all possible 4 melody slope sequences in the queries to do the searching. We manually examine whether the correct melody slope sequence in the correct target melody is identified as good candidate by the algorithm. The overall correct identification rate is 78%. But since each humming query usually contains 4 or 5 melody slope sequence candidate, 94 of the 100 queries have at least one slope sequence candidate correctly identified. 6 of the 100 queries failed to match the desired slope sequence candidates due to missing of correct melody slopes.

The correct identification rate can be increased by loosening the threshold values for $dist_{slope}$ and $corr_{slope}$. However, this will sacrifice the candidate rejection rate and the retrieval efficiency is degraded.

(2) Efficiency Evaluation

The proposed melody slope sequence matching also serves as a filter for locating likely candidate out of all possible candidates in the database. We did an experiment to find how many unlikely candidates are successfully rejected. The 100 humming queries in Q1 and Q2 are used in this experiment. Searching is conducted for all the candidates in the database.

Figure 7-2 shows the average number of candidates returned from the database versus the number of melody slopes used in a slope sequence matching. It is obvious that the more slopes are used, the less number of candidates are returned. It can be seen that for 3 melody slope sequence, about 1.7% of all possible slope sequences are returned. The number of candidate returned by 4 slope sequence is 0.5% of the population. In this experiment, we use the same threshold as in robustness testing.

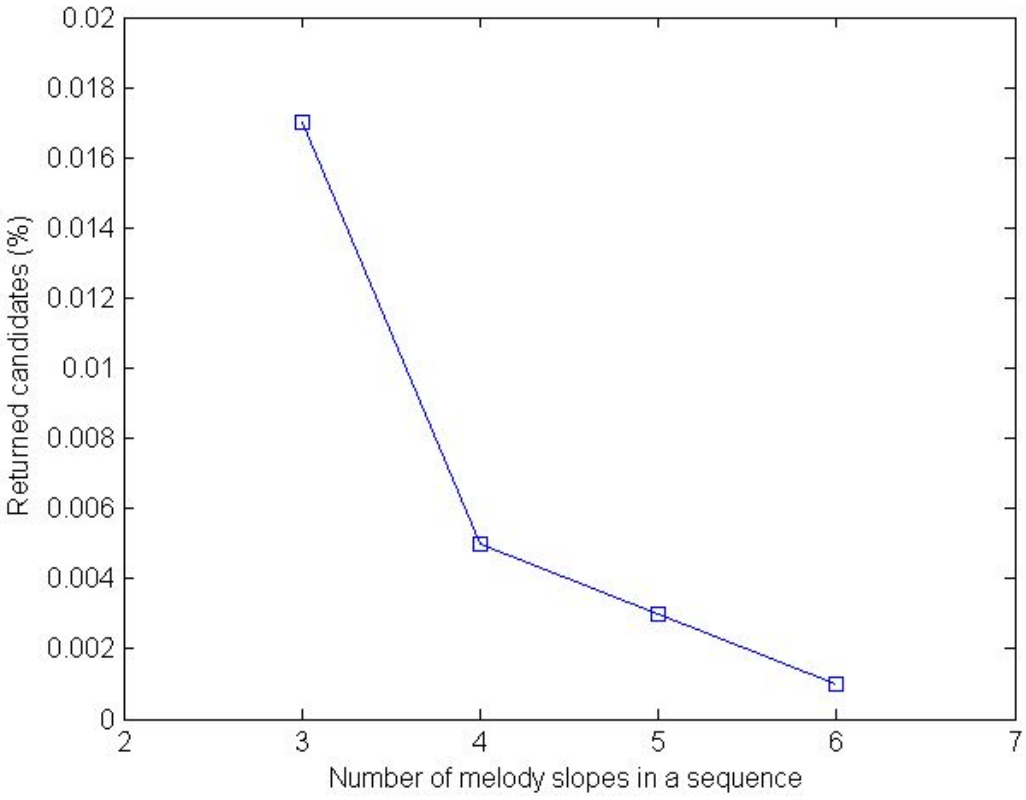

Figure 7-2: Number of returned candidates decreases when number of slope in a sequence increases

The result shows that by using melody slope sequence matching, large portions of the candidate in the database can be rejected, which greatly promote the efficiency in melody similarity matching.

7.3.2. Retrieval Performance Evaluation

To examine the retrieval performance by using melody slope, the 200 hummed queries in Q1, Q2, Q3 and Q4 are used in melody similarity measure. Melody alignment and pitch shifting are based on the first 4 melody slopes. In the retrieval, the thresholds are same as those mentioned before. The melody similarity metric D_1 is used in similarity measure.

Two types of retrieval performance evaluations have been conducted: one for matching only the beginning of a melody, and one for matching anywhere of a melody. For multiple

alignments in a single target melody, the alignment with highest similarity score is used for that melody.

The retrieval result for matching at melody beginning is shown in table 7-6. Each row shows the result for a query data set from Q1 to Q4 and the last row for the overall data set. The retrieval accuracy is in terms of percentage for rank lists from top 1 to top 10. The overall retrieval accuracy is 76% for top 10. However, it can also be seen that query data set Q4 has accuracy of 48%, which is much lower than the other data sets. This is due to the mismatching of slope sequence in the humming query, which contains inconsistent tempo variations. The overall results for data set Q1, Q2 and Q3 is above 85% for top 10.

Table 7-7 shows the retrieval accuracy for matching anywhere in the middle of target melodies. The overall result is 64% for top 10. And for the same reason, retrieval accuracy for Q4 (42%) is much lower than those for Q1 to Q3.

The average time for each query is about 3 seconds for matching anywhere of a melody, and less than one second for matching the beginning.

Table 7-6: Retrieval results for melody slope method: matching at the beginning

%	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top10
Q1	82	84	84	84	84	86	86	86	88	88
Q2	78	78	80	80	84	84	86	86	86	86
Q3	76	76	76	80	80	80	80	82	82	82
Q4	44	44	46	46	46	46	46	48	48	48
Avg	70	70.5	71.5	72.5	73.5	74	74.5	75.5	76	76

Table 7-7: Retrieval results for melody slope method: matching in the middle

%	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top10
Q1	58	62	66	69	70	70	72	72	74	74
Q2	54	58	60	61	62	66	66	68	68	68
Q3	58	60	62	64	64	64	64	68	68	72
Q4	38	38	40	40	42	42	42	42	42	42
Avg	52	54.5	57	58	59.5	60.5	61	62.5	63	64

The results shows that the slope matching method can handle well the consistent variation of tempo, like those in Q2 and Q3. But the performance deteriorates significantly for inconsistent tempo variations.

7.4. Melody Matching by Using Melody Skeleton

Melody matching by using melody skeleton is based on point sequence matching. The advantages of this approach are the invariance to tempo changes and robustness in melody skeleton matching.

In the experiments, the query data sets Q1, Q2, Q3 and Q4 are used. So there are totally 200 hummed queries. The hummed part is at the beginning of the query melodies, however the search is for anywhere in the middle of all the target melodies.

Robustness of the melody skeleton against (linear and non-linear) tempo and pitch variations is the main feature of this method. Figure 7-3 shows 6 hummed queries of the tune “Happy Birthday To You” using different tempos by different subjects. Figure (a) shows the query

hummed in normal speed. Figure (e) shows a faster tempo. Figure (b) and (c) shows slower tempos. Figure (d) and (f) shows inconsistent tempos.

Each figure in figure 7-3 shows the original query pitch curve, pitch lines and the value-run domain data points. The arrows in the figure indicate point errors (missing or additional). It can be seen that the errors increase with the variation, however the melody skeleton structure formed by extreme points is very robust, as it is almost identical for all the 6 queries.

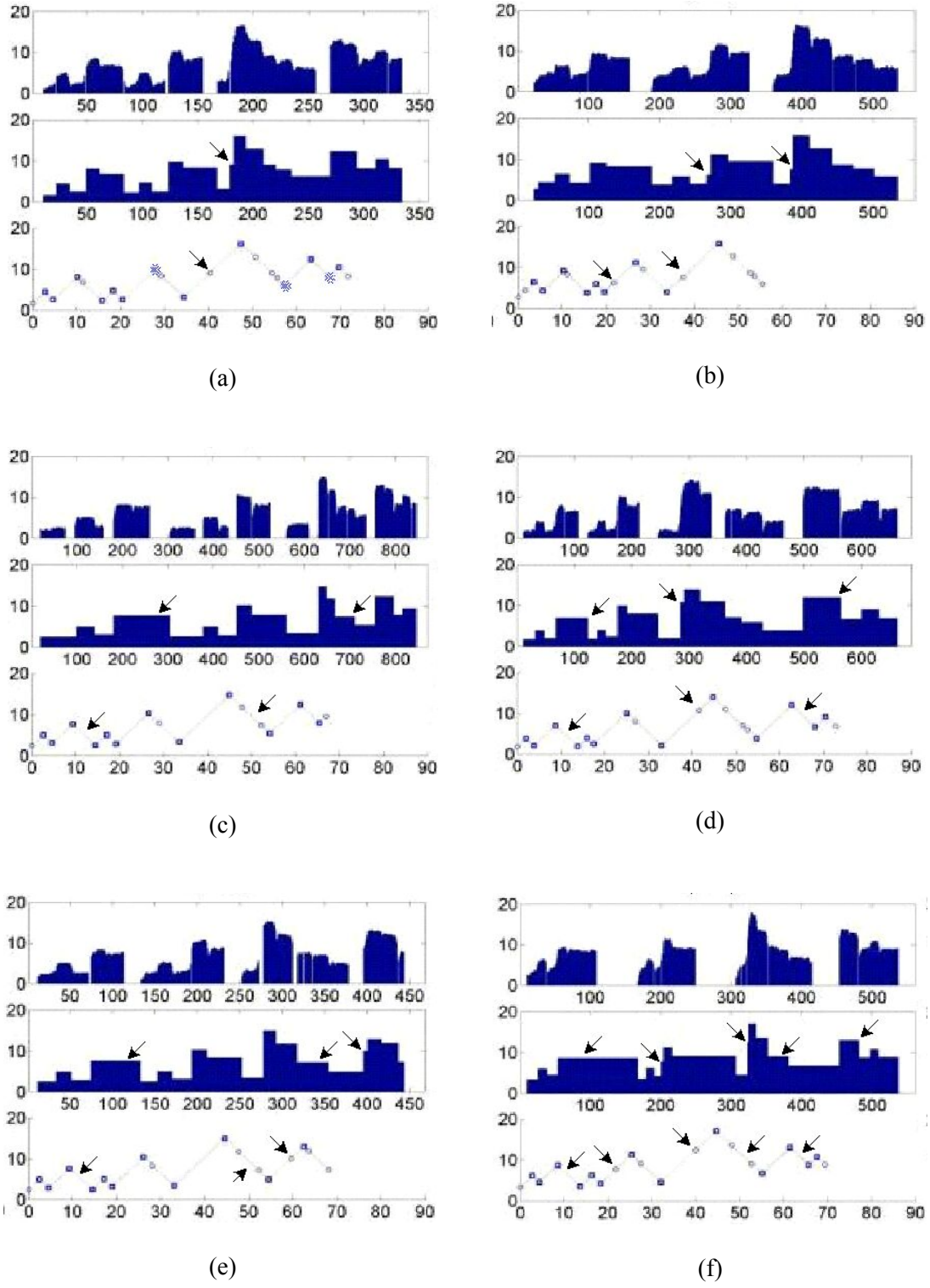

Figure 7-3: 6 hummed queries of a same tune "Happy Birthday To You" using different tempos by different persons (arrows indicates point errors)

7.4.1. Performance of Melody Skeleton Matching

To evaluate the performance of melody skeleton matching, we examine the correct hit rate and candidate elimination rate. The correct hit rate is the number of queries that have matches with the correct target divided by the total number of queries. The candidate elimination rate is average number of eliminated candidates divided by the total number of possible candidates.

The 2600 MIDI files in our database have a total of 240500 extreme points in the melody skeletons. Considering that the matching starts from only a peak point or a valley point, there are totally 240500/2=120125 possible matching candidates.

The average number of extreme points of the 200 queries is 9.5. But the first and last extreme points are not used, since they are more prone to have errors. So for each query, 7.5 extreme points on average are used.

94.5% (correct hit rate) of the queries can match with the correct target melodies. And on an average, 6643 candidates are returned for each query. So the candidate elimination rate is 1-(6643/120125)=94.47%.

7.4.2. Performance of Melody Retrieval

For the performance of melody retrieval, also two types of evaluation have been conducted: one for matching only at the beginning of songs and one for matching anywhere in the middle of songs.

The retrieval rate of the top 10 lists for matching is illustrated in table 7-8 and 7-9. For matching melody beginning (table 7-8), it has achieved 83% for top 1, and 93.5% for top 10. The result for matching anywhere (table 7-9) is 70% for top 1, and 84% for top 10. The retrieval performance is better than the slope method, and significantly for the results of query

data set Q4. This result demonstrates melody skeleton method is robust against non-liner tempo variations in the query.

The average time for each query is about 5.5 seconds for matching anywhere of a melody, and less than 2 second for matching the beginning.

Table 7-8: Retrieval results for melody skeleton method: matching at the beginning

%	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top10
Q1	86	86	88	90	90	92	94	94	94	94
Q2	82	82	86	88	90	92	92	94	94	94
Q3	84	84	86	88	90	90	92	92	92	94
Q4	80	82	82	86	86	86	90	90	92	92
Avg	83	83.5	85.5	88	89	90	92	92.5	93	93.5

Table 7-9: Retrieval results for melody skeleton method: matching in the middle

%	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top10
Q1	72	74	76	78	78	82	84	86	86	86
Q2	72	72	74	76	78	80	80	84	86	86
Q3	70	72	72	76	78	78	82	84	84	84
Q4	66	68	70	70	72	76	76	76	76	80
Avg	70	71.5	73	75	76.5	79	80.5	82.5	83	84

7.5. Melody Matching by Using Music Scale Estimation

Melody matching with music scale root is different from previous methods, where pitch shifting is done before melody alignment. The scale root is estimated by using the proposed scale modelling method.

To investigate the scale modelling approach, the scale root estimation is conducted for the 2600 melodies in the database. The results have shown that 96% of the melodies can fit to our scale model (Major/Minor), when at least one root note can be estimated (using the threshold: 5% of total number of notes of a melody). This result shows that most popular songs are written in Major/Minor scales. About 30% among the melodies fit to the scale model have more than one root estimated. These melodies are mostly short melodies with a small number of scale notes used. A small number of melodies (1%) contain key transpositions, and the root cannot be successfully estimated. For this case, local scale root estimation can be conducted for a window (say contains 10 different notes). And the key transposition can be detected by locating the discontinuity in the scale estimation result for all the windows. Melody portion before and after key transposition can then be indexed separately for matching.

The retrieval performance is evaluated by using query data set Q1, Q2, Q3 and Q4. The result is shown in table 7-10 and table 7-11. Table 7-10 is for matching only beginning of melodies. For top 1, the retrieval accuracy is 82%, and for top 10 the accuracy is 89.5%. Table 7-11 is for matching anywhere in the middle of melodies. For top 1, the retrieval accuracy is 69%, and for top 10 the accuracy is 77%.

The average time for each query is about 2 seconds for matching anywhere of a melody, and less than one second for matching the beginning.

Table 7-10: Retrieval results of music scale method: matching at beginning

%	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top10
Q1	84	84	86	86	88	90	90	92	92	94
Q2	82	82	82	84	84	84	84	88	88	90
Q3	82	82	84	84	86	86	86	88	88	88
Q4	80	80	80	82	82	82	84	84	84	86
Avg	82	82	83	84	85	85.5	86	88	88	89.5

Table 7-11: Retrieval results of music scale method: matching in the middle

%	Top 1	Top 2	Top 3	Top 4	Top 5	Top 6	Top 7	Top 8	Top 9	Top10
Q1	72	72	74	76	76	78	80	82	82	82
Q2	70	70	70	72	72	74	74	76	76	76
Q3	68	70	72	74	74	76	76	78	78	78
Q4	66	66	66	66	68	70	70	70	70	72
Avg	69	69.5	70.5	72	72.5	74.5	75	76.5	76.5	77

The retrieval results breaking down for each query data sets (Q1 to Q4) are presented in figure 7-4 to 7-7 to compare the three proposed matching methods. In the figures, solid lines correspond to matching beginning of a melody and dotted lines correspond to matching anywhere of a melody. Plots with circles are for melody slope method. Plots with up triangles

are for music scale method. Plots with upside-down triangles are for melody skeleton method. Figure 7-8 shows the results for overall query data set. It can be seen that melody skeleton method has highest retrieval accuracy performance. However, it has lowest efficiency performance (5.5 seconds compared with 2 seconds for the other 2 methods). This is reasonable since melody skeleton method is designed to tolerate large variations and errors with the price of higher computation in dynamic programming matching.

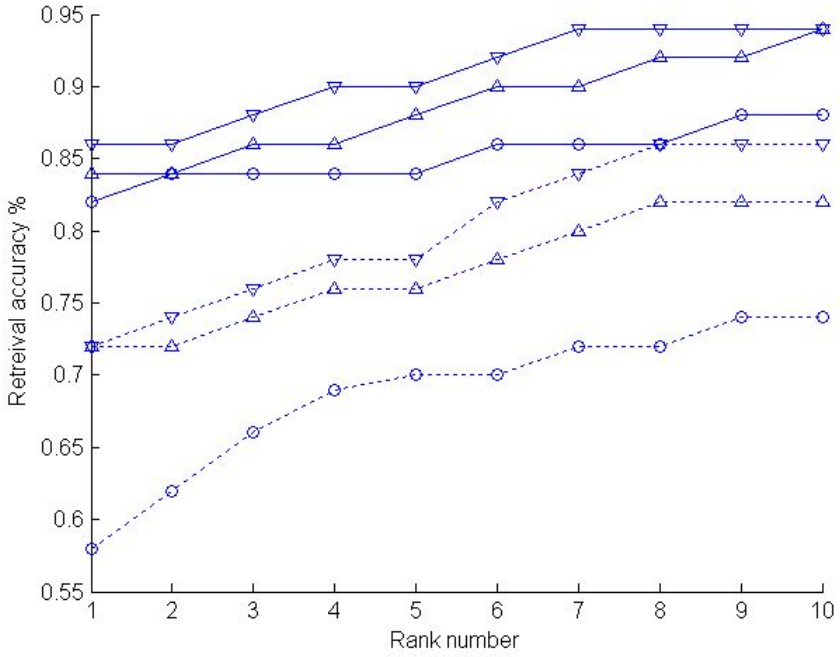

Figure 7-4: Retrieval results for query data set Q1

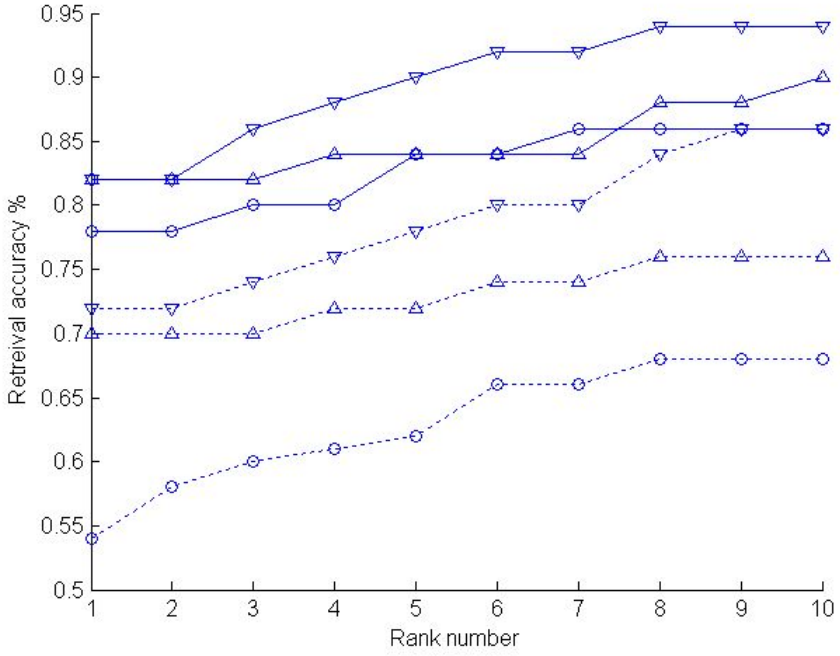

Figure 7-5: Retrieval results for query data set Q2

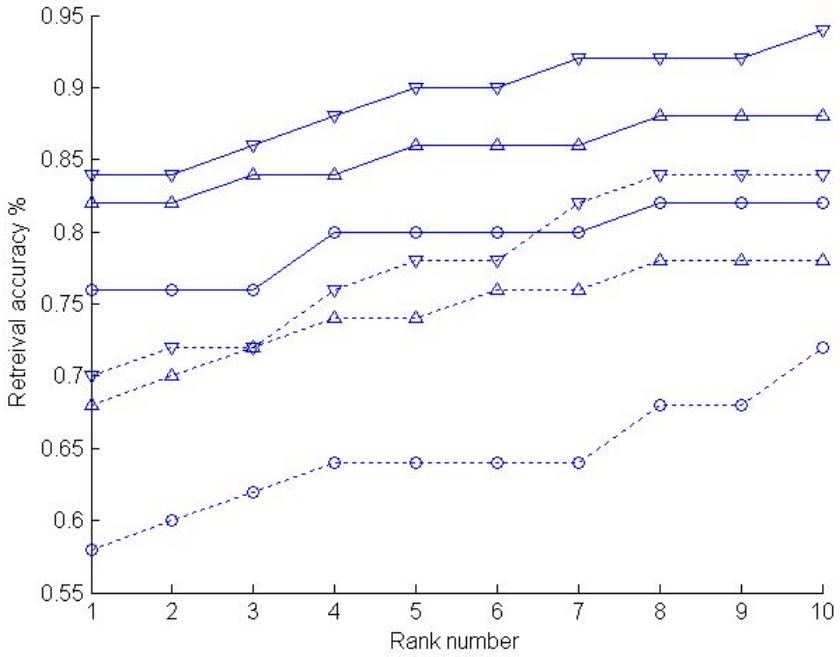

Figure 7-6: Retrieval results for query data set Q3

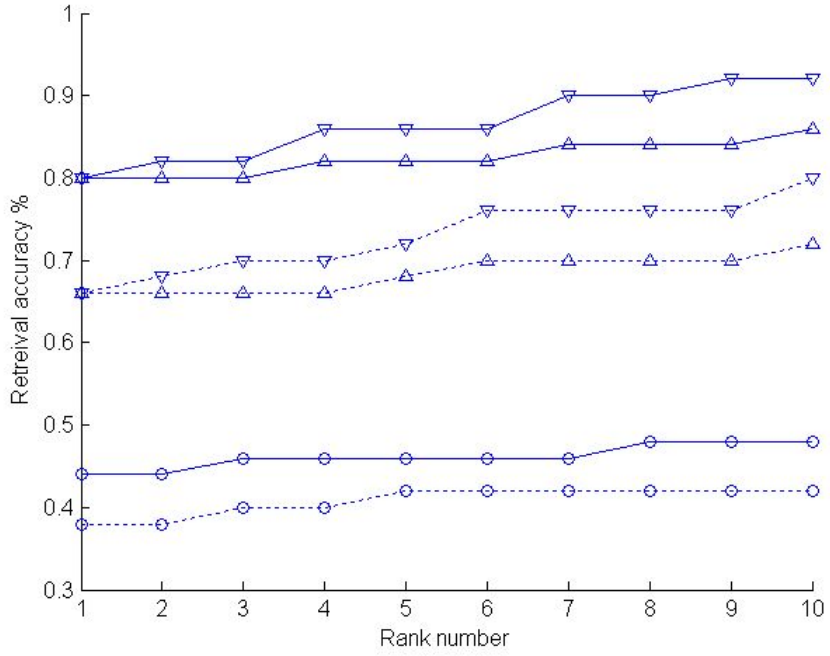

Figure 7-7: Retrieval results for query data set Q4

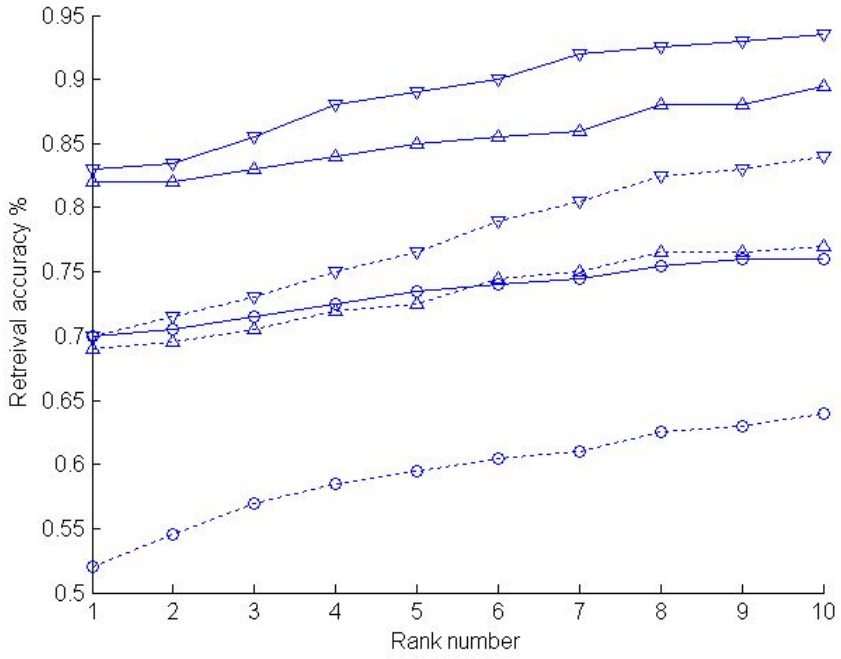

Figure 7-8: Retrieval results for overall query data

7.6. Similarity Metric Performance Comparison

We proposed 4 melody similarity metrics in chapter 4. To evaluation the performance of the metrics in music retrieval, an experiment is conducted to compare the retrieval results by using the 4 metrics. In this experiment, the melody alignment and pitch shifting is based on melody skeleton method. Q1 and Q2 are used as the query data set. For metric D_3 and D_4, the threshold Th_d in function $F(\ )$ is 0.5 semitone.

The retrieval accuracy result is illustrated in table 7-5.

Table 7-12: Retrieval performance comparison for different metrics

	Top 1	Top 5	Top 10	Top 20
Metric D_1	70%	74%	84%	85%
Metric D_2	71%	76%	85%	86%
Metric D_3	72%	75%	86%	86%
Metric D_4	75%	79%	85%	88%

The result shows that generally metric D_4 has the best retrieval performance.

7.7. Comparison with Previous Methods

The performance of the proposed methods could be better evaluated by comparison with the previous methods. However, due to the lack of standard melody database and standard humming queries, the comparison has been done based on the results reported by the respected papers.

The performance comparison is between the proposed melody skeleton method using metric D_1 with methods proposed by Jang [Jang_ISR00] and Nishimura [Nish_ISMIR01], which are based on pitch contour matching. The comparison is shown in table 7-13. It can be seen that the proposed melody skeleton method outperforms the previous methods in all the aspects.

Table 7-13: Performance comparison with previous methods

Music retrieval methods	Number of melodies in database	Matching melody beginning		Matching melody anywhere	
		Accuracy for top 10	Average retrieval time	Accuracy for top 10	Average retrieval time
Exhaustive search [Jang_ISR00]	800	85%	1.76 second	Not reported	Not reported
Search by start frame [Nish_ISMIR01]	20	Not reported	Not reported	73.3%	14.2 seconds
This thesis (melody skeleton, metric D_1)	2600	93%	2 seconds	85%	5.5 seconds

7.8. Prototype System Implementation

A prototype music retrieval system has been implemented for the experimentation purposes.

The system has a graphical user interface (GUI) that allows a user to hum a melody query through a microphone (figure 7-9. The system can show the intermediate pitch extraction results (figure 7-10, 7-11). The retrieval result is shown as a ranked list of music entries. Each

entry can be clicked so that the retrieved melody is played back from where it matches with the query (figure 7-12).

The system has been developed by using VC++ on Windows PC platform. The system has several modules: pitch extraction module, melody slope matching module, melody skeleton matching module, music scale estimation module, melody similarity measure module. All the modules are compiled into libraries and integrated in to the GUI.

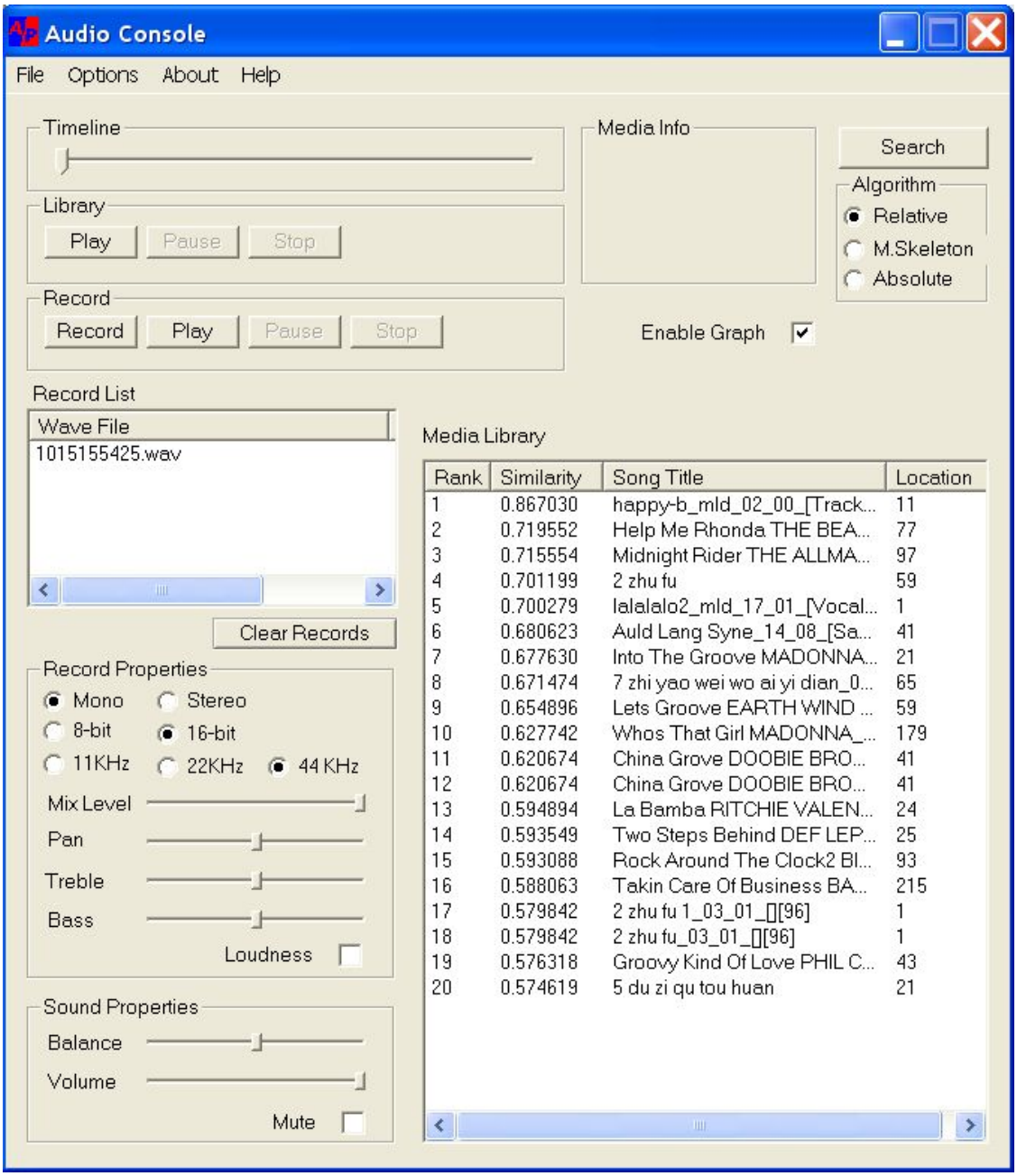

Figure 7-9: GUI of the prototype system

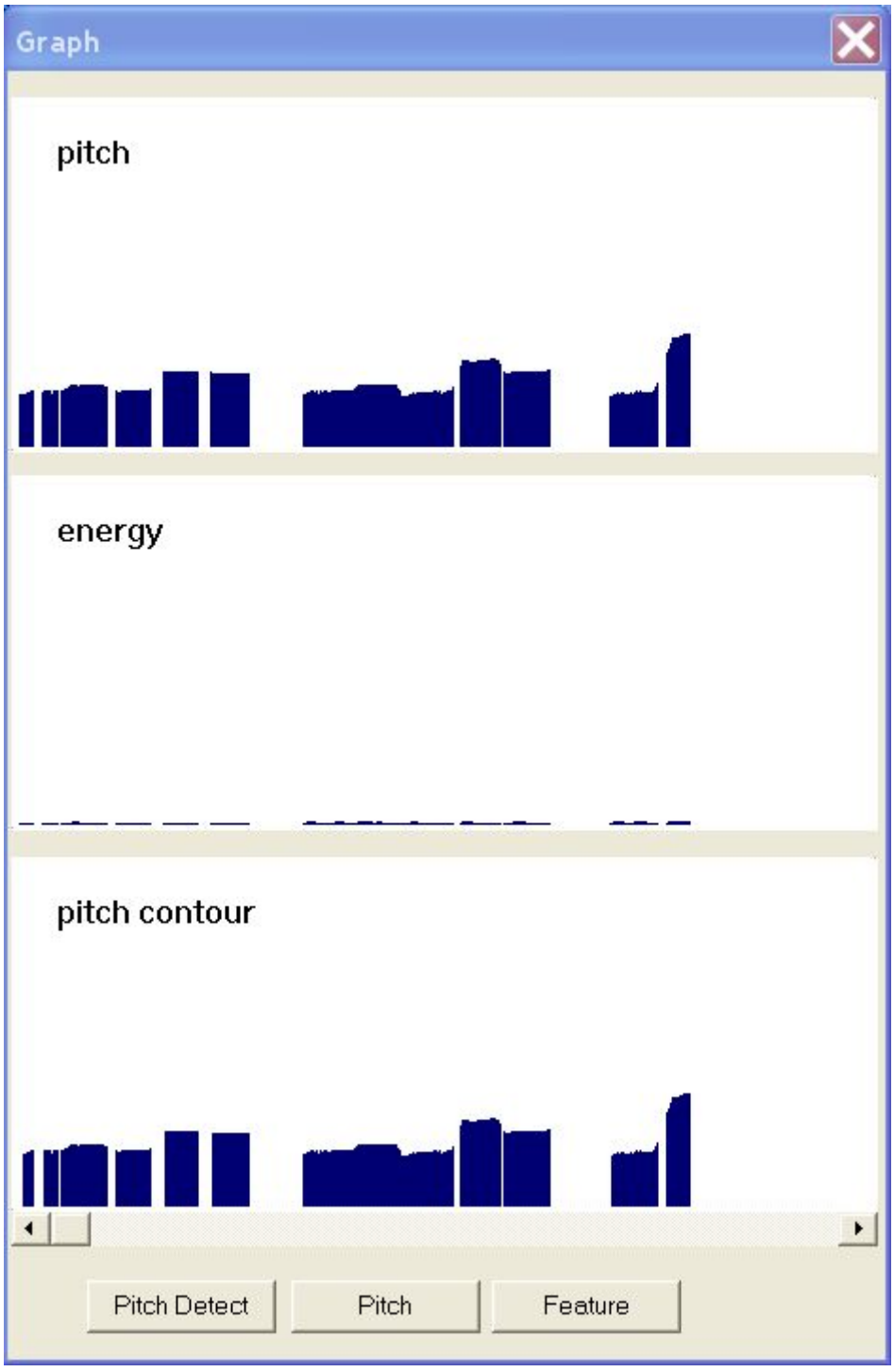

Figure 7-10: Intermediate result in music retrieval by humming

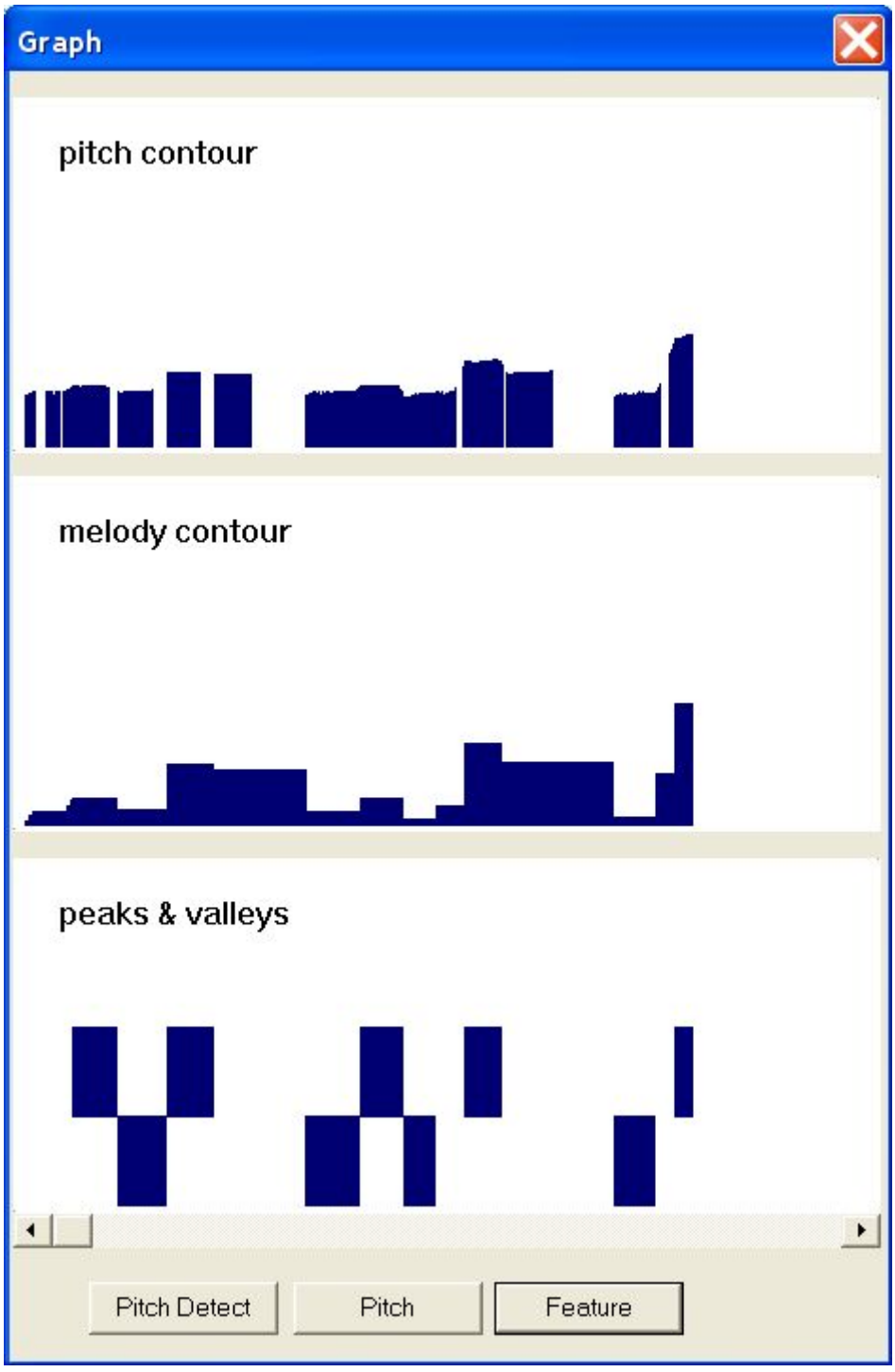

Figure 7-11: Intermediate result in music retrieval by humming

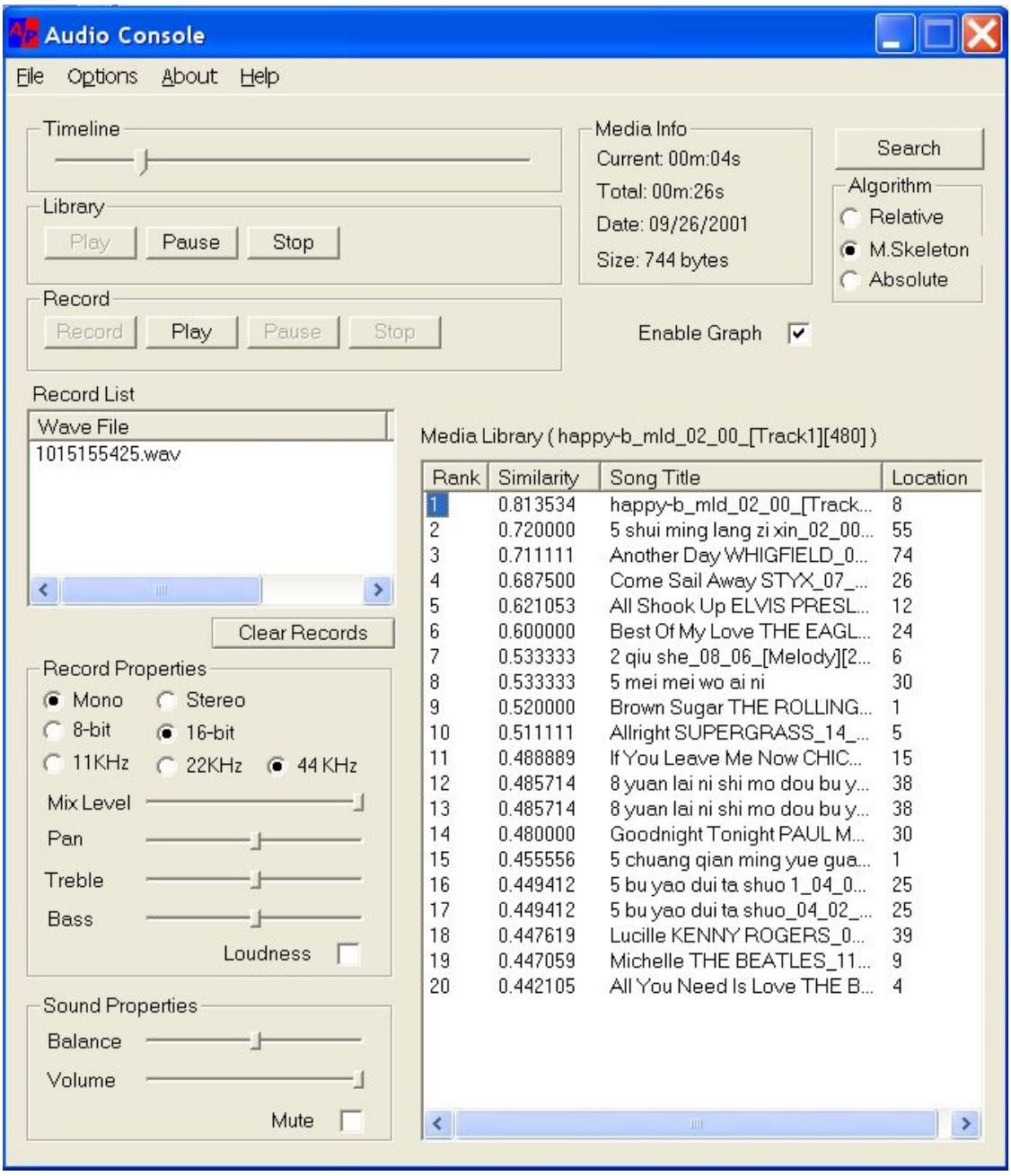

Figure 7-12: Music retrieval result of the prototype system

7.9. Discussion

The experimental results have demonstrated the performance of the proposed techniques.

The pitch extraction method used a novel approach of voice segmentation based on cepstral coefficient. The detected vowel onsets provided important cues for time frames with reliable

pitches, and the pitch contour can then be reliably tracked. The results showed that 96% of the vowel (note) articulation can be correctly detected for varied users, vocal conditions and syllables. Previous pitch detection methods using energy-based segmentation can only achieve 90% for our testing data. The pitch extraction result can be used for both continuous pitch contour matching and discrete note melody matching.

The slope-based approach for melody transposition and alignment was a novel method that utilized global shape features of melody. The slope sequence matching achieved average 94% correct transposition and alignment, which indicated the robustness of the method. The slope-based technique also had a very high efficiency performance, where 99.5% of the corpus candidates were rejected. The overall retrieval accuracy is 64% and 76% for top 10.

The humming query usually contains more than 4 melody slopes. In the melody slope method only the first 4 slopes are used in matching. We suggest a way to utilize all the slopes in the query and possibly improve the retrieval accuracy. Melody matching can be based on all the 4 slope subsequence, and every slope subsequence is searched and aligned with targets in the database. For each valid alignment, all the pitch line segments in the query are used in similarity measurement. This can be done, since a linear tempo change is assumed. Finally, the highest similarity value among all the subsequence alignment with a target melody is returned.

The skeleton-based method has demonstrated the capability of tolerating inconsistent tempo variation and large pitch errors, which could temper the global melody features. The novel point sequence matching method with point skipping mechanism was very robust against those variations and errors. The overall retrieval accuracy is 84% and 93.5% for top 10.

The scale root based method is the first approach to detect a reference pitch (scale root) from melody. The method performed extraordinarily well for pop songs, where 96% of the melodies can be correctly analyzed. One of the advantages of the method is making the melody

matching much more efficient than any previous method, since absolute pitch can be used in the matching. The overall retrieval accuracy is 77% and 89.5% for top 10.

Among the three proposed melody matching methods: melody slope, melody skeleton and scale method, it may be desirable to automatically identify a suitable matching method for a particular query. We suggest the following strategy for this. If the scale root detection has a high confidence, for instance very low model fitting error, then the scale root should be used in pitch transposition. If the note boundaries can clearly indicate the consistency in the tempo, then a melody slope method can be preferred. If otherwise, i.e. the query contains considerable pitch errors (high model fitting errors) and inconsistency of tempo, then the melody skeleton method should be the best choice.

The experiment on similarity measure performance has shown that the melody similarity metric can work well for music retrieval tasks. The comparison of different metrics showed that the non-liner metric (D4) outperforms the basic liner similarity metric (D1).

A comparison with the previous methods has shown that the proposed melody skeleton melody outperforms the previous methods both in accuracy and efficiency.

The implementation of the melody track identification tool and the music retrieval prototype system has demonstrated the feasibility of a real world music retrieval application using the proposed methods.

Chapter 8.
CONCLUSION AND FUTURE WORK

8.1. Summary

This thesis has presented a solution for content-based music retrieval by acoustic melody queries, specifically query-by-humming.

Due to the erroneous nature of note articulation in the hummed melody inputs by lay users, a melody representation that is least affected by note detection error is preferred. This thesis has proposed a time sequence based melody representation, called pitch line, which fulfils the above requirements. Pitch lines can be constructed in a straightforward manner from symbolic melody input. A robust pitch extraction technique has been proposed to detect the pitch value for each individual frame in the acoustic melody input, which may contain variations on volumes and vocal conditions. This technique utilizes the harmonic structure of the singing/humming voice signals to achieve robust pitch detection. Pitch line for acoustic input is finally constructed by a time sequence dimension reduction process, which effectively reduces the storage requirements while preserving the adequate pitch and time precisions.

Similarity matching of melody based on pitch line is the essential part of work in this thesis. A similarity metric for pitch line sequence is proposed. This metric is based on proper key transposition and sequence alignment. Pitch inaccuracy in the intonation normally presented in non-professional's singing/humming is considered in the metric.

In matching time sequences of absolute pitch, the key challenges are the key transposition and sequence alignment issues. This thesis proposed sequence alignment method based on global

features of the time sequence. A shape structure of the sequence, called melody slope, is first proposed for both key transposition and sequence alignment. The melody slope structure is robust against the pitch and speed variation in the acoustic input. Key transposition and sequence alignment for melody matching is based on matching of melody slope sequences. Relative pitch intervals and time duration of melody slope are utilized in the sequence matching. Melody slope sequences matching also serves as a filtering process, which can eliminate wrong candidates at an earlier stage thus increases the efficiency of melody matching. Key transposition is naturally achieved after the slope sequence can find a match. Sequence alignment of pitch line with in melody slope is based on the relative pitch values using a simple dynamic programming computation. Melody slope sequence matching is the first approach proposed for melody matching that utilizes global melody features.

A more sophisticated and robust approach for key transposition and sequence alignment has been proposed to deal with the possible although infrequent errors in melody slope. Errors on melody slope, like slope fragmentation or slope consolidation, could occur in the acoustic melody, if the melody is progressing by very small intervals. The technique is based on matching of melody skeleton represented by a points sequence in the pitch value run domain. The melody skeleton is a very compact representation of melody structure and is invariant to the tempo. A novel technique based on dynamic programming has been proposed for matching of melody skeletons. A point skipping mechanism can compensate the possible errors on the melody skeleton. Key transposition and sequence alignment using melody skeleton is even more robust than melody slopes and particularly can achieve invariance to the tempo in the acoustic melody input.

Key transposition and sequence alignment in the previous techniques are basically correlated, although it is to a less degree for melody slope and melody skeleton approaches. If the key transposition can be isolated from sequence alignment, the computation in melody matching can be drastically reduced. This thesis has been shown that it is achievable. In fact, most music

pieces including songs are composed using particular music scales, which is a subset of the chromatic scale. The notes of the melody are mainly from the specific scale. A music scale modelling technique is proposed to estimate the music scale type and its root in a melody, both symbolic and acoustic. A model for Major scale and Minor scale is proposed for song retrieval, since most songs are written in these scales. The scale estimation is done by fitting the notes or pitch values into the model, and the scale root is detected by a small fitting error. Key transposition within melody is handled by using a sliding window in the scale estimation, and the scale root is declared by grouping the results of all the windows. For key transposition in melody matching, multiple possible roots from acoustic melody are used to match with the single root estimated for symbolic melody. This is safe, because acoustic melody is a portion of the symbolic melody, and the estimated roots for acoustic melody would include the single root estimated for the complete symbolic melody. After key transposition, sequence alignment can be efficiently computed using the dynamic programming technique used in previous method.

Extensive experimentation has been conducted to evaluate the techniques proposed in this thesis. The experiments include the evaluation of pitch extraction from acoustic melody input, the evaluation of melody slope matching for key transposition and sequence alignment, the evaluation of melody skeleton based technique for key transposition and sequence alignment, the evaluation of music scale estimation from both symbolic melody and acoustic melody, the evaluation of the melody metric and retrieval performance for all the proposed key transposition and sequence alignment methods, the evaluation of the proposed metric; and a comparison of the proposed method with the previous methods.

A prototype system with friendly GUI has been implemented for query-by-humming, which can report intermediate results of each music file insertion and each melody query.

8.2. Contributions

The contributions of the thesis are listed as follows:

- A novel melody representation, called pitch line, has been proposed. Pitch line has shown to be sufficient, efficient and robust for representing melody from humming based inputs;

- A pitch extraction method has been proposed to reliably convert a humming signal into the proposed melody representation, which consists of segmentation, reliable pitch detection, pitch tracking and curve aggregation processes. The method has shown to be robust against variations in note articulation and vocal conditions of different users, where the pitch extraction accuracy is 96%.

- A general melody matching approach for pitch line has been proposed. The melody matching approach consists of three distinct processes: key transposition, local melody alignment and similarity measure. This general approach has shown to be more robust and efficient than the existing methods.

- Melody similarity measures using pitch line, defined for particular key transposition and melody alignment, have been proposed. The geometrical similarity measures have been designed with considerations to minimize the effect of pitch inaccuracies in the humming melody input.

- A melody alignment and transposition using global melody feature, called melody slope, has been proposed. Efficient melody alignment and key transposition has been achieved by using melody slope sequence matching. This process also acts as a filtering step that can efficiently reject wrong candidates (99.5%) in the matching of slope sequences. The retrieval accuracy is 76% and 64% for top 10 for matching beginning and anywhere of melodies.

- A melody matching using melody skeleton, an abstract of the melody representation, has been proposed to address the inconsistent tempo variation and occasional large pitch errors in the hummed melody. A novel point sequence matching using dynamic programming has been proposed for melody skeleton matching. This technique has shown to be very robust against tempo variations and large pitch errors (94.5%). The retrieval accuracy is 93.5% and 84% for top 10 for matching beginning and anywhere of melodies. The average retrieval time is 5.5 seconds (matching anywhere)..

- A melody scale and root estimation method has been proposed to assist melody matching. The scale root is estimated by using a scale modelling approach. The scale root of melody has been used for key transposition without local alignment in melody comparison, which has greatly reduced the computation requirement of melody matching, and promoted the efficiency performance of music retrieval. The retrieval accuracy is 89.5% and 77% for top 10 for matching beginning and the middle anywhere of melodies. The average retrieval time is 2 seconds.

- Extensive experiments have been conducted to evaluate the performance of the proposed techniques. The evaluation comprised pitch extraction, melody matching by melody slopes, melody matching by melody skeleton, and melody matching using scale root estimation. A comparison of the proposed methods with an existing method has also been presented.

- A research prototype system based on the proposed methods has been developed, which has facilitated evaluation of the proposed techniques and demonstrated the feasibility of commercial applications on music retrieval by humming.

8.3. Future Work

Content-based music retrieval is still at a very early stage. There are a lot of existing unsolved and emerging problems in content-based retrieval, which would be the focus of our future investigation.

Melody-based retrieval of polyphonic MIDI music is more challenging than monophonic music retrieval. The difficulty comes from the non-existence of a monophonic melody sometimes for the polyphonic music. And music retrieval has to be done by directly matching the melodic query with the polyphonic music note encoding, thus for a valid matching, one note of the melody query may match to any one of the multiple simultaneous notes of the music file. The computation complexity of similarity matching is much higher than melody matching. New techniques are needed to address the issues of robustness, accuracy and efficiency.

Retrieval of acoustic music using melody has a much wider range of application than retrieval of symbolic melody. But it is still no effective methods to extract symbols, like music notes, from polyphonic acoustic music, such as pop songs. Some work has been done to locate the singing portion of a polyphonic acoustic music. It would be an interesting task to convert the singing parts into note-like or pitch-like representation (even though polyphonic), such that a monophonic to polyphonic music matching (a problem discussed in the previous paragraph) can be conducted.

Music retrieval usually involves a very large corpus of music, thus the scalability of the retrieval technique is important. Indexing techniques from database seem to be a solution for this problem. However, special requirements are imposed for melody-based music retrieval, such as key transposition, tempo variation, pitch inaccuracy and etc, which call for novel indexing methodologies.

The music scale modelling technique proposed in this thesis is still limited to western style music, which prominently use Major and Minor scale. The scale modelling approach should be

able to be extended to other types of music. In fact, it would be very interesting and useful to automatically define the scale model of a music piece of any style. This work could lead to a music style classification method, which is very useful for content-based music retrieval.

The scale modelling technique can also be extended for scale or key detection from polyphonic music. For symbolic polyphonic music, the notes from different tracks can be fed to models of major scale, natural minor scale, harmonic minor scale and melodic minor scale, and the scale root and the key type can be detected. The result can be used to identify the melody track and the accompaniment tracks, as well as the key signature of the piece. For acoustic polyphonic music, a frequency domain spectrum analysis is needed to identify the main pitches in the signal, and then the pitch information can be fed into the scale models to estimate the scale or key of the piece of music. Furthermore, the technique can be extended to partition a music based on the key boundaries, where key change occurs.

REFERENCES

[Bain_DL99] Bainbridge D., Nevill-Manning C.G., Witten I.H., Smith L.A. McNab R.J. *Towards a digital library of popular music*. Proc. of the 4th ACM International Conference on Digital Libraries, 161-169.

[Bain_ISMIR00] Bainbridge D. *The role of music IR in the new Zealand digital library project*. In Proc. Of the first International Symposium on Music Information Retrieval, October, 2000.

[Bakh_CMJ97] Bakhmutova V., Gusev V.D., and Titkova T.N. *The search for adaptations in song melodies*. Computer Music Journal, 21(1):58-67, 1997.

[Bell_ISMIR00] Bello J.P., Monti G. and Sandler M. *Techniques for Automatic Music Transcription*, in International Symposium on Music Information Retrieval, 2000.

[Birm_ISMIR01] Birmingham W.P., Dannenberg R.B., Wakefield G.H. Bartsch M., Bykowski D., Mazzoni D. *MUSART: Music retrieval via aural queries*. Proc. of the 2nd Annual International Symposium on Music Information Retrieval, 73-81.

[Blac_MM98] Blackburn S. and DeRoure D. *A Tool for Content Based Navigation of Music*. Proceedings of ACM Multimedia, MM98, 1998.

[Blac_OHSW00] Blackburn S. and DeRoure D.D. *Music part classification in content based systems*. In 6th Open Hypermedia Systems Workshop, San Antonio, TX, 2000.

[Bloo_ES99] Boothooft G. and Pabon P. *Vocal registers revisited*, Proc. Eurospeech'99, Budapest, 423-426, 1999.

[Brow_JAS91] Brown J.C. and Zhang B. *Musical Frequency Tracking using the Methods of Conventional and 'Narrowed' Autocorrelation*. J. Acousti. Soc. Am. 89, 1991.

[Byrd_DL01] Byrd, D. *Music notation searching and digital libraries*. Proc. Of the 1st ACM/IEEE Joint Conference on Digital Libraries, 239-246.

[Cami_CRMM92] Camilleri L. *Computational theories of music*. Computer Representations and Models in Music, pages 171-185. Academic Press, 1992.

[Chai_MMCN02] Chai W. and Vercoe B. *Melody Retrieval On The Web*. Proc. Multimedia Computing and Networking, 2002.

[Chen_IWDE98] Chen J. and Chen A.L.P. *Query by rhythm an approach for song retrieval in music databases*, Proc. International Workshop on Research issues in Data Engineering, pages 139146, 1998.

[Chen_ICME00] Chen A.L.P., Chang M. and Chen J. *Query by Music Segments: An Efficient Approach for Song Retrieval*, Proc. International Conference on Multimedia and Expo. 2000.

[Croc_Hand] Crocker M.J. *Handbook of acoustics*, New York, Wiley 1998.

[Dann_ISMIR01] Mazzoni, D., and Dannenberg, R. B. *Melody matching directly from audio.* In J. S. Downie and D. Bainbridge (Eds.), Proceedings of the Second Annual International Symposium on Music Information Retrieval: ISMIR 2001.

[Dell_NYMP93] Deller J.R., Proakis J.G., Hansen J.H.L. *Discrete time processing of speech signals.* New York: Macmillan Pub. Co. 1993.

[Dowl_CH78] Dowling W. *Scale and contour: Two components of a theory of memory for melodies*. Computers and the Humanities, 16:107-117, 1978.

[Down_SIGIR99] Downie J.S. *Music retrieval as text retrieval: simple yet effective*. In Proc. Of the 22nd International Conference on Research and Development in Information Retrieval, Berkeley, CA, 1999.

[Dure_ISMIR01] Durey A.S. Clements M.A. *Melody spotting using hidden Markov models*. Proc. of the 2nd Annual International Symposium on Music Information Retrieval, 109-117.

[Fran_ICME00] Francu C. and Nevill-Manning C.G. *Distance Metrics and Indexing Strategies for a Digital Library of Popular Music*. Proc. International Conference on Multimedia and Expo. 2000.

[Gias_MM95] Ghias A., Logan J., Chamberlin D., Smith B.C., *Query By Humming: Musical Information Retrieval in An Audio Database*, Proceedings of ACM Multimedia, MM95, 1995.

[Hand_MIT89] Handel S. *Listening: An Introduction to the Perception of Auditory Events*. The MIT Press, 1989.

[Haus_ISMIR01] Haus G. and Pollastri E. *An Audio Front End for Query-by-Humming Systems*, International Symposium on Music Information Retrieval, 2001.

[Jang_ISR00] Jang J.S.R. and Gao M.Y. *A Query-by-Singing System based on Dynamic Programming*. International Workshop on Intelligent Systems Resolutions, Dec 2000.

[Jang_ICME01] Jang R.J.S. Lee H.R. Kao M.K. *Content-based music retrieval using linear scaling and branch-and-bound tree search.* Proc. of International Conference on Multimedia and Expo. 2001

[Jang_PCM01] Jang J.S.R., Lee H.R. and Yeh C.H. *Query-by-Tapping: A New Paradigm for Content-based music retrieval from acoustic input*. IEEE Pacific Rim Conference on Multimedia, Beijing 2001.

[Jang_MM01] Jang J.S.R. and Lee H.R. Hierarchical Filtering Method for Content-based Music Retrieval via Acoustic Input. In proceedings of ACM Multimedia, MM00, 2001.

[Kage_ICMC93] Kageyama T., Mochizuki K., and Takashima Y. *Melody retrieval with humming*. Proceedings of ICMC 1993.

[Kar_Download] *Download karaoke files and players.* http://www.schok.co.uk/kb/downloads.htm

[Kim_ISMIR00] Kim Y.E., Chai W., Garcia R. and Vercoe B. Analysis of a Contour-based Representation for Melody. International Symposium on Music Information Retrieval, 2000.

[Kosu_MM00] Kosugi N., Nishihara Y., Sakata T., Yamamuro M., and Kushima K. *A Practical Query-By-Humming System for a Large Music Database*. In proceedings of ACM Multimedia, MM00, 2000.

[Korn_CM98] Kornstadt A. *Themefinder: A web-based melodic search tool*. Computing in Musicology, 11:231-236, 1998.

[Lems_ICMC98] Lemstrom K. and Laine P. *Musical information retrieval using musical parameters*. In Proc. International Computer Music Conference, pp 341-348, Ann Arbour, 1998.

[Lems_ICMC99] Lemstrom K. Laine P. and Perttu S. *Using Relative Interval Slope in Music Information Retrieval*. Proc. International Computer Music Conference 1999.

[Lems_ISMIR00] Lemstrom K., Perttu S. *SEMEX: An efficient retrieval prototype*. Proc. of the 1st Annual International Symposium on Music Information Retrieval.

[Lind_MIT96] Lindsay A. T. *Using contour as a mid-level representation of melody*. Master's thesis, MIT Media Lab, 1996.

[McNab_DL96] McNab R.J., Smith L.A., Witten I.H., Henderson C.L., and Cunningham S.J. *Towards the digital music library: Tune retrieval from acoustic input*. Proceedings of Digital Libraries Conference, 1996.

[McNa_ICME00] McNab R.J., Smith L.A.. *Evaluation of a Melody Transcription System*. In proceedings of International Conference on Multimedia and Expo, 2000.

[Nish_ISMIR01] Nishimura T., Hashiguchi H., Takita J., Zhang J. X., Goto M., and Oka R., *Music Signal Spotting Retrieval by a Humming Query Using Start Frame Feature Dependent Continuous Dynamic Programming*, Proc. 3rd International Symposium on Music Information Retrieval, Indiana, USA, October 15-17, 2001.

[Melu_DL99] Melucci M., Orio N. *Music information retrieval using melodic surface*. Proc of the 4th ACM Conference on Digital Libraries, 152-160.

[Mong_CH90] Mongeau M. and Sankoff D. *Comparison of musical sequences*. Computers and the Humanities, 24:161-175, 1990.

[OMai_CM98] O'Maidin D. *A geometrical algorithm for melodic difference*. Computing in Musicology, 11:65-72, 1998.

[Poll_ICME02] Pollastri E., *A Pitch Tracking System Dedicated to Process Singing Voice for Music Retrieval*, In Pro. IEEE Int. Conf. on Multimedia and Expo, ICME2002

[Prec_CHI01] Prechelt L., Typke R. *An interface for melody input*. ACM Transactions on Computer-Human Interaction, 8(2), 133-149.

[Rabi_PH93] Rabiner LR, Juang BH. *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[Roll_MM99] Rolland P.Y., Raskinis G. and Ganascia J.G. *Musical content-based retrieval: an overview of the melodiscov approach and system*. Proc. ACM Multimedia 99, 1999.

[Pick_SIGIR01] Pickens J. *A Survey of Feature Selection Techniques for Music Information Retrieval*, SIGIR 2001, New Orleans, USA, Sept. 10-12, 2001.

[Poll_ICME02] Pollastri E. *A Pitch Tracking System Dedicated to Processing Singing Voice for Music Retrieval*. Proc. International Conference on Multimedia and Expo, 2002.

[Sodr_ECIRR02] Sodring T., Smeaton A. *Evaluating a melody extraction engine*. Proc of the 24th BCS-IRSG European Colloquium on IR Research. 2002

[Sono_ICMC00] Sonoda T., Muraoka Y. *A WWW-based music retrieval system: An indexing method for a large melody database*. Proc of the International Computer Music Conference (ICMC 2000), 170-173.

[Titze_PH94] Titze I.R. *Principles of Voice Production*, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[Tsen_SIGIR99] Tseng Y.H. *Content-based retrieval for music collections*. ACM-SIGIR, 1999.

[Uitd_MM98] Uitdenbogerd A.L. and Zobel J. *Manipulation of music for melody matching*. Proc. ACM Multimedia 98, 1998.

[Uitd_MM99] Uitdenbogerd A.L. and Zobel J. *Melodic Matching Techniques for Large Music Databases*. Proceedings of ACM Multimedia, MM99, 1999, pp. 57-66.

[Uitd_ACSC02] Uitdenbogerd A.L. and Zobel J. *Music Ranking Techniques Evaluated*. 2tth Australasian Computer Science Conference (ACSC2002), Melborne, Australia.

[Vanbasco] http://www.vanbasco.com/

[Wiki_Pitch] http://en.wikipedia.org/wiki/Pitch_(music)

[Wiki_Scale] http://en.wikipedia.org/wiki/Scale_(music)

[Yang_MM02] Yang C. *Efficient Acoustic Index for Music Retrieval with Various Degrees of Similarity*. Proceedings of ACM Multimedia, MM02, 2002.