Melody Retrieval On The Web

by

Wei Chai

B.S. Computer Science, Peking University, 1996
M.S. Computer Science, Peking University, 1999

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements of the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2001

Signature of Author _____
Program in Media Arts and Sciences
August 6, 2001

Certified By _____
Barry L. Vercoe
Professor of Media Arts and Sciences
Massachusetts Institute of Technology
Thesis Supervisor

Accepted By _____
Andrew B. Lippman
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences
Massachusetts Institute of Technology

Melody Retrieval On The Web

by

Wei Chai

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 6, 2001
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences

ABSTRACT

The emergence of digital music on the Internet requires new information retrieval methods adapted to specific characteristics and needs. While music retrieval based on the text information, such as title, composers, or subject classification, has been implemented in many existing systems, retrieval of a piece of music based on musical content, especially an incomplete, imperfect recall of a fragment of the music, has not yet been fully explored.

This thesis will explore both theoretical and practical issues involved in a web-based melody retrieval system. I built a query-by-humming system, which can find a piece of music in the digital music repository based on a few hummed notes. Since an input query (hummed melody) may have various errors due to uncertainty of the user's memory or the user's singing ability, the system should be able to tolerate errors. Furthermore, extracting melodies to build a melody database is also a complicated task. Therefore, melody representation, query construction, melody matching and melody extraction are critical for an efficient and robust query-by-humming system. Thus, these are the main tasks to be addressed in the thesis.

Compared to previous systems, a new and more effective melody representation and corresponding matching methods which combined both pitch and rhythmic information were adopted, a whole set of tools and deliverable software were implemented, and experiments were conducted to evaluate the system performance as well as to explore other melody perception issues. Experimental results demonstrate that our methods incorporating rhythmic information rather than previous pitch-only methods did help improving the effectiveness of a query-by-humming system.

Thesis Supervisor: Barry L. Vercoe
Title: Professor of Media Arts and Sciences

Melody Retrieval On The Web

by

Wei Chai

The following people served as readers for this thesis:

Thesis reader: _____

Joseph A. Paradiso
Principal Research Scientist, Media Laboratory
Massachusetts Institute of Technology

Thesis reader: _____

Christopher Schmandt
Principal Research Scientist, Media Laboratory
Massachusetts Institute of Technology

ACKNOWLEDGEMENTS

I have been very lucky to work in the Machine Listening Group of the Media Laboratory for the past two years. This allowed me to collaborate with many brilliant researchers and musicians. My period of graduate study at MIT has been one of the most challenging and memorable so far in my life. I am happy to have learned about many new technologies, new cultures, and especially the innovative ways people carry out research at MIT. This thesis work was funded under MIT Media Laboratory Digital Life Consortium. I would like to thank everyone who has made my research fruitful and this thesis possible.

I am indebted to my advisor, Professor Barry Vercoe, for giving me the freedom to pursue my own interests and for his suggestions which inspired my thoughts. Also, I am grateful for the encouragement, help and insight I have received from current and past members of the Machine Listening Group, especially Bill Gardner, Ricardo Garcia, Youngmoo Kim, Nyssim Lefford, Keith Martin, J.C. Olsson, Joe Pompei, Rebecca Reich, Eric Scheirer and Paris Smaragdis. Additionally, I would like to thank my readers Dr. Joseph Paradiso and Christopher Schmandt for their advice, and Professor Judy Brown, Connie Van Rheenen and Elizabeth Marzloff for their help.

I would like to express my sincerest thanks to all the participants in my experiments. They gave me good suggestions and feedback for improving the system. I would like to especially thank Donncha Ó Maidín who provided me his music corpus and the handy tool he developed to process those score data.

My previous advisors and colleagues at the Database Laboratory of Peking University and the National Laboratory on Machine Perception all gave me their support and help when I was in China. I would also like to thank them especially Professor Shiwei Tang, Professor Dongqing Yang, Professor Fang Pei and Jian Pei.

Finally, I would like to thank my family and all my friends for their encouragement.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

Technology has been changing the world rapidly and dramatically. The recent controversies surrounding the Napster phenomenon have brought me to believe that people's strong desire to obtain and share music freely, along with the irresistible trend of technology, will ultimately break the traditional models of the music industry and even people's music appreciation habits.

1.1 Motivation

The Internet has become an important source for people to obtain music for entertainment, education, business, or other purposes. This requires new information retrieval methods adapted to specific characteristics and needs. Although there are many websites or Internet based agents for music sale, advertisement and sharing, the interfaces are not as convenient for finding desired music; only category-based browsing and/or text-based searching are supported. To find a piece of music, the user needs to know its title, composer, artist or other text information, so that the search engine can search the database based on that information; otherwise the user needs to browse the whole category. This procedure might be very time-consuming. Therefore, developing new methods to help people retrieve music on the Internet is of great value.

A natural way of searching for music is to identify it by its content (e.g., melody) rather than its secondary features (e.g., title), because the content is usually more memorable and a more robust feature of the musical work.

My thesis is to build a query-by-humming system (called the QBH system), which can find a piece of music in the digital music repository based on a few hummed notes. When the user does not know the title or any other text information about the music, he is still able to search for music by humming the melody.

Query-by-humming is a much friendlier interface than existing systems for music searching on the Internet. In addition, although the system I built is a web-based system, it can certainly be moved to other application fields, for example, Audio-On-Demand (AOD) or Karaoke systems.

Besides the above application value, the query-by-humming system is also an interesting topic from a scientific point of view. Identifying a musical work from a melodic fragment is a task that most people are able to accomplish with relative ease. However, how people achieve this is still unclear, i.e., how do people extract melody from a complex music piece and convert it to a representation that could be memorized and retrieved easily and accurately with tolerance of some transpositions? Although this whole question is beyond the scope of this thesis, we will build a system that performs like a human: it can "extract" melodies from music; it can convert the melodies into an efficient representation and store them in its "memory"; when a user asks for a piece of music by humming the melody, it can first "hear" the query and then search in its "memory" for the piece that it "thinks" most similar to the query.

Given the application value and the scientific interest of the research, I propose to explore the melody retrieval problem from two perspectives: as a practical solution to a query-by-humming system, and as a scientific inquiry into the nature of the melody perception process.

The main features of this system as compared with other existing systems are:

- A new melody representation, which combines both pitch and rhythmic information.
- New approximate melody matching algorithms based on the representation.
- A set of automatic transcription techniques customized for the query-by-humming system to obtain both pitch and rhythmic information.
- A handy tool to build a melody database from various symbolic representations including score formats and MIDI format.
- A deliverable query-by-humming system including both the server application and the client application.

1.2 Methodology

There are several important issues in building such a query-by-humming system. One is that an input query (hummed melody) may have various errors due to uncertainty of the user's memory or the user's singing ability. Even if the hummed queries are perfect, it is still difficult to implement a 100% accurate system for transcribing the hummed signals into musical symbols, which are needed for melody matching in the next step. To tolerate these errors, we need effective representation and musically reasonable approximate matching method. Additionally, extracting melody information from an existing music corpus is not a trivial task.

Therefore, I divide the whole problem into four sub-problems: the melody representation problem, the melody matching problem, the melody extraction problem and the query construction problem. I believe the above four problems are the key points for a complete, robust and efficient query-by-humming system, and thus these are the focuses of my thesis.

To solve the four problems and build the system, several theories, approaches or techniques will be explored:

- Music psychology, especially related to melody perception.
- Musical signal processing, particularly pitch tracking and beat tracking.
- Pattern recognition, focusing on approximate matching and time-series analysis.
- Other techniques, such as computer networks, database systems and object-oriented analysis.

1.3 Organization

My thesis is divided into eight chapters. In chapter 2, *Background*, I review the status, problems and approaches of music information retrieval in general, and then specifically the research and approaches of existing query-by-humming systems.

Chapter 3, *Melody As A Significant Music Feature*, presents the music psychology principles and theories that are important for query-by-humming systems, for example, melody

perception, music memory, melodic organization, melodic similarity, etc. An interesting experiment of classifying folk music from different countries based on their melodies using hidden Markov models (HMMs), which illustrates that melody itself carries some statistical features to distinguish the styles of music, is also presented in this chapter. This experiment also points out some melody representation issues that will be focused on in later chapters.

Chapter 4, *System Architecture*, presents an overview of the system architecture, the functionality of each part and the communication between different parts.

Chapter 5, *Approaches and Algorithms*, presents our solutions to solve the melody representation problem, the melody matching problem, the melody extraction problem and the query construction problem respectively. Several new algorithms are proposed in this chapter and in Chapter 7 will be shown to be more effective than previous algorithms.

Chapter 6, *Implementation Issues*, presents some technical details to build the system efficiently.

Chapter 7, *Experiments and Evaluation*, presents the experiments for evaluating the system and their results. The effectiveness of different algorithms and interfaces is compared in the experiments. Several melody perception issues are explored as well.

Chapter 8, *Conclusions*, summarizes the approaches I used in building the query-by-humming system, the melody perception issues involved in such a system, and the contribution made in the thesis. Directions for further research are suggested.

CHAPTER 2 BACKGROUND

This chapter reviews the status, problems and approaches of music information retrieval in general, and then specifically the research and approaches of existing query-by-humming systems.

2.1 Music Information Retrieval

With the emergence of digital music on the Internet, automating access to music information through the use of computers has intrigued music fans, librarians, computer scientists, information scientists, engineers, musicologists, cognitive scientists, music psychologists, business managers and so on. However, current methods and techniques for building real-world music information retrieval systems are far from satisfactory.

The dilemma was pointed out by Huron (2000). Music librarians and cataloguers have traditionally created indexes that allow users to access musical works using standard reference information, such as the name of the composer or the title of the work. While this basic information remains important, these standard reference tags have surprisingly limited applicability in most music-related queries.

Music is used for an extraordinary variety of purposes: the restaurateur seeks music that targets certain clientele; the aerobics instructor seeks a certain tempo; the film director seeks music conveying a certain mood; an advertiser seeks a tune that is highly memorable; the physiotherapist seeks music that will motivate a patient; the truck driver seeks music that will keep him alert; the music lover seeks music that can entertain him. Although there are many other uses for music, music's preeminent functions are social and psychological. The most useful retrieval methods are those that facilitate searching according to such social and psychological functions.

In attempting to build systems supporting such retrieval methods, two general questions arise:

- What is the best taxonomic system by which to classify moods, styles, and other musical characteristics?

- How can we create automated systems that will reliably characterize recordings or scores?

The first question is related to the *feature selection* problem, which involves defining a comprehensive set of features including low level and high level features, objective and perceptual features, that can describe music well for the retrieval purpose. The second question is related to the *feature extraction* problem, which involves building computer systems to compute those features automatically.

Regarding the feature selection question, the features that have been proposed can be categorized as follows:

- *Reference-related features*, such as title, composer, performer, genre, country, date of composition, date of first performance publisher, copyright owner and status, etc.

- *Content-related features*, such as tonality, contour, meter, tempo, rhythm, harmony, lyrics, libretti, instrumentation, mood, style, similarity, etc.

Regarding the feature extraction question, current research goes in several directions:

- *Extracting content-related features from symbolic musical data*. For example, current query-by-humming systems aim to retrieve symbolic music data based on similarity. Thus, they belong to this category. How to extract melody (Uitdenbogerd, 1998) or motive (Liu, 1999) from the polyphonic scores has also been explored. Dannenberg (1997) built a real-time performance style classifier based on MIDI data. Brown (1993) presented a way of using autocorrelation to determine the meter of music scores. Chai (2000) attempted to extract several perceptual features from the MIDI files.

- *Extracting content-related features from acoustic musical data*. For example, determining pitch contour of audio signals has been extensively studied (Rabiner, 1976) (Roads, 1994). Tempo and beat tracking from acoustic musical signals has also been explored (Goto, 1994) (Goto, 1997) (Scheirer, 1998). Music summarization, i.e., attempting to extract musical themes or hooks from musical recording for identifying or recognizing a work, has been studied recently (Huron, 2000) (Chu, 2000). Instrument identification is fully explored by Martin (1999). Automatic music transcription is also widely studied.

- *Extracting reference-related features from symbolic musical data*. For example, the project I did for classifying folk music from different countries based on the melodies belongs to this category. It will be presented in detail in Section 3.5.

- *Extracting reference-related features from acoustic musical data*. Tzanetakis (2000) reported a result of 75% classification accuracy to classify classical, modern (rock, pop) and jazz using different features (FFT, MPEG filterbank, LPC and MFCC) and two classification techniques (Gaussian Mixture Model and K-NN). Pye (2000) reported a result of 92% classification accuracy to classify blues, easy listening, classical, opera, dance (techno) and Indie rock, using two different features (MFCC, MP3CEP) and two classification techniques (Gaussian Mixture Model and TreeQ).

In applications with symbolic music data, techniques used in text retrieval systems have been widely adopted. In applications with acoustic musical data, many signal processing techniques are employed.

General methods that can manipulate both symbolic data and acoustic data have also been proposed. For example, a new representation called Self-similarity Visualization was proposed by (Foote, 1999). Both raw audio and symbolic audio can be represented in this way. Doing music analysis and classification based on this representation is still being studied.

So far there are no standard evaluation criteria for these systems.

2.2 Query-by-humming systems

The query-by-humming system was first proposed by Ghias *et al*. (1995). Following Ghias *et al.*, several research groups including the MELDEX project by McNab *et al.* (1996), the Themefinder project by Stanford University, the TuneServer project by University of Karlsruhe, the MiDiLiB project by University of Bonn, etc., are working in this area.

Existing systems use a variety of representations for melody, and usually aim for flexibility in order to accommodate variations in data and query format. Symbolic music corpora, such as MIDI, are generally used. The data used in different systems varies greatly, but consists primarily of classical and folksong repertoires, since the copyright on most of these works has expired and they are now in the public domain. The data is taken almost exclusively from Western music, or at least music using the Western tonal system (12 half-steps per octave). The reasons for this are primarily practical, since MIDI and other machine-based formats for storing notation were designed for Western tuning, and there are neither standard formats nor standardized methods of adapting existing formats for non-Western tuning.

The system by Ghias *et al.* (1995) is the first full query-by-humming system, which included processing of an audio input signal in order to extract the necessary query information. However, this system was too simple in its melody representation and matching method. It used only 3-level contour information (U/D/S indicating that the interval goes up, down or remains the same) to represent melodies. Baesa-Yates and Perleberg algorithm was adopted for approximate matching with k mismatches. The size of the melody database and experiments for evaluating the system based on different human subjects were not reported.

MELDEX is the most well-known query-by-humming system so far besides Ghias' system. It also implemented a full system and was the first one, as far as we know, whose client application was put on the web. MELDEX allowed a variety of different contour representations, such as exact interval or finer (>3-level) contour information. It also attempted to incorporate rhythmic information (mainly represented as note durations), but the corresponding matching method was not effective. For example, absolute note durations (e.g., a quarter note at 90bpm) were used for melody matching. In practice, almost no users can hum in such a precise way. And MELDEX attempted to identify only the beginnings of melodies. In addition, the client-server division of MELDEX was not efficient. The client application only did the recording and then sent the recorded waveform data to the server side to do transcription and matching. Dividing in this way cannot employ the computational ability of the client side machine, while increasing the server side burden. It also increases the network bandwidth requirement. Two experiments were done according to (McNab *et al.*, 1996). One was to explore human performance in melody recall; the other was to match perfect queries (not real hummed queries) of different lengths with some folksong corpora to evaluate effectiveness of different melody representations. Experiments for evaluating the system based on real hummed queries were not reported.

Strictly speaking, the Themefinder project is not a query-by-humming system, because it only supported text format queries, i.e., the users had to manually input the text string using some predefined format to represent a query. But the Themefinder project is quite famous in its symbolic data processing and organization.

The TuneServer project also used only 3-level contour information to represent melodies. Although this system did not propose any new methods, its melody database was quite large, which was based on a book by *Denys Parsons: The Directory of Tunes and Musical Themes, Spencer Brown, 1975*. This book does not only include the full database with 10,370 classical tunes and musical themes, but also more than 5,000 popular tunes from the time between 1920 and 1975, as well as a list of national anthems.

The MiDiLiB project allowed a variety of different contour representations, such as exact interval or finer (>3-level) contour information. Similarly, this system did not propose any new

methods, but its melody database was very large, which contained 20,000 MIDI files. Some sophisticated melody extraction tools and algorithms were used in building this database.

As far as I know, there are no accuracy and effectiveness evaluations of the above systems reported based on real hummed queries. Since they are not public accessible or not working at this time (e.g. MELDEX, TuneServer), it is also impossible to evaluate these systems by the author.

CHAPTER 3 MELODY AS A SIGNIFICANT MUSICAL FEATURE

This chapter presents the music psychology principles and theories that are important for query-by-humming systems, for example, melody perception, music memory, melodic organization, melodic similarity, etc. An interesting experiment of classifying folk music from different countries based on their melodies using hidden Markov models (HMMs), which illustrates that melody itself carries some statistical features to distinguish the styles of music, is also presented in this chapter. This experiment also points out some melody representation issues that will be focused on in later chapters.

3.1 Melody Perception

How do we understand what we hear? How do we make sense of what we hear as music? One of the most evident features of music is melody. A definition of melody is that it is a coherent succession of single pitches. In such a melody, the succession of notes seems to hold together in a meaningful, interesting way – interesting, and also emotional. Of all music's structures, melody is the one that moves people the most and seems to evoke human sentiment most directly. Familiar melodies "register" simple qualities of feeling instantly and strongly.

The most familiar type of melody is a *tune* – a simple, easily singable, catchy melody such as a folksong, or a dance. A tune is a special kind of melody. Melody is a term that includes tunes, but also much else. A *motive* is a distinctive fragment of melody, distinctive enough so that it will be easily recognized when it returns again and again within a long composition. Motives are shorter than tunes, shorter even than phrases of tunes; they can be as short as two notes. *Theme* is the most general term for the basic subject matter of longer pieces of music. A single melodic "line" in time is enough to qualify as music: sometimes, indeed, as great music.

Texture is the term used to refer to the blend of the various sounds and melodic lines occurring simultaneously in music. *Monophony* is the term for the simplest texture, a single unaccompanied melody. When there is only one melody of real interest and it is combined with other sounds, the texture is called *homophony*. When two or more melodies are played or sung simultaneously, the texture is described as *polyphony*.

Melody is a perceptual feature of music. Sometimes what one person perceives to be the melody is not what another perceives. What makes melodies interesting and memorable is a combination of things: the overall shape- what is sometimes called "line" or "melodic contour"; the rhythm; and the type of movement (i.e., leaping or stepwise). In a broader sense, a melody can be pitched or purely rhythmic, such as a percussion riff. However, our research does not attempt to address all of these cases and is limited in scope to pitched, monophonic melodies. We assume that all the pieces in our music corpora have monophonic melodies, which the users can easily and consistently identify. Furthermore, the user should be fairly familiar with the melody he wants to query, though perfect singing skill is not required.

Several aspects of melody perception need to be considered when developing the melody retrieval systems.

(1) What is the type of query that users will present? In the case of someone trying to locate a half-remembered fragment of music, it is useful to understand how people remember music, and in particular, how they remember melodies and how accurately people can produce (e.g., hum) melodies.

(2) Since most music that we hear contains both melody and accompaniment, it is necessary to determine what would be perceived as melody in an accompanied musical work.

(3) Since many music queries involve finding similar but not exact matches to melodies, we need to decide what similarity means in terms of music perception.

The first aspect is related to the Melody Representation problem (Section 5.1) and the Query Construction problem (Section 5.3). The second aspect is related to the Melody Extraction problem (Section 5.2). The third aspect is related to the Melody Matching problem (Section 5.1).

3.2 Music Memory

There has been much research on how people build mental structures while listening to music and on how music is remembered. Dowling (1978) discovered that melody contour is easier to remember than exact melodies. Contour refers to the shape of the melody, indicating whether the next note goes up, down, or stays at the same pitch. He postulates that contour and scale are stored separately in memory and that melodies are mapped onto an overlearned scale structure, so that if presented a melody with an unknown scale, we map it to the scale structure with which we are familiar.

Edworthy (1985), however, discovered that the length of a novel melody determines whether it is easier to detect changes in contour or changes in intervals. There are also differences in terms of short-term and long-term memory for melodies: it is easy for people to reproduce a well-known melody with exact intervals than to do so for new melodies. In terms of music database queries, a user's query melody will probably be in long-term memory, so exact intervals will be important. Memory tasks are generally performed better by experienced musicians than by those with less experience.

Dowling (1986) presented evidence supporting the notion that schematic representations of familiar tunes in long-term memory consist of (rhythmically organized) sets of relative pitch chromas and that such representations can be accessed by means of labels and such global melodic features as contour. Short-term (episodic) memory, long-term (semantic) memory, and cognitive development were investigated with respect to the roles played by pitch, intervals, contour, and tonal scales.

3.3 Melodic Organization and Gestalt Principles

Melody seems to exist as a succession of discrete pitches in time, yet it is likely to be heard as one thing, one integrated entity, as pitch moving in time. The fact that a melody can be heard as pitch moving in time when all that a listener is confronted with is a sequence of separate pitches is something that has perplexed philosophers for centuries.

An answer was hinted by the Gestalt psychologists earlier in this century. They were trying to account for the perception of visual shape, but their theories seemed also to apply to melody, to

auditory shape. They suggested that certain laws seemed to underlie our perception of form, laws of proximity, similarity, good continuation and so on. These laws governed our tendencies to group elements in a visual field so as to constitute shape or form at a larger scale than the individual elements of the pattern.

Gestalt principles can serve as useful rules of thumb, indicating what sort of pattern organization will be perceived given certain stimulus features. They are especially amenable to translation from visual to auditory space where the relatively brief time spans of local stimulus organization are involved. (Dowling, 1986)

- *Proximity*: Items placed in close (spatial) proximity tend to be grouped together as a unit. The importance of pitch proximity in audition is reflected in the fact that melodies all over the world use small pitch intervals from note to note (this will be shown by the interval change histogram of our music corpora in Section 5.2). Violations of proximity have been used in various periods and genres of both Western and non-Western music for a variety of effects. For example, fission based on pitch proximity was used to enrich the texture so that out of a single succession of notes, two melodic lines could be heard.

- *Similarity*: Similar items tend to be grouped together as a unit. For example, similarity of timbre can be used to group sequences of sounds so that the melody will be perceived. Similar amplitude or loudness of notes was found to be fairly unimportant in the perception of musical parts compared to other rules, but is also used for grouping of notes.

- *Prägnanz (Good continuation)*: In perception we tend to continue contours whenever the elements of a pattern establish an implied direction. Here, good means symmetrical, simple, regular, but most importantly familiar. The listener could attend to notes of a familiar melody even though they are not differentiated from their background on any other basis than being parts of a meaningful configuration.

Additionally, the listeners usually pay more attention to louder and higher pitch notes as in a melody line.

The above rules were referred to as local pattern organization by Dowling (1986). The listener's appreciation of the global aspects of a piece of music is in terms of invariants -- structural constancies underlying surface change in local pattern features. Invariants that might occur over relatively long-time periods in a piece are regularities of temporal organization such as the beat, of tonal scale organization such as the key, and of instrumentation and density of note spacing in pitch and time. One way in which the comprehension of global invariants of the structure might function as we listen to a piece is by affecting our expectancies from moment to moment.

Dowling (1986) also proposed that the local features are related to each other and to global patterns by some hierarchical structure.

3.4 Melodic Similarity

The ability to recognize melodic similarity lies at the heart of many of the questions most commonly asked about music. It is melody that enables us to distinguish one work from another. It is melody that human beings are innately able to reproduce by singing, humming,

and whistling. It is melody that makes music memorable: we are likely to recall a tune long after we have forgotten its text. (Hewlett, 1998)

(Hewlett, 1998) reviews various theories about melodic similarity. In this book, conceptual and representational issues in melodic comparison are reviewed by Selfridge-Field; different melody matching methods are proposed, such as Geometrical algorithm, Dynamic Programming algorithm and other string-matching algorithms.

Uitdenbogerd (1998) suggested the following ordering of factors in music similarity, from good evidence of similarity to poor. Rhythmic aspect was not considered in this ordered list.

1) Exact transposition in the same key.
2) Exact transposition in a closely related key.
3) Exact transposition in a distant key.
4) Exact transposition except for a chromatically altered note.
5) Exact transposition except for a diatonically altered note.
6) Same contour and tonality.
7) Same contour but atonal.
8) Same note values but different contour.
9) Different contour, different note values.

3.5 Melody as a Feature for Folk Music Classification

There is another perspective from which to think about melodic similarity: melodic discrimination. That means to use melody as a feature classifying music. One interesting experiment we did (Chai, 2001) was to classify folk music from different countries based on their monophonic melodies using hidden Markov models. This section will present our experiment, which suggests to us that melodic similarity may have statistical explanations and melody representation is very important for melodic discrimination. Please note that the methods and the detailed results presented in this section will not be directly related to the rest of the thesis; readers may skip this section if they want.

Here is the description about our experiment. We chose folk music as our experiment corpus, because

- Most folk music pieces have obvious monophonic melody lines, which can be easily modeled by HMM. “Monophonic” here means only one single tone is heard at a time, and there is no accompaniment or multiple lines going simultaneously.

- Melodies of folk music from different countries may have some significant statistical difference, which should be able to be captured by an appropriate statistical model.

Bruno Nettl mentioned in his book (Nettl, 1973) that melody is the aspect of music that has been of the greatest interest to folk music study, but is also probably the most difficult part.

In the classification experiment, 187 Irish folk music pieces, 200 German folk music pieces and 104 Austrian folk music pieces were used. We don’t have specific reasons for choosing these three countries except for the availability of data. The data were obtained from two corpora, which are also contained in the QBH melody database:

- Helmut Schaffrath's Essen Folksong Collection which contains over 7,000 European folk melodies encoded between 1982 and 1994;

- Donncha Ó Maidín's Irish Dance Music Collection.

All pieces have monophonic melodies encoded in either of the two symbolic formats: **kern and EsAC.

In the experiment, we represented melodies in four different ways.

(A) Absolute pitch representation. A melody is converted into a pitch sequence by normalizing the pitches into one octave from C4 (Middle C) to B4, that is, pitches in different octaves but of same chroma are encoded with the same symbol in the sequence and thus there is a total of 12 symbols.

(B) Absolute pitch with duration representation. To incorporate rhythm information, we make use of the concept behind the representation for rhythmic sequences employed by Carpinteiro (1998). Briefly, we simply repeat each note multiple times to represent the duration, e.g., in our experiment, how many half-beats the note lasts.

(C) Interval representation. A melody is converted into a sequence of intervals. An interval is the difference of the current note and the previous note in semitones. There are 27 symbols indicating –13 to 13 semitones (intervals larger than 13 semitones are all indicated by +/-13).

(D) Contour representation. This is similar to Interval representation, but we quantize interval changes into five levels, 0 for no change, +/- for ascending/descending 1 or 2 semitones, ++/-- for ascending/descending 3 or more semitones. Thus, there is a total of 5 symbols. This representation is fairly compact and fault-tolerant in melody identification applications (Kim, 2000).

For example,

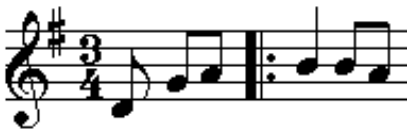
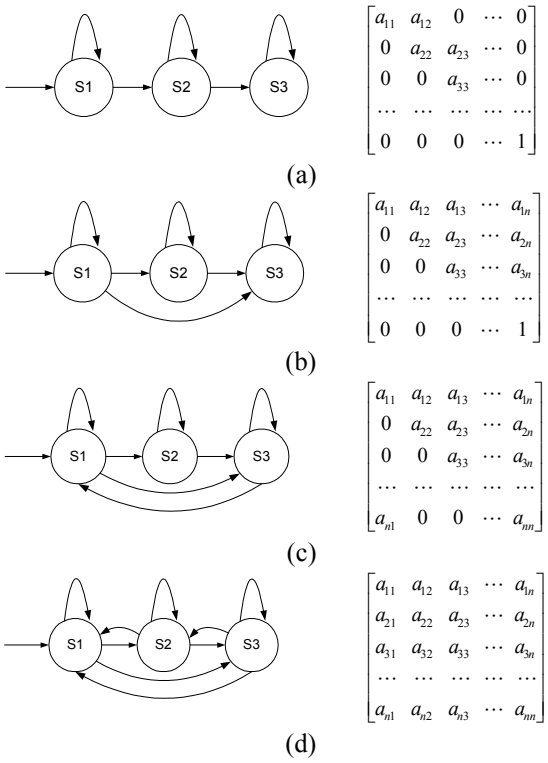


Figure 3-1: *An example for demonstrating different representations*

Given the example in Figure 3-1, the sequence in representation A will be {2,7,9,11,11,9}. The sequence in representation B will be {2,7,9,**11**,**11**,11,9}. The sequence in representation C will be {5,2,2,0,-2}. The sequence in representation D will be {++,+,+,0,-}.

HMM (Hidden Markov Model) is a very powerful tool to statistically model a process that varies in time. It can be seen as a doubly embedded stochastic process with a process that is not observable (hidden process) and can only be observed through another stochastic process (observable process) that produces the time set of observations. An HMM can be fully specified by (1) N, the number of states in the model; (2) M, the number of distinct observation symbols per state; (3) $A = \{a_{ij}\}$, the state transition probability distribution; (4) $B = \{b_j(k)\}$, the observation symbol probability distribution; and (5) $\Pi = \{\pi_i\}$, the initial state distribution. (Rabiner, 1989)

Because the number of hidden states and the structure may impact the classification performance, we use HMMs with different numbers of hidden states and different structures to do classification, and then compare their performances. Here are the different structures used and compared in our experiment (Figure 3-2).



$$\begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ 0 & a_{22} & a_{23} & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

Figure 3-2: *HMMs used in the experiment (a) A strict left-right model, each state can transfer to itself and the next one state. (b) A left-right model, each state can transfer to itself and any state to the right of it. (c) A variation of the left-right model, additional to (b), the last state can transfer to the first state. (d) A fully connected model.*

The Baum-Welch reestimation method was implemented to train a hidden Markov model for each country using the training set. To identify the country of an unknown melody, the Viterbi algorithm was used to decode the sequence and compute its log probabilities respectively using HMMs trained for different countries. We then assign the melody to the country with the highest log probability.

The results were obtained in this way: All the data were split randomly into training set (70%) and test set (30%). Each result was cross-validated with 17 trials using the 70%/30% splits.

The classification performances are shown in Figure 3-3 (i) – (iv). The first three figures show the generalization performances of 2-way classifications. The last figure shows the generalization performance of the 3-way classification. The X-axis in all the figures indicates different HMMs, whose corresponding structures are shown in Table 3-1.

The results show that, in general, the state number (2, 3, 4 or 6) didn't impact the classification performance significantly. The strict left-right HMMs (a) and the left-right HMMs (b) outperformed the other two HMMs (c/d). The representation C generally performs better than the representation A, B or D. Performances of the 6-state left-right HMM, for example, are

shown in Table 3-2. It achieved classification accuracies of 75%, 77% and 66% for 2-way classifications and 63% for the 3-way classification using representation C.

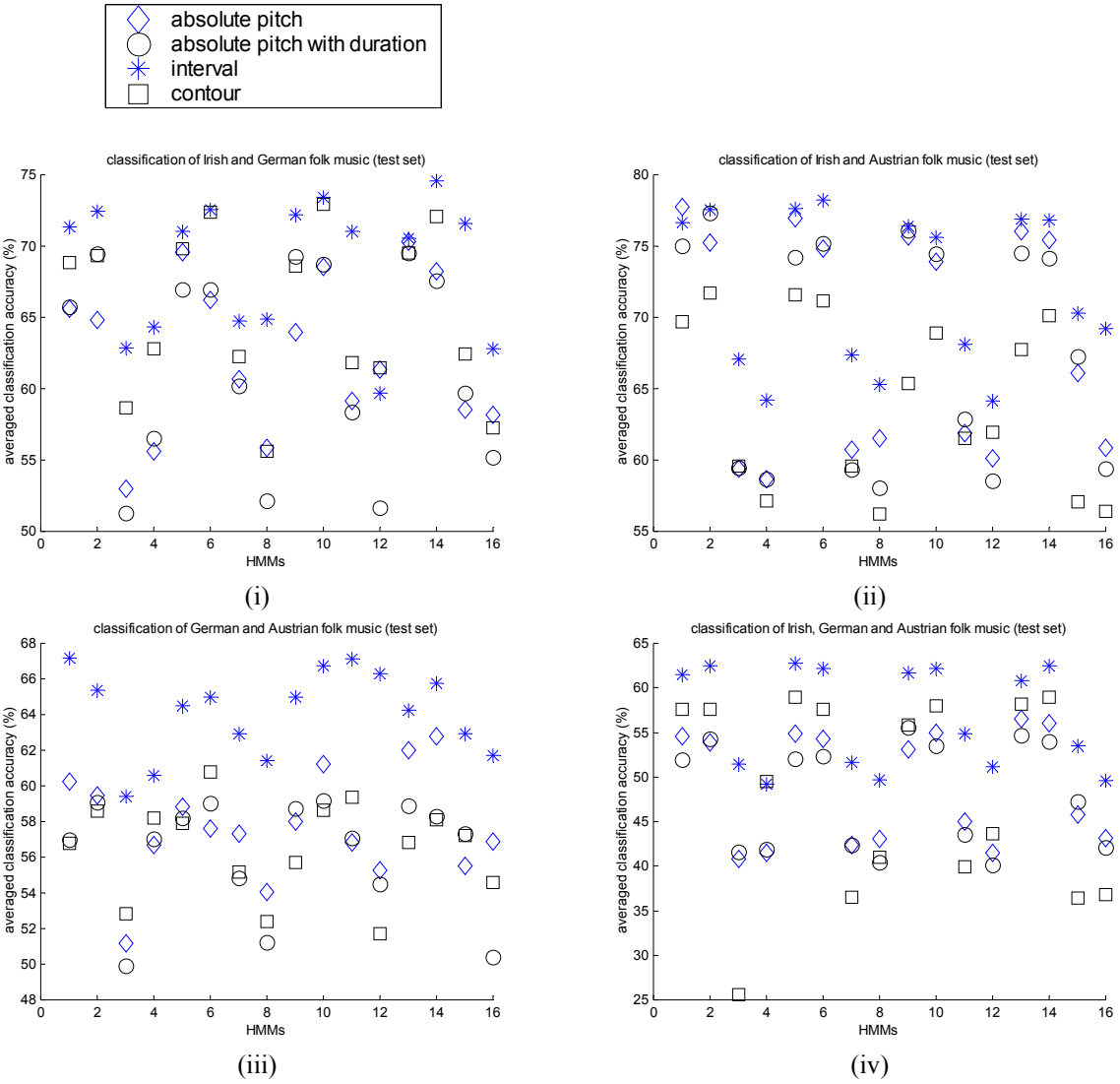


Figure 3-3: *Classification performances using different representations and HMMs. (i), (ii) and (iii) correspond to 2-way classifications. (iv) corresponds to the 3-way classification.*

Table 3-1: *16 HMMs with different structures and numbers of hidden states used in the experiment (see Figure 3-2 for the description of the structures).*

HMM	1	2	3	4	5	6	7	8
N	2	2	2	2	3	3	3	3
STRUC	a	b	c	d	a	b	c	d
HMM	9	10	11	12	13	14	15	16
N	4	4	4	4	6	6	6	6
STRUC	a	b	c	d	a	b	c	d

Table 3-2: *Classification performances of 6-state left-right HMM using different representations. The first three rows correspond to 2-way classifications. The last row corresponds to the 3-way classification. I: Irish music; G: German music; A: Austrian music.*

Classes	rep. A	rep. B	rep. C	rep. D
I-G	68%	68%	75%	72%
I-A	75%	74%	77%	70%
G-A	63%	58%	66%	58%
I-G-A	56%	54%	63%	59%

The performances of 2-way classifications are consistent with our intuition that it is harder to discriminate between German music and Austrian music than between German music and Irish music or between Austrian music and Irish music. Therefore, we expect that the result will improve if we use a more discriminable data set.

The results suggest to us a new way to think about musical style similarity. Nettl (1973) pointed out that it is very hard to state concretely just how much difference there is between one kind or style of music and another. As he suggested, one way of telling that a musical style is similar to another, which you already recognize, is that this styles also appeals to you. This has to do with the fact that folk music styles, like languages, exhibit greater or lesser degrees of relationship. Just as it is usually easier to learn a language that is closely related in structure and vocabulary to one's own, it is easier to understand and appreciate a folk music style similar to one that is already familiar. Here we presented a method to measure the musical style similarity quantitatively. The two styles that are less discriminable in classification are deemed more similar. The method can be based on the classification accuracy, as was done here, or the distance of their statistical models directly, for example, the distance of two HMMs (Juang, 1985).

The representation is very important for classification. The interval representation outperforming the absolute pitch representation is also consistent with humans' perception of melody. Although the absolute pitch method can represent the original work more objectively, the interval method is more compatible with humans' perception, since when people memorize, distinguish or sing a melody, they usually do it based only on the interval information.

The experiment shows that the contour representation was significantly worse than the interval representation for folk music classification. This indicates that although contour-based representation is fairly compact for identifying a melody, the quantization procedure may cause the features for style discrimination to be reduced.

The fact that the representation with duration did not outperform the representation without duration is not what would be expected. It seems to be inconsistent with humans' perception. We argue that it doesn't mean rhythmic information is useless for classification; instead, we suggest that the rhythmic encoding used (through repeated notes) in fact destroyed some characteristics of the melody, thus reducing the discrimination.

The experiment shows that melody is a significant musical feature. Although melody is not sufficient for music classification on its own, music in different styles (here, from different countries) does have significant statistical difference in their respective melodies.

CHAPTER 4 SYSTEM ARCHITECTURE

This chapter presents an overview of our system architecture, the functionality of each part and the communication between different parts.

4.1 Client-server Architecture

The QBH system adopts the client-server architecture and consists of four main parts (Figure 4-1):

- *Music database*: This includes the source data and the target data. The source data are the original music corpora, from which we extract melodies and generate the target data. The source data are in various symbolic representations (i.e., scores and MIDI files). The target data are the data that the end user can play back at the client side. The target data in the system are all in MIDI format.

- *Melody description objects*: These are binary persistent objects that capsulate the melody information based on our melody representation and play the role of an efficient indexing of the music corpus.

- *QBH Server*: This receives the query from QBH Client, matches it with the melodies in the melody description objects, and retrieves the target songs with the highest matching scores.

- *QBH Client*: This computes the pitch contour and beat information from the user's humming signals, constructs the query, sends the query to QBH Server via CGI, receives the results and plays them back.

Besides the above four parts, we need to develop some tools to build the whole system. They include:

- *Score to MIDI tool*
  It converts score files (in .krn, .alm, .esc or .niff formats; see Section 5.2.1 for detailed description about these formats) into MIDI files for playback.

- *Melody extraction tool*
  It extracts the melody information and constructs the melody description objects.

We have two client implementations: QBH Client and QBH AXClient. QBH Client is a standalone application, which means that the user needs to install it first. AXClient is an ActiveX control implementation, which can be embedded in a web page and the user will be able to run it by clicking the link in the web page without explicitly installing it.

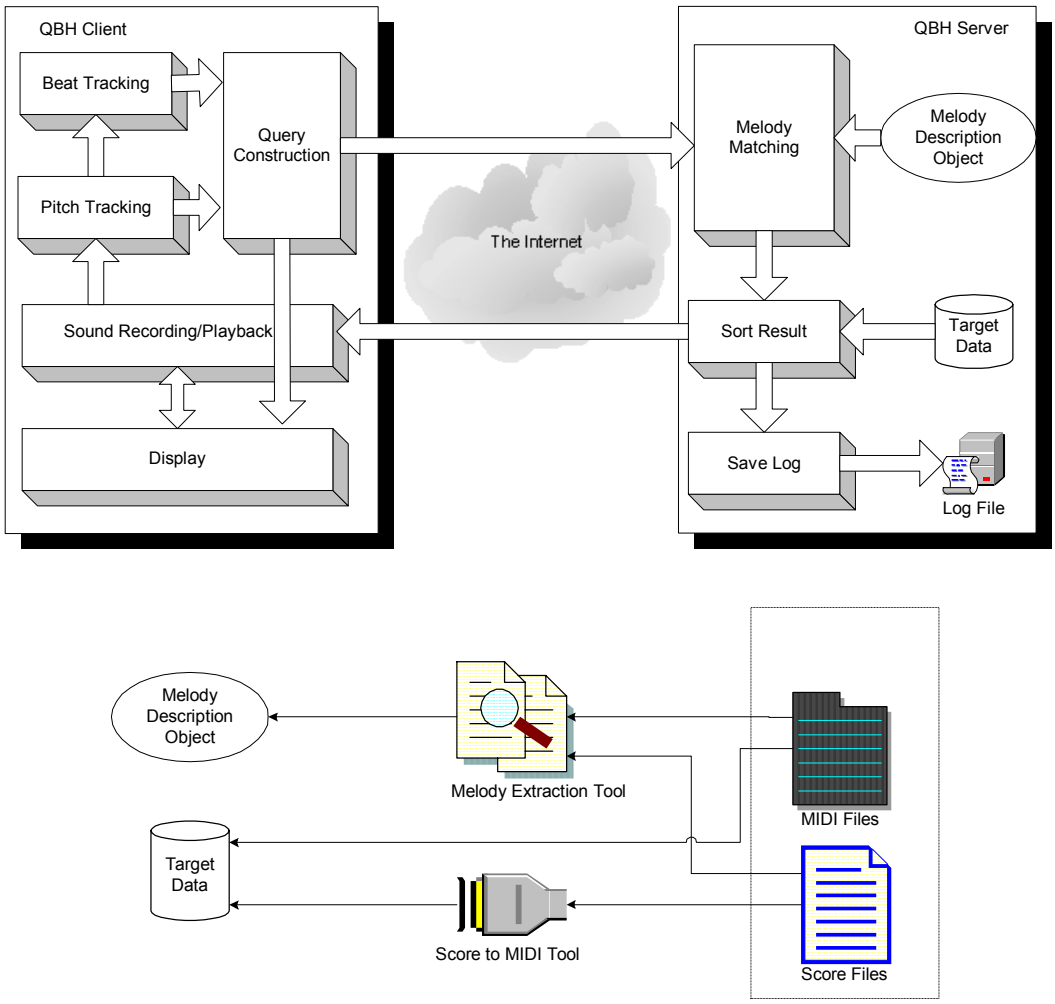


Figure 4-1: *QBH system architecture*

In our system, the query analysis part, which is also the most computationally complicated part, is located at the client side. This is different from some other systems, e.g., MELDEX. There are several advantages to dividing the system in this way:

- It can fully employ the computational ability of the client side machine, while reducing the server side burden.

- It can reduce the network bandwidth requirement. If we put the query analysis part at the server side, the humming signals must be transmitted to the server, which requires much more bandwidth compared with transmitting only the description data of the query, e.g., the pitch and rhythmic information in our system.

- It protects the user's privacy, because people may dislike transmitting their hummed queries to the server in that others might be able to hear them.

The client and server parts need to manipulate and transmit data in real-time, while all the tools can be used off-line to build the system.

4.2 Web-based Communication

All the communication between client and server uses HTTP protocol, e.g., URL, CGI (Figure 4-2). Its specification will be described in detail in Section 5.3.5. This made the whole system more flexible and open. With the interface unchanged, we can easily have a different implementation of either the client or the server without changing the other side. So here we provide not only an implementation of the system, but also a basis of a standard for such systems, which can support cross-platform operations.

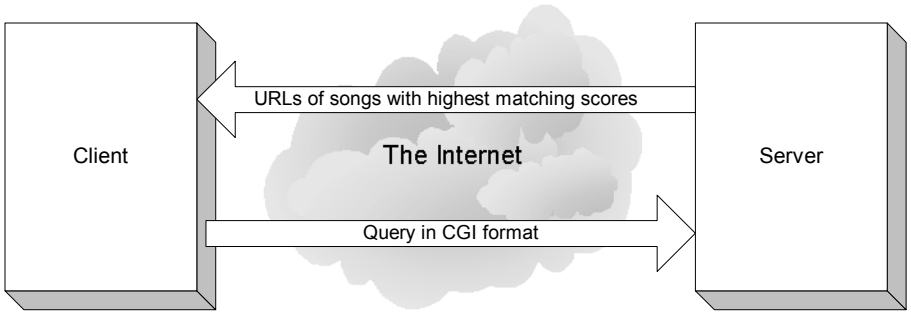


Figure 4-2: *Communication between the client and the server*

CHAPTER 5 APPROACHES AND ALGORITHMS

This chapter presents our solutions to solve the Melody Representation problem, the Melody Matching problem, the Melody Extraction problem and the Query Construction problem respectively. We propose a melody representation combining both pitch and rhythmic information. Two new melody matching methods corresponding to this representation are presented; for comparison, Dynamic Programming Method, which is most widely used in existing query-by-humming systems, is presented as well. Methods to extract melodies from various symbolic representations are presented; a handy tool was developed to build our melody database, which contains more than 8,000 songs currently. We also present our automatic transcription methods customized for the QBH system, including note segmentation, pitch tracking, interfaces for obtaining rhythmic information, and algorithmic beat determination. Our methods will be shown to be more effective in Chapter 7.

5.1 Melody Representation and Matching

5.1.1 Problems

What are the significant features people use to identify a melody or to distinguish between melodies? How can the melodies be represented sufficiently as well as concisely? Dowling (1986) proposed that a melody schema is not likely to be a literal mental copy of the melody. An exact copy would have to be translated – expanded, contracted, and shifted both in time and in pitch – to fit any actual instance of the melody that might be perceived. Therefore, it seems likely that a melody schema should represent more general higher-order information than specific pitches at specific temporal intervals. Previous query-by-humming systems mostly propose using pitch contours to represent melodies. They seldom use rhythm in their melody representation. However, rhythm is obviously important, because when identifying a melody, the listener perceives not only the pitch/interval information in the melody, but also how those notes correspond to particular moments in time. Rhythm is one dimension in which melodies in general cannot be transformed intact. A representation combining both pitch and rhythm information, which we call *TPB representation*, was proposed by Kim (2000) and adopted in the QBH system.

Additionally, how do people measure the similarity of melody? Or, how can we retrieve melody from our mind so easily even after it is transformed in some way? This problem is closely related to the representation problem. Levitin (1999) described melody as an "auditory object that maintains its identity under certain transformations … along the six dimensions of pitch, tempo, timbre, loudness, spatial location, and reverberant environment; sometimes with changes in rhythm; but rarely with changes in contour." Although rather broad, this definition highlights several important properties of melody. People are able to recognize melodies even when they are played on different instruments, at different volumes, and at different tempi (within a reasonable range). Based on the TPB representation, we also proposed new approximate string matching methods to do the melody matching task. The algorithms did not only take robustness into account but also efficiency.

The following sections present the melody representation and the melody matching methods we used in the QBH system.

5.1.2 Pitch contour

It is clear that some type of interval information is important for representing melody, since melodic matching is invariant to transposition. However, instead of representing each interval exactly (e.g., ascending minor sixth, descending perfect fourth, etc.), the literature suggests that a coarser melodic contour description is more important to listeners in determining melodic similarity (Handel, 1989). Experiments have shown that interval direction alone (i.e., the 3-level +/-/0 contour representation) is an important element of melody recognition (Dowling, 1978).

One possible reason for the importance of melodic contour is that this information is more easily processed and is at a higher (more general) level than interval information. But as one becomes more familiar with a melody and gains more musical experience, the specific intervals have greater perceptual significance (Edworthy, 1985) (Levitin, 1999).

There is, of course, anecdotal and experimental evidence that humans use more than just interval direction (a 3-level contour) in assessing melodic similarity. When recalling a melody from memory, most of us (not all!) are able to present information more precise than just interval direction. In an experiment by Lindsay (1996), subjects were asked to repeat (sing) a melody that was played for them. He found that while there was some correlation between sung interval accuracy and musical experience, even musically inexperienced subjects were able to negotiate different interval sizes fairly successfully. From a practical standpoint, a 3-level representation will generally require longer queries to arrive at a unique match.

Given the perceptual and practical considerations, we chose to explore finer (5- and 7-level) contour divisions for our representation.

5.1.3 Rhythmic information

It is first useful to define some terms. *Duration* is the psychological correlate of time. *Beat* refers to a perceived pulse marking off equal durational units. *Tempo* refers to the rate at which beats occur, and *meter* imposes an accent structure on beats (as in *"one,* two, three, *one,* two, three . . . "). *Meter* thus refers to the most basic level of rhythmic organization and does not generally involve durational contrasts. *Rhythm* refers to a temporally extended pattern of durational and accentual relationships.

In (Kerman, 2000), music is defined as “the art of sound in time”. Sound exists in time, and any sound we hear has its duration – the length of time we hear it in minutes, seconds, or microseconds. Though duration is not an actual property of sound, like frequency, amplitude, and other of sound’s attributes that are taught in physics courses, it is obviously of central importance for music. The broad term for the time aspect of music is rhythm. Rhythm is the driving force in the vast majority of music both popular and classical, music of all ages and all cultures.

The following simple example illustrates the importance of consistent rhythmic information in melodic description.

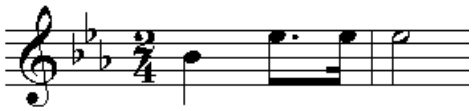


Figure 5-1: *First four notes of the Bridal Chorus from Lohengrin (Wagner), i.e., Here Comes the Bride.*

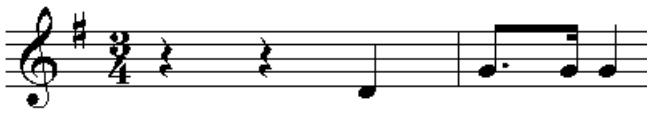


Figure 5-2: *First four notes of O Tannenbaum.*

It is apparent that these are two very distinct melodies, yet the sounding intervals and note durations are identical. The difference lies not only in the respective meters (time signatures) of the songs, but also in the position of the notes relative to the metric structure of each piece. The time signature of the first example is 2/4, and the strong beats occur on the first beat of a measure, which correspond to the first note of the piece. The second example has a time signature of 3/4, and the strong beat is on the first beat of a measure, corresponding to the second note. From this example, we clearly see the advantages of incorporating rhythmic information in a melodic representation.

5.1.4 Proposed melody representation

We use a triple <T, P, B> to represent each melody, which we will refer to as *TPB representation*.

T is the time signature of the song, which can change, but often does not. *P* is the pitch contour vector, and *B* is the absolute beat number vector. The range of values of *P* will vary depending on the number of contour levels used, but will follow the pattern of 0, +, -, ++, --, +++, etc. The first value of *B* is the location of the first note within its measure in beats (according to the time signature). Successive values of *B* are incremented according to the number of beats between successive notes. Values of *B* are quantized to the nearest whole beat. Clearly, the length of *B* will be one greater than the length of *P* because of the initial value.

In case a melody has multiple time signatures, we can break the melody into multiple melody segments, within each segment the time signature doesn't change. We can then apply the TPB representation to each melody segment.

Additionally, we use a vector *Q* to represent different contour resolutions and quantization boundaries. The length of *Q* indirectly reveals the number of levels of contour being used, and the components of *Q* indicate the absolute value of the quantization boundaries (in number of half-steps). For example, *Q*=[0 1] represents that we quantize interval changes into three levels, 0 for no change, + for an ascending interval (a boundary at one half-step or more), and - for a descending interval. This representation is equivalent to the popular +/-/0 or U/D/R (up/down/repeat) representation. *Q*=[0 1 3] represents a quantization of intervals into five levels, 0 for no change, + for an ascending half-step or whole-step (1 or 2 half-steps), ++ for ascending at least a minor third (3 or more half-steps), - for a descending half-step or whole-step, and -- for a descent of at least a minor third.

Thus, given a melody segment M and a resolution vector Q, we can get a unique contour representation:

$$getTPB(M,Q) =< T,P,B > \qquad \text{(Eq. 5-1)}$$

For example, the TPB representation for the example in Figure 5-1 using Q=[0 1 3] is <[2 4], [* 2 0 0], [1 2 2 3]>. The TPB representation for the example in Figure 5-2 using Q=[0 1 3] is <[3 4], [* 2 0 0], [3 4 4 5]>.

In our system, both the symbolic music corpora and hummed queries are converted into the TPB format, so that the similarity between the song and the query can be measured by matching the strings in TPB representations. Please note that the beat numbers in the query are not necessarily to be absolute beat numbers as in the melody segments, but they must be relatively correct (See Section 5.3.5).

5.1.5 Melody matching algorithms

All the following matching algorithms have been implemented in our QBH system, whose performances will be compared in Chapter 7.

Dynamic Programming Matching Algorithm (DPM)

Dynamic programming has been used in most existing query-by-humming systems. The algorithm only uses the pitch information for matching the query with melodies.

Dynamic programming is able to determine the best match of the two strings on a local basis. For example, using the two melody fragments (typically one is a query and the other is a melody segment), we would fill a matrix according to the following formula, where c represents the matrix; q and p represent the query melody string and the piece to match against respectively; index i ranges from 0 to query length and index j ranges from 0 to piece length:

$$c[i,j] = \min\begin{cases} c[i-1,j]+d & (i \geq 1) \\ c[i,j-1]+d & (j \geq 1) \\ c[i-1,j-1]+e & (q[i]=p[j] \; and \; i,j \geq 1) \\ c[i-1,j-1]+m & (q[i] \neq p[j]) \\ 0 & o.w. \end{cases} \qquad \text{(Eq. 5-2)}$$

where d is the cost of an insert or delete, e is the cost of an exact match, and m is the cost of a mismatch. We used $d = 1$, $e = 0$ and $m = 1$.

		Melody String							
		*	0	0	-2	1	0	2	-1
	*	0	**0**	0	0	0	0	0	0
	0	1	0	**0**	1	1	0	1	1
Query String	**-2**	2	1	1	**0**	1	1	1	2
	1	3	2	2	1	**0**	1	2	2
	2	4	3	3	2	**1**	1	1	2
	0	5	4	3	3	2	**1**	2	2

Figure 5-3: *Dynamic Programming Matching of the melody segment [* 0 0 -2 1 0 2 –1] and the query [* 0 -2 1 2 0]. The optimal matching is marked by the arrows, which represent how the value pointed to was calculated and can be followed to determine the parts that have been aligned.*

Figure 5-3 gives an example of using DPM to match the melody segment [* 0 0 -2 1 0 2 –1] and the query [* 0 -2 1 2 0]. The optimal matching has a minimum cost of 1, which occurs in one location. Tracing the path that led to the local minima results in the following matches of the strings:

```
[   *   0   0   -2   1        0   2   -1   ]
    [   *   0   -2   1   2    0   ]
```

DPM algorithm was evaluated to be the best similarity measure compared with all the other previous matching methods proposed for the query-by-humming systems including n-gram counting, the longest common subsequence and the Ukkonen measure. (Uitdenbogerd, 1999)

In our QBH system, to get the similarity score instead of the above cost measure, we simply invert the sign of cost and normalize it to be between 0 and 1.

$$score = (-cost + QueryLength) / QueryLength \qquad \text{(Eq. 5-3)}$$

TPB Matching Algorithm I (TPBM-I)

This is an algorithm we proposed to compute the similarity score *s* of a melody segment $m =< T_m, P_m, B_m >$ and a query $q =< T_q, P_q, B_q >$ to evaluate how well they match. The main difference between it and the DPM algorithm is that it considers both the rhythmic information and the pitch information. A higher score indicates a better match. Both the melody and the query must be converted into the TPB representation using the same quantization vector *Q*.

TPBM-I algorithm:

(1) If the numerators of T_m and T_q are not equal, then return 0.

(2) Initialize the measure number, $n = 1$.

(3) Align P_q with P_m from the n^{th} measure of *m*.

(4) Calculate the beat similarity scores for each beat (see below).
(5) Average the beat similarity scores over the total number of beats in the query, resulting in the overall similarity score starting at measure *n*: s_n.
(6) If *n* is not at the end of *m*, then $n = n + 1$ and repeat step 3.
(7) Return $s = \max\{s_n\}$, the best overall similarity score starting at a particular measure.

Algorithm for calculating the beat similarity score:
(1) Get the subsets of P_q and P_m that fall within the current beat as S_q and S_m.
(2) i=1; j=1; s=0;
(3) while $i \le |S_q|$ and $j \le |S_m|$
    a. if $S_q[i] = S_m[j]$ then
        s=s+1; i=i+1; j=j+1;
    b. else
        k=j;
        if $S_q[i] \ne 0$ then j=j+1;
        if $S_m[k] \ne 0$ then i=i+1;
(4) return $s = s / |S_q|$.

Thus, the beat similarity score is between 0 and 1.

TPB Matching Algorithm II (TPBM-II)

TPBM-II is a generalized version of TPBM-I. Basically it ignores the time signature and meter information. Only pitch contour and relative beat numbers are used in this algorithm.

TPBM-II algorithm:
(1) Initialize the beat number, n = 1.
(2) Align P_q with P_m from the n^{th} beat of *m*.
(3) Calculate the beat similarity score as above.
(4) Average the beat similarity scores over the total number of beats in the query, resulting in the overall similarity score starting at beat *n*: s_n.
(5) If *n* is not at the end of *m*, then $n = n + 1$ and repeat step 2.
(6) Return $s = \max\{s_n\}$, the best overall similarity score starting at a particular beat.

5.1.6 Effectiveness of the representation and matching algorithms

We first need to propose some measures for evaluating the effectiveness of a melody representation and a matching algorithm for melody retrieval purpose.

Two measures have been generally used in evaluating retrieval effectiveness in document retrieval systems.

- *Precision* is defined as the proportion of retrieved documents which are relevant. Precision can be taken as the ratio of the number of documents that are judged relevant for a particular query over the total number of documents retrieved. For instance, if, for a particular search query, the system retrieves two documents and the user finds one of them relevant, then the precision ratio for this search would be 50%.

- *Recall* is defined as the proportion of relevant documents retrieved. Recall is considerably more difficult to calculate than precision because it requires finding relevant documents that will not be retrieved during users' initial searches. Recall can be taken as the ratio of the number of relevant documents retrieved over the total number of relevant documents in the collection. Take the above example. The user judged one of the two retrieved documents to be relevant. Suppose that later three more relevant documents that the original search query failed to retrieve were found in the collection. The system retrieved only one out of the four relevant documents from the database. The recall ratio would then be equal to 25% for this particular search.

Figure 5-4 shows the concepts of precision and recall. B={relevant documents}, C={retrieved documents} and $A = B \cap C$. Thus, from the above definition,

$$Precision = \frac{A}{C} \qquad \text{(Eq. 5-4)}$$

$$Recall = \frac{A}{B} \qquad \text{(Eq. 5-5)}$$

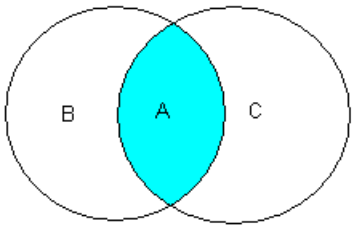


Figure 5-4: *Concepts of Precision and Recall.*

Theoretically, we would like to minimize C but maximize A to achieve maximum precision and recall. In practice, however, increasing one measure may cause the other one to decrease. So there is a trade-off between the two measures. The measures are affected by the matching algorithm (similarity between the query and the document) and how many documents are retrieved (similarity threshold).

In our particular case, we can still use the concepts while some special things need to be considered. Since for our melody retrieval system, our main goal is to retrieve the song that is exactly the one the user wants to look for and not the songs that sound similar in some sense to the query, there should be only one song contained in B except that there might be several copies in the database corresponding to the same song. If the right song is contained in the retrieved songs, then *recall=1*. Ideally, the right song and only the song should have the highest score, so that if we retrieve only one song with highest matching score, we will also get *precision=1*.

However, such good performance is hard to achieve in practice. On one hand, we need the representation and matching method to be strict enough, so that when the query in perfect, it can guarantee that only the right song will get the matching score 1 and thus *precision=1*. To this goal, exact matching is a good choice. On the other hand, we need the representation and matching method to be loose enough, so that even if there are some errors in the query – it is typically true – it still can get high matching score with the right song, so that it might be retrieved and thus *recall=1*. To this goal, approximate matching is good.

Therefore, to solve the tradeoff, we need a good representation and matching algorithm that can best characterize melodies and melodic similarity.

In (Kim, 2000), we did a complete evaluation of the TPB melody representation and TPBM-I matching algorithm based on a small data set. Only perfect queries randomly generated from the data set were used in the experiment. The results we got include:

Results I: Importance of rhythmic information.

In spite of anecdotal evidence (such as the examples from Section 5.1.3), we wanted to explicitly verify the usefulness of rhythmic information in comparing melodic similarity. To test this, we used the simplest contour (3-levels, Q=[0 1]) representations with and without the rhythmic information. Our results clearly indicate that rhythmic information allows for much shorter (and thus more effective) queries (Figure 5-5). A fewer number of matches indicates better performance.

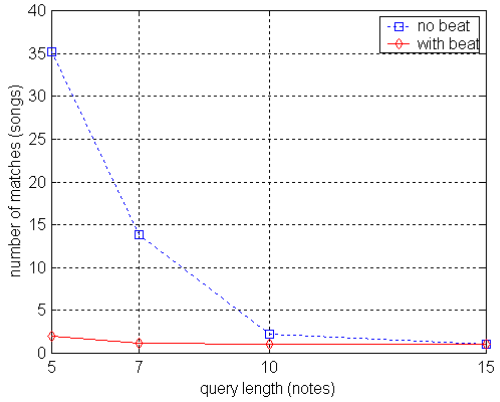
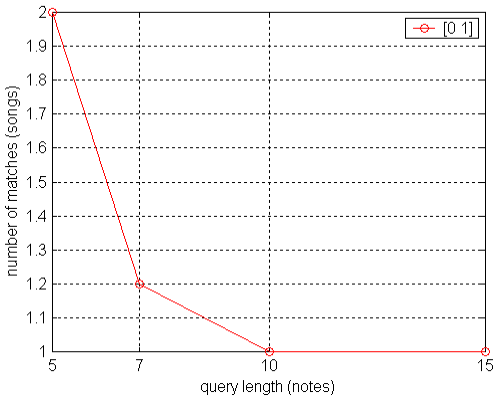
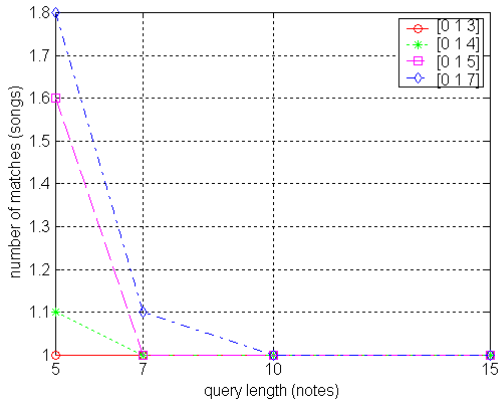


Figure 5-5: *Importance of rhythmic information from (Kim, 2000).*

Results II: Comparison of different quantization boundaries.

We examined 3-, 5-, and 7-level contour representations. For the 5- and 7-level contours, we also examined a variety of quantization boundaries (different quantization vectors Q). The results, in terms of average number of matches vs. query length (number of notes) are presented below in Figures 5-6. It is clear that the performance of 5-level contours are generally better than the 3-level contour, and 7-levels is better than that. For quantization vectors, we limited our search to $Q_k = [0\,1\,x\,...]$ cases only. Other values would have caused repeated notes (no interval change) to be grouped in the same quantization level as some amount of interval change, which does not make sense perceptually.





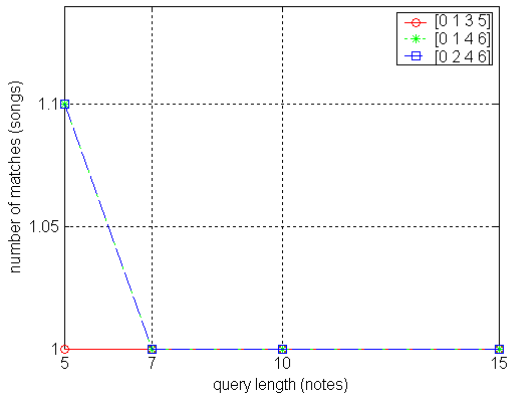


Figure 5-6: *Comparison of different quantization boundaries from (Kim, 2000).*

It is an obvious result that greater numbers of levels in general result in more efficient searches. Clearly, more levels means more information, meaning less notes are needed to converge to a unique solution. What is more illuminating is that the best 5-level contour was able to equal the performance of the 7-level contour based on the rather small data set. This suggests that a 5-level contour may be an optimal tradeoff between efficiency and robustness to query variation (more levels will cause more variations in queries).

Given our results, it is especially revealing to look at the histogram of interval occurrences in our data set (Figure 5-7). From this histogram, it is clear why certain quantization levels perform better than others. An optimal quantizer would divide the histogram into sections of equal area. Thus, for a 5-level contour we would like each level to contain 20% of the data. This is approximately true for the Q=[0 1 3] case, which has the best performance. No interval change (0) occurs about 23% of the time. Ascending half-steps and whole-steps (+1 and +2) are about 21% of the intervals, whereas descending half- and whole-steps (-1 and -2) represent approximately 23%. Other choices for quantization boundaries clearly have less-optimal probability distributions, which is why they do not perform as well.

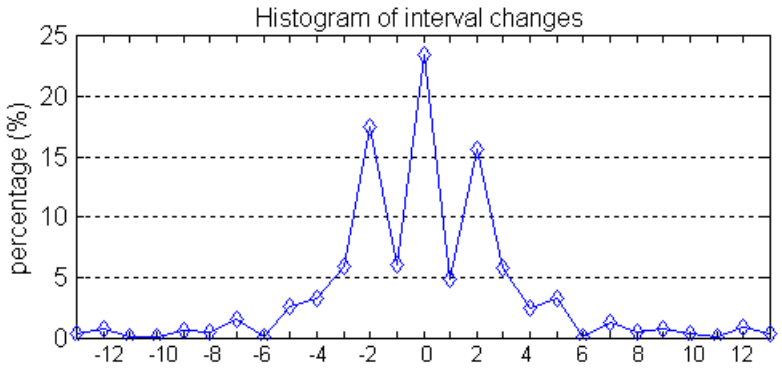


Figure 5-7: *The interval histogram from (Kim, 2000).*

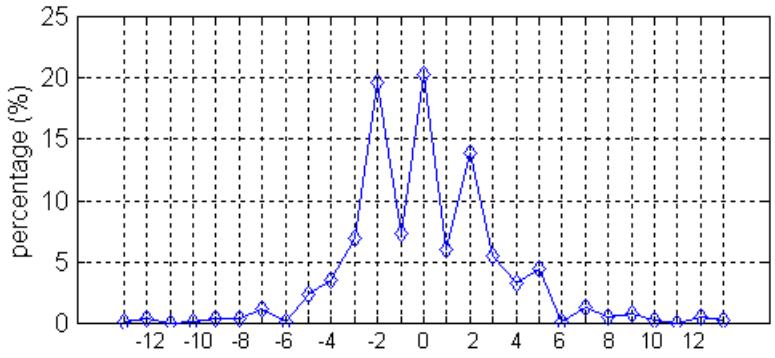


Figure 5-8: *The interval histogram based on the QBH corpora (same as Figure 5-13-c).*

Although the above experimental results were based on a very small data set, the interval distribution of our current melody database (Figure 5-8) is amazingly similar to the smaller data set we used in (Kim, 2000). And the statistics of folk music from different countries did not reveal a significant cultural bias in the distribution of intervals (See Section 5.2.5). Compared with results in Section 3.5, we may conclude that first-order pitch information (interval distribution) usually cannot capture the statistical difference between music from different cultures, while higher-order pitch information (e.g., modeled by HMM) can significantly do better.

The demonstration of effectiveness of our representation and matching method in (Kim, 2000) is limited though, because only perfect and randomly generated queries were used. Experimental results based on much larger data set and real queries will be presented in Chapter 7.

5.2 Melody Extraction

5.2.1 Problems

What is melody and how do humans perceive melody from a complex piece of music? Is it possible to extract melodies from existing digital music corpora by machine and use them to build a melody database for the retrieval purpose?

There could be two kinds of sources from which we extract melodies and build our melody database: symbolic musical data (e.g., scores and MIDI) or acoustic musical data (i.e., waveform representation). In this system, we only use symbolic corpora, because extracting melody from waveform representation involves the automatic transcription problem, which has not yet been solved well.

All the source data are in various symbolic formats including NIFF (Notation Interchange File Format), ALMA (Alphameric Language for Music Analysis), EsAC (Essen Associative Code), **kern (a subset of Humdrum format) and MIDI (Musical Instrument Digital Interface). They are widely known in computer music community. (See Table 5-1 for detailed information.)

Table 5-1: *Symbolic formats used in the QBH system*

FORMATS	ASCII/BINARY	DEVELOPED BY
NIFF	Binary	several manufacturers
ALMA	Ascii	Murray Gould and George Logemann
**kern	Ascii	David Huron
EsAC	Ascii	Helmut Schaffrath
MIDI	Binary	MIDI Manufacturers Association

Extracting melody information from symbolic representations with a separate monophonic melody track is relatively easy. I have developed a tool to manipulate this kind of corpora. However, in many cases, a melody is contained in a polyphonic track. Extracting melody information from such a polyphonic track is quite hard. It involves the melody perception issue. When humans hear music, which is typically polyphonic, it is easy to identify where the melody line is and to memorize it instead of the every detail of the music. The role that Gestalt rules play in melody perception has been discussed in Section 3.3. We can view this problem as a symbolic version of auditory scene analysis: how should we group the notes together to construct the melody lines or more generally a perceptual object, once we can successfully obtain the lower-level note events?

The following sections will describe our methods dealing with different types of symbolic corpora.

5.2.2 Manipulating monophonic scores

All the score files we used to extract melody and build the melody database are monophonic, corresponding to the melodies. They are from the same two sources used in Section 3.5:

- Helmut Schaffrath's Essen Folksong Collection which contains over 7,000 European folk melodies encoded between 1982 and 1994;

- Donncha Ó Maidín's Irish Dance Music Collection.

These source data are in symbolic formats including NIFF, ALMA, EsAC and **kern.

We developed a tool based on the CPN View implemented by University of Limerick (Maidín, 1998) to extract the pitch and duration information from the above various score formats. CPN View (Common Practice Notation View) is a library for building and manipulating representations of notated scores in C++. It is designed for use in building end-user software such as notation systems, music browsing systems and computer aided composition systems. It is also suitable for building algorithms for music analysis. Currently, CPN View supports symbolic formats including ALMA, Kern, EsAC and NIFF.

The information we want to extract, as described in Section 5.1, includes time signature, pitch contour and absolute beat numbers. The algorithm is described in Figure 5-9.

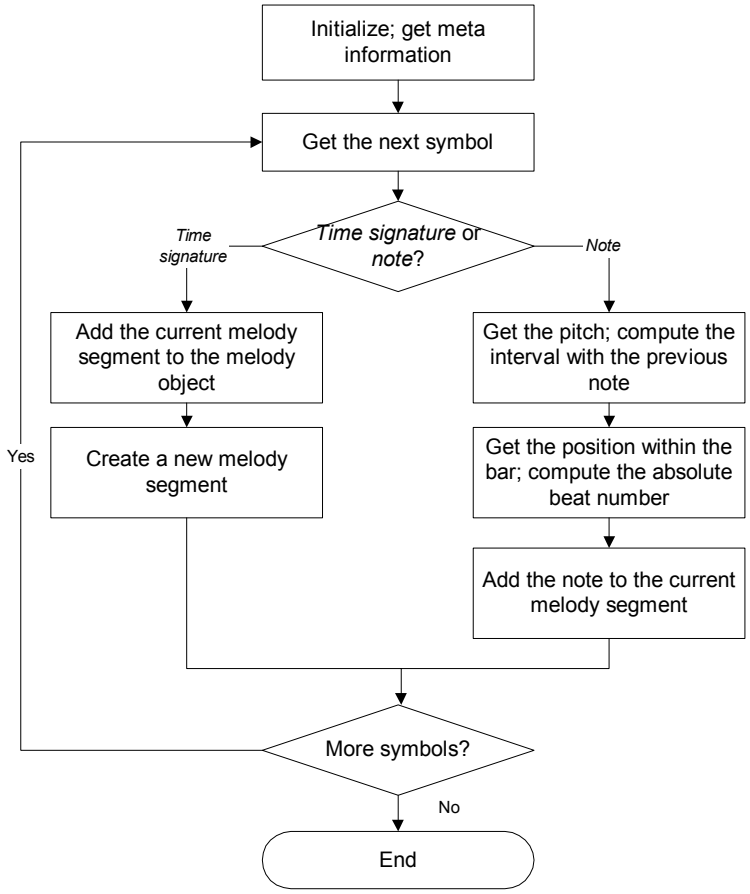


Figure 5-9: *Algorithm for extracting melody from monophonic scores.*

5.2.3 Manipulating MIDI files with monophonic melody track

Most of the MIDI files we downloaded from the Internet have a separate monophonic melody track and a special track name e.g., "melody", "vocal", "lead" etc. to identify this track. So dealing with this kind of corpora is similar to working with monophonic scores. We need to determine in which track the melody line is located according to the track name, and then extract the time signature, pitch contour and beat number from it. The only differences are (1) MIDI representation may not notate the music as accurately as the scores. Thus, to determine the beat number of each note, we need to quantize the beat. (2) *Time signature* events are usually in a different track instead of the melody track. The algorithm is described in Figure 5-10.

All the MIDI files we deal with are in MIDI format 1. The absolute beat number is quantized in the following way:

$$TicksPerBeat = PPQN * 4 / dd \qquad \text{(Eq. 5-6)}$$

where dd is the denominator of the time signature.

$$BeatPosition = TickNumber / TicksPerBeat - \lfloor TickNumber / TicksPerBeat \rfloor \qquad \text{(Eq. 5-7)}$$

$$BeatNumber = \begin{cases} \lfloor TickNumber / TicksPerBeat \rfloor + 1; & if\ BeatPosition \le QuantizeRat \\ \lfloor TickNumber / TicksPerBeat \rfloor + 2; & if\ BeatPosition > QuantizeRat \end{cases} \quad \text{(Eq. 5-8)}$$

where QuantizeRat is a constant, which means the note appearing after the corresponding position within the beat will be quantized into the next beat. It is set to be 15/16 in our system.

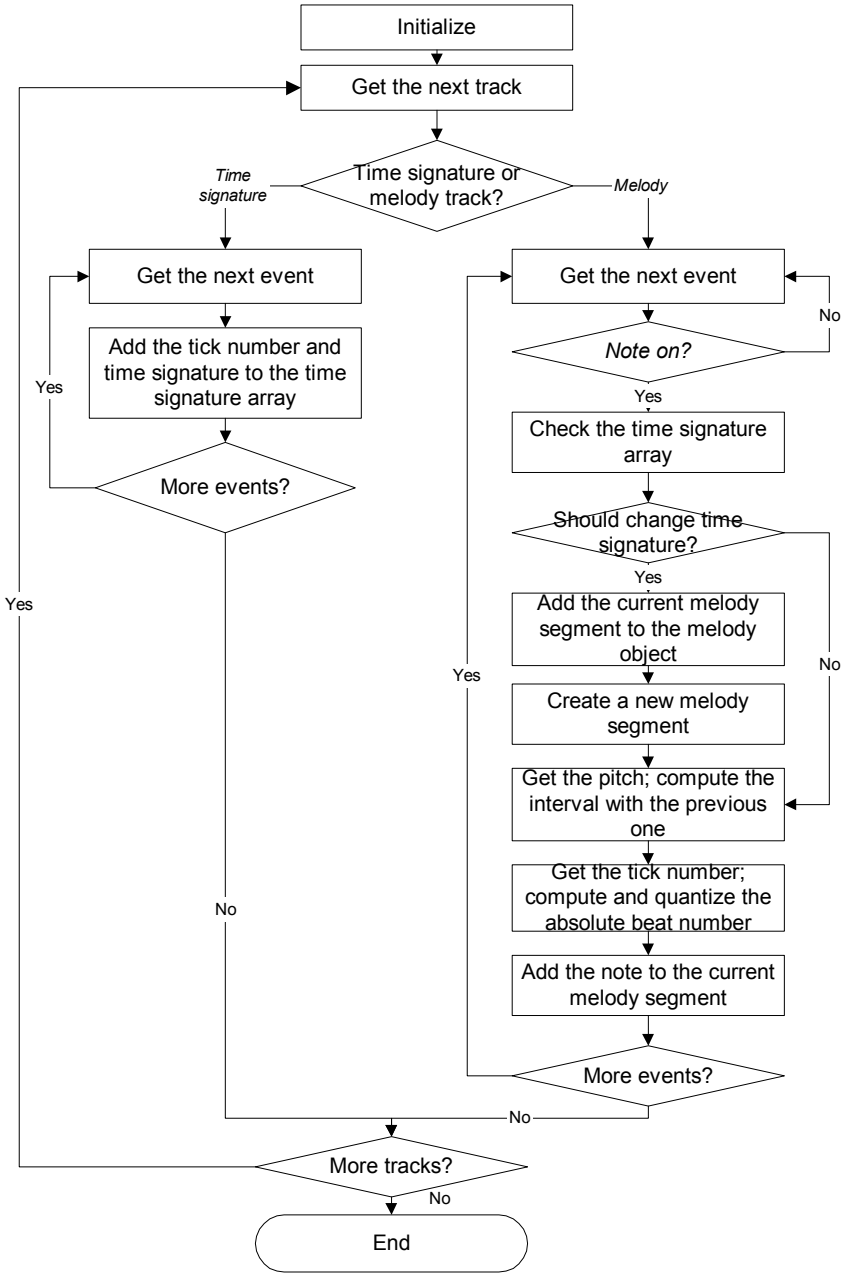


Figure 5-10: *Algorithm for extracting melody from MIDI files with monophonic melody track.*

5.2.4 Manipulating MIDI files with polyphonic melody track

In order to extract melodies reliably from polyphonic music files it is necessary to determine what a person would perceive as the melody when he listens to music. The aspects about melody perception have been presented in Section 3.3. The rules are rather complicated, while in practice we have some straightforward methods.

Several papers have explored the way that groups of notes are perceived. Two simple rules are generally used:

- A melody is heard as the figure if it is higher than the accompanying parts.
- Since melody is only considered as monophonic pitch sequence here, notes within a melody should have no time overlap and usually be close to each other in time.

Although these rules are far from robust for all kinds of music, for example, if the upper notes are constant and the lower notes form a more interesting pattern, then the lower notes will be heard as the figure, it is effective for some music corpus. Uitdenbogerd (1998) and MidiLib project proposed algorithms based on this simple rule, sometimes called the *skyline algorithms*, which extract the highest pitch line as the melody line. The basic algorithm is: For each onset time (a point on the time axis with at least one note onset) choose the note with the most distinct feature (e.g., highest pitch). If necessary, this note's duration is shortened such that it does not exceed the time between the actual and the next onset time. In principle, all variants of the pitch based skyline algorithm extract sequences from the higher pitched regions of a piece of music. Frequently, the computed extract jumps between a primary melody and accompanying notes.

Another algorithm called *melody-lines algorithm* is also proposed by MidiLib project, which aims at partitioning the polyphonic piece into a set of monophonic melodies. For this reason, notes close to each other in time are grouped together. For each onset time, each note is assigned with this onset time an existing melody line, which is "inaudible" at this time stamp. In this, *inaudible at a time stamp* means that the last note of that melody line must have ended before that time stamp. A basic criterion for the assignment of notes to melody lines is the minimization of pitch differences of subsequent notes. Notes that cannot be assigned to existing melody lines form new melody lines. In extending existing melodies, we only append one note to one melody at one time stamp. Furthermore one has to take care not to introduce long intervals containing pauses. The problems of this algorithm are: (1) it doesn't work with homophonic music, in which separate melody lines don't exist at all; (2) local errors (with regard to certain onset times) sometimes result in very chaotic melody extractions.

The algorithm we developed in our system to extract melody from MIDI files with separate polyphonic melody track is a variation of the skyline algorithm. The revised skyline algorithm adds one parameter called *time overlap parameter (TOP)*, which can make the recognized melody much cleaner than the classical skyline algorithm based on the test set provided by Uitdenbogerd. The algorithm is described in Figure 5-11.

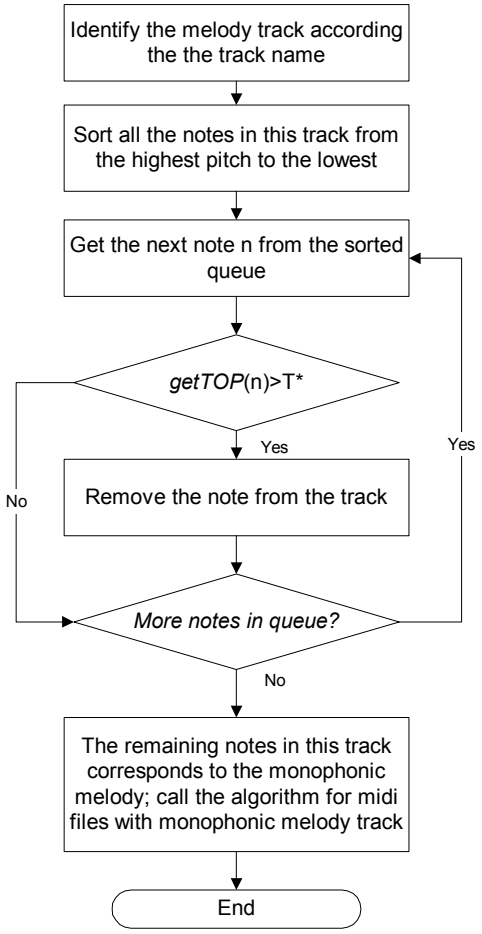
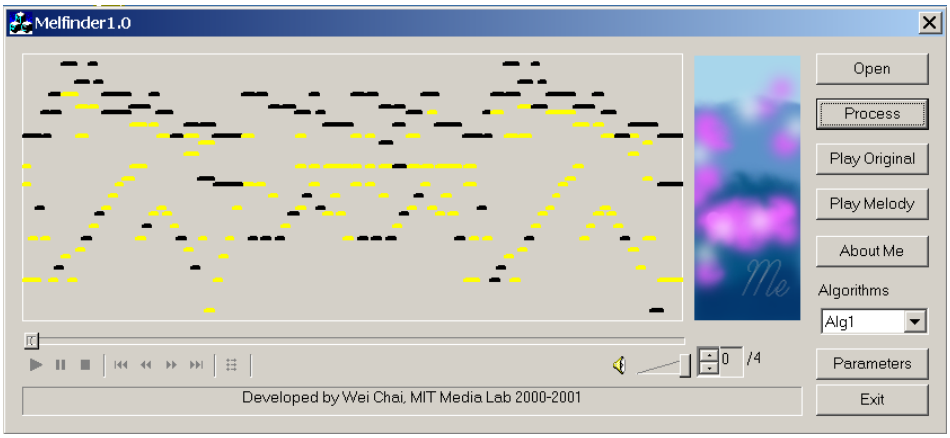


Figure 5-11: *The revised skyline algorithm for extracting melody from MIDI files with polyphonic melody track.*

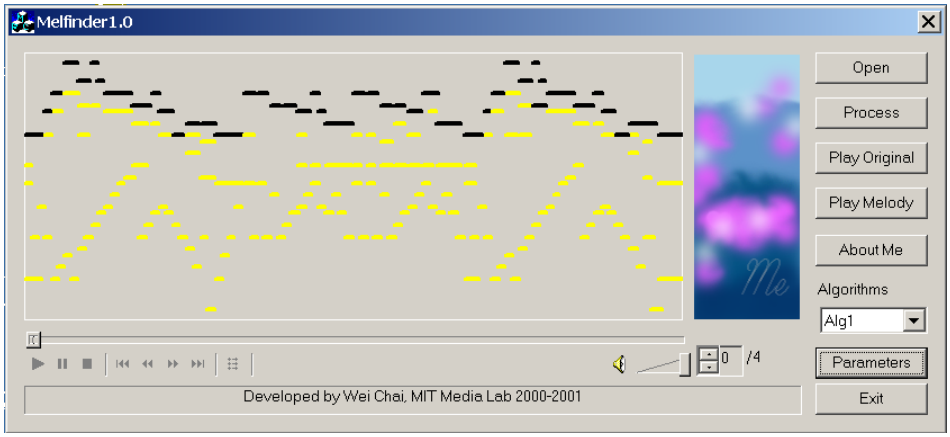
$$getTOP(n) = \frac{the\ time\ overlap\ between\ note\ n\ and\ current\ melody\ line}{duration\ of\ \ note\ n} \quad \text{(Eq. 5-9)}$$

T^* is a constant, which means that if the note overlaps too much with the current melody line, it will not belong to the melody. It is configurable and was set to be 0.5 in our system.

One example of the classical skyline algorithm and the revised skyline algorithm is shown in Figure 5-12. It is one song from the test set provided by Uitdenbogerd. The figure shows that without considering the time overlap, the classical skyline algorithm extracted a very chaotic melody with many accompanying notes.



(a)



(b)

Figure 5-12: *One example of the classical skyline algorithm (a) and the revised skyline algorithm (b).*

In our system, for the MIDI corpora that might contain polyphonic melody tracks, we simply use the polyphonic tool to process them first and then use the monophonic tool (as shown in Section 5.2.3) to extract melody information.

5.2.5 Symbolic music corpora

This section gives some statistics of the symbolic music corpora we used in the QBH system. The size of the melody database is shown in Table 5-2.

Table 5-2: *Music corpora used in current QBH system.*

Formats	Suffixes	Genres	Quantity	Sources
**kern	.krn	World folk	5109	Essen; Limerick
EsAC	.esc	World folk	1816	Essen; Limerick
ALMA	.alm	World folk	714	Essen; Limerick
NIFF	.nif	World folk	34	Essen; Limerick
MIDI	.mid	Classical, pop/rock	399	Internet
TOTAL:		8072 songs	>536,181 notes contained in melodies	

The histogram of the interval distribution is shown in Figure 5-13.

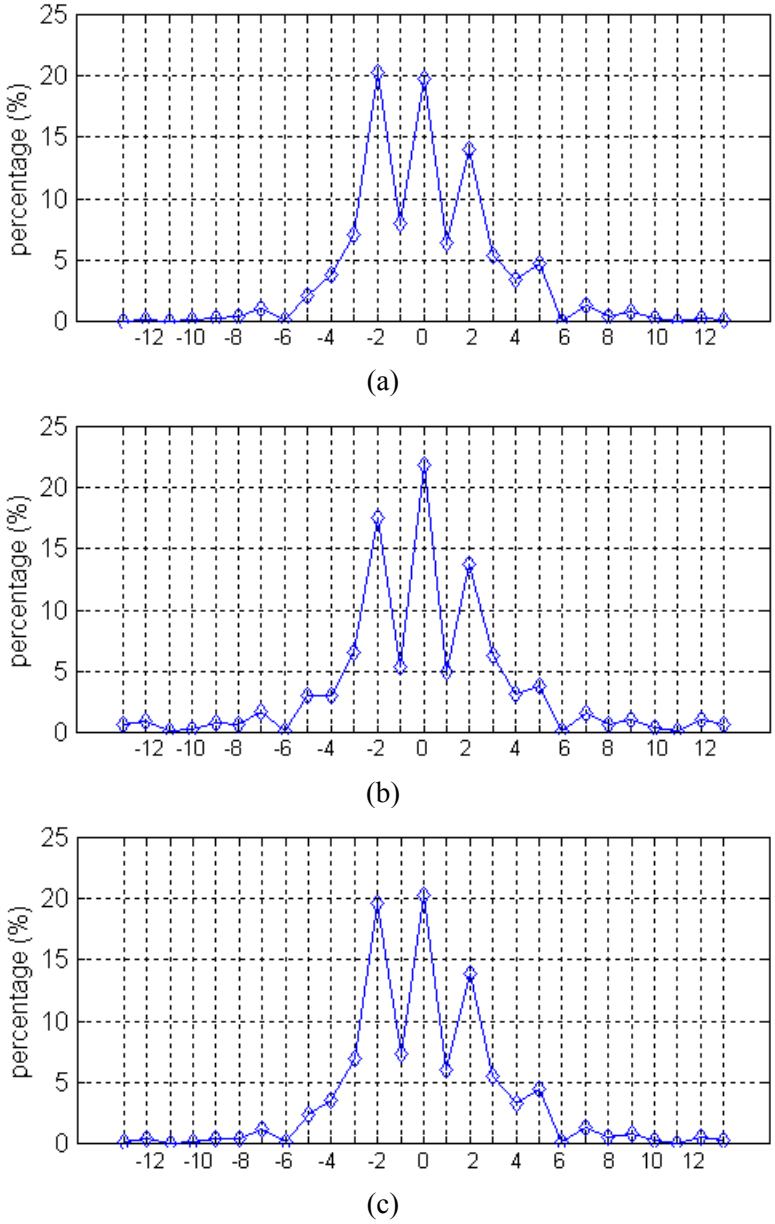


Figure 5-13: *Interval histograms. (a) The histogram of the world folk music scores (b) The histogram of the MIDI files (c) The histogram of the whole music corpora.*

5.3 Query Construction

Query construction here means to obtain the pitch contour and rhythmic information from the hummed query.

The algorithm to obtain pitch contour must be real-time and quite robust. Human voice is the stimuli to be tracked. Note segmentation is very important for the effectiveness of the system. A revised amplitude-based note segmentation algorithm and the autocorrelation pitch tracking algorithm have been implemented in the QBH system.

To obtain rhythm from user's query is more complicated. We developed three different ways to obtain the rhythmic information from hummed queries. The effectiveness, usability and friendliness will be compared later in Chapter 7.

All the examples mentioned in the following sections were recorded using 8000Hz sampling rate and obtained in the experiment that will be described in detail in Chapter 7. The parameters were set experimentally based on this sampling rate.

5.3.1 Note segmentation

The purpose of note segmentation is to identify each note's onset and offset boundaries within the acoustic signal. In order to allow segmentation based on the signal's amplitude, we ask the user to sing using the syllables like *da*, thus separating notes by the short drop in amplitude caused by the stop consonant. The following presents our note segmentation algorithm. Its performance will be reported in Chapter 7.

Amplitude-based Note Segmentation Algorithm (ANS)

The representation used in this segmentation method is similar to the RMS power of the signal. The amplitude is calculated by computing the spectrogram (e.g., 128 samples window length for 8kHz sampling rate; 64 samples overlap; hanning window), summing the absolute values of amplitude within human voice's frequency range (approximately below 1000Hz in our system) in each window. Thus, we get an amplitude function $A[w]$ where w is the window number (Figure 5-14, 5-15, 5-16, 5-17).

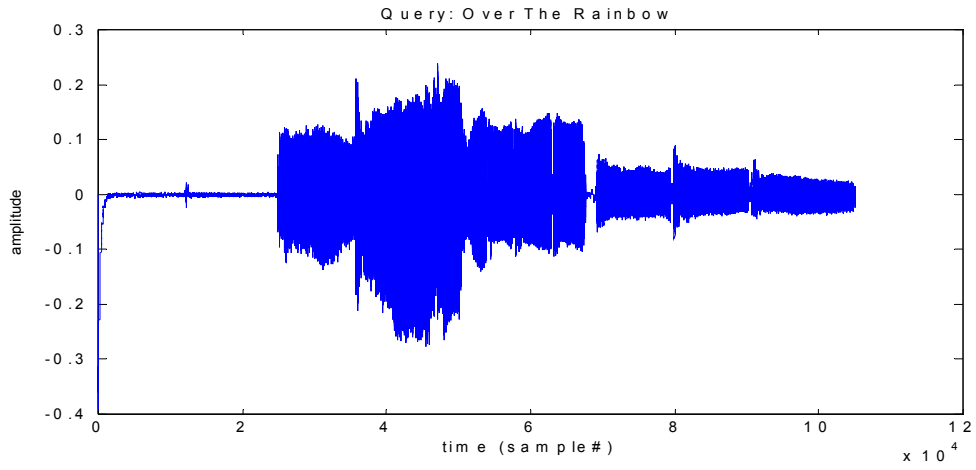


Figure 5-14: *The waveform of one query: Over The Rainbow.*

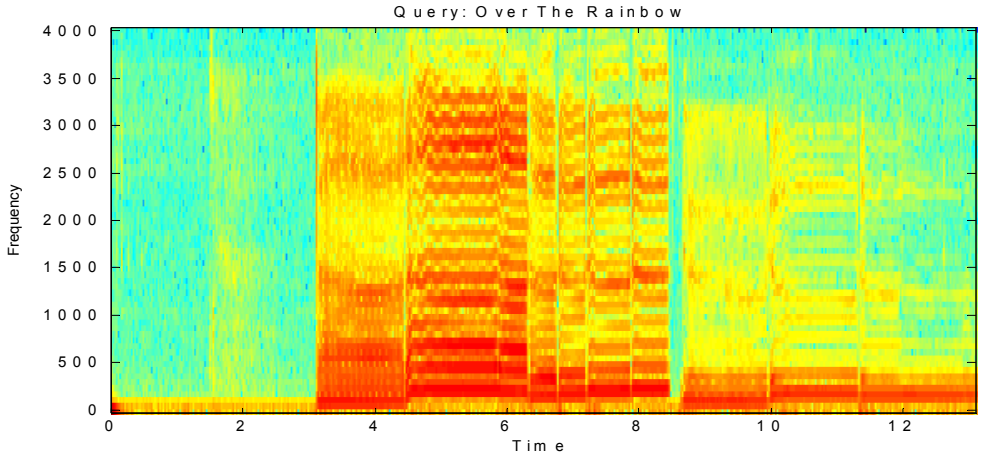


Figure 5-15: *The spectrogram of the query in Figure 5-14.*

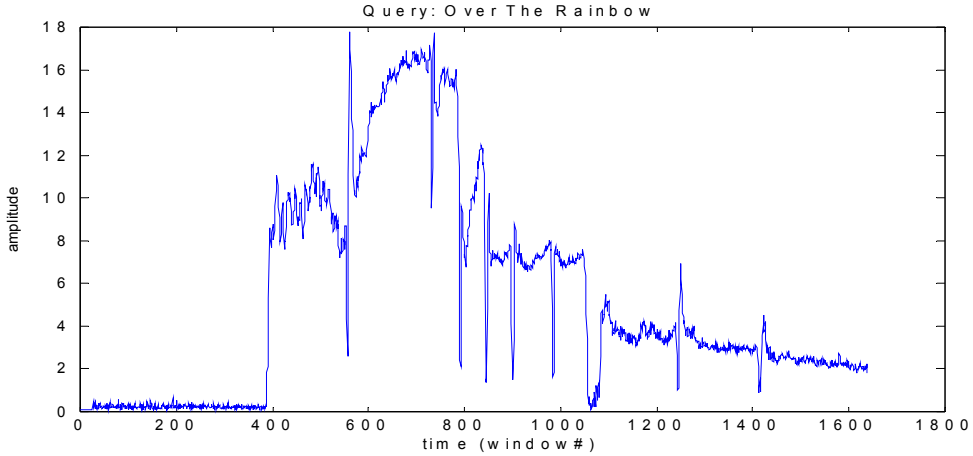


Figure 5-16: *Amplitude function A[w] of the query in Figure 5-14.*

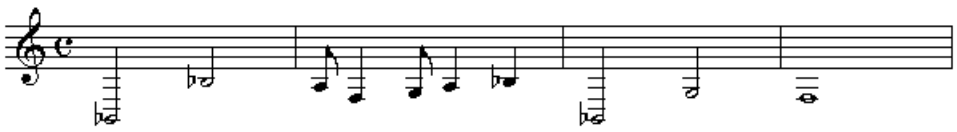


Figure 5-17: *The score of the query in Figure 5-14.*

A main difference of our method compared with previous amplitude based note segmentation methods is that a set of dynamic thresholds $\hat{a}[w]$ instead of one or two fixed thresholds are used, with a note onset recorded when the power exceeds the threshold and a note offset recorded when the power drops below the threshold.

In case the amplitude may exceed the thresholds for a short time due to the burst noise coming from the microphone, we defined another threshold $\hat{l}$ (e.g., 10 windows, 88ms, i.e., a sixteen note at a tempo of 180 beats per minute). Segments shorter than this threshold are discarded.

The segmentation algorithm is as follows:

(1) Compute the global threshold

$$\hat{a}_G = 0.3 \cdot \frac{1}{|w|} \cdot \sum_{w} A[w] \qquad \text{(Eq. 5-10)}$$

(2) Set the dynamic size $\hat{d}$ (e.g., 125).
(3) Set the current starting point $v = 1$.
(4) From v to $|w|$, divide $A[w]$ into n fragments, each of which contains $\hat{d}$ windows (except the last fragment which may have less than $\hat{d}$ windows).
(5) Compute the threshold for each fragment i. Assume that it contains d_i windows.

$$\hat{a}_F[i] = \max\{0.7 \cdot \frac{1}{d_i} \cdot \sum_{\substack{w \text{ within} \\ \text{segment } i}} A[w], \quad \hat{a}_G\} \qquad \text{(Eq. 5-11)}$$

(6) Compute the threshold for each window w. Assume that window w is contained in the fragment i.

$$\hat{a}[w] = \hat{a}_F[i] \qquad \text{(Eq. 5-12)}$$

(7) Use $\hat{a}[w]$ to segment the first note from v to $|w|$, with the note onset w_1 recorded when the power exceeds the threshold and the note offset w_2 recorded when the power drops below the threshold.
(8) Set the current starting point $v = w_2 + 1$. Go to step (4).

The segmentation process is illustrated in Figure 5-18 and Figure 5-19. Thresholds $\hat{a}[w]$ for segmenting each note are shown in the Figure 5-18 by horizontal lines.

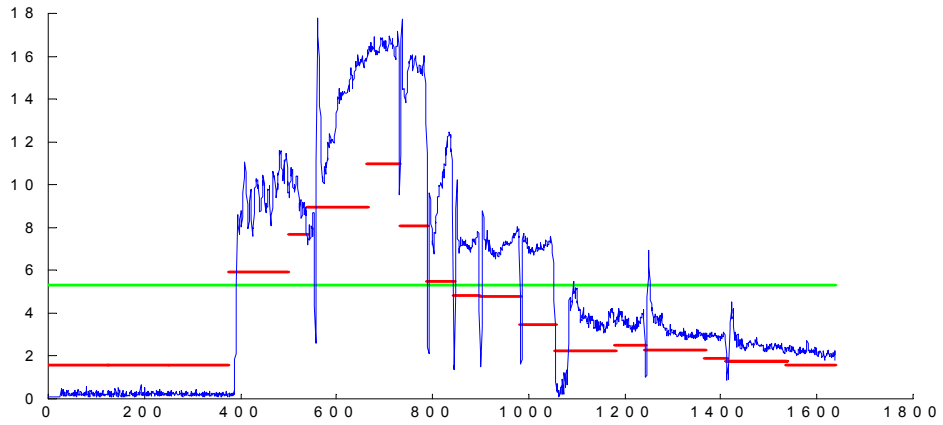


Figure 5-18: *The dynamic segmentation thresholds of the query in Figure 5-14.*

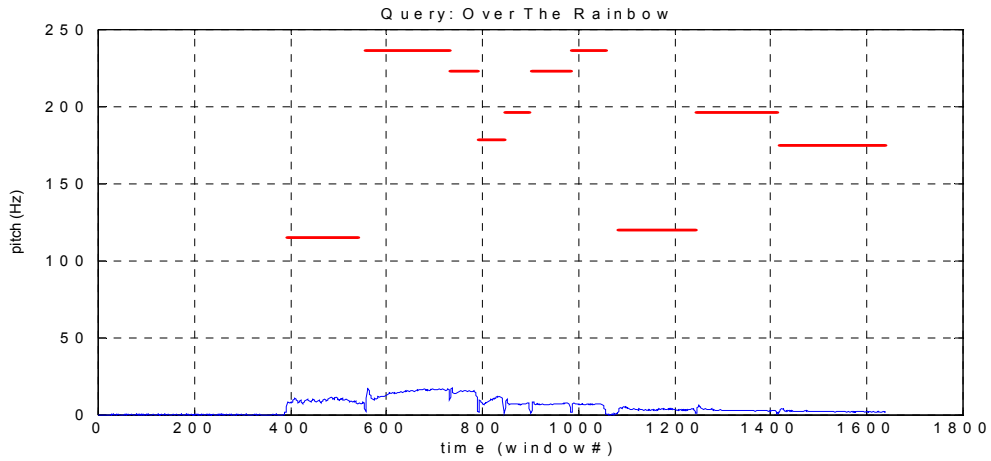


Figure 5-19: *The note segmentation and the pitch tracking results of the query in Figure 5-14 using ANS.*

This example shows that using one fixed threshold (indicated by the longest horizontal line in Figure 5-18) instead of the dynamic thresholds cannot segment all the notes correctly, because the amplitude may vary during the query; one best threshold for the whole query does not exist.

Amplitude-based note segmentation algorithms have add/drop errors occasionally. For example, when the amplitude of one note is not stable, which may rise and fall so as to cross the threshold several times, this note will be segmented into several notes and thus add errors occur. When drop in amplitude between the consecutive notes is not clear, drop errors occur. Using dynamic thresholds can significantly reduce add/drop errors, but cannot eliminate all of them.

5.3.2 Pitch tracking

Pitch is actually a perceptual concept, but in this thesis pitch tracking means to find the fundamental frequency (f_0) of the acoustic signals.

Considerations in choosing a pitch tracker include frequency range, frame rate, computation time, error rate, and resistance to noise. Some system specifications also affect the accuracy of the pitch tracker, including dynamic range of the microphone, sampling frequency, and amplitude quantization.

The pitch tracking algorithms can be divided into three categories (Rabiner, 1976) (Roads, 1994):

(1) *Time domain algorithms*, which operate directly on the waveform to estimate the pitch period. For these pitch trackers the measurements most often made are peak and valley measurements, zero-crossing measurements, and autocorrelation measurements. The basic assumption is that if a quasi-periodic signal has been suitably processed to minimize the effects of the formant structure then simple time-domain measurements will provide good estimates of the period. Classical algorithms include zero-crossing periodicity detector, peak periodicity detector, autocorrelation pitch detector, etc.

(2) *Frequency domain algorithms*, which use the property that if the signal is periodic in the time domain, then the frequency spectrum of the signal will consist of a series of impulses

at the fundamental frequency and its harmonics. Thus simple measurements can be made on the frequency spectrum of the signal (or a nonlinearly transformed version of it as in the cepstral pitch detector) to estimate the period of the signal. Classical algorithms include STFT-based pitch detector, adaptive filter pitch detector, tracking phase vocoder analysis, cepstrum analysis, etc.

(3) *Time- and frequency-domain algorithms*, which incorporate features of both the time-domain and the frequency-domain approaches to pitch tracking. For example, a hybrid pitch tracker might use frequency-domain techniques to provide a spectrally flattened time waveform, and then use autocorrelation measurements to estimate the pitch period.

Since developing robust and fast pitch tracking algorithm is not the focus of this thesis and a universal pitch tracking algorithm has not been developed – each algorithm has its own advantages and disadvantages, we chose a most widely used algorithm *Autocorrelation* in our system. The resulting pitch sequence will be quantized for melody matching according to the quantization vector Q (refer to Section 5.1.4). For example, the frequencies of two consecutive notes are f_1 and f_2, and $Q = [0 \quad 1 \quad 3]$. The pitch contour between these two notes will be

$$intv(f_1, f_2) = \lfloor 12 \cdot \log_2(f_2 / f_1) + 0.5 \rfloor \qquad \text{(Eq. 5-13)}$$

$$cont(f_1, f_2) = \begin{cases} 0 & , intv = 0 \\ sign(intv(f_1, f_2)) & , intv = \pm 1, \pm 2 \\ sign(intv(f_1, f_2)) \cdot 2 & , o.w. \end{cases} \qquad \text{(Eq. 5-14)}$$

Autocorrelation (AUTO)

Autocorrelation is one of the classical pitch trackers. The autocorrelation function is computed by Equation 5-15:

$$r_N[d] = \frac{1}{N} \sum_{n=1}^{N-d} x[n]x[n+d] \qquad \text{(Eq. 5-15)}$$

where N is the frame size,
d is a positive delay,
$x[n]$ is the signal, which is must be zero outside the frame, and
$r_N[d]$ is the autocorrelation at delay d.

By calculating the correlation between the signal and an increasingly delayed version of itself, the highest level of correlation can be found. This maximum in the autocorrelation function will occur at multiples of the pitch period length, allowing the frequency of the signal to be calculated. By itself, autocorrelation is prone to be influenced by the strong formant structure in speech, making it subject to harmonic errors (Hess, 1983). In our system, some simple rules were used to reduce the harmonic errors.

5.3.3 Interfaces for obtaining beat information

Two interfaces have been implemented to obtain the beat information from hummed queries.

Interface I (INTF-I)

The usage of this interface is as follows:

(1) The user inputs the numerator of the time signature and a proper tempo for a query.
(2) When the user begins to record a query, there will be a drum track playing at the same time based on the time signature and the tempo. Drum signals on strong beats are different from those on weak beats, so that the user can easily identify measures based on the rhythmic pattern.
(3) The user hums the melody with the drum signals. It can start from any measure, not necessarily from the first measure.
(4) Since most current sound cards support to record only the sound from the microphone, the recorded queries will not include the drum signals. Ideally, the users can use headphone to hear the drum signals, so that the drum signals will not be recorded in the hummed queries. The time when each drum signal occurs will be stored in a separate file for later query construction.

Thus, we can obtain all the information necessary for melody matching using TPBM-I from the recorded waveform query and the stored timing information, including

- Meter, which is actually the numerator of the time signature
- Pitch contour
- Relative beat number of each note, which remains the relative position within the measure

Interface II (INTF-II)

The usage of this interface is as follows. When the user begins to record a query, he needs to indicate the onset of each beat by clicking the mouse while he is humming. The drum signal will be played when the user clicks the mouse. It simulates people's habit of using hand or foot tapping the beat when humming songs. Similarly, the recorded queries will not include the drum signals. The time when the user clicks each beat will also be stored in a separate file for later query construction.

By this interface, we can only obtain the pitch contour information and the relative beat numbers, which may not remain the relative positions within measures, from the recorded waveform query and the stored timing information. Thus, TPBM-I matching method cannot be used for queries by interface INTF-II, while only TPBM-II or DPM matching method can be used.

5.3.4 Automatic beat tracking

The possibility of automatically extracting rhythmic information from the hummed queries was also explored in the QBH project. An *Optimal Alignment Method (OAM)* is proposed as follows.

Previous research on tempo or rhythm analysis of acoustic musical signals or music scores was mainly based on fairly long music fragments, while rhythm analysis of short hummed queries has not yet been conducted.

Additionally, building a system that can make rapid judgment about rhythm from a few hummed notes can help people learn more about humans' perception of rhythm.

The Optimal Alignment Method (OAM) described in the following extracts beat information that consists of the tempo of the query and the temporal position of each beat. In this thesis, we call one possible arrangement of temporal positions of beats as an *Alignment*. It can be fully specified by two parameters: tempo (τ) and the onset of the first beat (μ) as shown in Figure 5-20. Please note that the alignment shown in Figure 5-20 is apparently a poor one. Figure 5-21 shows alignments with different τ and μ.

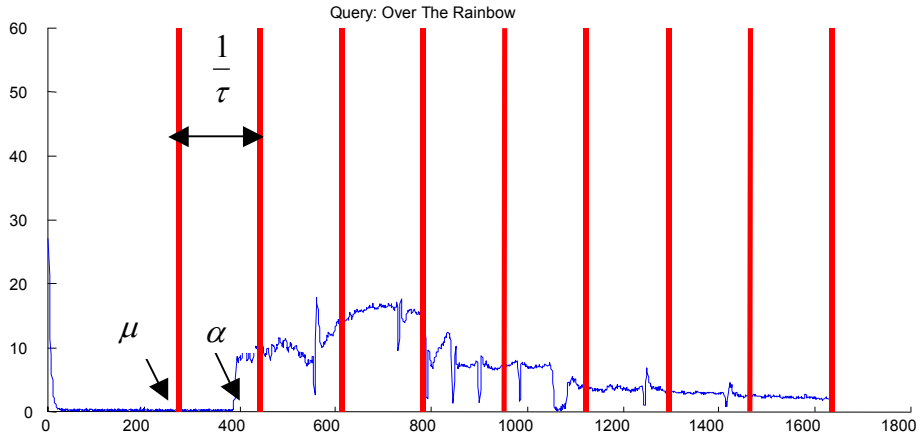


Figure 5-20: *The description of τ (the tempo), μ (the onset of the first beat) and α (the onset of the first note).*

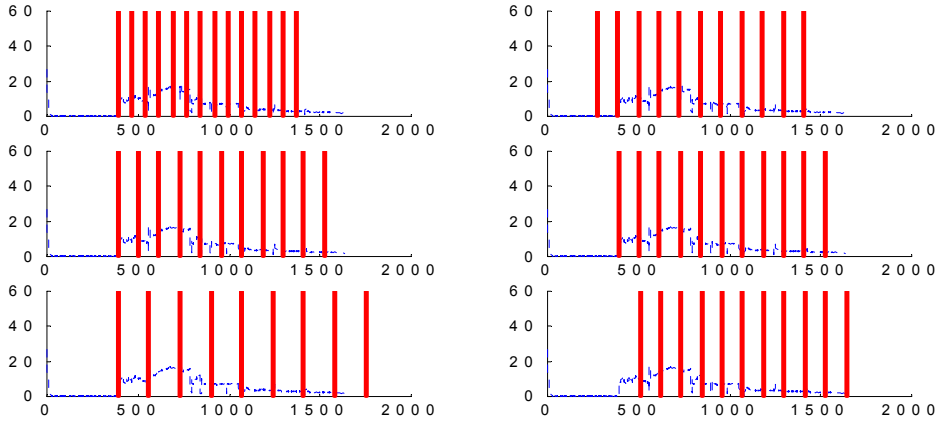


Figure 5-21: *In the first column are three different alignments whose onsets of the first beat are the same, but tempos are different. In the second column are three different alignments whose tempos are the same, but onsets of the first beat are different.*

Assumptions

The hummed queries are usually very short; they may last for only a few seconds and include several notes. Since the information is limited, we need some assumptions as our prior knowledge for solving the problem.

Assumption 1:
Since the query is short, we can assume that the tempo remains constant across the whole query. That means, τ doesn't change.

Assumption 2:
Due to the limitation of humans' singing, the tempo of hummed queries tends not to be too fast or too slow.

Assumption 3:
The onset of the first note is also the onset of the first beat in the query. Thus, μ equals α.

Optimal Alignment Method I (OAM-I)

The basic idea of this method is to find the most likely alignment. In OAM-I, all the three above assumptions are assumed to be true. Thus, μ will be the onset of the first note and we only need to find the optimal tempo τ. The following gives the description of OAM-I.

Step 1: Extract the note onsets from the hummed query using ANS

Step 2: Calculate the likelihood of each note onset given an alignment

The key part of our problem is to define the likelihood of an alignment. Given the alignment and the onset position of one note, we first need to compute which beat and at what point relatively within that beat the note will fall into and then calculate the likelihood of such a position. The principle we used for calculating the likelihood is that the note onsets are not equally likely to occur at any positions within a beat. For example, more notes would occur at the beginning of the beat than at one-third part from the beginning of the beat. Thus, we attempted to incorporate this prior knowledge into our method. Empirically, we set the likelihood $L_c(t)$ of each position within one beat to be as shown in Figure 5-22. It can be calculated using Eq. 5-16 to Eq. 5-19.

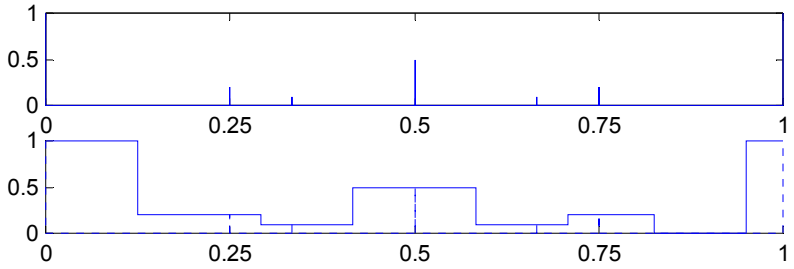


Figure 5-22: *Onset likelihoods within one beat. Above:* $L_d(t)$ *Below:* $L_c(t)$

$L_c(t)$ denotes the onset likelihoods within one beat:

$$L_c(t) = L_d(NN(t)) \qquad \text{(Eq. 5-16)}$$

where $t \in [0,1)$ is the relative onset position within one beat. It is computed from the onset's absolute time t_a, the tempo τ and the onset of the first beat μ, as shown in Eq. 5-17. Please note that μ here is fixed to be the onset time of the first note α in OAM-I.

$$t = (t_a - \mu) * \tau - \lfloor (t_a - \mu) * \tau \rfloor \qquad \text{(Eq. 5-17)}$$

NN(t) is used to find the nearest neighbor of *t* in *T*={0, 1/4, 1/3, 1/2, 2/3, 3/4, 9/10, 1}:

$$NN(t) = \arg\min_{s \in T}(abs(t - s)) \qquad \text{(Eq. 5-18)}$$

$L_d(t)$ denotes the onset likelihood within one beat at positions in *T*:

$$\begin{aligned} L_d(t) = & 1 \cdot \delta(t) + 0.5 \cdot \delta(t - 1/2) \\ & + 0.2 \cdot \delta(t - 1/4) + 0.2 \cdot \delta(t - 3/4) \\ & + 0.1 \cdot \delta(t - 1/3) + 0.1 \cdot \delta(t - 2/3) \\ & + 0 \cdot \delta(t - 9/10) + 1 \cdot \delta(t - 1) \end{aligned} \qquad \text{(Eq. 5-19)}$$

where $\delta(t) = \begin{cases} 1, & t = 0 \\ 0, & t \neq 0 \end{cases}$

The intuitive meaning of this definition is: notes occur most often at the beginning of beats, less often the half way positions, then the quarter positions, and rarely the one-third positions (for triples). We did not divide further because note onsets in hummed queries may not be obtained precisely and usually hummed queries will not be that complicated. For positions other than T={0, 1/4, 1/3, 1/2, 2/3, 3/4, 9/10, 1}, we simply find its nearest neighbor in *T* and assign its value. Please note that the weights used in $L_d(t)$ were empirically set, but it could also be computed statistically from some music collections.

Step 3: Calculate the likelihood of an alignment

Here, we want to incorporate our second assumption: the tempo τ should not be too fast or too slow. We introduced a tempo preference function, which is shown in Figure 5-23 and Equation 5-20. It is a Gaussian function centered at an average tempo $\bar{\tau}$, e.g., 1.6 beat per second (96bpm) in our configuration.

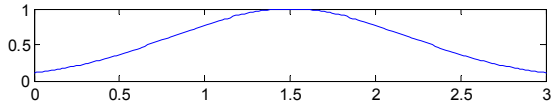


Figure 5-23: *Tempo preference function* $L_t(\tau)$

$$L_t(\tau) = e^{-(\tau - \bar{\tau})^2} \qquad \text{(Eq. 5-20)}$$

Now, to calculate the likelihood of the alignment, we can average the likelihood of each onset we have got in *step 2* and then weight it by the tempo preference function, as shown in Eq. 5-21. The results of this step are shown in Figure 5-24.

$$L(\tau) = \frac{\sum_{i=1}^{n} L_c(NN(t_i))}{n} \cdot L_t(\tau) \qquad \text{(Eq. 5-21)}$$

where *n* is the number of notes in the query.

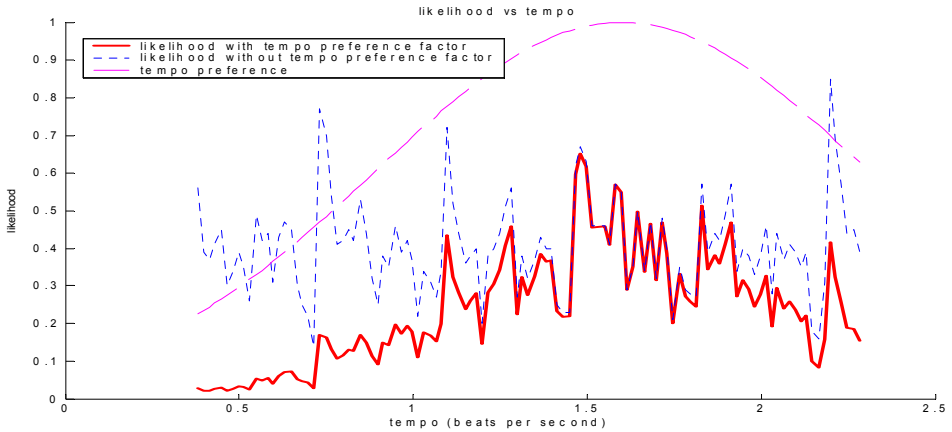


Figure 5-24: *Alignment likelihoods $L(\tau)$ – the maximum corresponds to the optimal alignment.*

Step 4: Choose the alignment with the maximum likelihood

We can calculate the likelihoods of a set of alignments (here means different tempos) given a query and then the alignment with the maximum likelihood will be chosen as the optimal alignment. Figure 5-25 shows the optimal alignment we chose for the example in Figure 5-14, which corresponds to the maximum in Figure 5-24.

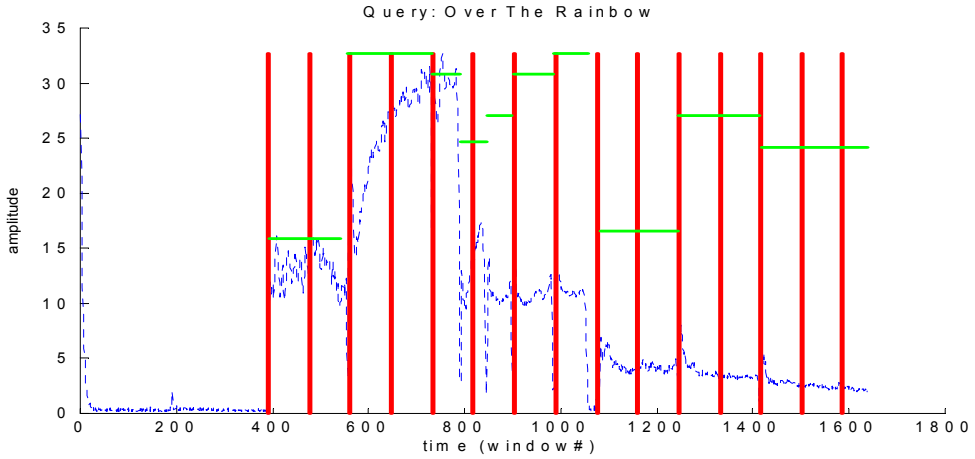


Figure 5-25: *The optimal alignment of the query in Figure 5-14 using OAM-I.*

Optimal Alignment Method II (OAM-II)

In *Optimal Alignment Method II (OAM-II)*, the third assumption may not hold (It rarely happens though). Thus, we need to choose both the optimal tempo (τ) and the optimal starting position (μ). The algorithm is similar to OAM-I except that when we try different alignments, we should not only change the tempos, but also the onset of the first beat.

$$L(\tau,\mu)=\frac{\sum_{i=1}^{n}L_c(NN(t_i))}{n}\cdot L_t(\tau) \qquad \text{(Eq. 5-22)}$$

where $t_i \in [0,1)$ is the relative onset position within one beat. It can also be computed using Eq. 5-17. But μ here can vary from $\alpha - \frac{1}{\tau}$ to α.

The result using OAM-II on the above example is shown in Figure 5-26. Please note that the likelihoods in the figure at $X = 0 \Rightarrow \mu = \alpha$ are the same as the results using OAM-I. Thus, OAM-II is a generalized version of OAM-I. When the onset of the first note is also the onset of the first beat in the query (it is typically true), the results of OAM-I and OAM-II should be equal, though OAM-II will be slower. Otherwise, if the assumption is not true (usually because the onset position of the first note is not determined precisely or the first note in the query does start in the middle of a beat), OAM-II will get different results.

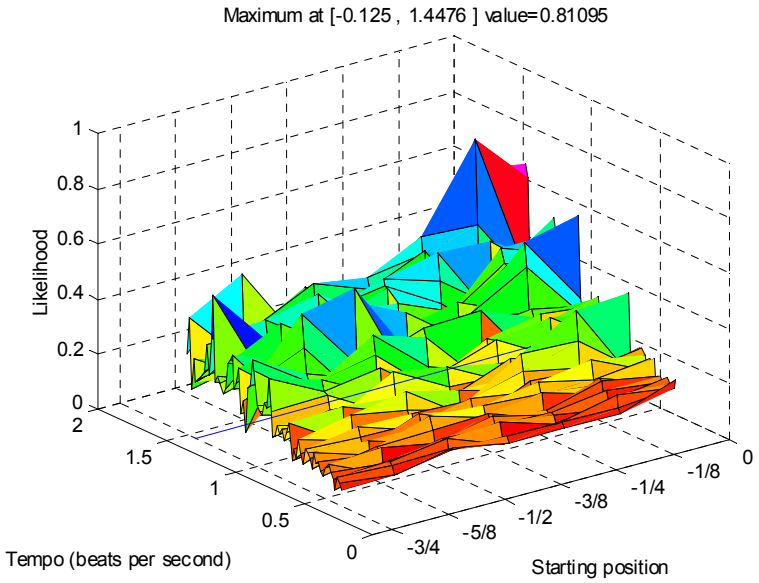


Figure 5-26: *Alignment likelihoods of OAM-II for the above example. X: starting position* $(\mu - \alpha) * \tau$ *; Y: tempo* τ *.*

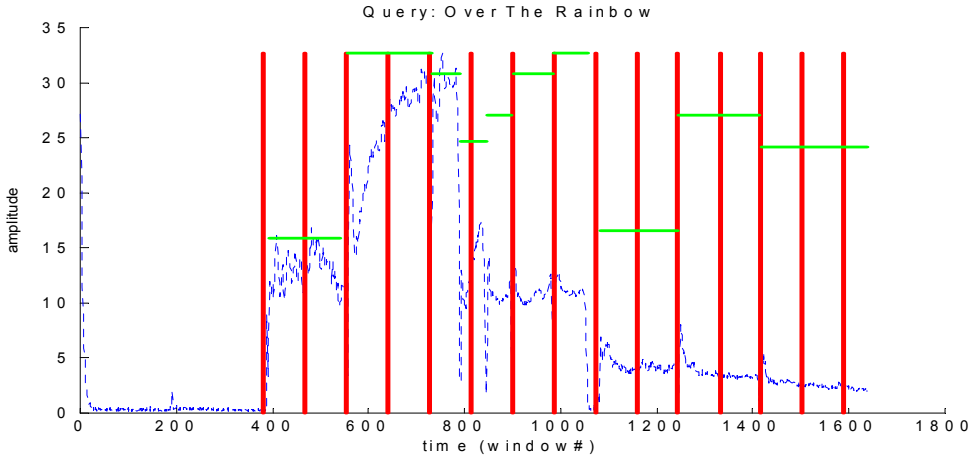


Figure 5-27: *The optimal alignment of the query in Figure 5-14 using OAM-II.*

Similar to INTF-II, we can only obtain the pitch contour information and the relative beat numbers from the recorded waveform queries, which may not remain the relative positions within measures. Thus, TPBM-I matching method cannot be used for queries by OAM, while only TPBM-II or DPM matching method can be used.

5.3 5 Query representation

The final query will be converted into ASCII CGI format to be transmitted to the server side for matching. The information capsulated in this format includes:

(1) Which matching algorithm to be used for this query: 1 for TPBM-I; 2 for TPBM-II; and 3 for DPM.
(2) How many most similar songs should be returned.
(3) Quantization vector Q.
(4) Time signature of the song. For TPBM-II or DPM, this field will be ignored.
(5) Pitch contour vector.
(6) Beat numbers vector. For DPM, this field will be ignored.

For example, the Over The Rainbow query in CGI format would be

http://bop.media.mit.edu/scripts/server.exe?mtype=1&maxnum=10&levels=0_1_3&timesig=4_4&pitch=100_2_-1_-2_1_1_1_-2_2_-1&beat=1_3_5_5_6_7_8_9_11_13

which indicates that

(1) TPBM-I algorithm will be used for matching ("mtype=1");
(2) Ten most similar songs will be returned ("maxnum=10");
(3) Quantization vector $Q = [0 \quad 1 \quad 3]$ ("levels=0_1_3");
(4) Time signature is 4/4 ("timesig=4_4");
(5) Pitch contour vector is [100 2 –1 –2 1 1 1 –2 2 –1] where 100 indicates the beginning ("pitch=100_2_-1_-2_1_1_1_-2_2_-1");
(6) Beat numbers vector is [1 3 5 5 6 7 8 9 11 13] ("beat=1_3_5_5_6_7_8_9_11_13"). The beat numbers should still remain the relative positions within measures, but not necessarily to be the same as the absolute beat numbers. That is, beat numbers [1 3 5 5 6 7 8 9 11 13] or [5 7 9 9 10 11 12 13 15 17] will give the exactly the same results given the meter is 4.

CHAPTER 6 IMPLEMENTATION ISSUES

This chapter presents some technical details to build the QBH system efficiently, including the database indexing, the server implementation and the client implementation. We also built an ActiveX control version of QBH Client, so that we can embed our client application into web pages.

6.1 Database Indexing

To make the system more time efficient, we built a set of persistent objects, *melody description objects*, containing all the information needed for matching. Once QBH Client sends a request to QBH Server, the server will load all the objects into memory first and then do the matching. Thus, the persistent objects act as an efficient indexing of the song database. Please note, *database* in this thesis is in its broader sense -- it means all the target songs and their indices that are currently stored in the file system, not a strictly speaking database. However, for a much larger music corpus, using a real database system instead of the file system should be more efficient and easier to maintain. In that case, we can still use our persistent objects as indexing by adding one more field pointing to the key of the song in the database; or use the indexing mechanism supported by some database systems in a similar way.

The structure of the melody description objects is shown in Figure 6-1. The objects were built using the melody extraction tool we developed in Section 5.2.

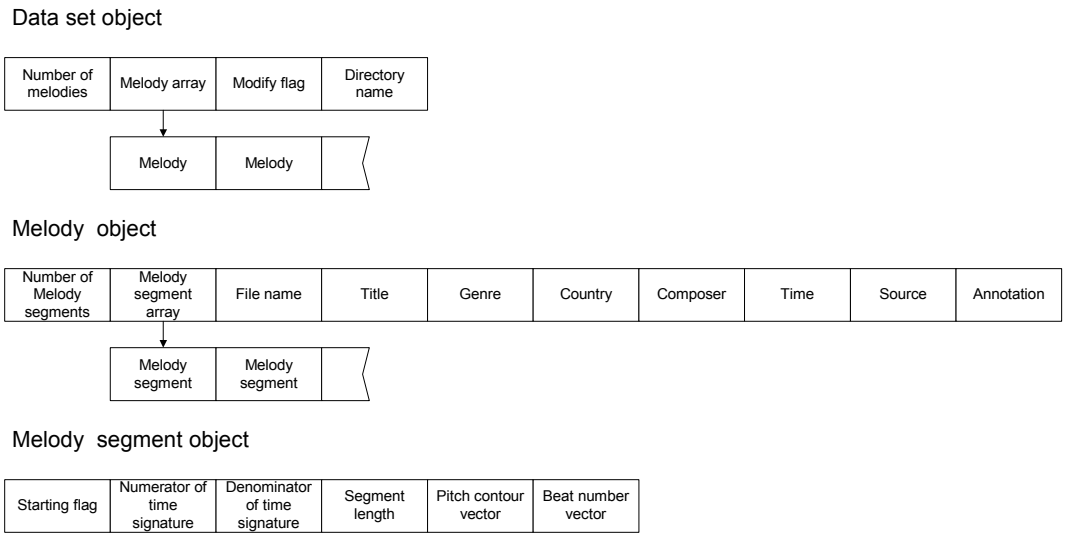


Figure 6-1: *Persistent melody description objects for database indexing.*

6.2 QBH Server Implementation

Both QBH Server and QBH Client were implemented on MS Windows platform and developed using Microsoft Visual C++.

We are currently using Microsoft Internet Information Server (IIS) as our web server, but other web servers on Windows platform supporting CGI can certainly be adopted as well. QBH Server performs as a CGI program, which will be called by the web server. Thus, the web

server obtains the query from QBH Client and passes the query to QBH Server. Then, QBH Server performs as shown in Figure 6-2:

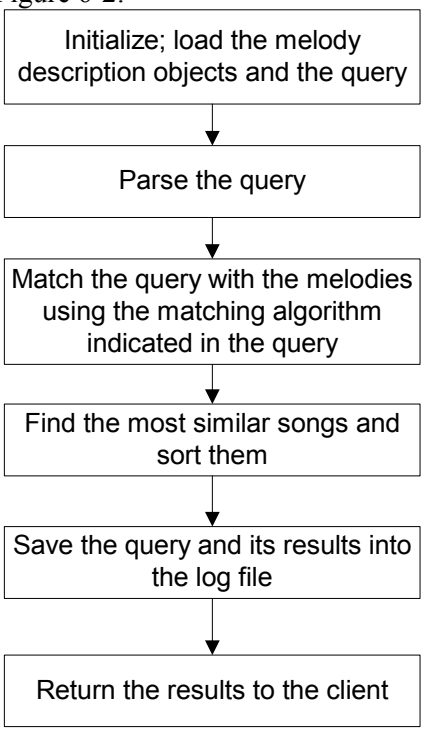


Figure 6-2: *Procedure of QBH Server.*

6.3 QBH Client Implementation

QBH Client (Figure 6-3) records the hummed melodies, constructs queries, sends them to QBH Server and plays the results. It performs as shown in Figure 6-4. The waveform recording and playback part was developed using MS DirectSound. The MIDI playback was developed using Media-player ActiveX control. All the pitch tracking and beat tracking algorithms were implemented using Matlab; and then the Matlab programs were compiled using Matlab MCC and embedded into C++ code.

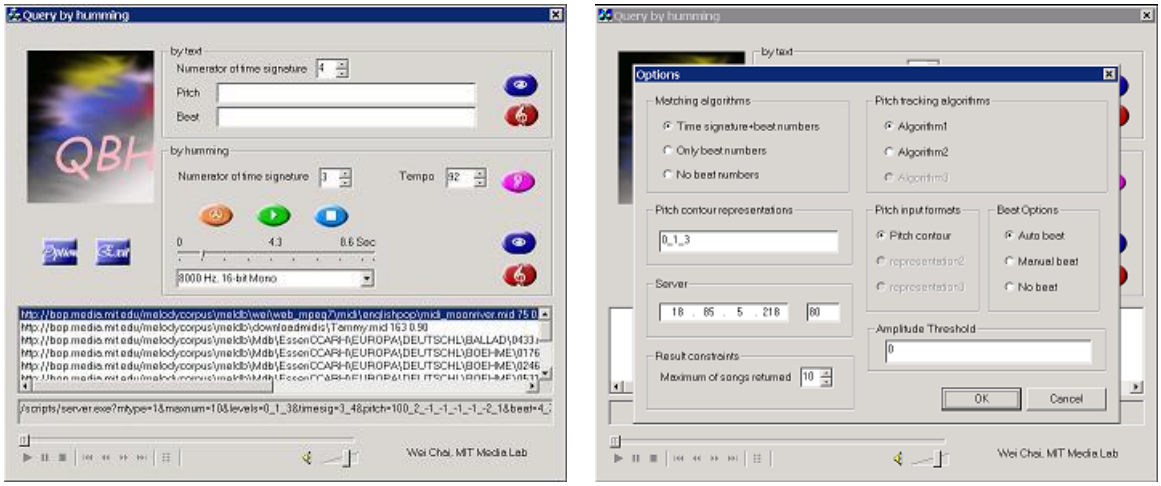


Figure 6-3: *Interface of QBH Client.*

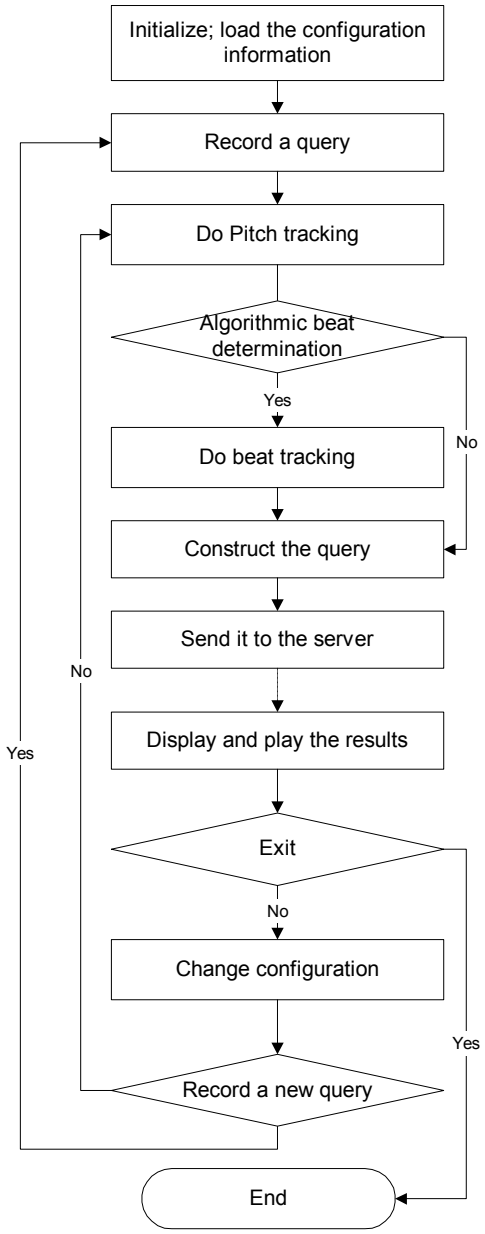


Figure 6-4: *Procedure of QBH Client.*

6.4 AXClient – AxtiveX Implementation of QBH Client

We also implemented an ActiveX version of QBH Client – AXClient. AXClient can be embedded directly into the web page. Users can run AXClient by clicking the link in a web page using Windows Internet Explorer without installing it in advance (Figure 6-5).

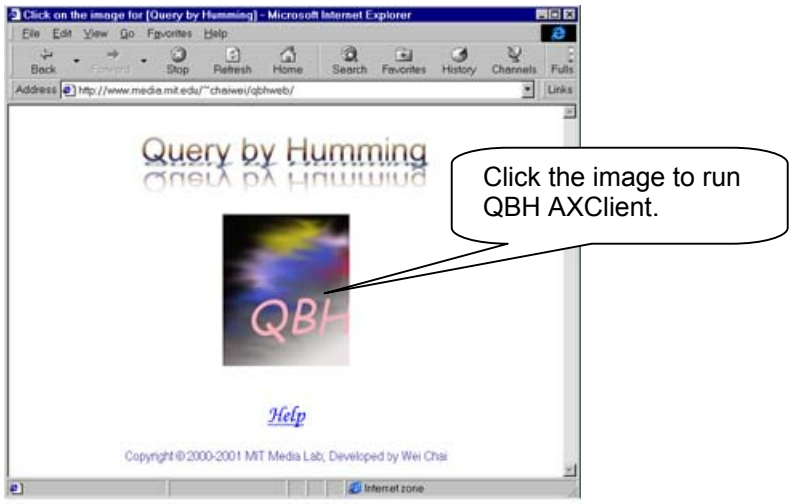


Figure 6-5: *The web page embedded with QBH AXClient.*

ActiveX controls are the third version of OLE controls (OCX), providing a number of enhancements specifically designed to facilitate distribution of components over high-latency networks and to provide integration of controls into Web browsers.

We chose ActiveX instead of Java applet to fulfill the goal of allowing QBH Client to be embedded into web pages, because a Java applet has more security constraints, which make it difficult to manipulating client side hardware (sound card) directly and write the temporary sound files in the local machine. Since sound recording in Java applets is implemented by JVM, currently, the sound quality recorded by most major JVMs is very poor. On the other hand, sound recording in ActiveX control is by direct system call, so the recording sound quality is good. Additionally, ActiveX (executable binary code) runs faster than Java applets (bytecode interpreted by JVM).

The structure of QBH AXClient is illustrated in Figure 6-6. We built QBH AXClient by wrapping the standalone QBH Client in the ActiveX control shell. Then we packed QBH AXClient in a compressed cabinet file ("QBHweb.cab") along with other necessary files, such as Matlab libraries, and signed the file, so that users can download it faster and verify it before running.

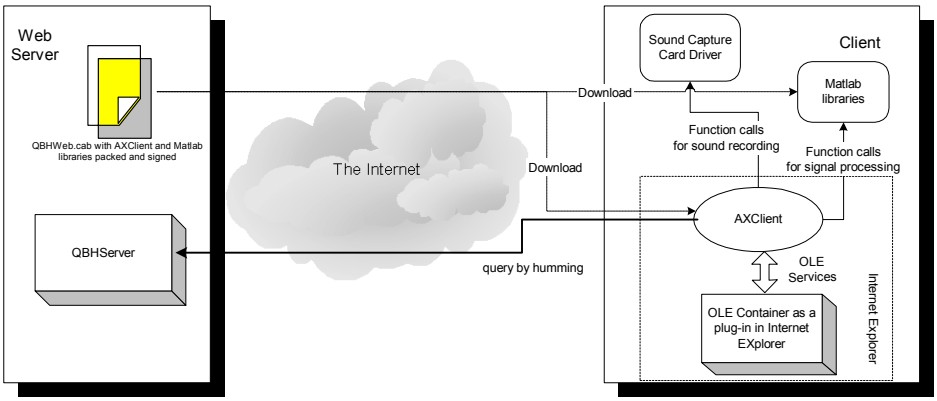


Figure 6-6: *Structure of QBH AXClient.*

CHAPTER 7 EXPERIMENTS AND EVALUATION

This chapter presents our experiments and the results for evaluating the system. The experiment procedure, the data set, apparatus and subjects are described. The accuracy of our note segmentation, pitch tracking and beat tracking algorithms is presented. The statistics of the hummed queries, including histograms of pitches, intervals, query lengths and tempi, are shown. The effectiveness of interfaces and algorithms based on both perfect queries and real queries is presented at last. Experimental results demonstrate that our methods combining both pitch and rhythmic information do help improving the effectiveness of the query-by-humming system.

7.1 Method

7.1.1 Procedure

Ten musically trained and untrained subjects participated in the experiment. There are two primary goals of this experiment. One is to collect some hummed queries to adjust the parameters used in the algorithms and to test the QBH system. The other is to explore the melody perception issues involved in such a system. For example, what is the typical tempo and length the subjects hum as a query? How accurate can the subjects memorize and hum the melody in terms of both pitch and rhythm? Are there any significant differences between professionals and amateurs?

The procedure of the experiment was as follows.

1) The experimenter explained the procedure and the purposes of the experiment, demonstrated how to run QBH Client, e.g., how to choose different interfaces or algorithms, record queries and search. At least one practice trial was used to familiarize the subject with the experimental procedure and to set the amplification at a comfortable listening level.

2) The subject chose at least 5 familiar songs from a web page including hundreds of well-known pop or classical songs. Then he wrote down the titles, genres and how well he knew the songs.

3) For each song, the subject hummed the melody to let the system search the song in the whole database (about 8000 songs). All the subjects were told to hum the queries with syllable *da*. The subject could determine from where and how long he hummed the melody for each query. In the query process, different algorithms might be used.

4) The computer returned ten songs with highest matching scores. The subject judged which one was correct. If the correct song was among the returned songs, the subject went on to the next song in his query list; otherwise, if the correct song was not among the retuned songs, he could either skip the current song to search the next one in his query list or hum the current one once more to repeat searching.

5) Finally, the subject was asked to fill in a questionnaire.

The subject was allowed to play the songs linked in the web page before he hummed the melody to make sure that it was the song he knew, but he was not encouraged to do so. The subjects were encouraged to proceed at their own pace. All the subjects took one to two hours to complete the experiment, which depended on the number of queries.

All the hummed queries, constructed ASCII queries and the configurations (algorithms, interfaces, parameters that were configured in the option dialog) the subjects chose for the queries correspondingly were stored in the local machine, so that the experimenter can analyze all the data offline.

All the subjects were randomly divided into two groups with 5 subjects each. Subjects in Group A participated in the pilot experiments, whose goal was to test the experiment equipment and the stability of the system. Subjects in Group A were told to use any combinations of algorithms and interfaces they prefer for each song. The experimenter was present when the subject did the queries to check the experiment pace and problems involved.

Subjects in Group B participated in the formal experiments. They were told to use four different combinations of algorithms/interfaces for each song. The four combinations were: (1) using TPBM-I for matching and INTF-I for obtaining rhythmic information. (2) using TPBM-II for matching and INTF-II for obtaining rhythmic information. (3) using TPBM-II for matching and OAM-I for obtaining rhythmic information. (4) using DPM for matching and thus rhythmic information was not necessary. Therefore, the subjects needed to hum at least three times for one queried song, corresponding to the first three combinations respectively, and the last combination can use any of the three queries (say, the third query) by just changing the matching algorithm without humming once more. The experimenter was not present when the subject started the queries after the practical trial demonstrating the subject had already been able to use the software.

7.1.2 Query data set

All the songs linked in the experiment web page were MIDI files downloaded from the Internet. They were classified into English pop (about 300 pieces), Chinese pop (about 100 pieces), Christmas songs (about 20 pieces) and classical songs (about 10 pieces). The subject could first go into a specific category through the corresponding link and then browse to find his familiar songs. All the songs within one category were listed in alphabet order based on the titles and with a MIDI files linked to each title. Most of the songs were well-known pieces, especially for subjects from English-speaking countries or Chinese-speaking countries.

All these MIDI files were assumed to have monophonic melody tracks and inserted into the melody database using the algorithm described in Figure 5-10. Through the experiment, we found several MIDI files only had polyphonic melody tracks or had no melody tracks at all, which caused corresponding search failures. But this kind of files were no more than 10% of all the MIDI files contained in the experiment web page by analyzing the experiment result and no more than 1% of all the files contained in the melody database because MIDI files converted from score files (>7000/8000) did not have such errors. In the following, the failures caused by this kind of score errors will be denoted as *SCORE ERR*.

7.1.3 Apparatus

Both QBH Client and QBH Server run on the same laptop PC machine for all the experiments. The client and the server communicated locally. The experiments were done at a soundproof booth to prevent environmental noise.

All the experiments for Group B were done using exactly the same facilities. The same headset with microphone for PCs was used for sound recording and playback.
In the pilot experiments for Group A, the subjects were using different microphones. Some of the subjects used headphones, while others did not.

7.1.4 Subjects

The subjects were recruited with emails. All subjects were from colleges or companies around Cambridge. Table 7-1 to Table 7-7 show the demographic information about the ten subjects' profile and musical background. The statistics of the subjects show that most subjects participating in our experiments did not have special singing training or professional music background. They were typically amateurs who like singing. We can expect that for future query-by-humming applications these people will be the major part of the users.

Table 7-1: *Subjects' gender.*

	Male	Female
Group A	3	2
Group B	2	3

Table 7-2: *Subjects' age.*

	23-30	31-40	41-50	51-60	>60
Group A	2	1	1	0	1
Group B	3	0	2	0	0

Table 7-3: *Subjects' occupation.*

	Undergraduate student	Graduate student	Professor	Other academic	Non-academic
Group A	0	1	2	1	1
Group B	0	4	0	0	1

Table 7-4: *Subjects' native language.*

	English speaker	Chinese speaker	Spanish speaker
Group A	3	1	1
Group B	3	2	0

Table 7-5: *Subjects' music experience.*

	Music major	Amateur who likes practicing	Amateur who only likes appreciating	Not a music fan
Group A	2	1	2	0
Group B	1	4	0	0

Table 7-6: *Subjects' singing experience.*

	Professional	Amateur who likes singing and practices a lot	Amateur who likes singing but seldom practices	Someone who does not like singing
Group A	1	1	3	0
Group B	0	3	2	0

Table 7-7: *Subjects' score reading experience.*

	0 (none)	1	2	3	4 (proficient)
Group A	1	1	1	0	2
Group B	0	0	2	2	1

7.2 Results

7.2.1 Note segmentation and pitch tracking

It is difficult to precisely evaluate the accuracy of the note segmentation and pitch tracking algorithm, because the procedure cannot be automatic. My method is that I listened to each query and then compared it with the transcription done by the computer. Thus, the note segmentation errors and harmonic errors in pitch tracking can be easily found, while the precision errors in pitch tracking were hard to be found. The main criteria for evaluating the pitch tracking accuracy is that the relationship between continuous notes (intervals) should be roughly correct. In this way, the errors that were not detected should be very limited.

Table 7-8 shows the statistics of the note segmentation and the pitch tracking error rate based on hummed queries by subjects in Group B. The errors are divided into three categories: drop errors (losing notes or merging notes), add errors (adding notes) and harmonic/pitch errors. Please note that the drop errors are mainly due to the deficit of the note segmentation algorithm, but they might also be caused by the pitch tracking errors (especially harmonic errors), because in our note segmentation algorithm, if the note pitch is detected to be out of range between 100Hz and 1000Hz, the note will be discarded. It happened rarely that some subjects hummed notes beyond this frequency range.

Table 7-8: *Note segmentation and pitch tracking errors based on hummed queries by subjects in Group B.*

	Drop errors	Add errors	Harmonic/pitch errors	Total errors
Sub. 1	2.7%	0.4%	0	3.1%
Sub. 2	7.7%	0.7%	0.5%	8.9%
Sub. 3	3.6%	0	3.6%	7.2%
Sub. 4	1.4%	1.7%	1.4%	4.5%
Sub. 5	2.3%	0.5%	0.6%	3.4%
Average	3.4%	0.9%	0.9%	5.2%

7.2.2 Automatic beat tracking

It is also hard to precisely define and evaluate the accuracy of the beat tracking algorithm. My method is that I listened to each query, judged the rhythmic structure of the query (where beats should occur) and then compared it with the beat tracking result computed by the computer. The beat tracking results were represented in two ways:

(1) Visual representation: The query is represented as horizontal lines indicating notes obtained from the note segmentation and pitch tracking algorithm, and vertical lines indicating beats computed by the beat tracking algorithm.

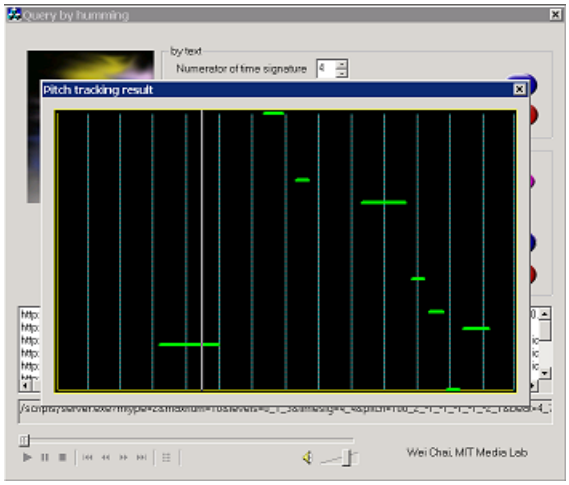


Figure 7-1: *The interface for displaying pitch tracking and beat tracking results.*

(2) Audio representation: The query can be played back with drum signals added at the positions where beats occur according to the algorithm, so that the experimenter or users can easily judge if the beat tracking result is correct.

Table 7-9 shows the statistics of the beat tracking performance using different algorithms based on hummed queries by subjects in Group B. If there are multiple queries corresponding to one target song and the beat tracking results of one or more of the queries are correct (may not be all), then the beat tracking result of the trail for the song is counted as correct. Besides OAM-I and OAM-II, we also tried OAM-I with simplified $L_c(t)$ (shown in Figure 7-2) and OAM-I without considering the tempo preference factor. The results demonstrate that both OAM-I and OAM-II performed above 70% correct, while III and IV performed slightly worse.

Table 7-9: *The performances of the algorithms. I: OAM-I; II: OAM-II; III: OAM-I with simplified onset position likelihood within one beat; IV: OAM-I with equally likely tempos. √: correct; X: wrong; 2: the tracked tempo is twice faster; ½: the tracked tempo is twice slower; 1/3: the tracked tempo is three times slower.*

		Note#	I	II	III	IV
Subject 1	1	16	√	√	√	√
	2	36	½	½	½	½
	3	18	√	X	√	2
	4	19	√	√	1/3	√
	5	21	X	X	X	X
	6	50	√	X	X	√
	7	32	½	½	½	√
Subject 2	1	21	X	√	2	2
	2	22	√	√	√	2
	3	22	√	√	√	2
	4	31	√	√	√	2
	5	18	√	√	√	√
	6	14	X	√	X	X
	7	38	√	√	√	√
	8	15	√	√	√	√

	9	16	√	√	√	√
	10	16	√	√	√	√
	11	15	√	√	√	√
Subject 3	1	22	√	√	√	2
	2	9	√	√	√	√
	3	17	X	X	X	X
	4	17	√	√	√	√
	5	10	√	√	√	√
	6	15	X	√	X	√
	7	8	√	X	√	√
Subject 4	1	15	√	√	√	√
	2	31	√	X	√	√
	3	31	√	√	√	√
	4	30	√	√	√	√
	5	23	√	√	√	√
	6	31	√	X	√	√
	7	24	√	√	√	√
Subject 5	1	16	X	X	X	X
	2	13	√	√	√	√
	3	25	X	X	2	X
	4	19	√	√	X	√
	5	16	X	X	√	X
	6	15	X	√	√	X
	7	14	√	√	√	2
	8	16	√	√	√	√
	9	30	√	√	√	√
	10	25	2	X	2	X
	11	24	√	√	√	√
	12	22	X	√	X	2
	13	15	√	√	√	√
Accuracy			71%	71%	69%	62%

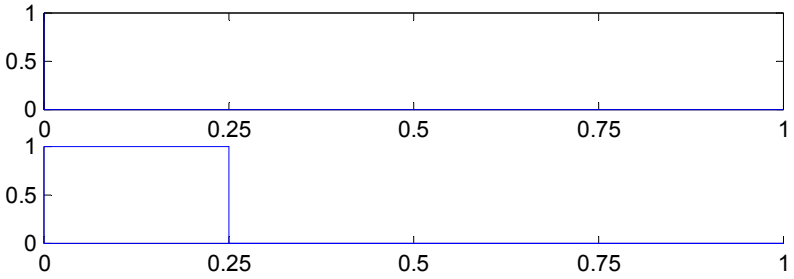


Figure 7-2: *Simplified onset likelihood within one beat.*

Please note that the rhythmic patterns of the queries were often but not always consistent with rhythmic patterns of the queried songs, because the subjects sometimes transformed the rhythm

of the original piece. Later results (Section 7.2.4) will show that some subjects tend to do so due to inaccuracy of the long-time music memory.

The performance of Optimal Alignment Method (OAM) demonstrates that computer determination of beat information from hummed queries is feasible.

We have also tried to determine the musical meter from the hummed queries: the time signature and each beat's position within a measure based on principles that the number of note occurrences depends upon the meter, the highest number is at the position of the beginning of the measure, and listeners attempt to place long notes on strong beats (Brown, 1993) (Howell, 1985). However, the results did not turn out to be accurate. One reason is that the query is too short to accumulate sufficient information for such judgment. This is consistent with humans' perception: although listeners attempt to establish a metrical analysis as soon as possible (Howell, 1985), sufficient clues and long enough time are needed.

One interesting thing is that when human listeners listen to hummed queries, they feel easier to perceive the rhythmic patterns of some queries, while they feel those of others somewhat ambiguous. This could be determined by the singer's singing ability, the song's rhythm and the listener's familiarity with the song (music memory). In our experiment, the likelihood L gives a measure of certainty of the rhythm. The correlation between this measurement and humans' actual rhythm perception can be explored in the future.

7.2.3 Statistics of the hummed queries

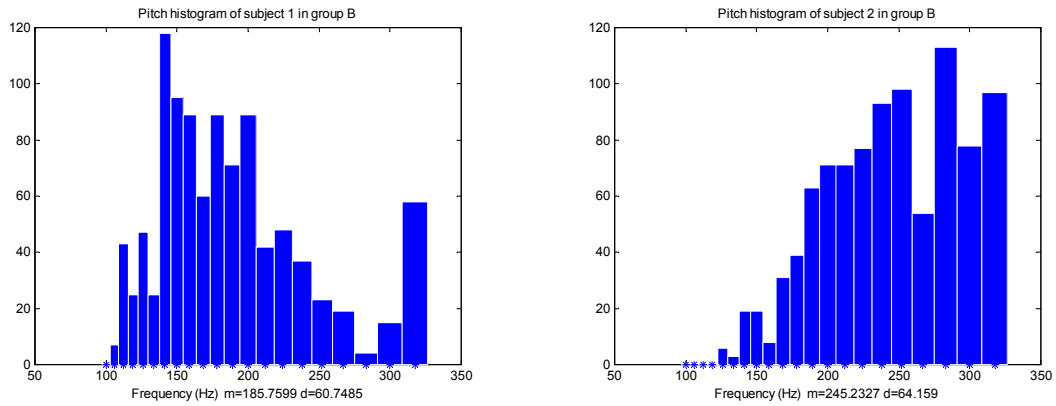
The following statistics is based on the hummed queries by subjects in Group B. All the data were computed automatically using ANS note segmentation and AUTO pitch tracking algorithms. The tempo related data were obtained using INTF-I, INTF-II and OAM-I respectively according to the subjects' options. The error rates of these algorithms have been presented in Section 7.2.1 and Section 7.2.2.

Pitch histograms (in Hz) of the hummed queries

Since the perceived pitch is a logarithmic transformation of frequency, the bins of the histogram were not equally spaced, instead, they were divided in this way:

$$\frac{C_{i+1}}{C_i} = 2^{1/12} \qquad \text{(Eq. 7-1)}$$

where C_{i+1} and C_i are the centers of the $(i+1)^{th}$ and the i^{th} bins respectively. Thus, the range of each bin is roughly one semitone.



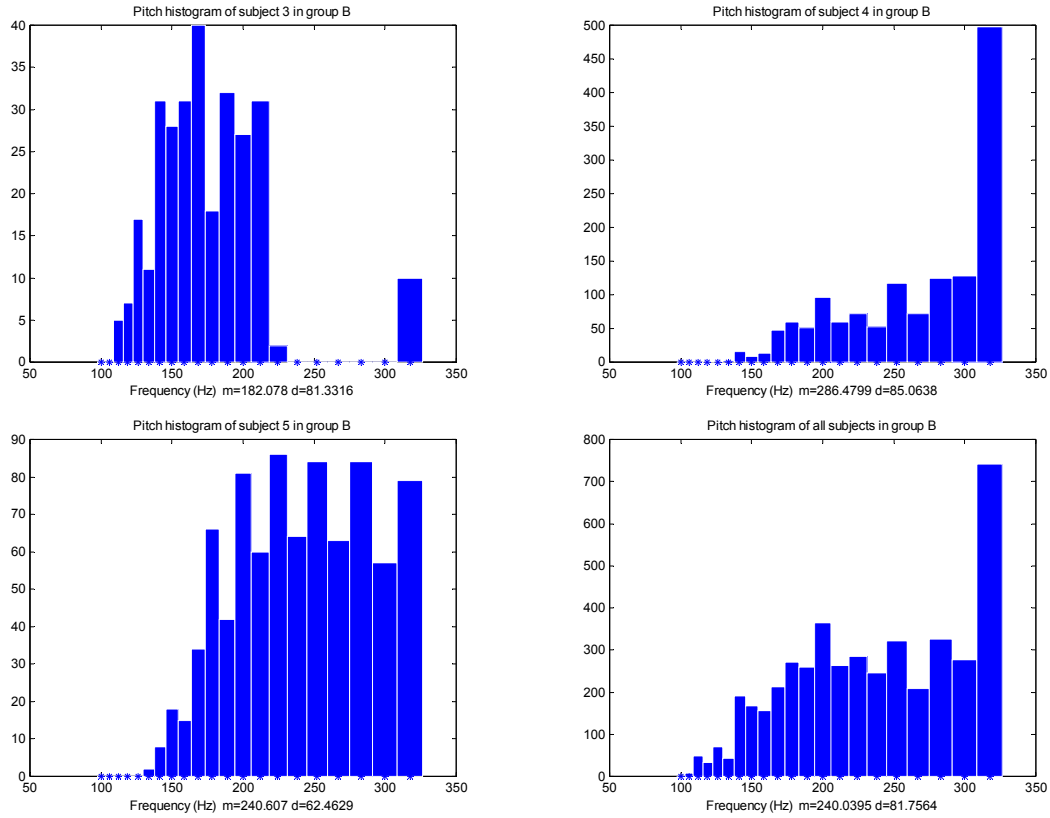
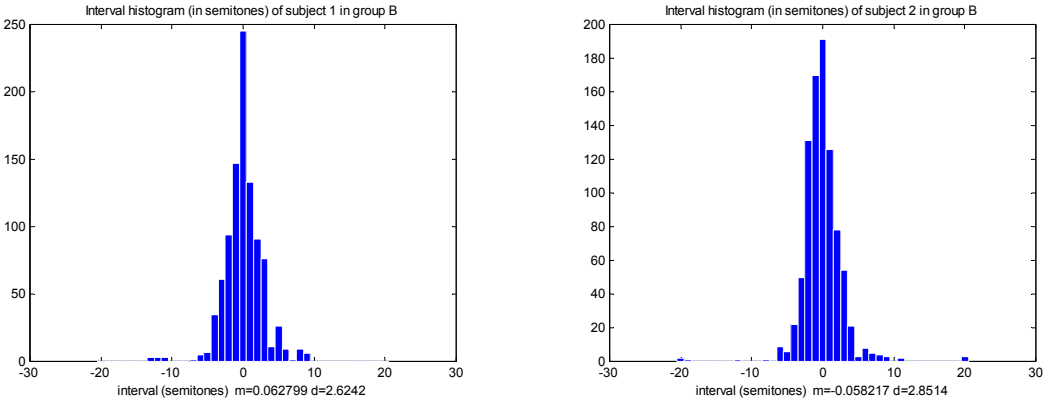


Figure 7-3: *Pitch histograms of the hummed queries by subjects in Group B. Subjects 1 and 3 are male. Subjects 2, 4 and 5 are female.*

Figure 7-3 shows that usually people do not hum too high or too low for the queries. 100Hz to 350Hz might be the typical range for both female and male users.

Interval histograms (in semitones) of the hummed queries



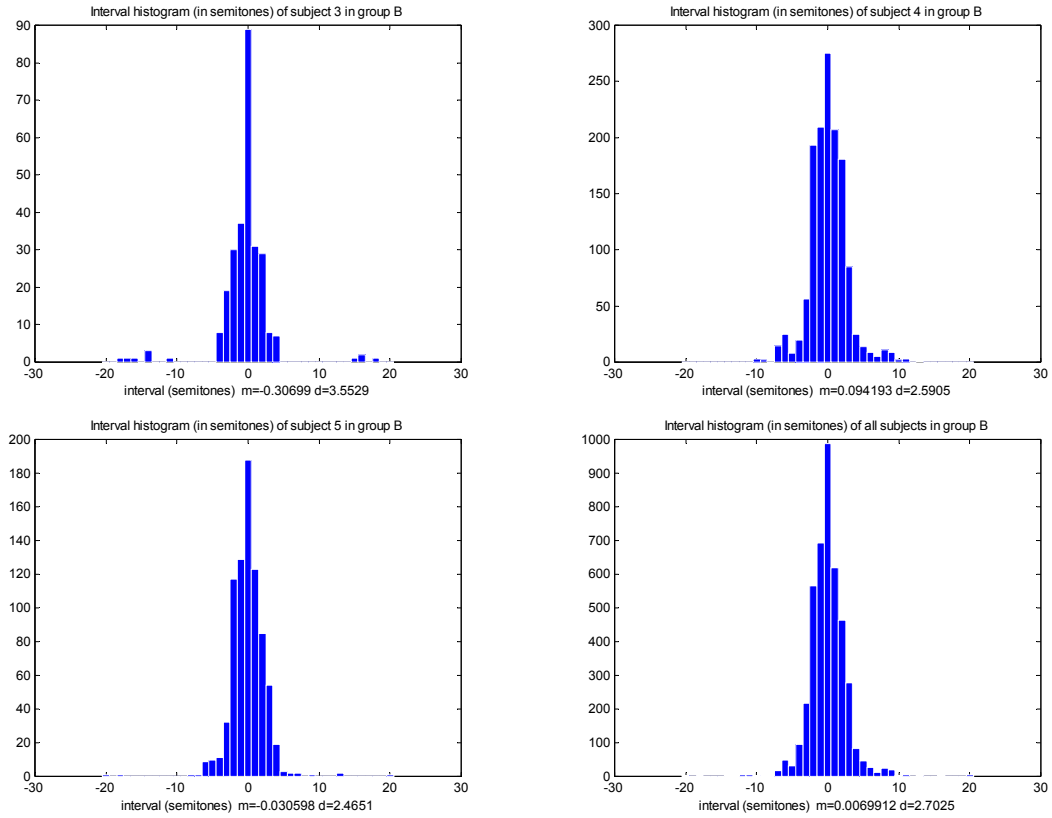


Figure 7-4: *Interval histograms of the hummed queries by subjects in Group B.*

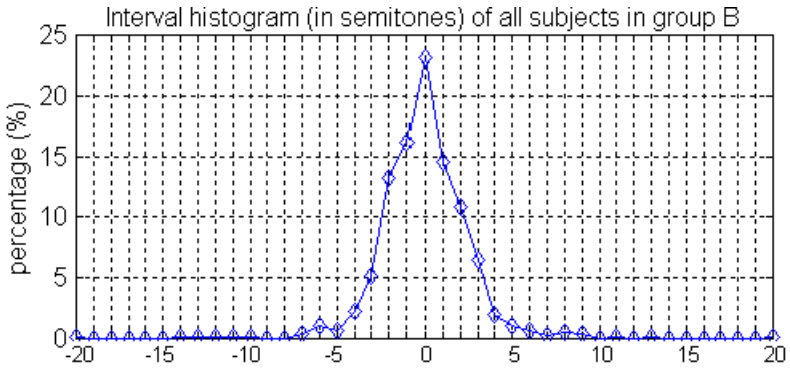


Figure 7-5: *The average interval histogram of the hummed queries by subjects in Group B (equivalent to the last figure in Figure 7-4).*

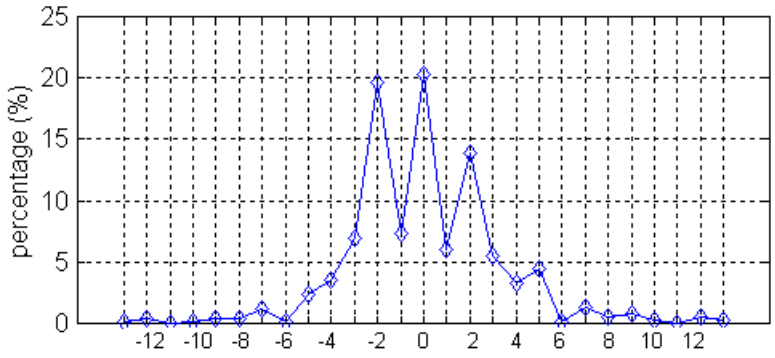


Figure 7-6: *The interval histogram of the whole music corpora (same as Figure 5-13-c).*

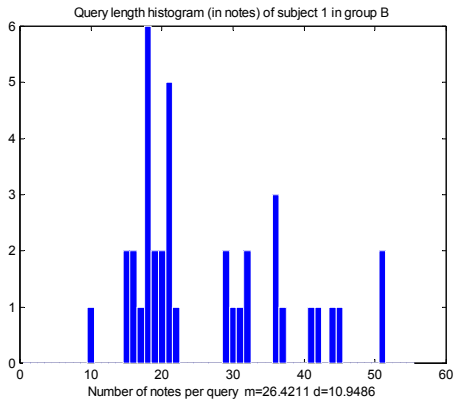
Comparing the interval histogram of the hummed queries with that of the whole music corpora, it is very interesting to see that the queries tend to have smaller interval changes. The reasons for this phenomenon might be:

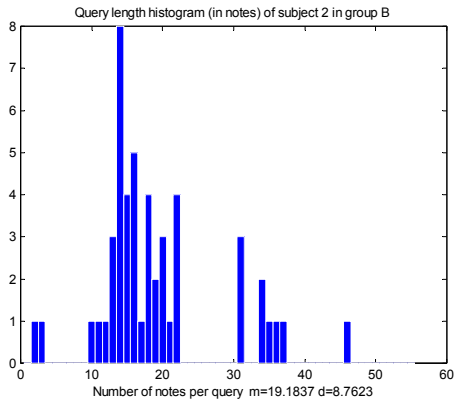
- It is easier for people to memorize the parts of melodies with smaller interval changes, which should also be the "hook" of the melody.
- People tend to pick the parts with smaller interval changes for query, because it is easier to sing.
- People tend to compress wide intervals and stretch small ones.

It is not clear though which reason contributes more to this phenomenon.

There are more intervals of minor second (one semitone) appearing in the hummed queries. It should be caused by the inaccuracies of the users' singing and the pitch tracking algorithm.

Query length histograms (number of notes per query) of the hummed queries





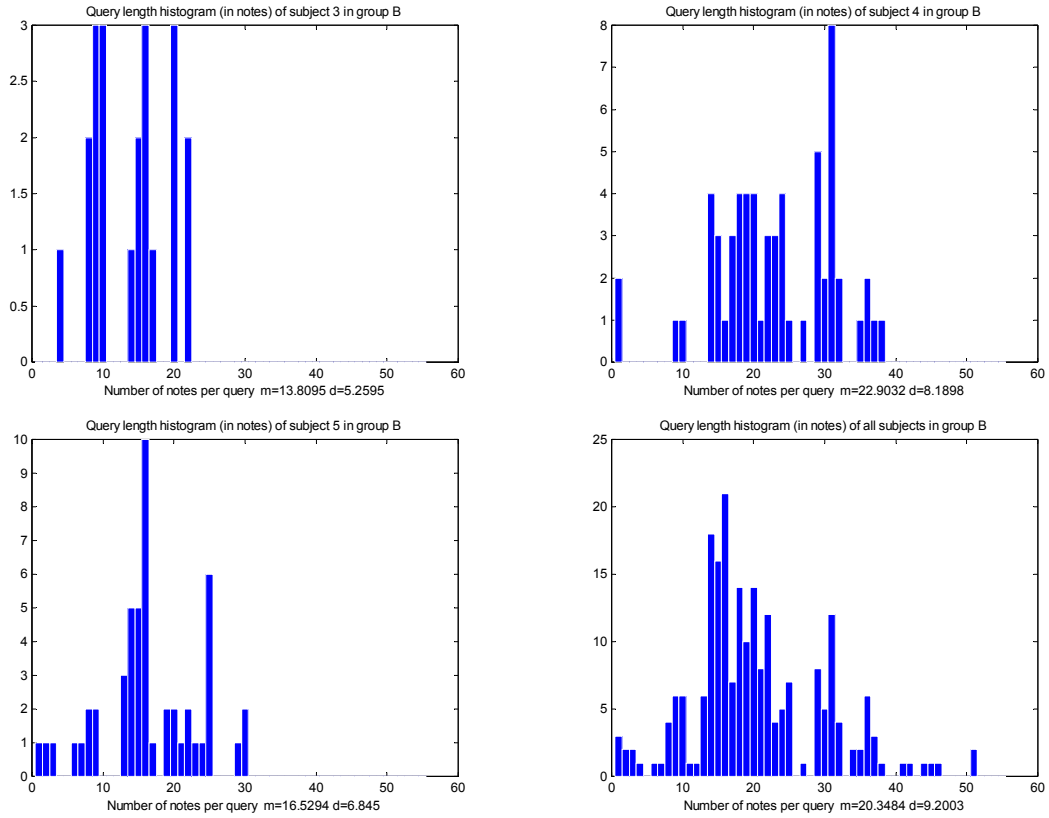
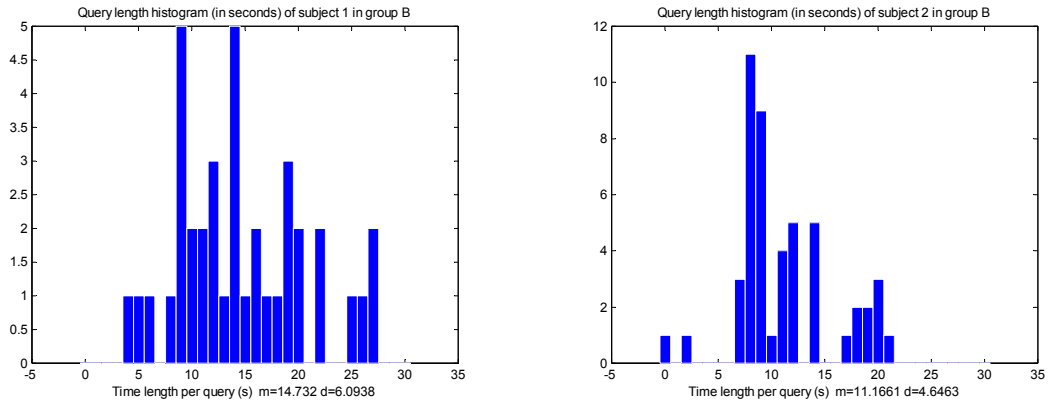


Figure 7-7: *Query length histograms of the hummed queries by subjects in Group B.*

Please note that the queries with no notes correspond to the situations that the subjects clicked the recording button but hummed nothing. The histograms show that a typical query contains ten to thirty and often less than twenty notes.

Query length histograms (seconds per query) of the hummed queries



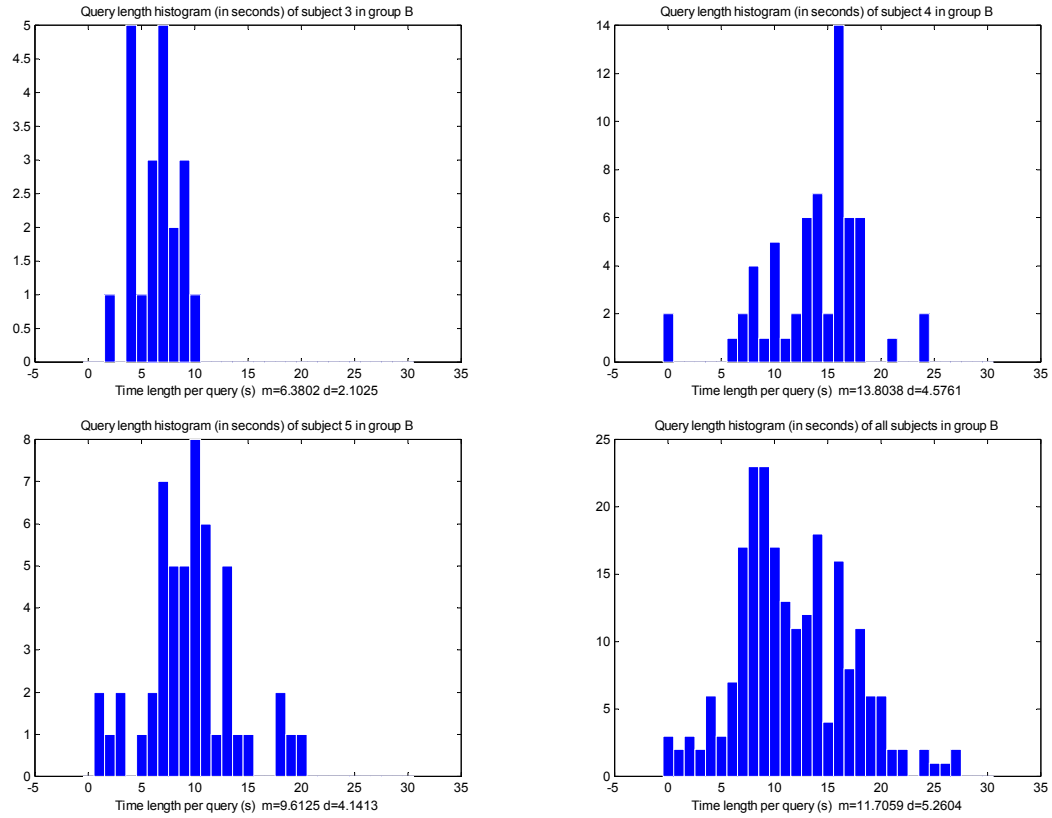
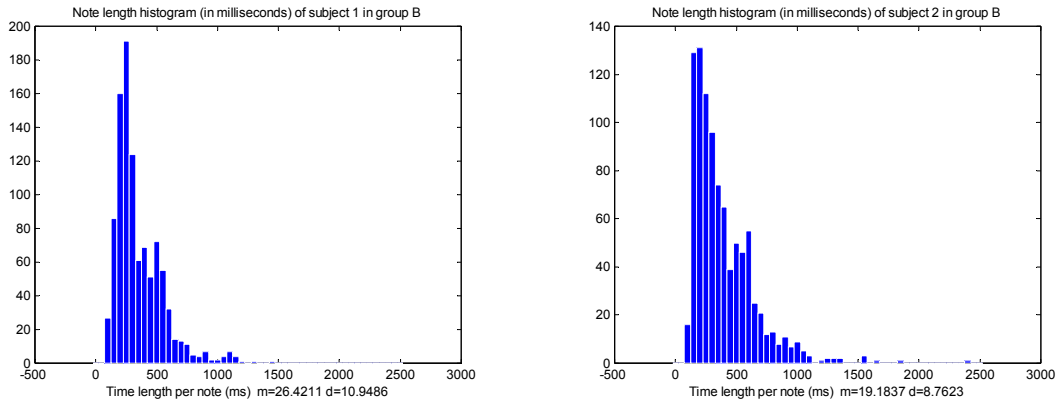


Figure 7-8: *Query length histograms of the hummed queries by subjects in Group B.*

Similar to the above, the queries with 0 second correspond to the situations that the subjects clicked the recording button but hummed nothing. The histograms show that a typical query is five to twenty and often about ten seconds long.

Note length histograms (milliseconds per note) of the hummed queries



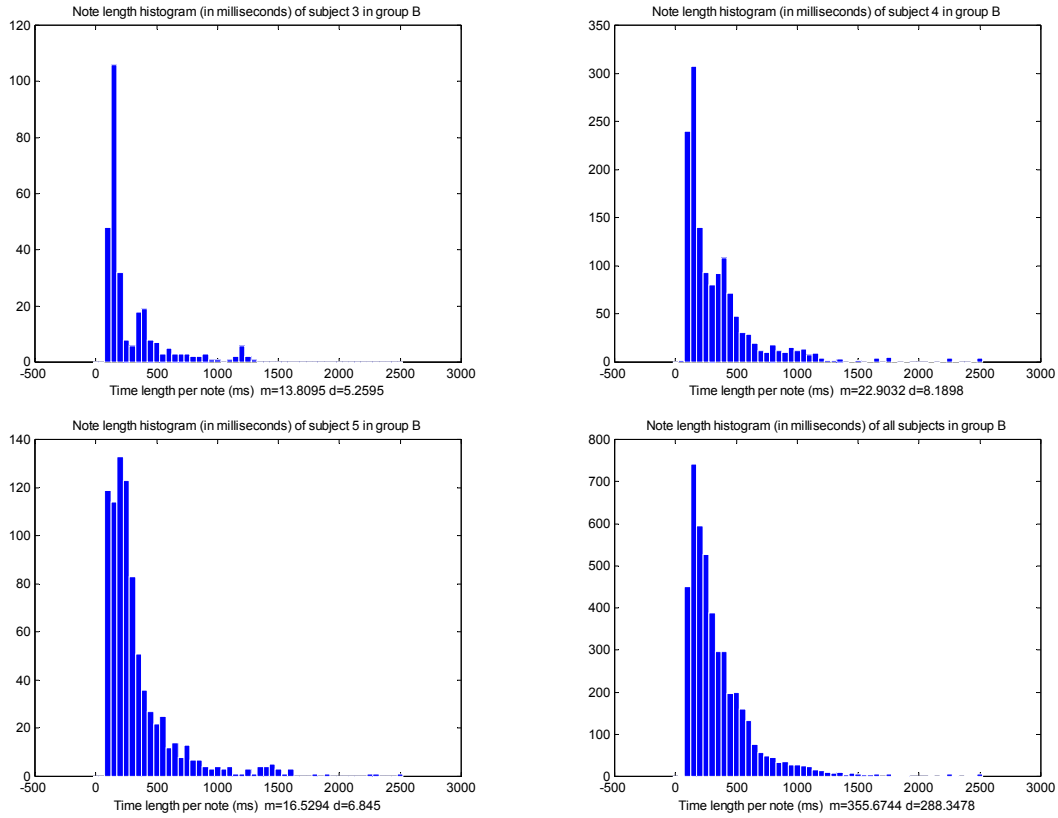
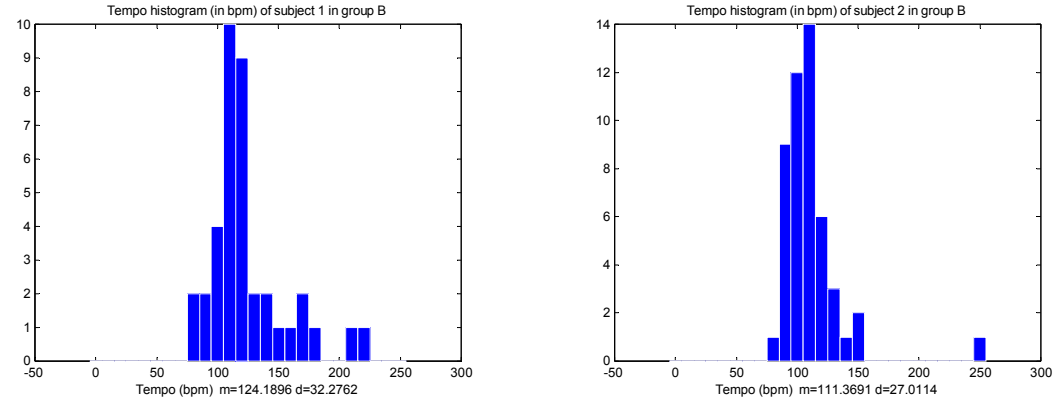


Figure 7-9: *Note length histograms of the hummed queries by subjects in Group B.*

Tempo histograms (beats per minute) of the hummed queries



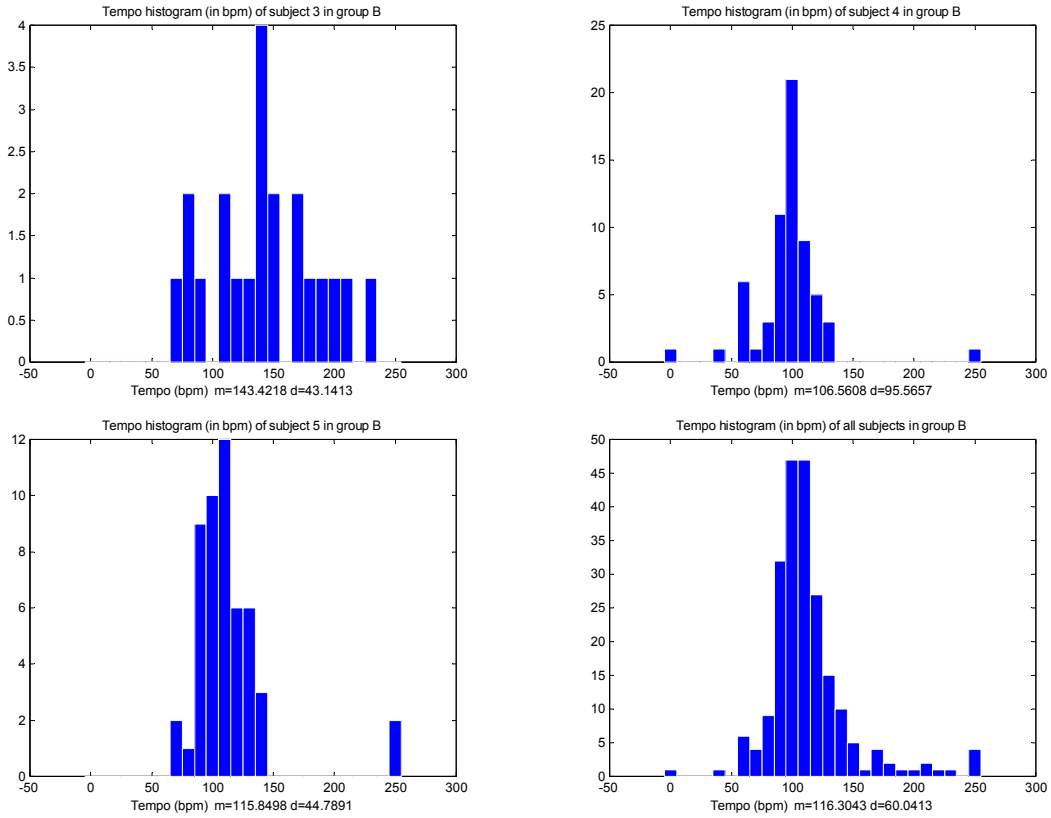


Figure 7-10: *Tempo histograms of the hummed queries by subjects in Group B.*

The histograms show that a typical query is as fast as about 100 bmp. The shape of the tempo histogram is roughly a Gaussian distribution, which is consistent with our tempo preference function (see Section 5.3.4).

7.2.4 Effectiveness of interfaces and algorithms

This section shows the effectiveness of our interfaces and algorithms based on both perfect queries and real queries. When using perfect queries, we use *margin* to evaluate the effectiveness of our methods incorporating rhythmic information compared with the most widely used method without rhythmic information (DPM matching method). When using real perfect queries, we use *first hit rate* to evaluate the effectiveness of our methods compared with the pitch-only method. Both measures indicate that our methods (especially *TPBM-I/INTF-I and TPBM-II/INTF-II*) outperform the previous method (*DPM*).

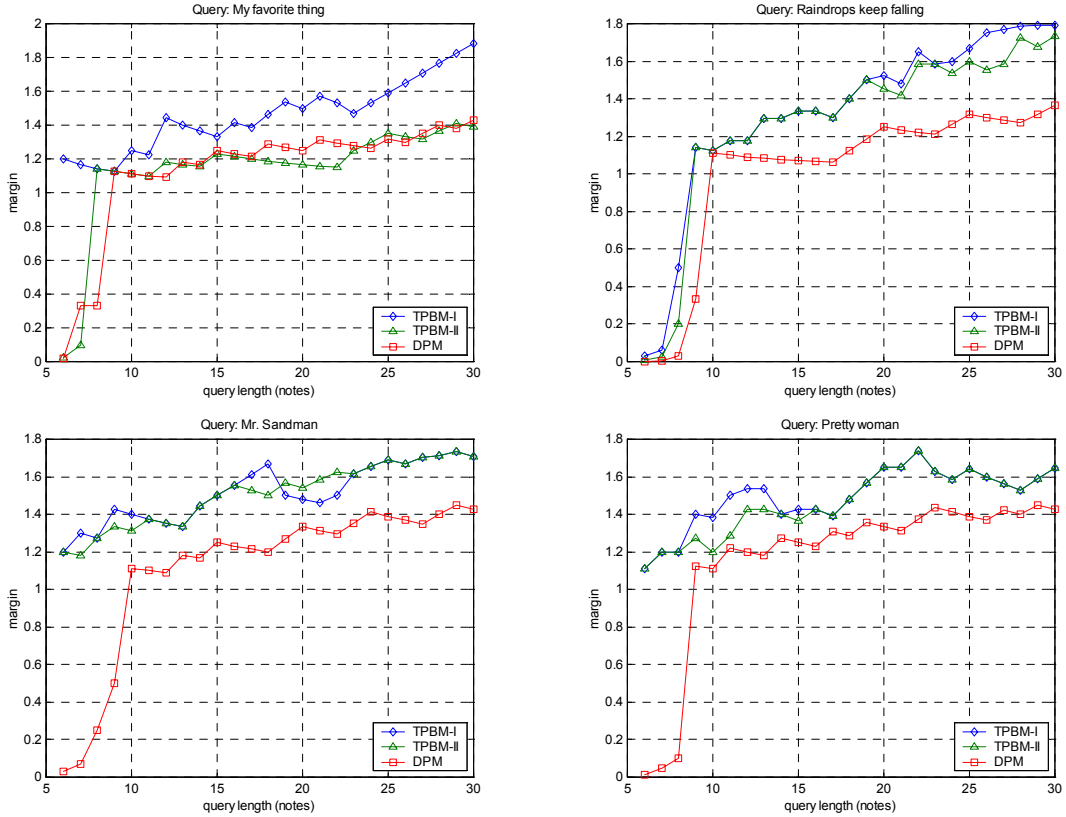
Using perfect queries

Similar to the method we used in Section 5.1.6, we first use perfect queries to test the precision of different representations and matching algorithms, but this time the queries were not randomly generated from any portion of any song, instead, they were actually the seventh queries by each subject in Group B, thus totally five queries corresponding to five different songs. The queries were truncated from the corresponding songs starting at the note the subjects first hummed and ending at different positions in the songs to construct perfect queries with various lengths. We can still use the number of songs with perfect match (matching score equals 1) as a measure, but after the query is sufficiently long, the precision will be 1 no matter which representation and matching method is used. Given that, we propose to use *Margin* to measure the precision.

$$Margin = \begin{cases} \frac{1}{n}, & n > 1 \\ \frac{1}{second\ highest\ score}, & n = 1 \end{cases} \qquad \text{(Eq. 7-1)}$$

where n is the number of songs with perfect match. Thus. If there are multiple songs matching the query perfectly, margin will be less than 1; otherwise, margin will be greater than 1. In either case, the greater the margin, the more likely the right song will be returned uniquely and, thus, the better performance.

The margin varying with the query length (notes per query) is shown in Figure 7-11. The performance of each perfect query is shown first and then the average performance is shown at last. Please note that although it is generally true that the margin increases when the query length increases, it does not increase monotonically.



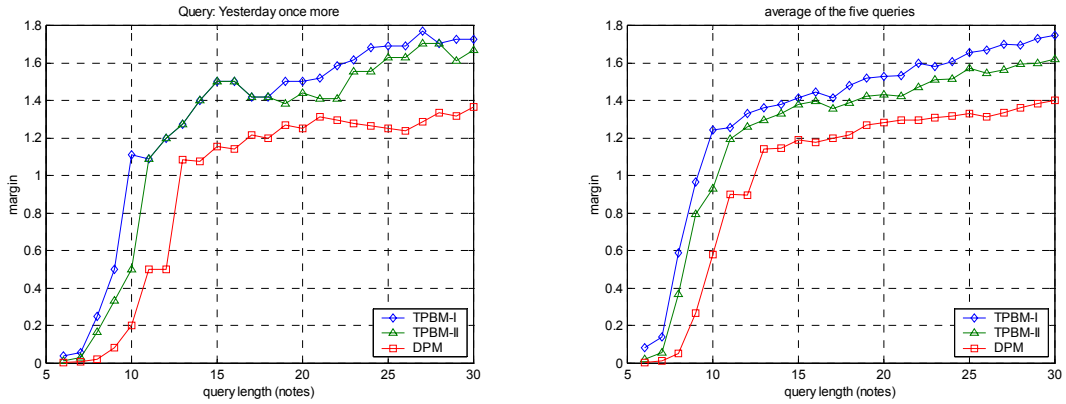


Figure 7-11: *Margin vs query length based on perfect queries.*

The experiment shows that on the average, TPBM-I outperforms TPBM-II, and TPBM-II outperforms DPM, which is the most widely used pitch-only method. And it is almost impossible to get unique perfect match (*precision=1*) without considering the rhythmic information if the query length is shorter than ten notes. We can imagine that the query will need to be much longer to get unique result if the melody database is ten or even hundred of times bigger, which would be true for a real commercial system. The users, on the other hand, would not like to hum too long to get the right song, especially sometimes they can memorize only very short phrases of the song (say, hooks or motives).

Using real queries

The margin using perfect queries is an important measure describing the effectiveness of a representation and matching method, but it is not sufficient, because we also need to consider the fault tolerance issue. For example, the exact matching method (*score=1* if the query and the song exactly match; otherwise *score=0*) can always get the maximum margin performance, but it is not fault tolerant at all. Therefore, we still need evaluate the system performance based on real queries. Reports to evaluate other existing query-by-humming systems in this way were never seen before.

Table 7-10 shows the statistics of query performance using different algorithms based on hummed queries by subjects in Group B. If there are multiple queries corresponding to one target song and the best result among the trails for the song is reported in the table. In the table, (n, m) indicates that the right song appeared n^{th} in the resulting songs sorted by matching scores from high to low and songs from the nth to the m^{th} got the same scores. So we want both n and m to be as small as possible and ideally *(1,1)*, indicating the right song and only the right song got the highest matching score, which is called *first hit*. *FHR (First Hit Ratio)* in this table means how many of the queried songs got first hits. If the right song did not show up in the top ten songs with highest matching scores, it is indicated by "-". Score errors (See Section 7.1.2) are indicated by *SCORE ERR*. The songs with score errors were not counted in computing FHR.

Table 7-10: *Query performance by subjects in Group B.*

		TPBM-I INTF-I	TPBM-II INTF-II	TPBM-II OAM-I	DPM	
Subject 1	1	(1,1)	(2,2)	(1,1)	(2,10)	
	2	(2,2)	(1,1)	-	(2,2)	
	3	(1,1)	(1,1)	(1,1)	(1,1)	
	4	(1,1)	(1,1)	(1,1)	(1,1)	
	5	(1,1)	(1,1)	(1,1)	(1,1)	
	6	(2,4)	-	-	-	
	7	(1,1)	(1,1)	-	(1,1)	
	FHR	71%	71%	57%	57%	
Subject 2	1	(1,1)	-	-	(1,1)	
	2	(1,1)	(1,1)	-	-	
	3	-	-	(1,1)	(1,1)	
	4	(1,1)	(1,1)	-	(1,1)	
	5	(1,1)	(1,1)	(1,1)	-	
	6	-	-	-	-	*SCORE ERR*
	7	(1,1)	(1,1)	(1,1)	(2,4)	
	8	(1,1)	-	-	-	
	9	-	-	-	-	
	10	-	-	-	-	
	11	(1,1)	-	(3,10)	-	
	FHR	70%	40%	30%	30%	
Subject 3	1	(1,1)	(1,5)	(8,10)	(7,10)	
	2	-	-	-	(1,4)	
	3	-	-	-	-	
	4	-	-	-	-	*SCORE ERR*
	5	(1,1)	-	-	-	
	6	(1,1)	-	-	(1,1)	
	7	-	-	-	-	
	FHR	50%	0	0	17%	
Subject 4	1	-	-	-	(1,1)	
	2	(1,1)	-	(1,3)	(1,1)	
	3	-	(1,1)	-	(1,1)	
	4	-	-	(1,1)	(1,1)	
	5	-	-	-	(1,1)	
	6	-	-	-	-	*SCORE ERR*
	7	-	-	(1,1)	(1,1)	
	FHR	17%	17%	33%	100%	
Subject 5	1	(1,2)	(1,1)	-	-	
	2	(1,1)	(1,1)	(1,1)	(1,1)	
	3	(2,3)	(1,1)	-	(1,1)	
	4	-	-	(1,2)	-	
	5	(8,8)	(1,1)	(2,6)	-	
	6	(1,1)	(1,1)	-	(1,1)	
	7	(1,1)	(1,1)	(1,1)	-	

8	(1,1)	(1,1)	(1,1)	(1,1)
9	(1,1)	(1,1)	(1,1)	(1,1)
10	-	-	-	(1,1)
11	(1,1)	(1,1)	(1,1)	-
12	(1,1)	(1,1)	-	-
13	(1,1)	(1,1)	(1,1)	(1,1)
	62%	85%	46%	54%

The performance shows that most subjects (subjects 1, 2, 3 and 5) did improve their query effectiveness using some method incorporating rhythmic information (TPBM-I/INTF-I or TPBM-II/INTF-II). The combination of TPBM-II and OAM did not perform as well as it was estimated (compare the beat tracking performance shown in Table 7-9), because sometimes even the beat tracking result is correct according to the query, it does not mean that the subject hummed the query in a rhythmic pattern same as the original song. This might have two reasons: some songs have different versions with different rhythmic patterns; the subjects could not memorize the rhythm of the song accurately.

Additionally, pitch tracking errors impact the performance of every combination.

Only one subject in Group B did better using pitch-only method, because this subject hummed the queries very accurately in pitch but always transformed the rhythms of the original songs.

Actually, from the experiments, we found the difference between musically trained and untrained people is significant in terms of long-term musical memory. For very familiar and frequently heard songs, e.g., Christmas songs, all users can hum very accurately both in pitch and in rhythm, while for songs that have not been heard for a long time, musically trained people usually still can hum very accurately, but musically untrained people hum with large transformation both in pitch and in rhythm.

We also found that by using different interfaces to obtain rhythmic information, the subjects may change the way they hummed. For example, some subjects tended to hum slower when they used INTF-II, but hum faster when they used the algorithmic beat tracking method.

According to the questionnaire from the subjects in Group A and Group B, the preferences of using rhythmic information for query and the methods for obtaining those rhythmic information are shown in Table 7-11 and Table 7-12.

Table 7-11: *Subjects' opinion with regard to using the rhythmic information for query. Question: "Did you feel it useful to add beat information for query?"*

	0 (useless)	1	2	3	4 (very useful)
The number of subjects who chose the corresponding scale	1	0	1	5	3

Table 7-12: *Subjects' preference regarding the interfaces for obtaining the rhythmic information. Question: "Which beat tracking interface do you prefer?"*

	INTF-I	INTF-II	OAM-I
The number of subjects who preferred the corresponding interface	4	1	5

Although INTF-I or INTF-II performed better than OAM-I, many users preferred the algorithmic beat tracking method for obtaining rhythmic information, because they wanted the interface for obtaining rhythmic information to be more natural and friendlier; in that case, automatic beat tracking is the best choice. Most users did not like INTF-II because they felt it hard to get used to the interface of indicating beats by clicking the mouse, although it is natural for people to clap hands along with music.

On the whole, most users felt the rhythmic information useful and experimental results demonstrate that rhythmic information did help improving the effectiveness. We found that properly incorporating the rhythmic information will not only narrow down the search, but also make the system more fault tolerant to the pitch related errors due to the users' singing skill and the deficiency of the pitch tracking algorithm.

7.3 Summary

This chapter presents our experiments and the results for evaluating the QBH system. Most subjects had no special singing training or professional music background. Normal instead of professional apparatus was used for recording and running the system. Thus, the experiments were designed to simulate the real-world task of searching for music on the Internet with a query-by-humming interface.

The statistics of the hummed queries, including histograms of pitches, intervals, query lengths and tempi, give us an overview of typical hummed queries. This information will be very helpful for adjusting parameters, refining algorithms or even developing personalized interfaces so as to improve the effectiveness of a query-by-humming system. Additionally, it gives us some observations about humans' melody perception process, e.g., what is the typical tempo, length or portion of a song the subjects hum as a query? How accurate can the subjects memorize and hum the melody in terms of both pitch and rhythm? Are there any significant differences between professionals and amateurs?

We also give statistics of the effectiveness of our algorithms, the interfaces and the system as a whole. Two measures are proposed to evaluate the performance of a query-by-humming system: *margin* based on perfect queries and *first hit rate* based on real queries. We expect greater margin and first hit rate, which corresponds to better precision and recall (concepts mostly used in text-retrieval systems). Experimental results show that both of the measures improve significantly using our methods incorporating rhythmic information compared with previous pitch-only method (refer to Figure 7-11 and Table 7-10). Therefore, for most users using our methods will improve their chance to find the songs that they search for, even if they cannot sing perfectly. The experiments based on real queries as in this thesis were not reported before.

CHAPTER 8 CONCLUSIONS

This thesis has explored the melody retrieval problem from two perspectives: as a practical solution to a query-by-humming system, and as a scientific inquiry into the nature of the melody perception process.

A new melody representation, which combines both pitch and rhythmic information, and new approximate melody matching algorithms were proposed and shown to be more effective than previous pitch-only methods. A set of practical and real-time automatic transcription techniques customized for the query-by-humming system to obtain both pitch and rhythmic information were adopted in the system. A useful tool to build a melody database from various symbolic representations including score formats and MIDI format was developed. A deliverable query-by-humming system including both the server application and the client application has been built and is accessible online.

In our experiment, the melody representation and the melody matching methods with rhythmic information outperformed the method without rhythmic information for most users. This result demonstrates that incorporating rhythmic information properly can significantly improve the effectiveness of query-by-humming systems. Meanwhile, users want the interface for obtaining rhythmic information to be more natural and friendlier; in that case, automatic beat tracking is the best choice. Although algorithmic beat tracking proposed in this thesis for automatically obtaining rhythmic information from hummed queries did not perform as well as INTF-I or INTF-II, its accuracy is still promising and at least it should work very well for musically trained users.

We also have other interesting findings from our experiments, though the quantitative statistics are not shown in the thesis. For example, the difference between musically trained and untrained people is significant in terms of long-term musical memory. For very familiar and frequently heard songs, e.g., Christmas songs, all users can hum very accurately both in pitch and in rhythm, while for songs that have not been heard for a long time, musically trained people usually still can hum very accurately, but musically untrained people hum with large transformation both in pitch and in rhythm. These findings suggest to us that a query-by-humming system may use different matching methods customized to different users according to their music background and singing skill to achieve best performance.

Finally, I would like to answer the question asked by one subject, “When will the system be available in music stores so you can easily find the music you have in mind?” To make query-by-humming systems commercialized and practical, the main difficulty is how we can extract and tag the melodies of existing songs in waveform, which may be located anywhere over the web, and build efficient indices pointing to these songs based on the melodic information. While automatically extracting melodies from songs in waveform is still too hard, we can manually tag the melodies of songs by music providers and/or record companies using some standard format, so that search engines can easily understand the melody description information and use the techniques described in this thesis to find music for the users. Actually, the standardization task is in process by the MPEG-7 committee; I sincerely hope commercial systems will show up and serve all music lovers very soon.

References

Brown, Judith C. “Determination of the meter of musical scores by autocorrelation”. J. Acoust. Soc. Am. 94:4, Oct. 1993.

Chai, Wei and Vercoe, Barry. “Using user models in music information retrieval systems.” Proc. International Symposium on Music Information Retrieval, Oct. 2000.

Chai, Wei and Vercoe, Barry. “Folk music classification using hidden Markov models.” Accepted by International Conference on Artificial Intelligence, June 2001.

Chen, J. C. C. and Chen, A. L. P. “Query by rhythm: an approach for song retrieval in music databases” Proc. Eighth International Workshop on Reasearch Issues In Data Engineering, 1998.

Chou, T. C.; Chen, A. L. P. and Liu, C. C. “Music database: indexing techniques and implementation.” Proc. International Workshop on Multimedia Database Management Systems, 1996.

Chu, Stephen and Logan, Beth. “Music Summary Using Key Phrases”. CRL Technical Report, 2000.

Dannenberg, Roger B., Thom, Belinda, and Watson, David. “A Machine Learning Approach to Musical Style Recognition”. International Computer Music Conference, 1997, pp. 344-347.

Dowling, W. J. “Scale and contour: Two components of a theory of memory for melodies.” Psychological Review, vol. 85, no. 4, pp. 341-354, 1978.

Dowling, W. J. and Harwood, D.L. *Music Cognition.* Academic Press, 1986.

Edworthy, Judy. “Melodic contour and musical structure.” *Musical Structure and Cognition.* Academic Press, 1985.

Foote, Jonathan. “Methods for the automatic analysis of music and audio”. FXPAL Technical Report FXPAL-TR-99-038.

Foote, Jonathan. “An overview of audio information retrieval”. In Multimedia Systems, vol. 7 no. 1, pp. 2-11, ACM Press/Springer-Verlag, January 1999.

Ghias, A.; Logan, J.; Chamberlin, D. and Smith, B. C. “query by Humming: musical information retrieval in an audio database.” Proc. ACM Multimedia, San Francisco, 1995.

Goto, Masataka and Muraoka, Yoichi. “A beat tracking system for acoustic signals of music.” ACM Multimedia 94 Proceedings (Second ACM International Conference on Multimedia), pp.365-372, October 1994.

Goto, Masataka and Muraoka, Yoichi. “Real-time Rhythm Tracking for Drumless Audio Signals -- Chord Change Detection for Musical Decisions.” Working Notes of the IJCAI-97 Workshop on Computational Auditory Scene Analysis, pp.135-144, August 1997.

Handel, S. *Listening*. Cambridge, MA: MIT Press, 1989.

Hewlett, Walter B. and Selfridge-field, Eleanor. *Melodic Similarity: Concepts, Procedures, and Applications.* MIT Press, 1998.

Howell, Peter; Cross, Ian and West, Robert. Musical Structure and Cognition. Academic Press, 1985.

Huron, David. "Perceptual and cognitive applications in music information retrieval." International Symposium on Music Information Retrieval, October 23-25, 2000.

Juang, B. and Rabiner, L. "*A Probabilistic Distance Measure for Hidden Markov Models"*. The Bell System Technical Journal, vol. 64, pp. 391—408, 1985.

Kerman, Joseph. *Listen*. Brief 4th ed., c2000.

Kim, Youngmoo; Chai, Wei; Garcia, Ricardo and Vercoe, Barry. "Analysis of a contour-based representation for melody". Proc. International Symposium on Music Information Retrieval, Oct. 2000.

Kosugi, N.; Nishihara, Y.; Kon'ya, S.; Yamamuro, M. and Kushima, K. "Music retrieval by humming-using similarity retrieval over high dimensional feature vector space." Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1999.

Lee, C.S. "The rhythmic interpretation of simple musical sequences: towards a perceptual model." *Musical Structure and Cognition.* Academic Press, 1985.

Levitin, D. J. "Memory for Musical Attributes" from Music, Cognition, and computerized Sound, ed. Perry R. Cook. Cambridge, MA: MIT Press, 1999, 214-215.

Lindsay, A. T. Using contour as a mid-level representation of melody. MS thesis. MIT Media Lab, 1996.

Liu, C. C.; Hsu, J. L. and Chen, A. L. P. "Efficient theme and non-trivial repeating pattern discovering in music databases", Proc. 15th International Conference on Data Engineering, 1999.

Maidín, Donncha Ó. "Common practice notation view users' manual". Technical Report UL-CSIS-98-02, University of Limerick.

Martin, Keith D. *Sound-Source Recognition -- A Theory and Computational Model*. PhD dissertation. MIT Media Lab, 1999.

McNab, R. J.; Smith, L. A.; Witten, I. H.; Henderson, C. L. and Sunningham, S. J. "Toward the digital music library: tune retrieval from acoustic input." Proc. ACM Digital Libraries, Bethesda, 1996.

MiDiLiB, University of Bonn, http://leon.cs.uni-bonn.de/forschungprojekte/midilib/english.

Nettl, Bruno. *Folk and Traditioanl Music of the Western Continents.* 2d ed. Prentice-Hall, 1973.

Pollastri, E. “Melody-retrieval based on pitch-tracking and string-matching methods.” Proc. Colloquium on Musical Informatics, Gorizia, 1998 .

Pye, David. “Content-based methods for the management of digital music.” International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2000.

Rabiner, L. R.; Cheng, M. J.; Rosenberg, A. E. and McGonegal, C. A. “A comparative performance study of several pitch detection algorithms”, IEEE Trans. on Acoustics, Speech and Signal Processing, vol. ASSP-24, no.5, 1976, 399-418.

Rabiner, Lawrence R. “A tutorial on hidden Markov models and selected applications in speech recognition”. In Proc. of the IEEE Volume: 77 2 , Feb. 1989 , Page(s): 257-286.

Roads, Curtis. *The Computer Music Tutorial*. Cambridge, MA: MIT Press, c1994.

Scheirer, E. D. *Music-Listening Systems.* PhD dissertation. MIT Media Lab, 2000.

Scheirer, Eric D. “Tempo and beat analysis of acoustic musical signals.” J. Acoust. Soc. Am. 103:1, pp 588-601. Jan 1998.

Selfridge-Field, Eleanor. “Conceptual and representational issues in melodic comparison”. In Melodic Similarity, Concepts, Procedures, and Applications, MIT Press, 1998.

Sloboda, John A. and Parker, David H. H. “Immediate Recall of Melodies.” *Musical Structure and Cognition.* Academic Press, 1985.

Talupur, Muralidhar; Nath, Suman and Yan, Hong. “Classification of music genre.” Technical Report, 2001.

Themefinder, Stanford University, http://www.ccarh.org/themefinder.

Tseng, Y. H. “Content-based retrieval for music collections.” Proc. Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999.

TuneServer, University of Karlsruhe, http://wwwipd.ira.uka.de/tuneserver.

Tzanetakis, George and Cook, Perry. “A framework for audio analysis based on classification and temporal segmentation”. In Proc. Euromicro, Workshop on Music Technology and Audio processing, Milan, September 1999.

Tzanetakis, George and Cook, Perry. “Audio Information Retrieval (AIR) Tools”. In Proc. International Symposium on Music Information Retrieval, Oct. 2000.

Uitdenbogerd, A. L.and Zobel, J. “Manipulation of music for melody matching.” Proc. ACM International Conference on Multimedia, 1998.

Uitdenbogerd, Alexandra and Zobel, Justin. “Melodic Matching Techniques for Large Music Databases.” Proc. ACM International Conference on Multimedia, 1999.

Watkins, Anthony J. and Dyson, Mary C. “On the perceptual organization of tone sequences and melodies.” *Musical Structure and Cognition.* Academic Press, 1985.

Wold, E.; Blum, T.; Keislar, D. and Wheaten, J. “Content-based classification, search, and retrieval of audio.” IEEE Multimedia Volume: 33, Fall 1996, Page(s): 27 –36.