國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

歌聲檢索系統：改良式發端識別以及修正式旋律比對

Querying By Humming System：Improved Onset Detection and Modified Melody Matching

胡哲銘

Che-Ming Hu

指導教授：丁建均 博士

Advisor: Jian-Jiun Ding, Ph.D.

中華民國 99 年 6 月

June, 2010

國立臺灣大學（碩）博士學位論文
口試委員會審定書

歌聲檢索系統：改良式發端識別以及修正式旋律比對
Querying By Humming System：Improved Onset Detection and Modified Melody Matching

本論文係胡哲銘君（學號 R97942093）在國立臺灣大學電信工程學研究所，所完成之碩士學位論文，於民國 99 年 5 月 22 日承下列考試委員審查通過及口試及格，特此證明

口試委員：丁建均（簽名）

（指導教授）

王鵬華

葉敏宏　許文良

系主任、所長 陳光禎（簽名）

誌謝

能完成這篇論文，我要特別感謝我的指導教授丁建均老師，老師不僅提供我們研究的方向，也非常用心跟我們討論每一次的 meeting 給予我們意見，並且常常與我們聚餐，不只在課業上的指導，也給予我們求學與生活上的經驗，也關心我們的生活，在數位訊號處理實驗室過了兩年生活當中，我過了一個很充實也很滿足的生活，在求學知識上我學到了許多，與實驗室同學們也相處地相當融洽。

感謝畢業的學長們，在我碩一的時候，幫忙我許多，讓我能快速學得作研究的技巧以及看 paper 的能力，更為實驗室留下許多的研究資源，不只是對我也對之後的學弟們都非常有幫助，特別感謝阿德架了實驗室網站，也常常回實驗室幫忙大小事務。

感謝與我一起努力、研究、生活的同學們，在這研究生活的兩年當中，並不覺得研究生是個有壓力的生活，反倒是互相討論的過程中激發學習的動力，特別感謝維毅以及偉崙學弟，感謝維毅教了我許多程式的技巧以及課業的問題，也感謝偉崙激發了我許多研究上的想法。另外還感謝科傑、保言、文琦、維毅一起修課時互相討論以及研究上互相關照。感謝實驗室同學為我提供聲音檔供作實驗模擬的訊號。

另外感謝信慧學長和碩一學弟，雖然才短短相處一年，但是其間我們常常一起去運動，聚餐，唱歌，看電影等等，也培養出深厚的感情。實驗室氣氛很棒，這是由大家共同營造出來的，我很高興我能從這實驗室畢業，大家一起去溪頭參加的 CVGIP 研討會也是個珍貴的回憶與經驗。

最後，我最感謝的是我的爸爸媽媽以及我的哥哥姐姐們，你們鼓勵我求學向上，使我不用擔心生活上的經濟問題，能專心的作研究，因為有你們的支持，我才能完成碩士學位。我親愛的家人，謝謝你們。

中文摘要

近幾年, 音樂檢索的技術開始被深入的研究。傳統上，我們使用歌名或歌手等以文字為基礎的方式來進行音樂檢索。然而，假如我們忘記歌名跟歌手等文字線索那要怎麼辦呢？因此，接下來我們將介紹一個新穎的觀念，利用歌曲中的旋律內容來進行檢索。哼唱詢問(QBH)是人們跟電腦透過網際網路進行音樂搜尋的互動式觀念，而在一個龐大的音樂資料庫底下，此搜尋結果需要快速的搜尋時間。

大致上來說，我們嘗試著取出一段哼唱旋律的音頻並且比較其音頻在音樂資料庫的最大相似度。在資料庫中，高相似度的曲目將會被列出並且依照相似度由高往低排列。這篇論文將會介紹發端識別，音頻偵測，以及旋律比對的演算法。然而，由人們產生的歌聲總是不完美的，因此，仍然有許多困難之處留待我們去解決和改善。每一個人有不同的演唱方式且會導致各種不同型態的聲音波形，像是音調的高低，音準的準確性，尾音哼唱方式，演唱方式等等..，而這些因素也造成發端識別上的困難。

因此，在這篇論文中，我們主要提出了三種改善哼唱詢問(QBH)的方式，一種是針對發端識別做改善，而另外兩種則是針對旋律比對做改善，此外，在音頻偵測的方面，我們也採用了自己的方式。

最後，未來的研究應該更致力於改善哼唱詢問的效能，使它針對不同演唱風格的情況下能更具可適性以及信賴性。另外，在進行旋律比對方面上，我們要求更快的搜尋時間以及較低的複雜度，特別是在一個龐大的音樂資料庫底下。

關鍵字：哼唱詢問、發端識別、音頻偵測、旋律比對。

ABSTRACT

Music retrieval techniques have been investigated in recent years. Typically, we use the names of singers or songs as retrievals. However, how do we search songs when we forget the names of singers and songs? Hereunder, we will introduce a novel concept for music retrieval searching by using any melodic passage of a song. ‘Querying by humming’ (**QBH**) is a interaction concept for people to interact with computer through internet, and the searching results are revealed fast and orderly by comparing the sung input with a large database of known songs.

Generally speaking, we try to extract a series of the pitches form the humming input by a single individual, and compare these pitches with pitch interval of the known musical database. Melodies (theme) of the database are similar to the sung input are retrieved and listed orderly depending on its similarity score. This paper will present the algorithm for note onset detection (event detection), pitch detection, pitches quantization, melody encoding and melody matching (similarity matching or pattern matching).

However, Human reproduction of melodies is always imperfect, therefore, there are many difficulties for us to overcome and improve. Every individual has different singing style that results as a variety of patterns of sung inputs that make note onset detection become more difficult. Besides, the accuracy rate of onset detection also influences the performance of melody matching.

Therefore, in this thesis, we mainly proposed three methods to improve the QBH system, one is for improving onset detection and other two are for improving melody matching. Besides, we use our own method to extract the fundamental frequency.

For that, future research should make an effort to improve event detection, making it more adaptive and reliable to deal with various situations. Furthermore, research on measuring similarity between melodies in database and sung theme need to be further pursued to reduce the computation time and complexity while the amounts of musical database have been exploding nowadays.

Index Terms - Querying by humming, Onset detection, Pitch detection, melody matching.

CONTENTS

口試委員會審定書 .....#

誌謝 v

中文摘要 vii

ABSTRACT ix

CONTENTS xi

LIST OF FIGURES xv

LIST OF TABLES xix

Chapter 1 Introduction.....1

1.1 Background.....1

1.2 Primary Achievement of This Thesis.....2

1.3 Chapter Organization.....4

1.4 Abbreviations.....6

Chapter 2 Basic Musical Characteristics.....7

Chapter 3 Onset Detection 11

3.1 Magnitude Method 13

3.2 Short-Term Energy Method.....16

3.2.1 Implementation of Short-Term Energy Method.....17

3.2.2 Discussion of Short-Term Energy Method.....19

3.3 Surf Method.....20

3.3.1 Implementation of Surf Method.....21

3.3.2 Discussion of Surf Method.....23

3.4 High Frequency Content (HFC) method 24

3.4.1 Implementation of HFC Method 25
3.4.2 Discussion of HFC Method 26
3.5 Pitch method 27
3.6 Discussion between These Onset Detections 29
Chapter 4 Pitch Estimation 33
4.1 Autocorrelation Function for Pitch Tracking 33
4.2 Modified Hilbert-Huang Transform 34
4.2.1 The Over View of HHT 35
4.2.2 Music Pitch Analysis with Modified HHT 37
4.3 Sub-Harmonic Summation 38
4.4 Harmonic Elimination Method 39
4.4.1 Elimination of Harmonics 40
4.4.2 Extended Kalman Filter (EKF) 41
Chapter 5 Melody Matching 43
5.1 Dynamic Programming Algorithm 43
5.1.1 Implementation of Dynamic Programming Algorithm 44
5.1.2 Complexity of Dynamic Programming 46
5.2 Hidden Markov Model 47
5.2.1 Implementation of Hidden Markov Models 49
5.2.2 Complexity of Hidden Markov Model 54
Chapter 6 Representations for Musical Feature 57
6.1 Music Scale Modeling for Melody Matching 58
6.2 Pitch Interval and IOI for Melody Matching 61
6.3 Construction of Music Database 63
Chapter 7 Proposed Algorithm 65

7.1 Proposed Onset Detection 69
7.2 Proposed Pitch Estimation 75
7.3 Proposed Melody Matching 78
7.3.1 Modified Dynamic Programming 78
7.3.2 Modified Hidden Markov Model 84
Chapter 8 Experimental Result 87
8.1 Performance Standard 87
8.2 Comparison of Onset Detection Performance 88
8.3 Comparison of Melody Matching Performance 91
8.4 Comparison of Whole QBH System Performance 92
Chapter 9 Conclusion and Future Work 97
9.1 Conclusions 97
9.2 Future Work 98
REFERENCE 101

LIST OF FIGURES

Fig. 1.1 The system diagram of 'Query-By-Humming' system. 2

Fig. 2.1 The fundamental frequency and its fundamental frequency. 8

Fig. 2.2 The illustration for notes of a piece of music "兩隻老虎". 9

Fig. 2.3 "Attack", "transient", "decay", and "onset" in the ideal case of a single note. ...9

Fig. 3.1 The original waveform produced by a male person singing a passage of melody of the song "愛情釀的酒"of 12 notes. 11

Fig. 3.2 The real onset times are denoted by red circles and the numbers in circles mean the n-th notes in the signal. 12

Fig. 3.3 The system block diagram of onset detection. 13

Fig. 3.4 A simple illustration for the algorithm of magnitude method. 14

Fig. 3.5 The onset detection of magnitude method for the song "愛情釀的酒"..... 15

Fig. 3.6 The process of calculating energy by a sliding window. 16

Fig. 3.7 The pre-process of extracting the absolute value of the signal in Fig. 3.1. 17

Fig. 3.8 The envelope of signal in Fig. 3.7. 18

Fig. 3.9 The short-term energy detection result only detects 10 notes and misses two circled parts. 19

Fig. 3.10 The short-term energy detection result detects 14 notes and over detects two more circled parts. 20

Fig. 3.11. The slope of center of second-order polynomial. 21

Fig. 3.12 The slope of each frame of envelope $y(k)$ of the original signal. 23

Fig. 3.13 The surf detection result detects 14 notes and over detects two more circled parts. 24

Fig. 3.14 The HFC detection result detects 19 notes and over detects 7 more circled parts. 26
Fig. 3.15 The time-frequency distribution of input signal. 27
Fig. 3.16 The time-MIDI distribution of input signal. 28
Fig. 3.17 The time-MIDI distribution of input signal after onset detection. 29
Fig. 3.18 Comparison between these onset detections for the query “兩隻老虎”. 30
Fig. 4.1 The waveforms of input signal and its autocorrelation. 34
Fig. 4.2 The modified HHT process for fundamental frequency estimation. 38
Fig. 4.3 IMFs of C4 (261Hz) by shifting (a) without and (b) with the filterbank pre-processing [14]. 38
Fig. 4.4 System block diagram for harmonics elimination. 40
Fig. 5.1 Four maximal-scoring alignments. 46
Fig. 5.2 Features for a passage of a piece of music. 48
Fig. 5.3 Markov Model for the case from Fig. 5.2. 49
Fig. 5.4 Markov model (MM) for string $\{a,b,c,c,d\}$ 51
Fig. 5.5 Hidden Markov model (HMM) for string $\{a,b,c,c,d\}$ 52
Fig. 6.1 Pitch intervals in Major scale and Minor scale. 58
Fig. 6.2 A music scale model for Major and Minor scale. 59
Fig. 6.3 A time series of pitch of the same song produced by a male and a female. 61
Fig. 6.4 A time series of pitch interval of the same song produced by a male and a female. 62
Fig. 6.5 The numbered notes of the Chinese children’s song “兩隻老虎”. 63
Fig. 7.1 Over detected result in magnitude method. 65
Fig. 7.2 Under-detected result in short-term energy method. 66

Fig. 7.3 Misjudge in HFC method.....67
Fig. 7.4 Misjudge in Surf method.....68
Fig. 7.5 Time-frequency distribution in pitch method.....68
Fig. 7.6 The system block of our proposed onset detection. 69
Fig. 7.7 Envelope of absolute value of input signal. 70
Fig. 7.8 Assumed attacking signal.....71
Fig. 7.9 The matched filter of signal in Fig. 7.8.....71
Fig. 7.10 The envelope signal after normalizing.....72
Fig. 7.11 The power of 0.6 after normalizing.....73
Fig. 7.12 After convolution with the matched filter. 73
Fig. 7.13 the onset detection of proposed method.....74
Fig. 7.14 The autocorrelation function of a compound signal composed by a 2-Hz and a 5-Hz signal.....75
Fig. 7.15 After our proposed onset detection 76
Fig. 7.16 The harmonics of a sound produced by a male.....77
Fig. 7.17 The process of extracting the fundamental frequency.....77
Fig. 7.18 The kernel of comparing the max value.....79
Fig. 7.19 The priority of cell A, B and C.....81
Fig. 7.20 Alignment of target and query.....82
Fig. 7.21 The transition probability model from target 85
Fig. 7.22 The modeling process of HMM and MHMM.....85
Fig. 8.1 The performance of all onset detection methods for query “愛情釀的酒” which has 12 actual notes.....88

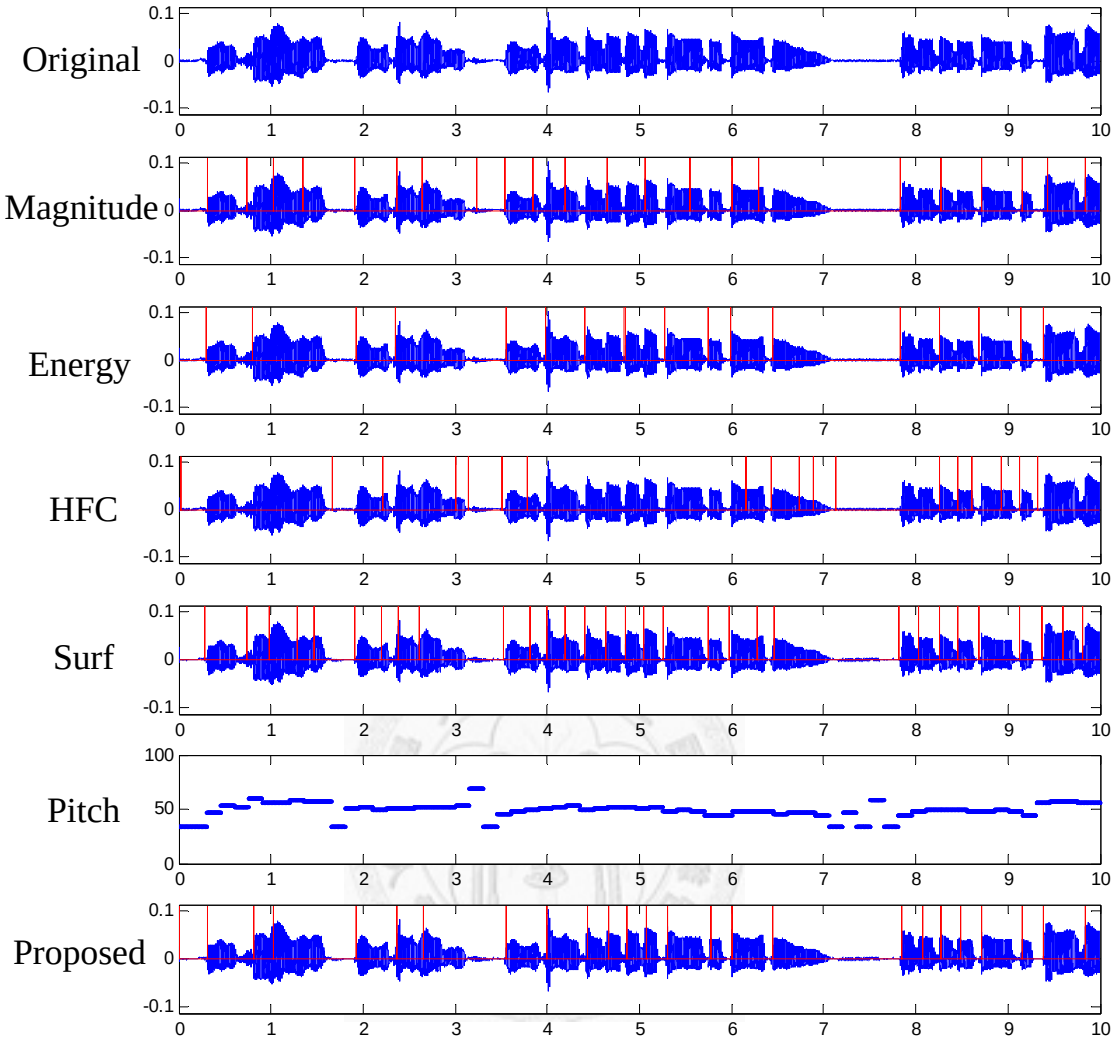


Fig. 8.2 The performance of all onset detection methods for query “挪威的森林” which has 24 actual notes. 89

LIST OF TABLES

Table 2.1 The relationship of fundamental frequency and MIDI number of notes in 3 consecutive octaves (from Oc3 to Oc5).....10

Table 3.1 Four different estimation situations and error type.....31

Table 3.2 Performance of known onset detection methods for 3 cases and each of them has 26 queries. 32

Table 5.1 Alignment matrix. 46

Table 5.2 Observation probability table. 53

Table 5.3 The result of realigning for Table 5.2. 53

Table 5.4 The result of normalizing for Table 5.3. 54

Table 6.1 Error fitting. 60

Table 6.2 The temporary number look-up table for numbered notes. 64

Table 7.1 the pitch interval matrix of target and query. 79

Table 7.2 The alignment matrix after function in (7.3). 80

Table 7.3 The routing of the proposed melody matching. 81

Table 8.1 Performance of all kinds of onset detection methods for 3 cases and each of them has 26 queries. 90

Table 8.2 MRRR and HR of 26 queries for 3 versions of length from 50 targets by using DP algorithm. 91

Table 8.3 The computation time of these four melody matching methods by using our proposed onset detection. 92

Table 8.4 Performance of 26 queries which are 10-second long from 50 targets. 93

Table 8.5 Performance of 26 queries which are 15-second long from 50 targets. 94

Table 8.6 Performance of 26 queries which are 20-second long from 50 targets. 95

Chapter 1 Introduction

1.1 Background

People always hum a melodic passage of a song without knowing its name of singer and song. It is inconvenient for user who doesn't know the title or the composer of the song to search the desired song. 'Querying-By-Humming' (**QBH**) [1][2][3] is an interaction concept, making it possible to retrieve a song when the user hums a familiar tune and in the mean while the searching system seeks the possible songs fast and orderly in a very short time.

The current algorithm for QBH system mainly includes three most significant parts: onset detection, pitch detection and melody matching. It's a linked combination of several techniques and requires sufficient backgrounds for basic music theory. The system organization is illustrated in Fig. 1.1 as a guide in this paper. The QBH system requires symbolic representations of the sung melodies which consist of a sequence of musical notes (e.g., the pitch names), the time onset of each note and sometimes may use time duration of each note as an additional feature.

The known onset detection [4][5][6][7][8] methods are magnitude method, short-term energy method, surf method, high-frequency content method and pitch method. The known pitch tracking [9][10][11][12][13][14][15][16][17][18] methods are sub-harmonic summation (**SHS**), autocorrelation function (**AFC**) pitch tracking and modified Hilbert-Huang transform (**MHHT)**. The known melody matching [20] methods are dynamic programming (**DP**) algorithm which is used to align DNA sequence originally and hidden Markov model method (**HMM**). We will present how these methods work in Chapter 3, Chapter 4, Chapter 5, respectively.

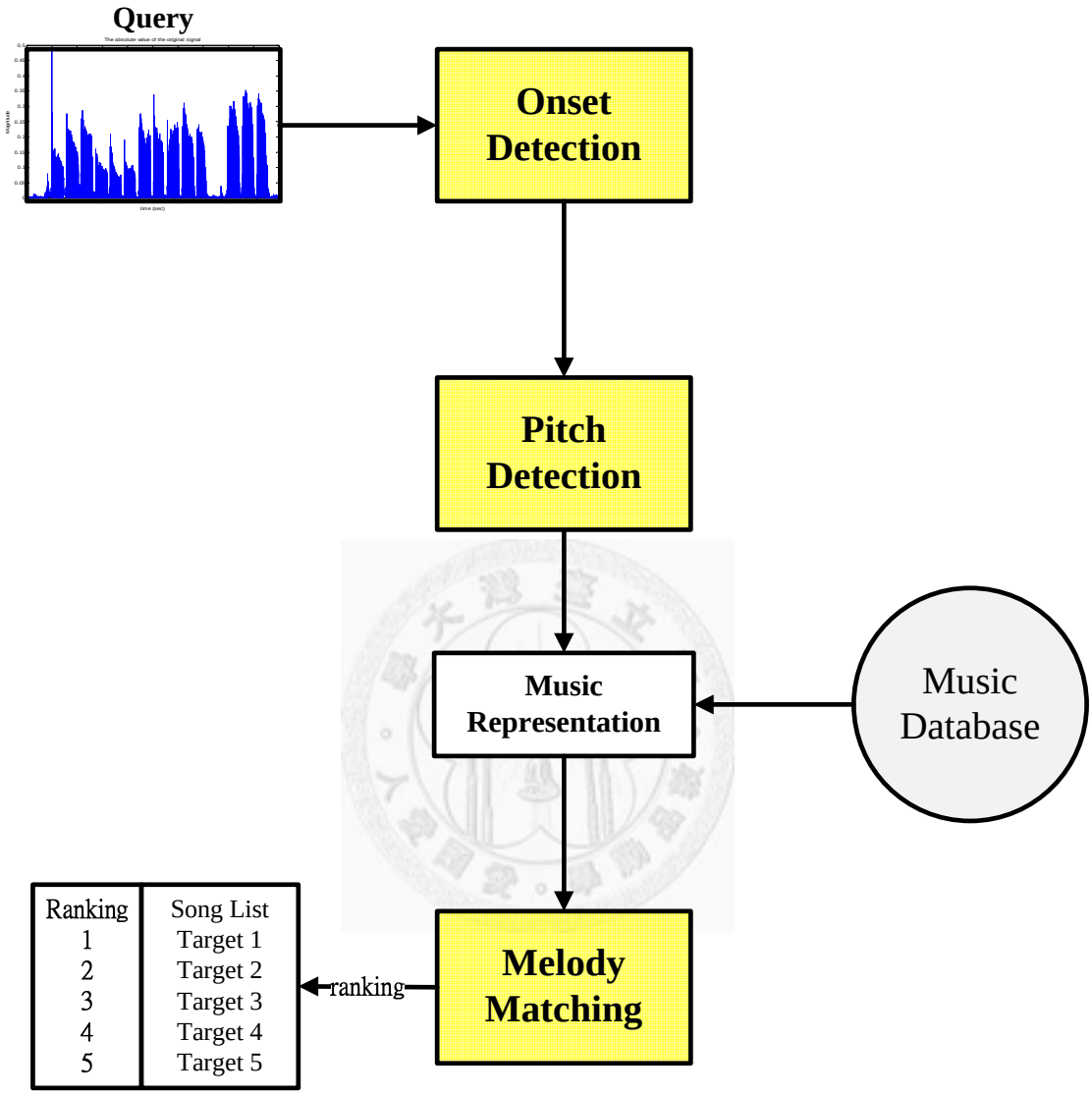


Fig. 1.1 The system diagram of ‘Query-By-Humming’ system.

1.2 Primary Achievement of This Thesis

According to the most three significant parts as shown in Fig. 1.1, we improve the performance of onset detection and melody matching. Besides, we use our own method to extract the fundamental frequency on pitch tracking. Unlike most theses, we don’t discuss only the performance of onset detection or melody matching but the whole QBH

system.

The true positive (**TP**) rate is obviously rising after using our proposed onset detection while the hit rate is also rising after using our proposed melody matching methods. Based on the known onset detection, the proposed onset detection is to make onset detection more robust and non-sensitive for thousands of variations of sung waveform to deal with free singing style from different individuals. In addition, we modified dynamic programming algorithm (**DP**) and hidden Markov model (**HMM**) to promote the hit rate (**HR**) and **MRRR** and name them as **MDP** and **MHMM**, respectively.

In our simulation, we use three versions of queries: 10-second length, 15-second length, 20-second length of queries. Each version has 26 queries and the number of targets in our music database is 50.

Our proposed onset detection maintains the high TP rate for all three versions. The known onset detection seems can't provide the stability. For example, Short-term Energy Method reaches 94.20% in TP rate but it performs worse in the version of 15-second and 20-second. Surf method reaches 94.70% and 97.05% in the version of 15-second and 20-second but it is under 90% in the version of 10-second. Our proposed onset detection exceeds 98% in TP rate for all three versions.

Our proposed melody matching methods: MDP and MHMM. MDP reduces the searching time of DP from 35.07 second to 3.8 second for each query in the version of 20-second. Moreover, MDP promote the hit rate (HR) from 64% to 100 % in the version of 20-second. MHMM needs a little more time than HMM, but it still cost less running time than MDP. For example, running time of MHMM is 1.52 second in the version of 20-second while running time of HMM is 1.14 second, but the searching time of MHMM is still very fast. The hit rate is rising from 44% to 84% by using MHMM.

1.3 Chapter Organization

In Chapter 1, we briefly introduce the background of **QBH** system in section 1.1 and our contribution in this thesis in section 1.2. Then, the chapter organization is described in section 1.3 and some abbreviations are listed in the final section 1.4.

In Chapter 2, there are some basic music characteristics we have to understand before doing this research. Table 2.1 shows the relationship between fundamental frequency and its MIDI number.

In Chapter 3, we discuss the known onset detection methods: magnitude method, short-term energy method, surf method, high-frequency content (**HFC**) method and pitch method. Magnitude method uses the differences of magnitude as the feature to judge if an onset happens or not. Short-tern energy method calculates the energy of each frame, if an energy band exceeds a threshold value and its position in time domain would be detected as an onset. Surf method use the large slope to detect the possible onset while the HFC method use the high frequency content to estimate what time onsets happen. Besides, pitch method let us observe the time-frequency distribution of the sung input.

In Chapter 4, we discuss some existing pitch tracking method. Pitches can be extracted from onset note by onset note which is divided by onset detection. There are several approaches to compute the pitches. Sub-harmonic summation (**SHS**) computes the sum of harmonically compressed spectral. Basically, the maximum sum is chosen as the pitch in that time frame. Pitches can be detected by using adaptive threshold value to filter the possible frequency band on spectrum. Autocorrelation function (**AFC**) of pitch tracking is to find the fundamental period and reverse it to get the fundamental frequency. Modified Hilbert-Huang transform (**MHHT)** are also used to estimate the

fundamental frequency as well. However, melodies are composed of not only the fundamental frequency but also its harmonics (partials). Harmonics always interrupt our estimation to the fundamental frequency.

In Chapter 5, not only onset detection causes the wrong estimation of pitches, but also people themselves sing wrong pitches for the song and sometimes sing more or less pitches than the original song. We discuss melody matching methods: dynamic programming algorithm and hidden Markov model. Because no reliable onset detection can exactly detect the perfect onsets as its actual onsets due to free singing styles, onset detection may only provide us an approximate onset detection result but not the exact one. Therefore, melody matching techniques are developed to modify the singing errors generated by onset detection.

In Chapter 6, we will introduce music representations for the feature extracted from each notes detected by onset detection. First, we describe music scale in section 6.1 to explain how to match one song in different tonality. In section 6.2, we introduce the concept of pitch interval which is the difference of contiguous pitches and it is really useful to match the same song in different key. Finally, we present how to construct our database in section 6.3. The pitches form sung input are quantized to a sequence of symbolic representations and are compared to known symbolic representations in the large database.

In Chapter 7, people always sing in different ways so the tempos and tonality (key) are also different of each person. Therefore, the onset detection must be robust and insensitive to the tonality. Instead of using absolute pitches and time intervals of each notes, we use relative pitches to do melody matching. In section 7.1, we will present our proposed onset detection. In 7.2, we will present our pitch estimation. Finally, we will present our proposed method: **MDP** and **MHMM** in detail.

In Chapter 8, we will show the experimental result of our QBH system and prove its performance is better than any other combinations of known onset detection and melody matching method. In Chapter 9, we make a conclusion of this research and set some future works for further research to make our QBH system as best a possible.

1.4 Abbreviations

There are some abbreviations which appear frequently in this thesis so we list these abbreviations for reference:

- **HR**: Hit rate
- **TP**: True Positive
- **DP**: Dynamic Programming
- **HMM**: Hidden Markov Model
- **HFC**: High Frequency Content
- **AFC**: Autocorrelation Function
- **SHS**: Sub-Harmonic Summation
- **QBH**: Querying-By-Humming System
- **MDP**: Modified Dynamic Programming
- **MRRR**: Mean Reciprocal Right Rank
- **MHMM**: Modified Hidden Markov Model
- **MHHT**: Modified Hilbert-Huang Transform

Chapter 2 Basic Musical Characteristics

When talking about music, there are some related musical term we have to know previously, like chord, loudness, pitch, pitch interval, note, rhythm, timbre, transcription and music scale. We have to understand the meaning of these musical tern, then we can do further work. Therefore, there are some basic definitions for some musical characteristics bellow:

- **Fundamental frequency**: The fundamental frequency of a single tone signal and it is usually highly correlated with pitch.
- **Harmonics**: Multiples of the fundamental frequency and are along with its **F0**.
- **Octave**: An octave contains 12 semitones. It can also be defined that the second harmonics of a certain frequency.
- **Loudness**: The energy intensity of a sound that is the primary psychological correlate of physical strength (amplitude) and is often represented by **db**.
- **Pitch**: Pitch is the perceived as the fundamental frequency of a sound. It has high correlation to the fundamental frequency of a single tone.
- **Note**: One is a symbol used in musical notations and the other meaning in Fig. 2.2 is a sound which is produced by a sound object.
- **Chord**: A chord is three or more different notes that sound simultaneously. It is always with some manners, like the difference of a chord's notes will be whole tone or whole tone's multiples.
- **Timbre**: Timbre can be view as a identification of a certain sound and is different form sound objects. The multiple relationship

- **Transcription**: Transcription is to convert a piece of a sound into note number.
- **Music scale**: Music scale is a particular set of notes that are defined by musicians as being appropriate for a song.
- **Pitch interval:** The pitch difference between any two contiguous notes, measured in semitones, is called the pitch interval
- **Onset:** An instant chosen to mark the temporally extended transient. In most cases, it means the start of the transient, or the earliest time at which the transient can be reliably detected in.
- **Attack:** the interval during which the amplitude envelope increases.
- **MIDI:** A popular music encoding format, in which the music notes are represented by numbers (from 0 to 127).

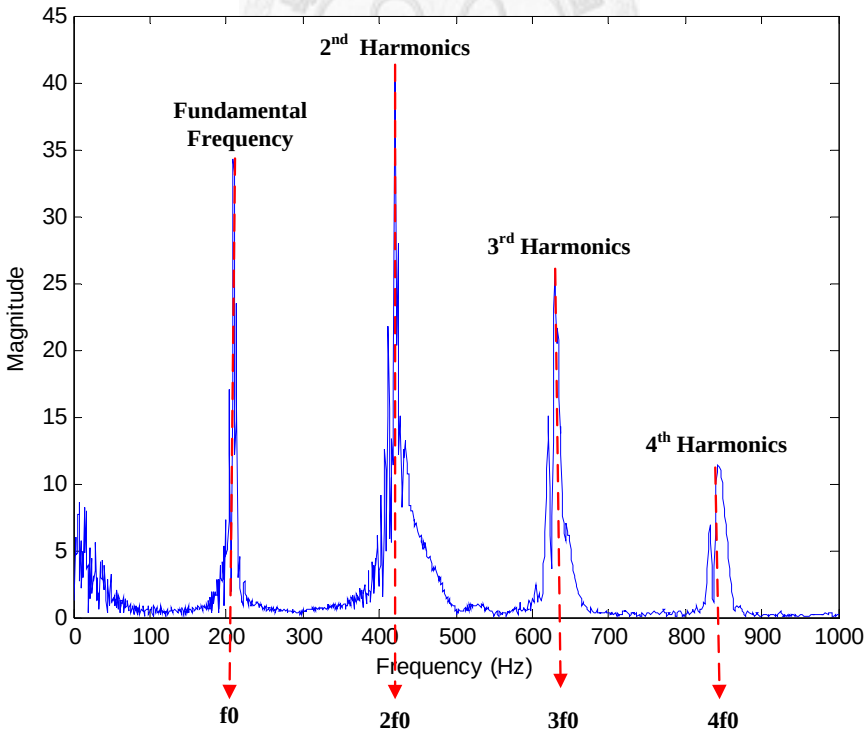


Fig. 2.1 The fundamental frequency and its fundamental frequency.

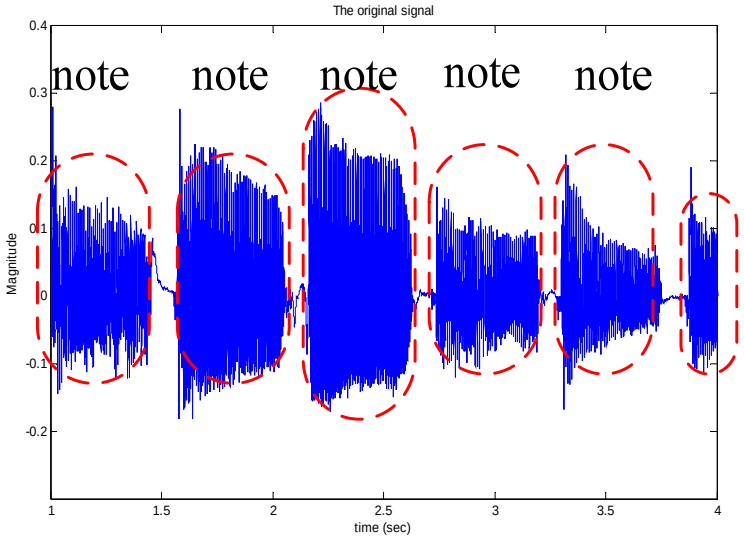


Fig. 2.2 The illustration for notes of a piece of music “兩隻老虎”.

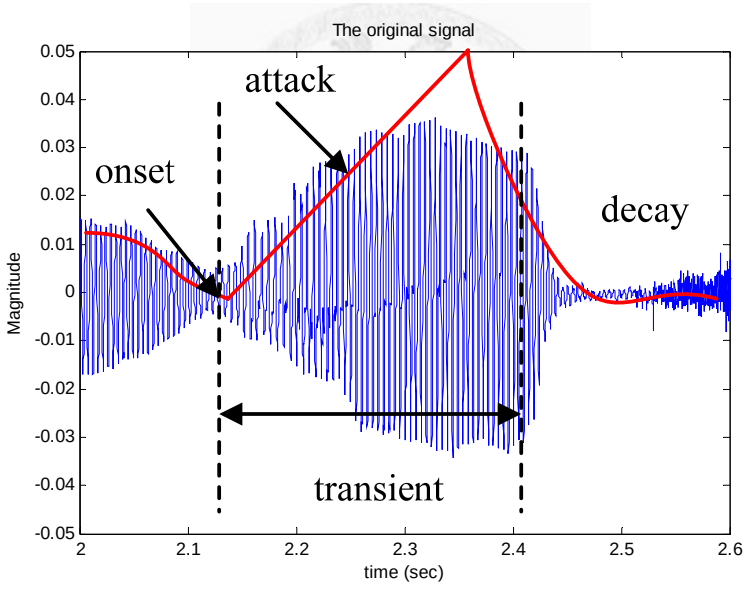


Fig. 2.3 “Attack”, “transient”, “decay”, and “onset” in the ideal case of a single note.

In addition, there is a relationship between fundamental frequency and its MIDI number. Equation (2.1) and (2.2) are equivalent basically.

$$f = 2^{n/12} \times 440\ Hz \tag{2.1}$$

The definition of MIDI note number from frequency is:

$$MIDI\ \text{number} = 69 + 12 \times \log_2\left(\frac{f}{440}\right) \qquad (2.2)$$

where n in (2.1) is also represent MIDI number.

Table 2.1 shows the relationship of note name, its MIDI number and its fundamental frequency form Octave#3 to Octave#5. Most pitches of pop music is in this range, so we list the table as follows:

Table 2.1 The relationship of fundamental frequency and MIDI number of notes in 3 consecutive octaves (from Oc3 to Oc5)

Note Name (Oc3)	**MIDI number**	**Fundamental frequency**	**Note Name (Oc4)**	**MIDI number**	**Fundamental frequency**	**Note Name (Oc5)**	**MIDI number**	**Fundamental frequency**
*Do*3	48	130.81	*Do*4	60	261.63	*Do*5	72	523.25
*Do*3#	49	138.59	*Do*4#	61	277.18	*Do*5#	73	554.37
Re 3	50	146.83	Re 4	62	293.66	Re 5	74	587.33
Re 3#	51	155.56	Re 4#	63	311.13	Re 5#	75	622.25
*Mi*3	52	164.81	*Mi*4	64	329.63	*Mi*5	76	659.26
*Fa*3	53	174.61	*Fa*4	65	349.23	*Fa*5	77	698.46
*Fa*3#	54	185.00	*Fa*4#	66	369.99	*Fa*5#	78	739.99
*Sol*3	55	196.00	*Sol*4	67	392.00	*Sol*5	79	783.99
*Sol*3#	56	207.65	*Sol*4#	68	415.30	*Sol*5#	80	830.61
*La*3	57	220.00	*La*4	69	440.00	*La*5	81	880.00
*La*3#	58	233.08	*La*4#	70	466.16	*La*5#	82	932.33
*Ci*3	59	246.94	*Ci*4	71	493.88	*Ci*5	83	987.77

Chapter 3 Onset Detection

Once we decide the onset time of notes of a query, then the sequence of pitches can be calculated. Because humming voice from human is a single tone at one time not multiple tones at one time, we don't have to take polyphonic melodies into our account. The definition of onset time is a significant characteristic which is an abrupt rise in energy of the signal.

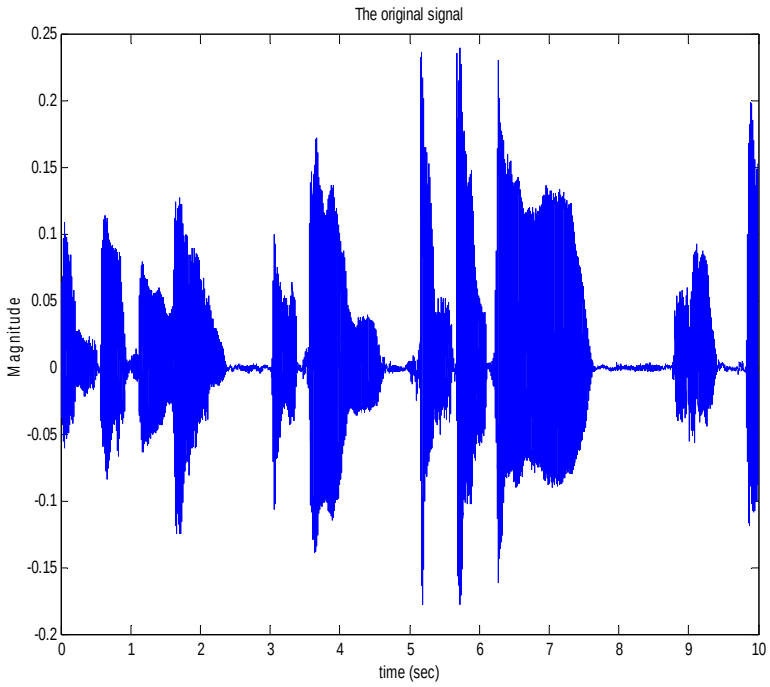


Fig. 3.1 The original waveform produced by a male person singing a passage of melody of the song "愛情釀的酒"of 12 notes.

In this Chapter, we use a query produced by a male person singing a passage of melody of the song "愛情釀的酒" of 12 notes by using the syllables 'da-da'. The original signal is shown in Fig. 3.2 and red circles are denoted as the real onset sounded

by a male person in.

From Fig. 3.1, we can tell the real onset time by listening to the recording query but can't tell what time do these notes sound accurately. In Fig. 3.2, the real onsets are denoted by red circles and the numbers in circles means the n-th onset of the signal.

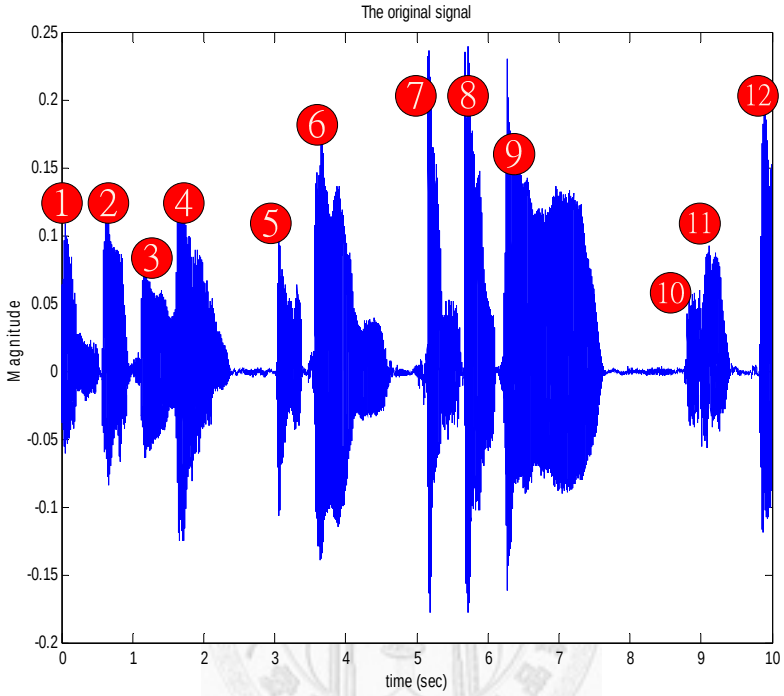


Fig. 3.2 The real onset times are denoted by red circles and the numbers in circles mean the n-th notes in the signal.

The input signal is denoted by x and the absolute value of x is denoted as X. X can be viewed as a pre-processing of the original signal and it is easier to deal with in the following different onset detection methods. Before detecting onsets, the input signal, x, has to pass a previous process to make the waveform more easily be dealt with. After detecting onsets, the signal has to eliminate redundant onsets due to the reason that it may detect any possible onsets. Therefore, the post process is to eliminate onsets which are too close to each other. In the following simulation, we set the

minimum onset time interval is 1500 sampling points. That means if the sampling rate is 8000 Hz, then the minimum time interval is 0.1875 second. Assumed that the minimum time duration between each sound is 0.2 second, so 1500 sampling points for minimum onset time interval is acceptable. The system block diagram is plot in Fig. 3.3.

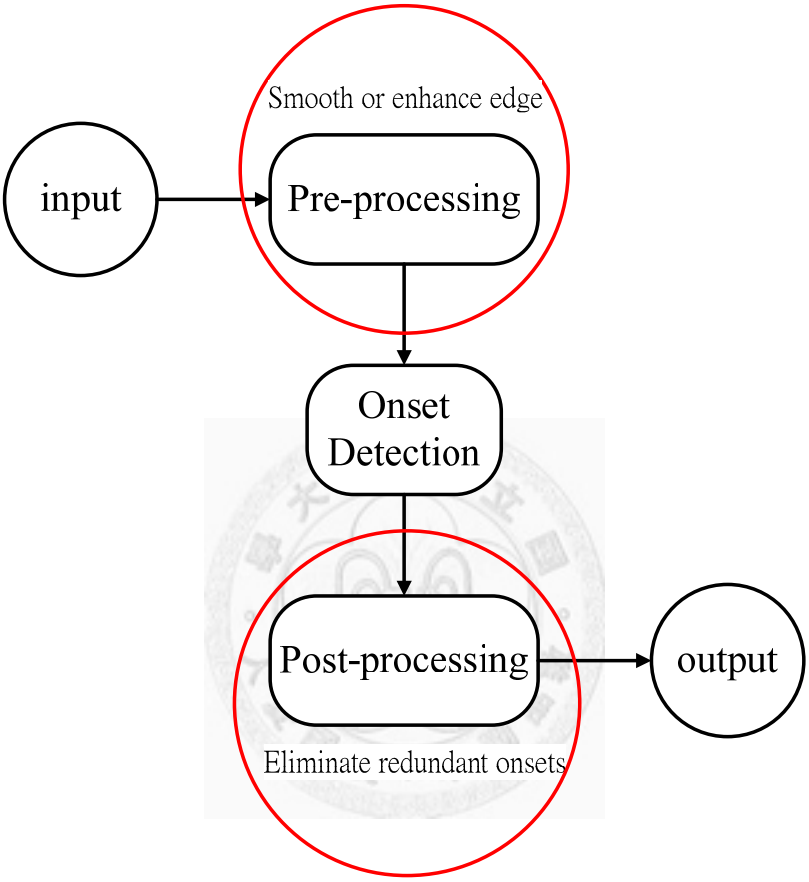


Fig. 3.3 The system block diagram of onset detection.

In the following section, we will present each onset detection method mentioned in detail. Using the system in Fig. 3.3 to simulate all the onset detection methods and the detection result will be compared later.

3.1 Magnitude Method

This approach is the most straightforward method which people can imagine intuitively.

Using the difference of two consecutive sampling points to detect the possible onset times is the basic concept here. In this method, the sampling rate is chosen to as 8000 samples per second and the length of the recording query is 10 seconds. That is, this recording query has 240000 sampling points. First, we choose a threshold by empirical test from the simulation of this method. Then, we set a buffer to store the magnitude of every 1000 sampling points before the processing point. That means every sampling point has its own buffer. Initially, we set the buffer as a zero array whose size is 1000 (sampling points). When we deal with the first sampling point, we reset the buffer as Fig. 3.4 shows. When we deal with the first sampling point whose value is 1, we reset the buffer by adding the difference value (e.g. 4) of the processing point in the end of the buffer and eliminate the first value of the buffer in order to maintain the buffer's size

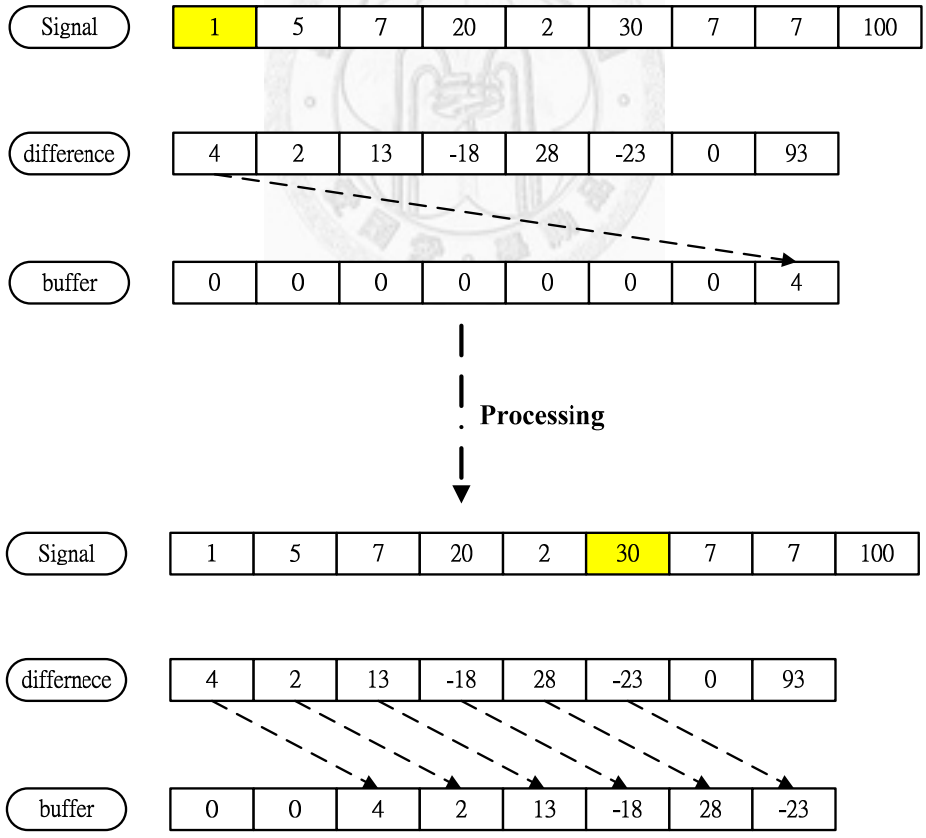


Fig. 3.4 A simple illustration for the algorithm of magnitude method.

When the processing point comes to the value of 30, the buffer has become as Fig. 3.4 shows. The differences of the consecutive sampling points are calculated all over the querying signal. Each difference which exceeds the threshold values we set before has to be checked again and will be selected as a possible onset if its value exceeds all the values stored in its processing buffer. For example, if the threshold value is set as 25, then the time index of the value of 30 will be selected as a possible onset. Because 30 exceeds 25, and 30 exceeds all the values in the buffer ([0 0 4 2 13 -18 28 -23]). Besides, the processing will keep going to the next sampling point until the end of the signal.

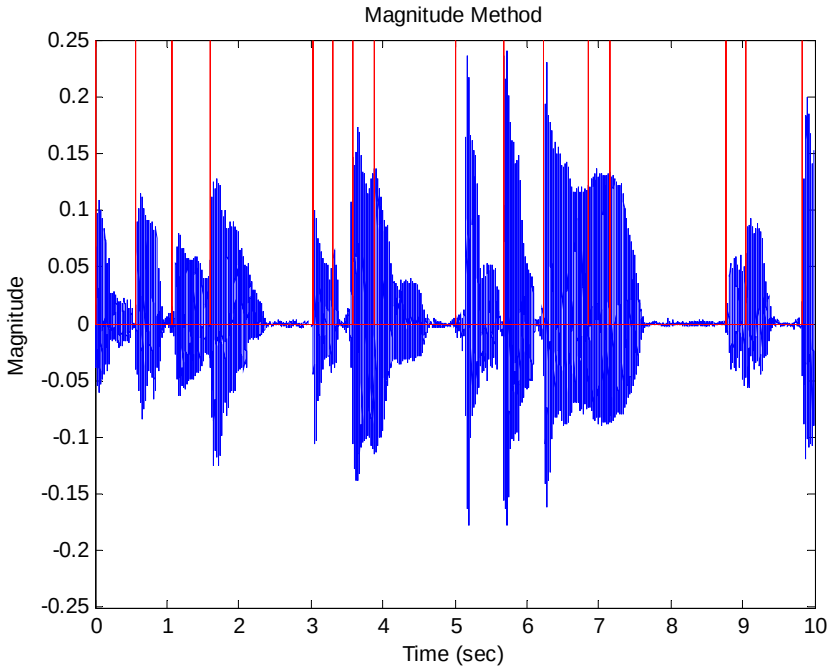


Fig. 3.5 The onset detection of magnitude method for the song “愛情釀的酒”.

The detection result is shown in Fig. 3.5. Compared with Fig. 3.2, we can found that this method detects 16 onsets and more than 12 actual onsets in Fig. 3.1. That because the threshold value we choose in magnitude is fixed. If we choose too small threshold value, the result will be over-detected while if we choose too large threshold value, the result will be under-detected.

3.2 Short-Term Energy Method

This approach [1] is another easy-realized method due to the reason that there are always silences between two consecutive sounds. For that, using the short-term energy to calculate the sum of energy in each window is another method to detect onset time. When short-term energy window slides from beginning to the end of the signal, the energy of each frame will be calculated. The energy value is judged if an onset time happens or not. The short-term energy E_k in frame k is estimated by

$$E_k = \sum_{n=1}^{N} x(n)^2 \tag{3.1}$$

The simple chart for short-term energy method is plot as in Fig. 3.6. E_k is the sum of the kth energy frame and is calculated by (3.1) and N is the frame size.

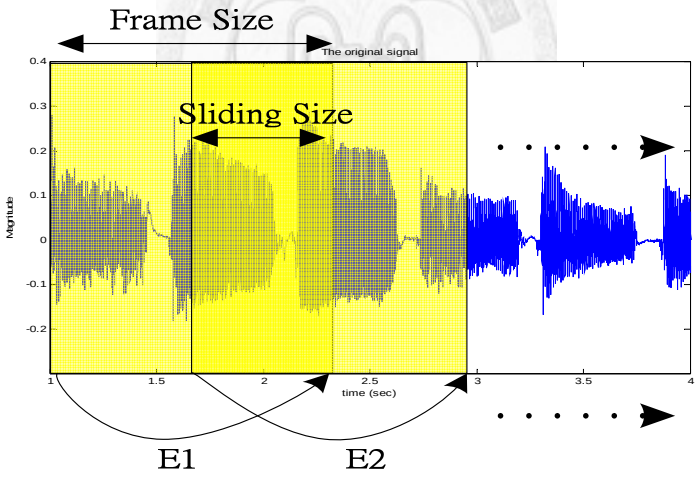


Fig. 3.6 The process of calculating energy by a sliding window.

Adaptive threshold values are used to determine onset times and offset times. For instance, the note onset threshold is set as 0.02 while the note offset threshold is set as 0.01 by empirical test. Energy in a frame exceeds the note onset threshold will be defined as a onset time while energy in a frame lower the note offset threshold will be

defined as a offset time. The window size is chosen to be 20ms and each motion slides 10ms in the following simulation.

3.2.1 Implementation of Short-Term Energy Method

As we can see from Fig. 3.1, silences exist between two contiguous notes. Because the energy of silence part is very small and close to zero, the threshold can be easily decided to judge an energy band is in silence or not.

In this method, we use the signal in Fig. 3.1 as an input. The pre-process of short-term energy method is to calculate the energy of each frame whose size is 20 ms and slides 10 ms per motion. Before calculating the energy, the signal is transform to the absolute value as in Fig. 3.7.

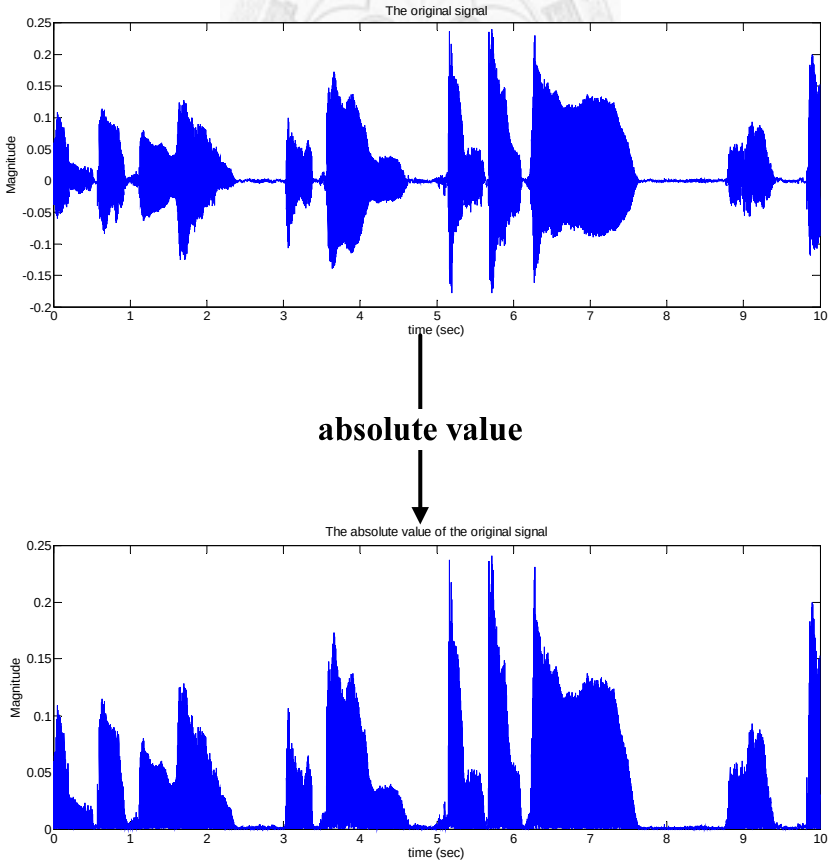


Fig. 3.7 The pre-process of extracting the absolute value of the signal in Fig. 3.1.

Then, the energy envelope is calculated frame by frame as Fig. 3.8 shows. It can be found that region whose energy is closed to zero is silence.

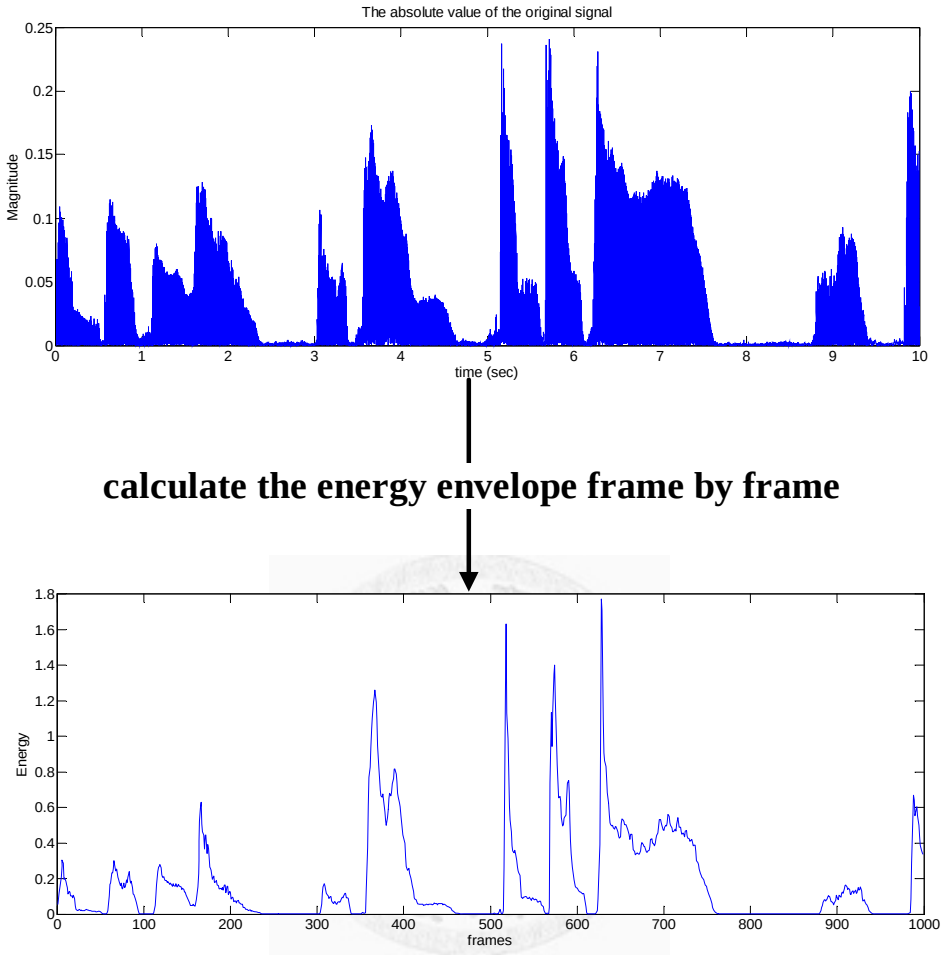


Fig. 3.8 The envelope of signal in Fig. 3.7.

How to choose a threshold value is the most difficult part in onset detection. Here, we have two implementations. The first one is to set a minimal threshold value to filter silences. In first implementation, energy frame which exceeds the threshold value is reset as 1 and under the threshold value is reset as 0. The resetting value of each frame only have 3 values: -1, 0, 1. The value 1 means the onset, the value -1 means the offset and the value 0 represents no obvious change in the signal. However, this implementation has a fatal drawback. If there is no silence between two notes, the

second onset would not be detected. The second implementation is to set a threshold value to filter abrupt rise of energy. However, the second method is almost similar to magnitude method.

3.2.2 Discussion of Short-Term Energy Method

Obviously, using the first implementation is not sufficient due to two missing detection in Fig. 3.9. Threshold value is 0.02 in Fig. 3.9.

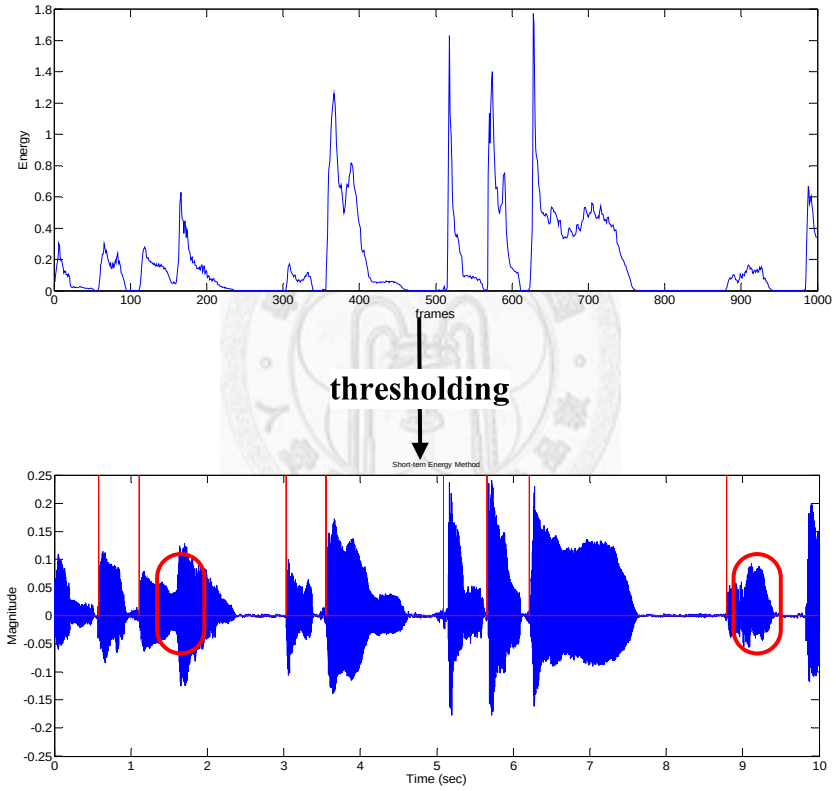


Fig. 3.9 The short-term energy detection result only detects 10 notes and misses two circled parts.

If we set the threshold value as 0.06, the detection result will become that as in Fig. 3.10. Therefore, choosing threshold is very difficult to handle. If we choose too small threshold value, the onsets will be under-detected. If we choose too large threshold

value, the onsets will be over-detected. In the following section, another features such as slope and frequency will be considered to detect the onset not only form magnitude and energy.

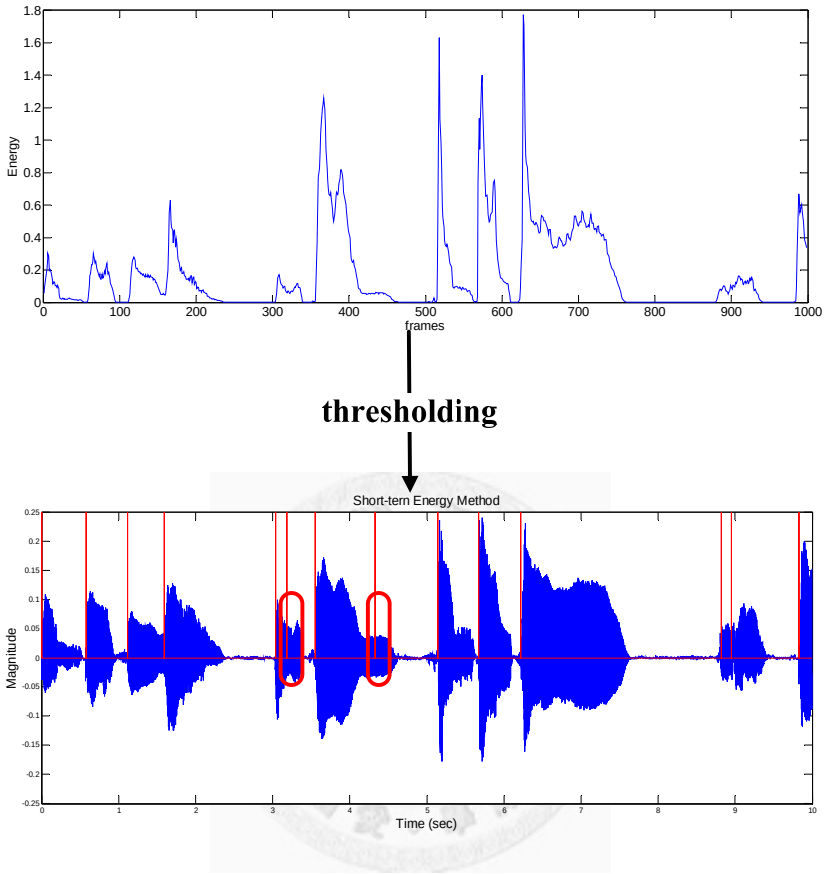


Fig. 3.10 The short-term energy detection result detects 14 notes and over detects two more circled parts.

3.3 Surf Method

The surf method [1] has been proposed from the techniques of **Schloss.** The signal is filtered through a first-order high pass filter to enhance the part of abrupt rise of energy and blocked into frames whose size is 20 ms with a 10 ms shifting motion per time. The envelope $y(k)$ is evaluated by only picking the local maximal value of *k-th* frame of the signal. The slope of the envelope is calculated by fitting a second-order polynomial

($a+bt+ct^2$). We denote a small finite range of envelope sequence around sample τ by $\{y(\tau+t)\}_{t=-M}^{M}$, where M represents the width of the polynomial interpolation. A polynomial approximation of the slope at center τ is then given by

$$\frac{\partial y(\tau+t)}{\partial t} \cong b = \sum_{t=-M}^{M} \frac{t \cdot y(\tau+t)}{\sum_{t=-M}^{M} t^2} \tag{3.2}$$

When M is set as 2, 5-point approximation is computed. Due to the characteristic of abrupt rise of energy of the signal, we look at the positive zero-crossings of the slope contour to pick up these onset times.

3.3.1 Implementation of Surf Method

First, we use a second-order polynomial in (3.3) to model a curve and then make first-order differential to it. Then we will get the coefficient of b, and b represents the slope of the tangent line of center. In our implementation, we use 5-point fitting to fit the slope of the center of every 5 points. Take Fig. 3.11 for example, the slope can be founded by fitting the 5 point.

$$f(t) = a+bt+ct^2 \tag{3.3}$$

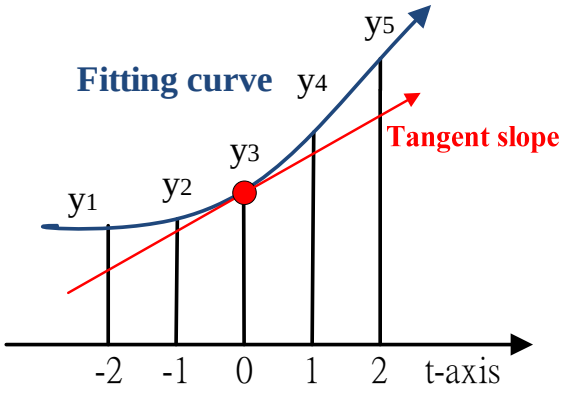


Fig. 3.11. The slope of center of second-order polynomial.

Each frame has 5 sampling points of signal and its slope will be calculated frame by frame. The set of coefficient {a, b, c} is the solution and the value of b is the slope of center in each frame. Equation (3.4) may have not the real solution, but the minimal solution can be calculated. Therefore, the solution, s, can be evaluated by (3.7).

$$As = y \tag{3.4}$$

$$A^T As = A^T y \tag{3.5}$$

$$(A^T A)^{-1} A^T As = (A^T A)^{-1} A^T y \tag{3.6}$$

$$s = (A^T A)^{-1} A^T y \tag{3.7}$$

Due to the third sampling point, y_3, is the center of each frame, it is located at original on t-axis as Fig. 3.11 shows. The first one is located at -2, the second one is located at -1, the fourth one is located at 1 and the fifth one is located at 2. The solution matrix, A, can be created by a set of simultaneous equations in (3.8).

$$\begin{Bmatrix} 1a - 2b + 4c = y_1 \\ 1a - 1b + 1c = y_2 \\ 1a + 0b + 0c = y_3 \\ 1a + 1b + 1c = y_4 \\ 1a + 2b + 4c = y_5 \end{Bmatrix} \tag{3.8}$$

Then, matrix A is created in (3.9) and the solution of the set of coefficient {a, b, c} is computed by (3.10). Then, the solution is calculated by (3.10).

$$A = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \right)^{-1} \bullet \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{bmatrix} \bullet \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} \quad (3.10)$$

3.3.2 Discussion of Surf Method

In Fig. 3.12, the slopes are calculated from the envelope of the original signal. It can be found that the onset always has the large slope.

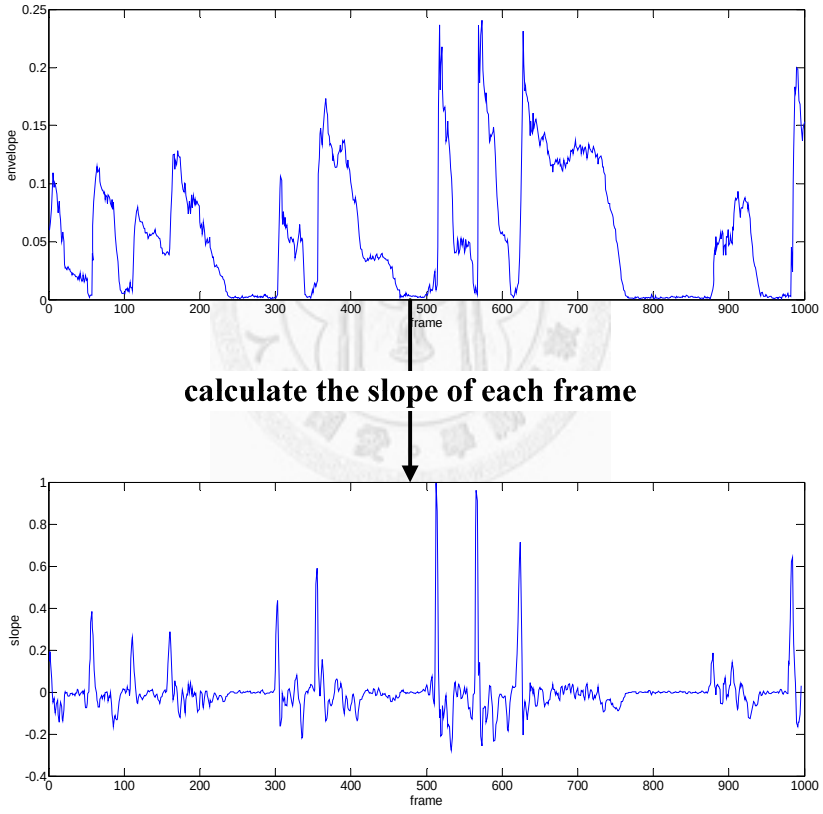


Fig. 3.12 The slope of each frame of envelope $y(k)$ of the original signal.

However, we can observe the result form Fig. 3.13 and found that this approach is over-detected. It is due to the reason that when people sound a pitch, slightly off-pitch in the end of a sound. The two more detected notes in Fig. 3.13 are estimated as notes due

to the magnitudes abruptly rise in the end of the sounds.

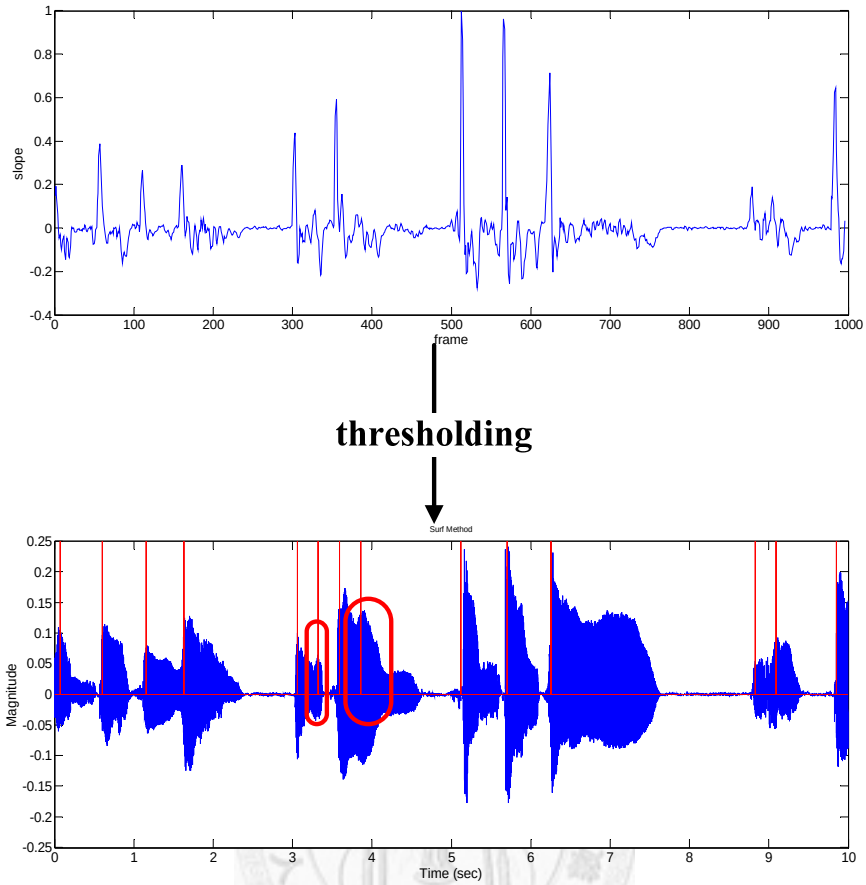


Fig. 3.13 The surf detection result detects 14 notes and over detects two more circled parts.

3.4 High Frequency Content (HFC) method

The high-frequency content (**HFC**) [1] method has been adopted for several years and can be found from several literatures. The algorithm combines both changes in overall energy and the energy concentration at high frequencies together. The signal is blocked into frames of 20 ms with a frame shift of 10 ms. The frame of the signal computes M-point FFT to generate a short-time DFT $X_k(m)$. The energy E_k in the frame k is defined as the sum of the squared magnitude of each FFT bin,

$$E_k = \sum_{m=m0}^{M/2+1} |X_k(m)|^2 \tag{3.11}$$

Denote that H_k is a weighted version of E_k and is linearly proportion to the higher frequencies,

$$H_k = \sum_{m=m0}^{M/2+1} m \cdot |X_k(m)|^2 \tag{3.12}$$

A detection function is designed to as the product of the rise in high frequency energy between two consecutive frames and the normalized high frequency content of the current frame *k*,

$$D_k = \frac{H_k}{H_{k-1}} \cdot \frac{H_k}{E_k} \tag{3.13}$$

Then, a threshold value is set to compare with D_k if a note onset or not. If D_k surpasses a given threshold, we indentify the detection function has found a note onset.

3.4.1 Implementation of HFC Method

Due to the shape of onset is like as an edge, using the high frequency content of an edge is an another way to detect onsets. Using the equation mentioned in (3.12), the weight frequency content is calculated frame by frame. High frequency content gains much more weighting than low frequency content.

$$\frac{H_k}{H_{k-1}} \tag{3.14}$$

$$\frac{H_k}{E_k} \tag{3.15}$$

Equation (3.14) represents trend degree to which frame and the function here is localization. Equation (3.15) represents normalization for each frame due to silence may have high frequency content with noise.

In this implementation, sampling rate is chosen as 8000 Hz. The size of each frame is 20ms and slides 10ms per motion. Frequency analysis (the **FFT**) is used to analyze the spectrum of each frame. Be careful, we only observe the positive frequency content, so the range which from 0 Hz to 4000 Hz are calculated.

3.4.2 Discussion of HFC Method

The simulation result is as in **Fig. 3.15.** From the deteciton result, 7 more redundant onsets are detected. In this approach, silences may be detected as onsets in Fig. 3.14.

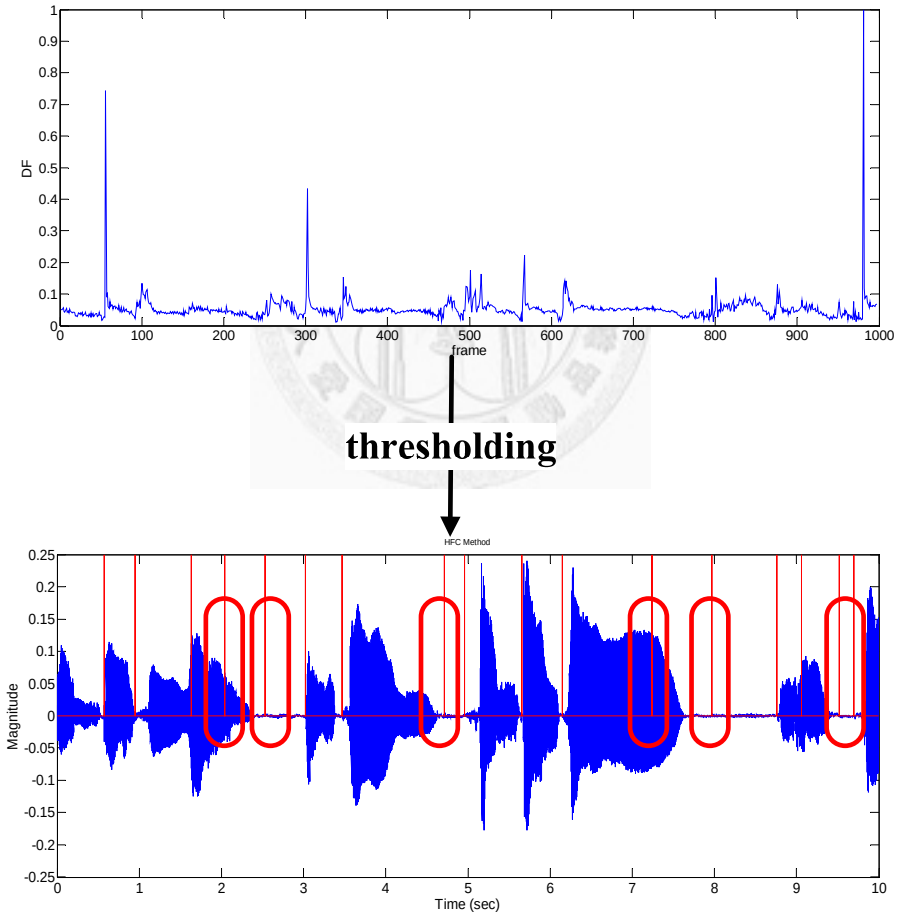


Fig. 3.14 The HFC detection result detects 19 notes and over detects 7 more circled parts.

It is due to the reason silences have slightly vibration of energy and that may cause

high frequency content and this parts of signal can be viewed as noise. Therefore, the high frequency content method seems not to perform well in onset detection.

3.5 Pitch method

This method [1] is to compute the pitch of the signal. The advantage is sung input doesn't need to be in an isolated manner but a free-style way for us to observe its changes of onsets and offsets. However, the drawbacks of this approach is that there too much frequency distribution so that we even can't distinguish what time do the pitch change. First, an input signal will be computed its frequency domain content by DFT and as Fig. 3.15 shows.

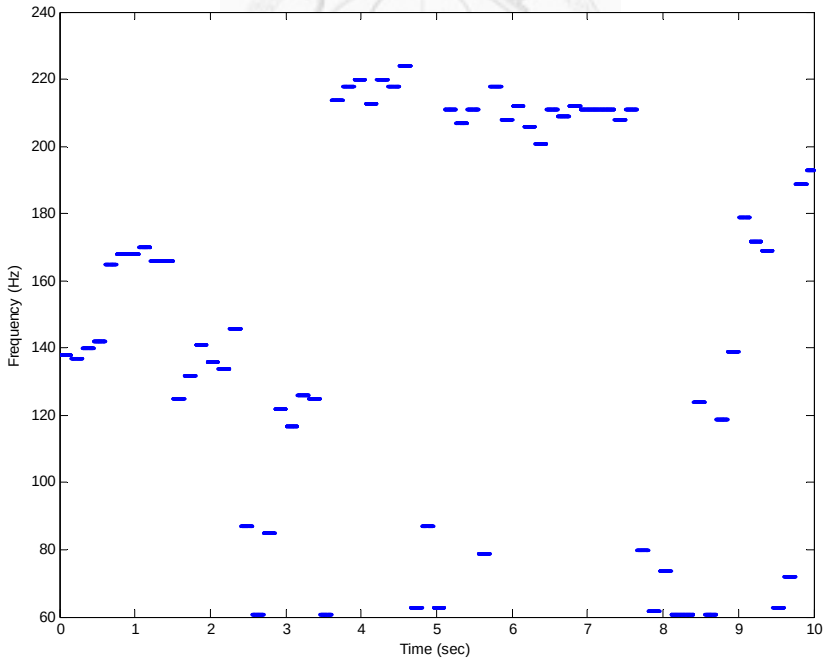


Fig. 3.15 The time-frequency distribution of input signal.

From above Fig. 3.15, we use a frame whose size is 15 ms due to our assumption that the minimum time duration of two consecutive notes is 30ms and the sampling rate

is 8000 Hz. Smaller the frame size is and higher clarity the frequency resolution are. It can't be identified clearly what frequency band belongs to the same MIDI number band. Therefore, the frequency should be transferred to the MIDI number by using equation (2.2) and the new figure is plot as follows:

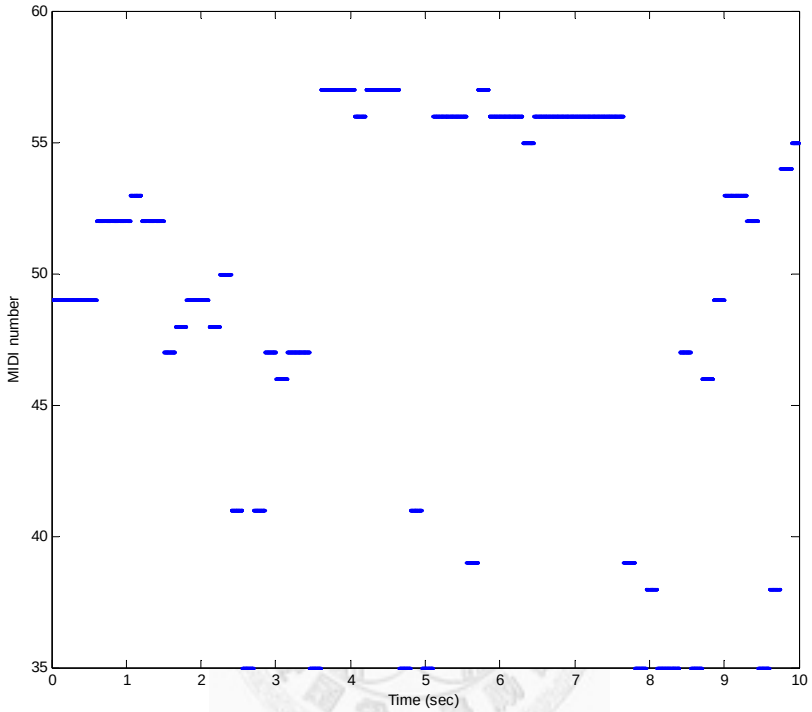


Fig. 3.16 The time-MIDI distribution of input signal.

The vertical axis represents MIDI number and the horizontal axis represents time. From Fig. 3.16, it can be founded that the pitch is more clearly and we can identify the melody contour more easily. However, there are still many parts that we can't identify well due to many small fragments of the frequency content. Hence, using this approach to detect onset time is not a good way because it detects the pitches too detailed. In Fig. 3.17, this figure is the time-MIDI distribution of input signal after onset detection. From that, we can compare Fig. 3.16 with Fig. 3.17 and found distribution in Fig. 3.16 is still

too fragmentary.

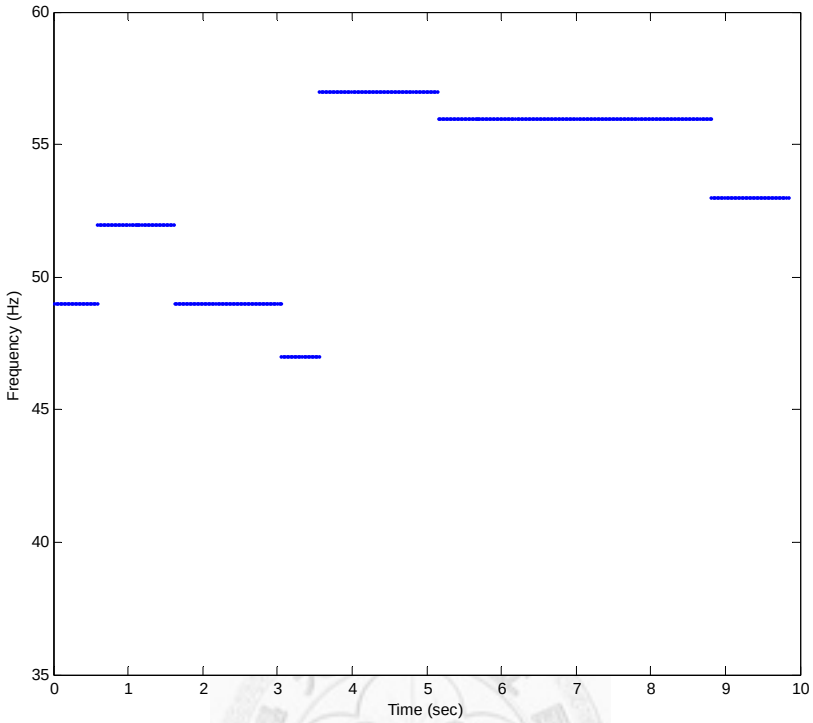


Fig. 3.17 The time-MIDI distribution of input signal after onset detection.

The conclusion is pitch method should be modified and then can be used to detect onset. Due to the property of the time duration of each note, we can assume that the minimum time duration is 300 ms. Each note has the property of maintaining the pitch for a time duration which exceeds 300 ms in our assumption. Maybe we can eliminate the frequency fragment in Fig. 3.16 which does not last for over 300 ms and emerge the fragment to the nearby fragment which meets our demand. Therefore, pitch method seems not a good way to detect onset times.

3.6 Discussion between These Onset Detections

The following figure shows the comparison result of these onset detections and the top

bin in this figure represents the waveform of original signal. The second bin represents onset detection result by using magnitude method. The third bin is the onset detection result of short-term energy method. The fourth bin is the onset detection result of HFC method. The fifth bin is the onset detection result of surf method and the last one is the onset detection result of pitch method.

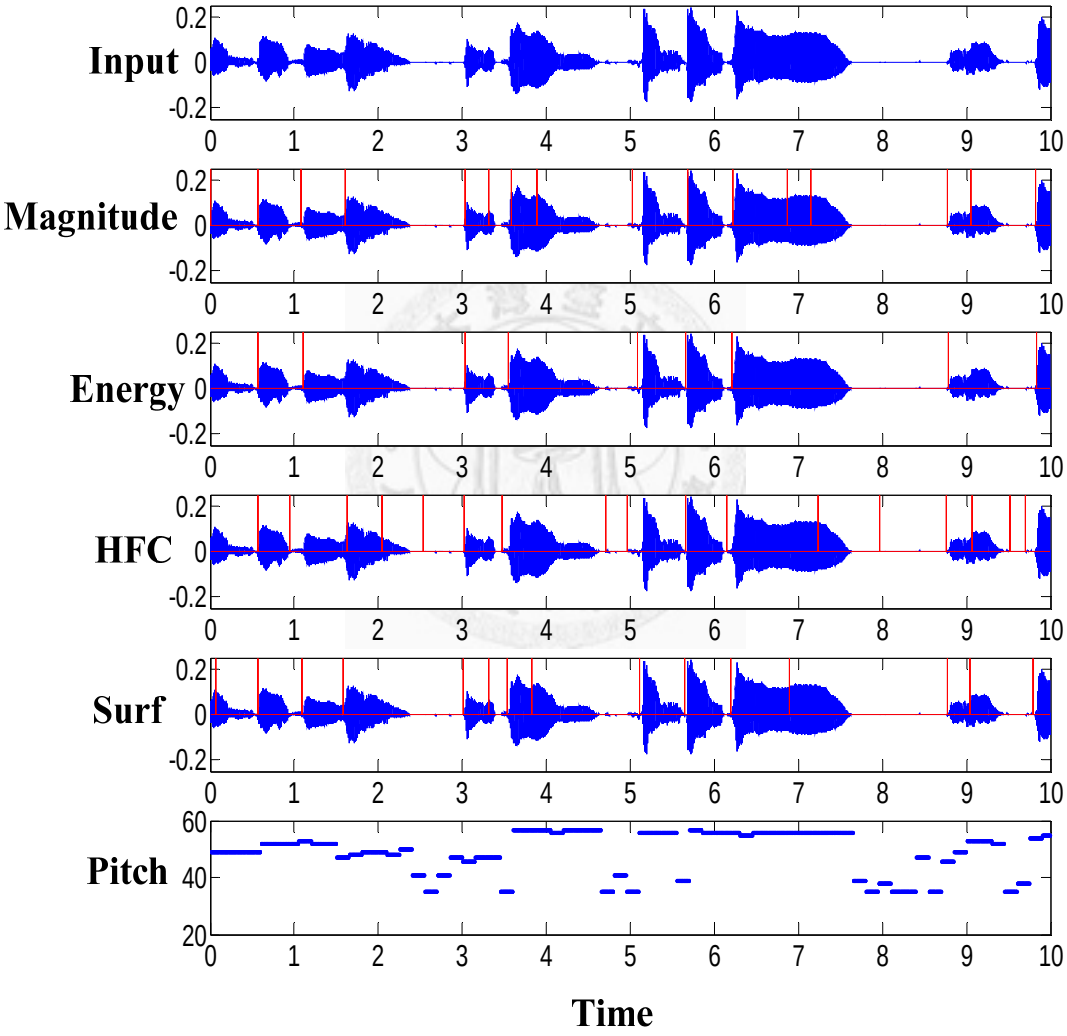


Fig. 3.18 Comparison between these onset detections for the query “兩隻老虎”.

Table 3.1 shows the error types and four different situations: First, an onset is

detected as an onset that is called true positive. Second, an onset is detected as a non-onset that is called false negative. Third, a non-onset is detected as an onset that is called a false positive. Last, a non-onset is detected as a non-onset that is called true negative.

Table 3.1 Four different estimation situations and error type.

Actual Situation / Estimation	Actual onset	Actual non-onset
Judged as an onset	***True Positive***	***False Positive***
Judged as a non-onset	***False Negative***	***True Negative***

Denote TP is true positive, FP is false positive and FN is false negative. Table 3.2 shows the performance of all kinds of onset detection methods mentioned in previous section for the query in Fig. 3.1. True positive rate and False positive rate are calculated by (3.16) and (3.17), respectively.

$$TP = \frac{number\ of\ (True\ Positive)}{number\ of\ (Actual\ Onset)} \tag{3.16}$$

$$FP = 1 - \frac{number\ of\ (True\ Positive)}{number\ of\ (Estimate\ Onset)} \tag{3.17}$$

The recognition rate is modified to suit the music property. The numbers of NFN and NFP are should be subtracted due to they are all wrong detections.

In Table 3.2, this table shows the TP rate and FP rate for all onset detection

introduced in previous section. Basically, TP rate which is under 90 % is bad performance. On the whole, surf method performs better than any other methods. Short-term energy method only performs well in 10-second version. Magnitude method and HFC method performs not well on onset detection.

Table 3.2 Performance of known onset detection methods for 3 cases and each of them has 26 queries.

Query / Method	10-second version	15-second version	20-second version
Magnitude Method	TP = 75.49% FP = 27.30%	TP = 87.03% FP = 16.04%	TP = 89.49% FP = 15.82%
Short-Term Energy Method	TP = 94.20% FP = 9.15%	TP = 78.63% FP = 23.56%	TP = 74.30% FP = 27.16%
HFC Method	TP = 77.32% FP = 26.00%	TP = 86.58% FP = 15.39%	TP = 87.59% FP = 14.29%
Surf Method	TP = 86.22% FP = 18.16%	TP = 94.70% FP = 10.76%	TP = 97.05% FP = 6.14%

Chapter 4 Pitch Estimation

After we detect all the note onsets as possible, the aim is to extract the fundamental frequency, the pitch, of each note. There are some approaches for computing the fundamental frequency (the pitch): autocorrelation pitch tracking, sub-harmonic summation (**SHS**) [12] and modified Hilbert-Huang transform. Sounds produced by humming are along with harmonics which interrupt our estimation to the fundamental frequency. How to efficiently eliminate the harmonics without changing the fundamental frequency is also discussed in this chapter. In processing, octave error often happens due to the harmonics of two octave pitches are same.

4.1 Autocorrelation Function for Pitch Tracking

Autocorrelation pitch tracking is robust at potential fundamental frequency detection, especially for weak fundamental situation. This function is particularly useful in estimating hidden periodicities in a signal. An estimate of the autocorrelation function [9] (**AFC**) of an N-length sequence $x(k)$ is given by:

$$r_{xx}(n) = \frac{1}{N-n-1} \sum_{k=0}^{N-n-1} x(k) \cdot x(k+n) \tag{4.1}$$

Where $r_{xx}(0)$ denotes the energy of N-sized frame, n is time lag value and $x(n)$ is a time domain signal. If the signal has high autocorrelation for a lag value, say K, it has the same high autocorrelation for the lag values, n*K. The lag value K is chosen to be the time interval of signal period and the fundamental frequency is the inverse of K.

For example, we take x as a input signal as shown in (4.2). This is a cosine signal

which is periodic. As same as almost sound, each sound has its own frequency which represents that a humming sound is a periodic signal.

$$x = \cos(2\pi f_i t) \tag{4.2}$$

In order to easily observe the variation of the autocorrelation, the equation (4.1) is modified as (4.3).

$$r_{xx}(n) = \frac{1}{N} \sum_{k=0}^{N-n-1} x(k) \cdot x(k+n) \tag{4.3}$$

Here, we observe time inside 1 second and sampling rate is 1000 Hz. The waveforms of (4.2) and its autocorrelation by using modified method (4.3) are shown as below:

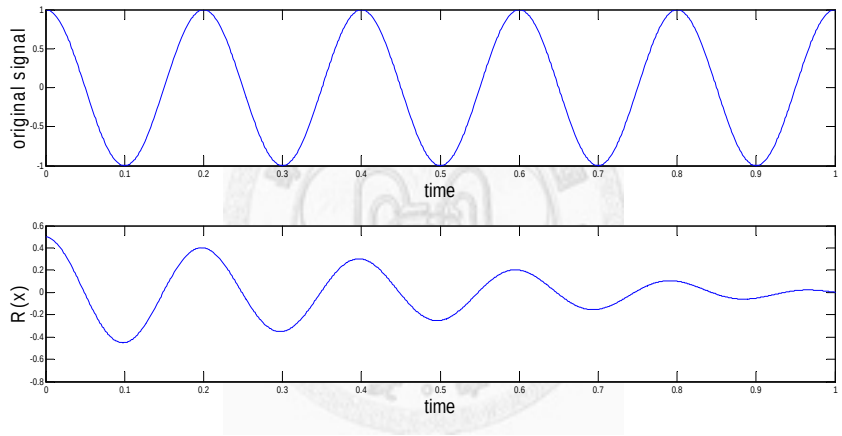


Fig. 4.1 The waveforms of input signal and its autocorrelation.

In Fig. 4.1, the top bin represents the waveform of input signal and the bottom bin represents the autocorrelation of input signal. From the autocorrelation, we can estimate the period of this signal is 0.2 second due to it is the first peak in autocorrelation (ignore the peak located at 0 second).

4.2 Modified Hilbert-Huang Transform

The autocorrelation-based method is not to suitable for detect multiple **F0**s (the

fundamental frequency) and high pitch estimation. Besides, it needs a longer term frame whose size has to be at least twice as long as the fundamental period. Also, the Fourier transform is poor at estimating low-pitch notes since it's not effective in handling low-frequency musical signals due to the logarithm scale notes. Because autocorrelation function (**AFC**) for pitch detection is not robust when the F0 exceeds 600Hz, another algorithm, Hilbert-Huang Transform (**HHT**), for pitch detection has been adopted in [18]. However, there are still some imperfect factors due to the HHT, so the Modified Hilbert-Huang Transform has been proposed in to improve the drawbacks in the original HHT.

4.2.1 The Over View of HHT

The Hilbert-Huang Transform is proposed by Huang, has proved to be certain data analysis application in mechanical and aerospace engineering field. The most important is HHT is a nonlinear analysis tool which is very suitable for natural phenomenon and non-stationary data as quasi-periodic music signals. HHT mainly consists of two parts:

- Step1: Empirical mode decomposition (**EMD**)
- Step2: The Hilbert-Huang Transform

EMD is the pre-processing step to the Hilbert-Huang Transform by using a nonlinear and iterative procedure to decompose the input signal to several *intrinsic mode function* (**IMFs**). An IMF is a function satisfying following two constraints:

1. The number of extrema and the number of zero-crossings should be equal or differ at most by one (i.e., the local maximum has to be positive and the local minimum

has to be negative).

2. The mean function (the mean of the local maximum and minimum) has to be close to zero (i.e., an IMF symmetrically oscillates around zero).

An input signal, $x(t)$, we first find its local maximum x_U and local minimum $x_L(t)$, then calculate their upper and lower envelope. The average function is calculated by the mean of x_U and $x_L(t)$. Denote that $h(t)$ is acquired via $x(t)-m(t)$. If $h(t)$ meets the two IMF criteria, we set the first IMF as $h(t)$. Otherwise, the residual $r(t)=x(t)-h(t)$ is then set to as the next new input signal and apply the same process until the first IMF is found. Each IMF is determined by a convergence threshold. If the energy of the residual is sufficiently small or the residual signal becomes a monotonic function, the iteration terminates. Otherwise, we go to find the next IMF.

The stopping criterion is defined by

$$SD=\sum_{t=0}^{T}[\frac{|h_{1,k-1}(t)-h_{1,k-1}(t)|^2}{h_{1,k-1}^2(t)}] \tag{4.4}$$

Where SD represents the threshold to determine if a IMF holds. In the processing of EMD, several IMF are obtained, then we apply the HHT to analysis every IMF to get the information of amplitude and instantaneous frequency. However, the original HHT does not result in good performance for music pitch detection. Hereunder, four difficulties are common encountered as below:

- *Boundary effect*
- *Intermittence and mode mixing*
- *Stability of EMD with respect to the signal perturbation*

- *Existence of sub-harmonics and partials*

4.2.2 Music Pitch Analysis with Modified HHT

In order to solve the problems mentioned in ***section 4.2.1,*** a proposed algorithm has been adopted by modifying the original HHT. First, we adopt a *mirror approach* to estimate the outside extrema in the EMD. Furthermore, we define a window around the original center whose size is about 50-75% of the original one by just evaluating the value with the effective range to addressing the boundary effect problem.

Second, to handle the mode mixing problem, we adopt Rilling's stopping criteria in the EMD processing and add a filter-bank pre-processing stage. The mode amplitude is $a(t) = (x_U(t) - x_L(t))/2$, and the evaluation function is $\sigma(t) = |m(t)/a(t)|$. Then the stopping criteria can be written as

- $\sigma(t) < \theta_1$ for some prescribed fraction $(1-\alpha)$ of the total duration
- $\sigma(t) > \theta_2$ for the remaining region.

In above, θ_1, θ_2, and α are three pre-decided parameters and are usually set to as 0.05, 0.5 and 0.05, respectively.

Third, to address the problem of sub-harmonics and partials, a post-processing has to be adopted. We discard some undesired frequency bands whose energy id less than 10 % of the original signal so that gives us the smaller range of finding the interested frequency band containing F0. The last step is use AFC to estimate every F0 in each IMF. The processing model is illustrated in **Fig. 4.2**. And the simulation results between the original HHT and the modified HHT is shown as **Fig. 4.3**.

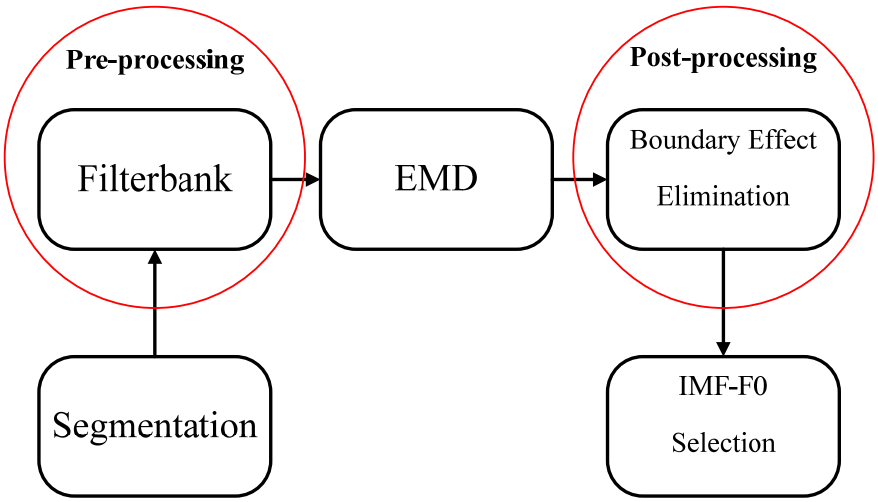


Fig. 4.2 The modified HHT process for fundamental frequency estimation.

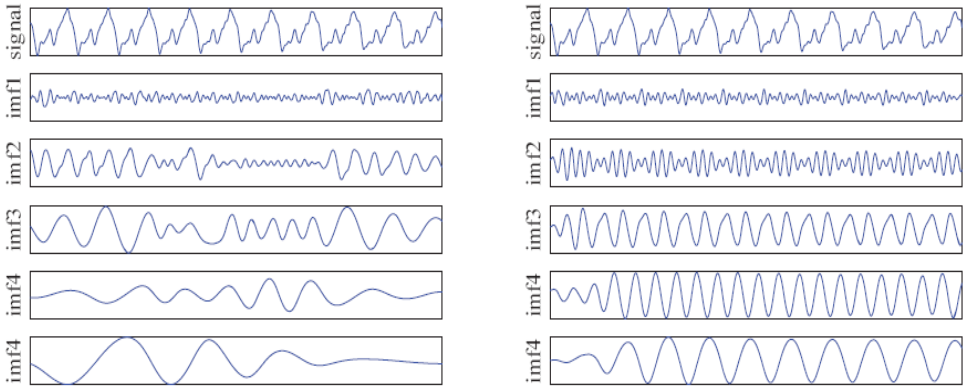


Fig. 4.3 IMFs of C4 (261Hz) by shifting (a) without and (b) with the filterbank pre-processing [14].

4.3 Sub-Harmonic Summation

This algorithm is proposed in reference and to detect the pitch of the signal. It origins from that each spectral peak increase the perception of a pitch corresponding to the frequency of the component. The point here is that higher spectral components (peak) of a signal contribute less in human perception than lower components do. Based on above principle, all these contributions add up in a sub-harmonic summation and the maximum of the sum is defined as the pitch. In common case, the fundamental

frequency is too weak for us to detect or misses in the signal. This mechanism creates a virtual pitch produced by this algorithm. Virtual pitch is common in human perception. The algorithm is implemented in original paper and given by a single expression,

$$H(s) = \sum_{n=1}^{N} h^{n-1} W(s + \log_2 n) \, | \, S(s + \log_2 n) \, | \tag{4.5}$$

Where $s = \log f$ denotes the logarithmic frequency which represents the sub-harmonic sum spectrum, N (e.g., 15) denotes the number of harmonics, n is the compression rank, h (e.g., 0.84) denotes the decreasing factor, $W(s)$ is an arc-tangent function which represents the transfer function of the auditory sensitivity filter, and $| S(s + \log_2 n) |$ denotes the compressed amplitude spectrum representation. An estimation of pitch is the value of $f = 2^s$ that S is chosen when $H(s)$ reaches the maximum value.

4.4 Harmonic Elimination Method

Each Pitch of music signals is composed of the F0 and its harmonics. However, we have to eliminate the harmonics of the fundamental frequency to simplify the analysis of music signals. As we know, harmonics of the note are integer multiples of this frequency. In this algorithm, it mentions two stages: the first stage eliminates the harmonics of the music signal and only reserves the fundamental part (the first harmonic). The second stage estimates and tracks the F0 of the music signals by means of an *Extended Kalman Filter* (**EKF**). In this step, we will identify pitch and duration time of each note. The best advantage for this method does not need to face window size problem and is very suitable for on-line music transcription.

4.4.1 Elimination of Harmonics

A. General Model

A well-known approach to model music signals that assumes a additive sinusoidal plus residual and this model can be formulated as

$$y(t) = s(t) + n(t) \tag{4.6}$$

where $n(t)$ is a broad-band stationary signal (e.g., a white noise) and

$$s(t) = \sum_{k=1}^{N} A_k(t)\cos(k\omega_k(t)\times t + \varphi_k(t)) \tag{4.7}$$

Denote that $s(t)$ by the pure musical part of signal, $A_k(t)$ and $\omega_k(t)$ by the representatives of time-varying amplitude and frequency of harmonics, respectively. $\omega_1(t)$ is the fundamental frequency, N is the number of harmonics, and where $\varphi_k(t)$ represents trimbre variations and performance effects.

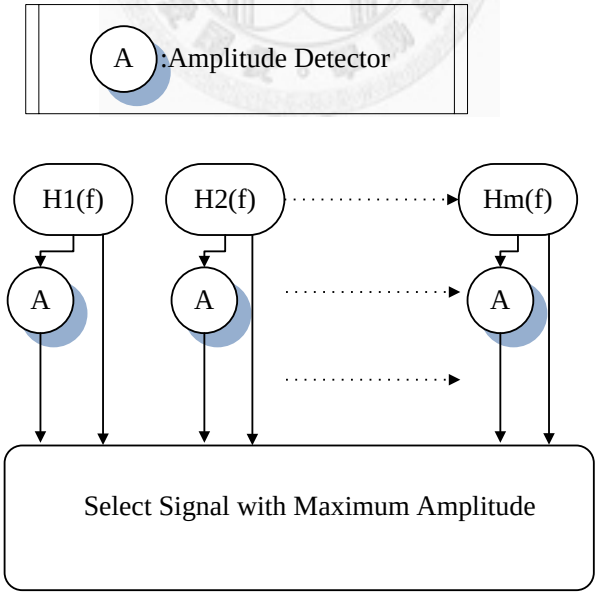


Fig. 4.4 System block diagram for harmonics elimination.

B. Elimination of Harmonics

In music notation, we sound each note with pronunciation as *Do Re Mi Fa So La Si* orderly with an octave. An octave is defined as from *Do* to *Si*. In [16], the frequency has the same note name with its octave. However, the frequency of its second harmonics of a note is not in the range of its octave. Therefore, we can use this feature to design many band-pass to filter the undesired frequency band. Therefore, the output of the system will be:

$$z(t) = A_1(t)\cos(\omega_1(t)\times t + \varphi_1(t)) + n'(t) \tag{4.8}$$

4.4.2 Extended Kalman Filter (EKF)

Frequency Tracking

Next, the step is to do frequency tracking. Given the state-space model as in (4.9), the EKF technique can be applied straightforwardly. And the complete set of equations of the EKF frequency tracker derived from (4.9) is here presented as in (4.10).

$$\begin{cases} x_1(t+1) = x_4(t)[x_1\cos(x_3(t)) - x_2\sin(x_3(t))] \\ x_2(t+1) = x_4(t)[x_1\sin(x_3(t)) + x_2\cos(x_3(t))] \\ x_3(t+1) = x_3(t) + w(t) \\ x_4(t+1) = x_4(t) + u(t) \\ \quad z(t) = x_1(t) + v(t) \end{cases} \tag{4.9}$$

$w(t)$, $u(t)$ and $v(t)$ are zero-mean uncorrelated white noises, having variances q , p and r , respectively. The state variable $x3$ (t) represents the unknown frequency $w(t)$ that determines the name of the note.

$$\begin{cases} x_1(t+1) = x_4(t)[x_1\cos(x_3(t)) - x_2\sin(x_3(t))] \\ x_2(t+1) = x_4(t)[x_1\sin(x_3(t)) + x_2\cos(x_3(t))] \\ x_3(t+1) = x_3(t) + w(t) \\ x_4(t+1) = x_4(t) + u(t) \\ z(t) = x_1(t) + v(t) \end{cases} \tag{4.10}$$

Where

$$f(\hat{x}(t|t)) = \begin{bmatrix} \hat{x}_4(t|t) = \hat{x}_4(t)[\hat{x}_1(t|t)\cos(\hat{x}_3(t|t)) - \hat{x}_2(t|t)\sin(\hat{x}_3(t|t))] \\ \hat{x}_4(t|t) = \hat{x}_4(t)[\hat{x}_1(t|t)\cos(\hat{x}_3(t|t)) + \hat{x}_2(t|t)\sin(\hat{x}_3(t|t))] \\ \hat{x}_3(t|t) \\ \hat{x}_4(t|t) \end{bmatrix} \tag{4.11}$$

$$H = [1\ 0\ 0\ 0] \quad C = [0\ 0\ 1\ 0]$$

And

$$\hat{x}(t|t) = \begin{bmatrix} \hat{x}_1(t|t) \\ \hat{x}_2(t|t) \\ \hat{x}_3(t|t) \\ \hat{x}_4(t|t) \end{bmatrix}, F(t) = \frac{\partial f(x)}{\partial x}\Big|_{x=\hat{x}(t-1|t-1)} \tag{4.12}$$

$$I_{11} = diag(0,0,1,1)$$

Finally, the fundamental frequency can be estimated by using the equation (4.12).

Chapter 5 Melody Matching

After the pitches of the sung signal are estimated, we can transfer the frequency to a certain MIDI number. Then, comparing the numeral sequence of sung input with those in database, if any song which gets the higher scores in similarity matching, it would be selected as the best-matching song. However, nothing is always perfect due to the distortion from human humming. Some common singing situations are listed below:

- **People sing any part of the melody**.
- **People sing at the wrong key**.
- **People sing at a reasonably correct global tempo**.
- **People sing too many or too few notes**.
- **People sing the wrong intervals or confuse some with others**.
- **People sing the contour reasonably accurately**.
- **People with singing experience sing better on some aspects than people without singing experience do**.
- **People sing familiar melodies better than less familiar ones**.

5.1 Dynamic Programming Algorithm

This method stems from the GA algorithm [20] to find the optimal alignment for DNA sequences. Likewise, it can use the same method to compare a number of sequences of MIDI number with its database to find the desired song. Unlike GA, dynamic programming for QBH is a slightly different from GA algorithm. The sequence in GA is all symbolic representations like {A, T, G, C}. If the symbolic representations do not hit

the same symbolic representation at proper positions, system will judge those are wrong hitting. Unlike GA, dynamic programming in music similarity matching is according to the difference degree of the pitches between two consecutive notes. In following, the common used dynamic programming for melody matching is introduced.

A string is a sequence of symbolic representations drawn from an alphabet. The best alignment is found by comparing the similarity between string A and string B with the lowest cost (or, equivalently, highest reward). Dynamic programming technique has been used to align gene sequence based on a common ancestor for over 30 years and has also been used widely for musical query matching in recent years.

5.1.1 Implementation of Dynamic Programming Algorithm

This algorithm is a typical string alignment, or string-matching algorithm. Denote the length of a string, *S*, as |S|. Let Q denote the querying string and T denote the target string. Based on the assumption, both two strings are composed of characters drawn from the same alphabet. Create a matrix *AlignScore* with |Q|+1 rows and |T|+1 columns where *AlignScore(i,j)* is the score of the best alignment between the initial segment q_1 through q_i of Q and the initial segment t_1 through t_i of T. This matrix is initialized by setting *AlignScore(0,0)=0.* The Other elements are filled in *AlignScore(i,j)* with a value, where i and j denote row and column, respectively. The value is decided by using (5.1). The top line in (5.1) give a reward to the cell (i,j) when q_i is equal to t_j and it also gives a penalty to the cell (i,j) when q_i does not match to t_j. The following two lines mean that skip happens to target elements or query elements. As we can see, the middle line calculates the penalty for skipping target element t_j and the lowest line gives the penalty for skipping query element q_i. $AlignScore(i, j)$, $matchScore(q_i, t_j)$

and *Penalty* are defined as follows,

$$\begin{aligned} &AlignScore(i, j) \\ &= \max\left\{\begin{array}{l} AlignScore(i-1, j-1) + matchScore(q_i, t_j) \\ AlignScore(i-1, j) - skiPenaltyT\arg et(t_j) \\ AlignScore(i, j-1) - skiPenaltyQuery(q_i) \end{array}\right\} \end{aligned} \tag{5.1}$$

$$matchScore(q_i, t_j) = \begin{cases} 2, & \text{if } q_i = t_j \\ -2, & \text{otherwise} \end{cases} \tag{5.2}$$

$$\begin{aligned} skipPenaltyQuery(q_i) &= 1 \\ skipPenaltyT\arg et(t_j) &= 1 \end{aligned} \tag{5.3}$$

Where $matchScore(q_i, t_j)$ function means the reward of a match or a mismatch for positive two points or negative two points, respectively, and Penalty cost in (5.3) are defined to punish that when a skip happens. An illustration is given in Table 5.1 that represents a *AlignScore* matrix for query sequence “G D A C B” and target sequence “G A B B”.

The directions which arrows point to can be viewed as the routes for tracing backwards through the series of parents used to generate the score. The score for the global alignment is calculated by the value in the right-hand, lower corner of the table. In this case, the score is 3 as the table shows. As we can see from the Table 5.1, there are four routes in this alignment matrix. Four maximal-scoring alignments between query and target are showed in Fig. 5.1. Here, a dash is denoted as a skip.

The directions which arrows point to can be viewed as the routes for tracing backwards through the series of parents used to generate the score. The score for the global alignment is calculated by the value in the right-hand, lower corner of the table. In this case, the score is 3 as the table shows.

Table 5.1 Alignment matrix.

Target / Query		G	A	B	B
	0	-1	-2	-3	-4
G	-1	**2**	1	0	-1
D	-2	**1**	0	-1	-2
A	-3	0	**3**	**2**	-1
C	-4	-1	**2**	**1**	0
B	-5	-2	1	**4**	**3**

As we can see from the Table 5.1, there are four routes in this alignment matrix. Four maximal-scoring alignments between query and target are showed in Fig. 5.2. Here, a dash is denoted as a skip.

TARGET:	G - A - B B	G - A - B B	G - A B B	G - A B - B
QUERY:	GDA C B -	GDA C - B	G DA C B	G DA - C B

Fig. 5.1 Four maximal-scoring alignments**.**

5.1.2 Complexity of Dynamic Programming

Denote that Q is a querying sequence and T is a target sequence. $\{T_1, T_2, T_3, ..., T_n\}$ is a set of target sequences form database. Each of T_i may be the optimal matching

sequence to the querying one, so every T_i needs to be compared with the querying sequence. Dynamic programming is used to run the alignment algorithm between Q and each target, T_i, and then a ordered set of values will be returned. The computational complexity of finding the best-matching sequence takes time of comparing each single target, and all the targets have to be checked, then the matching processing is finished. The complexity of finding the most similar target is given by *equation reference goes here*.

$$\begin{aligned} StringMatchComplexity &= O(\sum_{i=1}^{N} |Q| \bullet |T_i|) \\ &= O(N|Q|mean(|T|)) \end{aligned} \tag{5.4}$$

Where $|Q|$ is the length of Q, $|T|$ is the length of T and $|T_i|$ is the length of T_i. The complexity of dynamic programming is large and leads to long searching result.

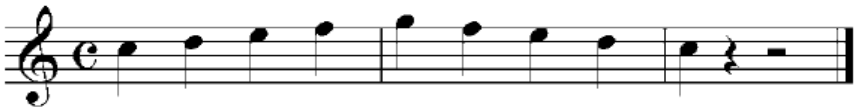
5.2 Hidden Markov Model

With the information of IOI and pitch interval, we regard a certain combination of IOI and pitch interval of one note as a situation. Because the notes are consecutive, there is a transition model in a piece of music. Therefore, every music piece has its own hidden Markov Model and we can use this model to establish the maximum likely-hood probability and search the optimal matching of our database. As we know, Markov Model is a probability-transition cycle. Markov Model (MM) [20] consists of a series of specific states which are characterized by duples < *pitchInterval, IOIratio* > mentioned in this section. Each state has a transition probability to the other states.

A Markov Model represents a process going through a sequence of discrete staes such as notes in a melody. Each song on database has its own Markov Model which is created by the feature of the notes of the song itself. Below are four basic elements to form a Markov Model:

- A set of states, $S = \{s_1, s_2, s_3 ....., s_n, \}$
- A set of transition probabilities, *T*, where each $t_{i,j}$ in *T* represents the probability of a transition from s_i to s_j.
- A probability distribution, π, where π_i is the probability the automation will begin in state s_i.
- *E*, subset of S containing the legal ending states.

In this model, the probability of transitioning from one state to another state is assumed to depend only on the transition of current state. That is the so-called Markov property. A simple graph is illustrated in Fig. 5.3 from the note events of Fig. 5.2.



Delta pitch	2	2	1	2	-2	-1	-2	-2
IOI	1	1	1	1	1	1	1	1
IOI ratio	1	1	1	1	1	1	1	1
State	α	α	β	α	χ	δ	χ	χ

Fig. 5.2 Features for a passage of a piece of music.

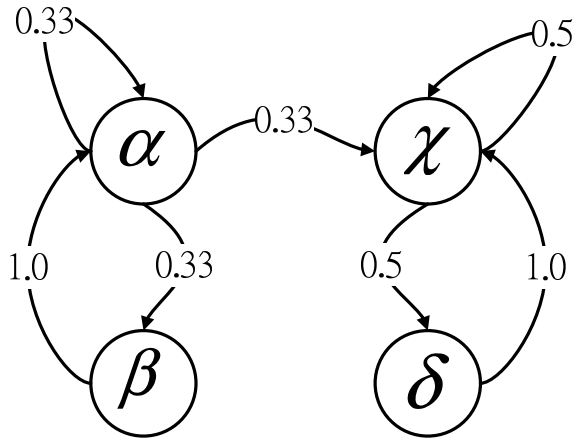


Fig. 5.3 Markov Model for the case from Fig. 5.2.

States are composed by duple, < *pitchInterval, IOIratio* >. In Fig. 5.3, nodes represent states and the edges indicate transition probabilities and the transition with zero probability are not shown. However, the transcription system may introduce pitch errors, such as errors from wrong onset detection and octave displacement. That is, if a zero-probability transition indeed happens, there would be no best estimation for the right matching. Therefore, constructing a hidden Markov model (**HMM**) is necessary.

5.2.1 Implementation of Hidden Markov Models

A model that explicitly maintains a probability distribution over all possible observations for each state is called a hidden Markov model (**HMM**). That means no zero-probability transition exists due to every transition may happens even though it has a very minimal probability. In addition to the four basic principles based on **MM**, there are two more constraints and requirements for the **HMM**.

- A set (alphabet) of possible observations, $A_{observe} = \{o_1, o_2, o_3, ..., o_n\}$
- A probability distribution over the set of observations for each state in S.

In this approach each musical theme in database are represented as hidden Markov model. Then, a query is calculated its own probability form the query's notes transition by referring to the hidden Markov model of itself. Each observation is a note-transition duple, < *pitchInterval, IOIratio* >.

From above paragraph mentioned, making a proper hidden Markov model from MIDI is very important. A query is a sequence of observations and $A_{observe}$ is used to correspond directly to A_{query} for the string matchers. In HMM system, observations consist of duples < *pitchInterval, IOIratio* >. Here, a simple example is taken in the following description. As we know, there are 12 musical half steps in an octave. It is assumed that every piece of music has no more than an octave between notes, that is there are 25 possible *pitchInterval* values, corresponding to all possible transitions whose range does not exceed an octave. In this case, *IOIratio* is quantized to five values. This means that there are 125 possible observations due to the product of 5 and 25. Therefore, the resulting table has over 15000 entries, exactly to say, it has 15625 entries inside a square matrix.

After establishing the HMM table, the probability of each entry has to be estimated from the duples of every music theme in database. Given a state, s, the probability of observation o_i may be estimated by the count of how many times does o_i happen and is compared to the total number of times s is encountered.

$$P(o_i \mid s) = \frac{count(o_j, s)}{\sum_{j=1}^{|o|} count(o_j, s)} \tag{5.5}$$

However, creating a large amount of data of observations and hidden states is not easy to deal with. Hence, it is more tractable by assuming conditional independence between

pitchInterval and *IOIratio.* There is no obvious correlation between *pitchInterval* and *IOIratio* so that we assumed both of them are independent. Then, the probability of encountering any observation duple can be derived form (5.5):

$$P(o \mid s) = P(pitchInterval_o \mid pitchInterval_s) \times P(IOIratio_o \mid IOIratio_s) \qquad (5.6)$$

From equation (5.6), we can see that the probability of evaluating from one observation to another observation is simplified. There are observations that do not occur in the training data and we could not assume this observation will never be observed in an actual query. Thus, a minimal probability, P_m, will be used to replace all cells whose value is zero in the resulting table. Probabilities are then normalized so that the sum of all observation within one state is maintained to be equal to one.

Here, we take an actual case for example. Now a sequence of strings is $\{a,b,c,c,d\}$ and its Markov model is constructed as follows. From the string $\{a,b,c,c,d\}$, the Markov transition can be illustrated as in Fig. 5.4.

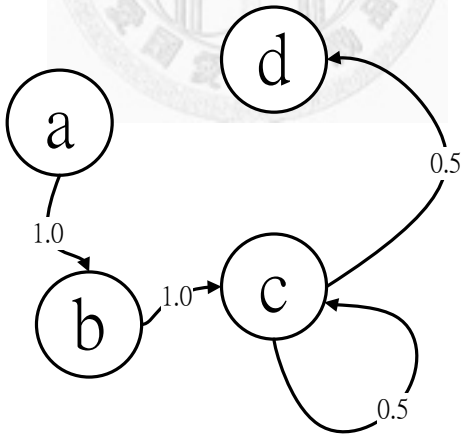


Fig. 5.4 Markov model (MM) for string $\{a,b,c,c,d\}$

Now, there are two querying string and they are $\{a,b,a,c,d\}$ and $\{b,b,e,c,a\}$, respectively. From these two query sequences, we assume that all elements are

$\{a,b,c,d,e\}$. That means there 5 elements in the whole event. However, the reference Markov model does not have state, “e”. It’s due to some reason that errors may happen from the source of reference string or some wrong estimation of input sequence. Hence, some observation may not estimated in querying string and the potential hidden state should be taken into account to recreate the model, hidden Markov model (HMM). The HMM is shown in Fig. 5.5 .

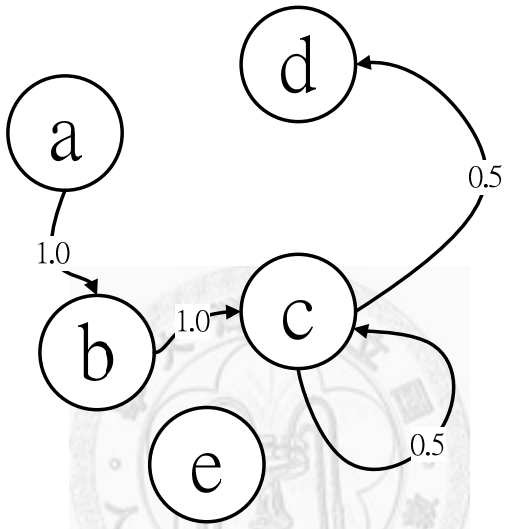


Fig. 5.5 Hidden Markov model (HMM) for string $\{a,b,c,c,d\}$

It seems no big difference from Fig. 5.4 but it represents a significant meaning. Edges in Fig. 5.5 mean the transition probability and direction. No transition edge means there is no transition probability between two states. According to this HMM, a probability-transition table can be generated. From Table 5.2, there are still many zero value in this resulting matrix. Zero value has to be replaced with a minimal value due to the transition may just be hidden and con not be observed directly. Therefore, setting a minimal probability is essential and the errors may be lower down and re-align the probability distribution in Table 5.3.

Table 5.2 Observation probability table.

From / To	a	b	c	d	e
a	0	0	0	0	0
b	1	0	0	0	0
c	0	1	1/2	0	0
d	0	0	1/2	0	0
e	0	0	0	0	0

Table 5.3 The result of realigning for Table 5.2.

From / To	a	b	c	d	e
a	0.1	0.1	0.1	0.2	0.2
b	1.0	0.1	0.1	0.2	0.2
c	0.1	1.0	0.5	0.2	0.2
d	0.1	0.1	0.5	0.2	0.2
e	0.1	0.1	0.1	0.2	0.2

What number to replace zero is worth to discussing. Here, we exchange 0 with a very small value. Take the case of Table 5.2 for instance, we set the value of P_m is 0.1. Because the row string means the states and column string means the state-to-state observations, the sum of probability has to be equal to 1. Finally, normalizing is necessary for maintain the Markov model's principle and the final result is shown in

Table 5.4.

Table 5.4 The result of normalizing for Table 5.3.

From / To	a	b	c	d	e
a	0.071	0.071	0.076	0.2	0.2
b	0.714	0.714	0.076	0.2	0.2
c	0.071	0.071	0.384	0.2	0.2
d	0.071	0.071	0.384	0.2	0.2
e	0.071	0.071	0.076	0.2	0.2

Given that the two queries are $\{a,b,a,c,d\}$ and $\{b,b,e,c,a\}$, we want to calculate which is more similar to the string $\{a,b,c,c,d\}$. From $\{a,b,a,c,d\}$, we know the routes are $a \to b$, $b \to a$, $a \to c$ and $c \to d$ where $a \to b$ means value in column a and row b. Therefore, the matching probability of $\{a,b,a,c,d\}$ is about 0.027. From $\{b,b,e,c,a\}$, we know the routes are $b \to b$, $b \to e$, $e \to c$ and $c \to a$. Calculated by the same rule, the matching probability of $\{b,b,e,c,a\}$ is about 0.015. Hence, string $\{a,b,a,c,d\}$ is judged to be more similar to the string, $\{a,b,c,c,d\}$.

5.2.2 Complexity of Hidden Markov Model

The themes in our database are encoded as HMMs and the query is treated as an observation sequence. Given this, we are interested in finding the most likely target from HMM to generate the query sequence.

Denote that Q is a querying sequence and T is a target sequence.

$\{T_1, T_2, T_3, ..., T_n\}$ is a set of target sequences of database. $|S_i|$ is the number of states in the hidden Markov model of target, T_i. Then, the HMM size of T_i is $|S_i|^2$.

Each HMM of T_i is constructed by its state transition and the size of HMM is $|S_i|^2$. According to the state transition of Q, we can get a transition probability from each HMM of T_i. Then, an ordered set of probability will be returned. The computational complexity of finding the best-matching sequence takes time of comparing each single target, and all the targets have to be checked, then the matching process is finished. The complexity of finding the most similar target is given by.

$$\begin{aligned} HMMmatchComplexity &= O(\sum_{i=1}^{N} |Q| \bullet |S_i|^2) \\ &= O(N|Q|mean(|S|)) \end{aligned} \tag{5.7}$$

Where $|Q|$ is the length of Q, $|T|$ is the length of T, $|T_i|$ is the length of T_i and N is the number of targets.

Chapter 6 Representations for Musical Feature

How to express a note has became significant in music information retrieval (**MIR**). From [35], the system block contains three important parts: onset detection, pitch estimation and melody matching. The features extracted by previous step can be used as a symbolic representation for music notes. However, humming queries are different form people even they sing the same song. The difference may come from the time duration of notes in melody, tonality of melody and pitch accuracy of melody.

In order to solve the problem of time duration between notes, we use onset detection in Chapter 3. Onset detection makes every note independent from any other notes. Assumed that the signal is cut in perfect onset interval, the difference of time duration for each person has become not significant. Because we can extract the fundamental frequency form each cut note and turn the segment of the signal into MIDI number. No matter what you sing in fast speed or in slow speed for a song, onset detection solves such problem.

There is another problem: how to compare two melodies if the tonality of query is different from the target but both are the same song. There are two methods: music scale modeling and using melody contour. We will introduce music melody modeling first and detail its method to match two songs with different tonality. Later, we will introduce how melody contour make melody matching robust.

The common features used in MIR are pitch and IOI. In this section, we mainly discuss how to use pitch to match melody and have a brief introduction to IOI. In this paper, our proposed is mainly based on pitch interval.

6.1 Music Scale Modeling for Melody Matching

Due to the key transposition issue in melody matching, the scale modeling method is proposed in [35] to find the proper key when doing melody matching. Most music pieces are created by some particular music scales. The most common used scales are Major scale and Minor scale.

Although the hummer may not know its music scale, the scale is decided when the tune is hummed. As we know, each octave has 12 notes mentioned in [35]. The 12 notes are conventionally named as: A, A#, B, C, C#, D, D#, E, F, F#, G, G#. A music scale is a series of notes defined by musicians as appropriate for a music piece. A particular scale is defined as a series of pitch intervals between the series of notes in one octave. The most widely used two scales are Major scale and Minor scale. Both of these two scale are on one octave. The pitch intervals for Major scale are: 2 2 1 2 2 2. And the pitch intervals for a Minor scale are: 2 1 2 2 1 2 2. Besides, a scale is usually referenced to a root scale (e.g. C). The illustration is shown in Fig. 6.1.

For example, Major C scale whose root note is note C consists note C, D, E, F, G, A , B, Major D scale consists note D, E, F#, G, A, B, C#, and Minor A scale consists note A B C D E F G.

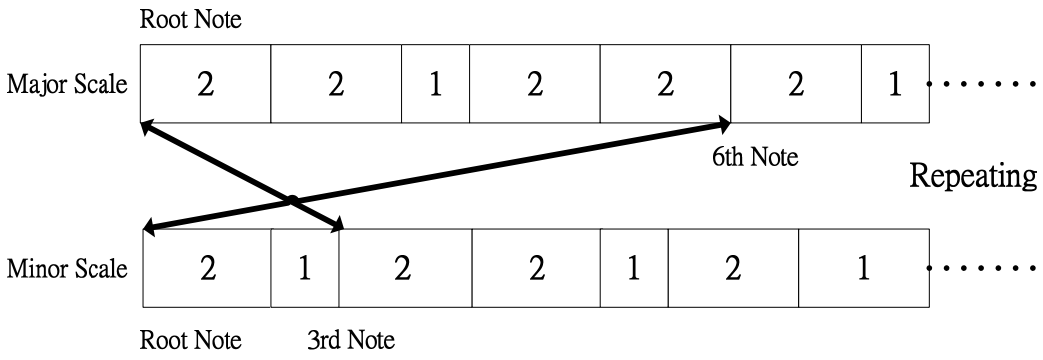


Fig. 6.1 Pitch intervals in Major scale and Minor scale.

Fig. 6.1 illustrates the Major scale and Minor scale. It can be seen as a Major scale

coincides with a Minor scale with its 6th note as the root note while a Minor scale coincides with a Major scale with its 3rd note as the root note. Due to this property, we only need to estimate the root of one scale (say the major scale) and use the root note to do key transposition. With rot note, we can adjust the key of query as same as the key of target. In the following figure, represents a music scale for Major and Minor scale.

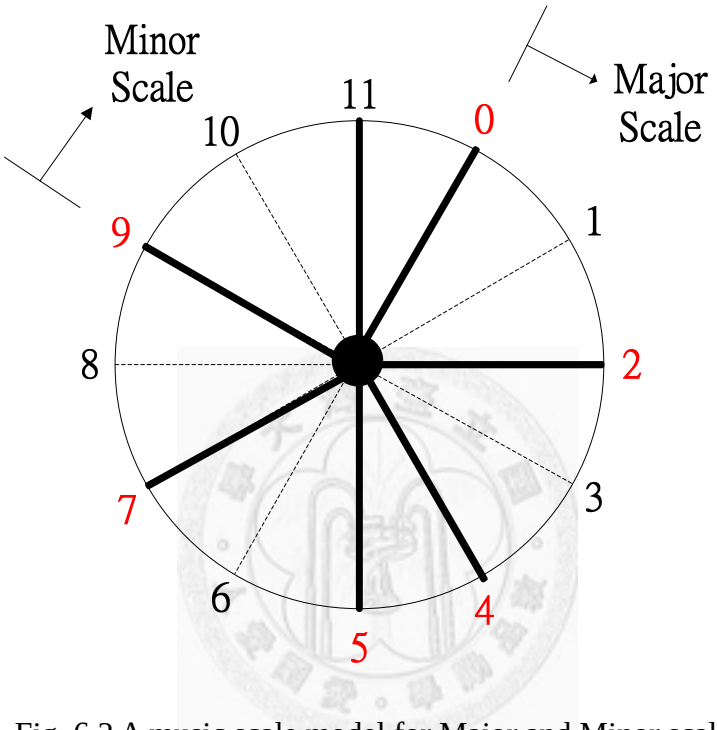


Fig. 6.2 A music scale model for Major and Minor scale.

The model in Fig. 6.2 is proposed in [35] and the music scale model is a circle of 12 equally spaced notes. Although the music scale information is usually not encoded in the music data file, ether in MIDI or waveform formats. The red highlighted notes labeled by (0, 2, 4, 5, 7, 9, 11) correspond to the note in the music scale (Major or Minor) which are called *scale notes*. The notes labeled by (1, 3, 6, 8, 10) correspond to the notes outside of the scale, and we call each of them a *non-scale* note.

A Major scale is starting from note 0 by clockwise enumeration while a Minor scale is the list starting from note 9. If a passage of a melody can fit into the model, the

note in the query corresponding to note 0 of the model is then the root note of a major scale. How to estimate the major scale root of a melody has two main formats: symbolic and acoustic. And both method consists 2 steps: (a) construction of note histogram for the melody; (b) comparing of the note histogram with the scale model and locating the root note.

The first step is since music scale is cyclic in an octave, the notes of melody are mapped into the model in Fig. 6.2 by constructing a note histogram with 12 bins which are numbered from 0 to 11 by using (6.1):

$$B = MOD(P,12) \tag{6.1}$$

where P is the note number (from 0 to 127), B is the bin number (from 0 to 11), and MOD is the modulus operation. In the second step, the note histogram is compared with the scale model by following a clockwise alignment of the histogram bins with the notes of the scale model. If the melody of a song is composed by the notes: 58, 63, 62, 63, 67, 65, 63, 65, 67, 65, 63, 67,…And the note histogram of this song is [23], [0], [3], [50], [0], [37], [0], [30],[0], [0], [27], [0].

Table 6.1 Error fitting.

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

The scale model fitting errors are zero for bin 2 in Table 6.1 and that means the note in bin 2 is estimated to be the root note for major scale. The fitting errors are also small for bin 4 and bin 9 so that notes in bin 4 and bin 9 are also estimated as the roots of major scale. Multiple root notes are claimed for this melody and it mainly contains 5 scale notes from the note histogram.

According the scale modeling method, the root note can be estimated so that we

can adjust the key of query to as same as target and for every possible root notes. However, we have to know the root note of targets first. Obviously, this method is still to be improved.

6.2 Pitch Interval and IOI for Melody Matching

Using pitch interval is a straightforward concept and is more easily implemented than scaling model. The definition of pitch interval is explained in [35]. But here, pitch interval for melody matching represents pitch interval between two contiguous notes only.

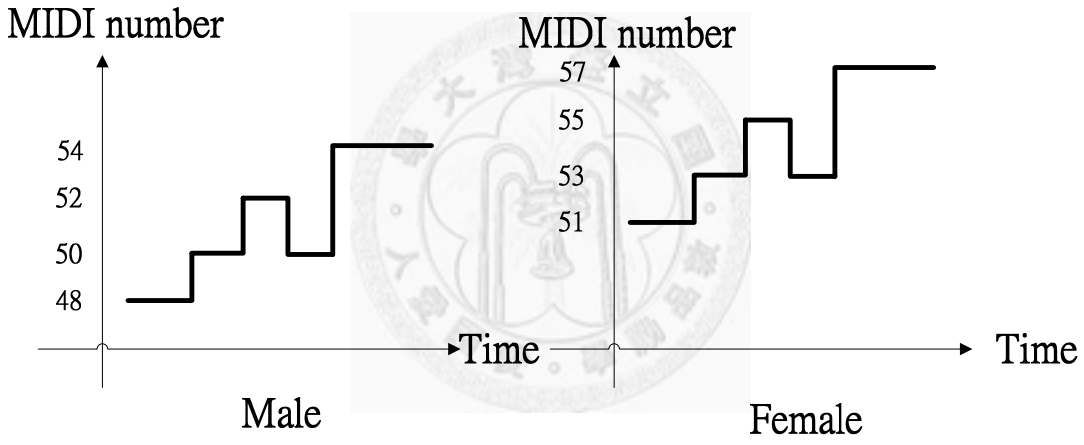


Fig. 6.3 A time series of pitch of the same song produced by a male and a female.

Figure above shows the time series of pitch by a male and a female person for the same song. Generally speaking, the key of a female is usually higher than the key produced by a male for the same song. If we use “pitch” as the feature, the situation in Fig. 6.3 won’t be detected as the same song. However, observing from this illustration, it can be found that the melody contour is the same by comparing two plots. Although query is always not perfect, the melody contour should be the most similar to its aim target from the music database.

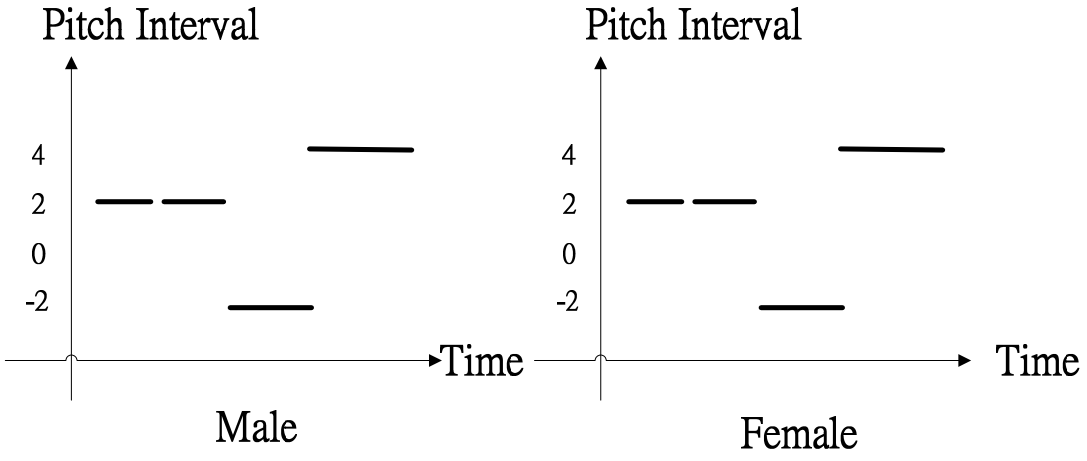


Fig. 6.4 A time series of pitch interval of the same song produced by a male and a female.

By subtracting every pitch by its previous contiguous pitch of a query, the series of pitch intervals is computed as shown in Fig. 6.4. Another feature, inter-onsets-intervals (**IOI**), is often used as a feature to do melody matching. As the same meaning as its name, IOI, it means a series of time duration of every two contiguous onsets. However, the speed of singing a song is different by people. Therefore, only using IOI is adequate, as the same concept as pitch interval, the feature is modified as IOI ratio. Pitch interval is robust for any tonality for the same song while IOI ratio is robust for different singing speed for the same song. Whether a song is sung in fast speed or slow speed, the ratio of the time duration of every two contiguous notes is changing slightly.

In our QBH system, we have to create a music database to provide targets for query to search. How to construct a music database is introduction in next section. We only use pitch interval as our feature due to the data of IOI ratio is not easy to obtain. Another reason is there is no pitch interval standard source for us to use and some sources even require charge.

6.3 Construction of Music Database

Because the pitch interval source is difficult to obtain, we construct a music database by ourselves. In targets, we use numbered notes as our database file as shown in Fig. 6.5.

‖1 2 3 1 | 1 2 3 1 | 3 4 5 - | 3 4 5 - ‖

兩隻老虎　兩隻老虎　跑得快　跑得快

‖56 54 3 1 | 56 54 3 1 | 1 5. 1 - | 1 5. 1 - ‖

一隻沒有眼睛 一隻沒有嘴吧 真奇怪 真奇怪

Fig. 6.5 The numbered notes of the Chinese children's song "兩隻老虎".

The number in Fig. 6.5 is so-called numbered notes which are easy for human to memorize the rhythm of a song. We assume the pitch range of a melody is usually within 3 octaves. In numbered notes, 1 denotes Do, 2 denotes Re, 3 denotes Mi, 4 denotes Fa, 5 denotes Sol, 6 denotes La and 7 denotes Ci. Therefore, the series of pitch interval can be found from the numbered notes. The only information we want to know is pitch interval, and we don't need to know the query's exact pitches. Bases on this principle, we calculate pitch difference by relative pitch interval in Table 6.2.

In Table 6.2, there are 6 columns and among them, column 1,3 and 5 represent the numbered notes within 3 octaves in our assumption. The temporary number has no sense itself but the pitch interval can be obtained by subtracting two temporary numbers. Take the case in Fig. 6.5 for example, this song is with 2 octaves. All notes are in current octave except note "**5.**" which denotes Sol in its lower octave. If we take the current octave as the second octave in Table 6.2, then we will get a series of temporary numbers as follows: 12 14 16 12 12 14 16 12 16 17 19 16 17 1919 21 19 17 19 21 19 17 16 12 19 21 19 17 16 12 12 7 12 12 7 12. After subtracting by two contiguous

temporary numbers, the pitch interval series are: 2 2 -4 0 2 2 -4 4 1 2 -3 1 2 0 2 -2 -2 -1 -4 7 2 -2 -2 -1 -4 0 -5 5 0 -5 5.

Table 6.2 The temporary number look-up table for numbered notes.

Numbered Note (1st)	Temporary number	Numbered Note (2nd)	Temporary number	Numbered Note (3rd)	Temporary number
1	0	1	12	1	24
2	2	2	14	2	26
3	4	3	16	3	28
4	5	4	17	4	29
5	7	5	19	5	31
6	9	6	21	6	33
7	11	7	23	7	35

In query, after detecting the onset times, we extract each pitch of note segmentation. The pitch is computed by using the FFT to estimate the fundamental frequency and the detail method will be explained in section 7.2. After estimating the fundamental frequency of each onset, the frequency can be transferred into MIDI number by using equation (2.2). With the information of a series of MIDI number, the pitch interval can be computed as well. Finally, the information of pitch interval of query and targets are known, the similar matching score can be evaluated by the third step, melody matching to find the best top n rank songs (n denotes the rank size).

Chapter 7 Proposed Algorithm

After introducing several onset detection methods and melody matching methods, we proposed approaches to improve onset detection to advance the true positive rate and improve the melody matching to tolerate some errors like missing few pitches, adding few pitches or singing few wrong pitches. At first part, we will present the proposed onset detection and present the proposed melody matching at the second part.

In Fig. 3.2, there are only 12 notes in this querying signal. Several onset detections are mentioned in Chapter 3 and their detection result are shown as we can see. In Fig. 3.5, the differences of magnitude method detects 16 notes and exceeds 4 more notes than the original one as circled by red circles in Fig. 7.1.

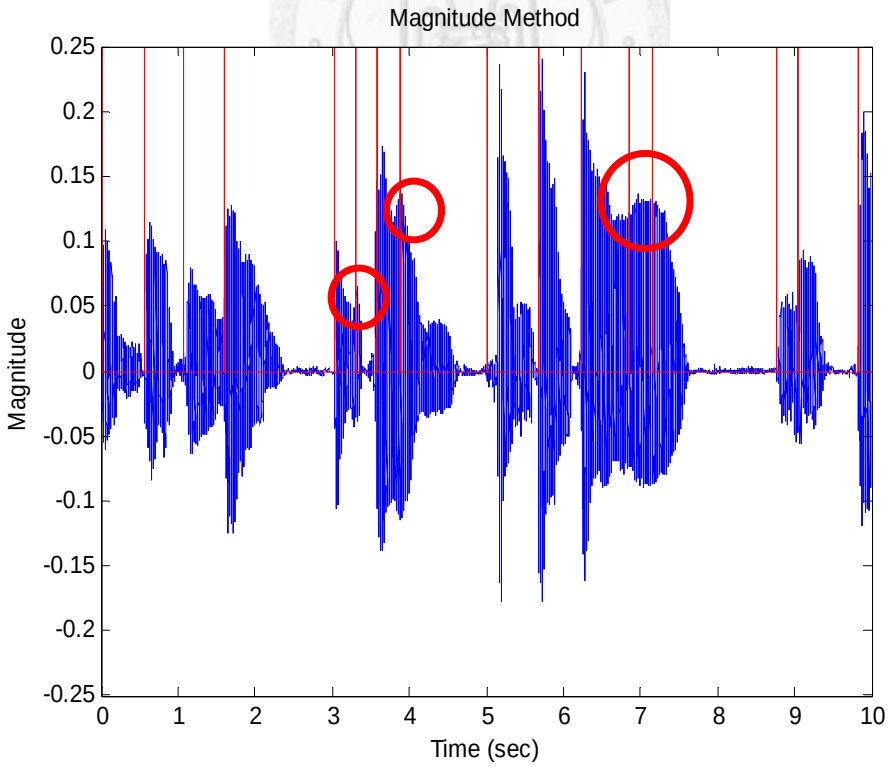


Fig. 7.1 Over detected result in magnitude method.

That is due to the reason that in this method, magnitude differences are highly

depended. Therefore, we found that the over-detected onsets are always have a abrupt rise. Even though these are just sounds in the end of pitches, they may cause wrong detections.

In short-tern energy method, it often causes under-detected result as in Fig. 7.2. That is because we use the property of silence whose energy is close to zero. Therefore, if there is no silence between two notes, then the onset may not be detected as Fig. 7.2 shows and the red circles are onsets which are missed.

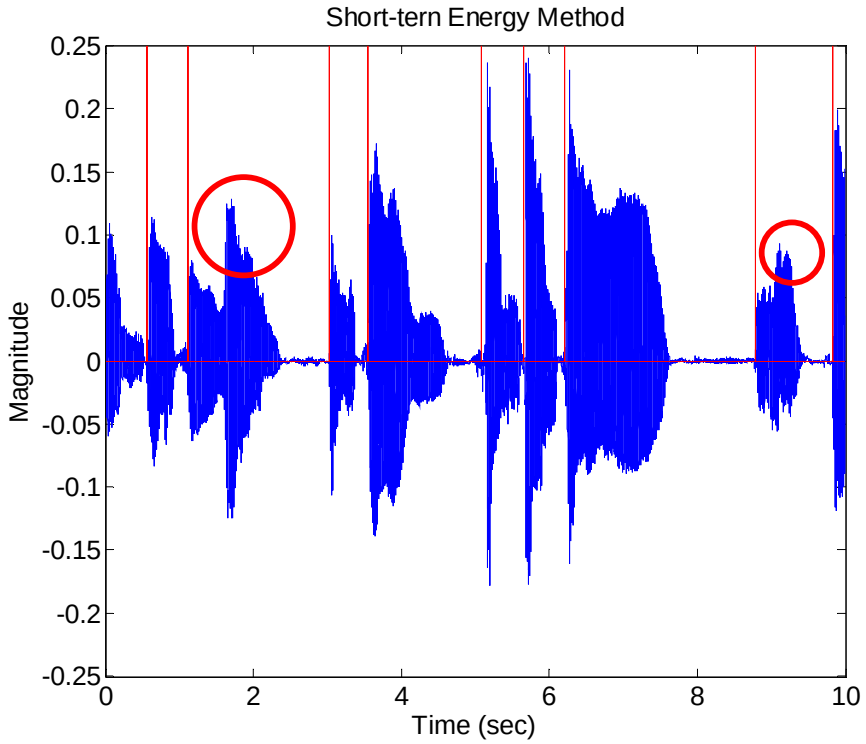


Fig. 7.2 Under-detected result in short-term energy method.

In HFC method, its computation time is more than previous two method mentioned above. It's due to this method calculate frequency spectrum for every frame whose size is much smaller than the original signal. That means this method evaluates FFT for each frame for lots of times. Besides, the HFC method detects location which may has high frequency content. It may detect the noise as a onset time either as shows Fig. 7.3.

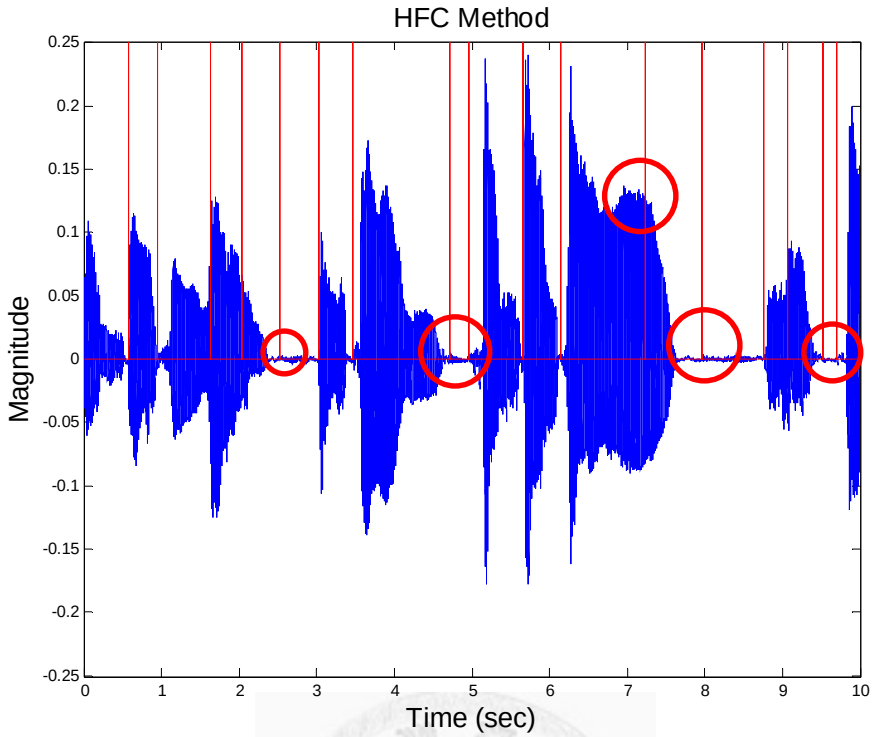


Fig. 7.3 Misjudge in HFC method

From Fig. 7.3, we can clearly point out the over-detected onsets and these detection are obviously not the real onsets. That is just because the signal has high frequency content at that moment.

Surf method calculates the slope by fitting every five points to a second-order polynomial. The detection result is shown in Fig. 7.4. As the similar problem to the first method in Fig. 7.1, this approach detects where may cause high slope as onsets. As we can see in Fig. 7.4, three more onsets are detected due to where cause high slope. Therefore, all methods mentioned above face the same problem: how to choose a threshold value to filter the real onset has become the most difficult issue.

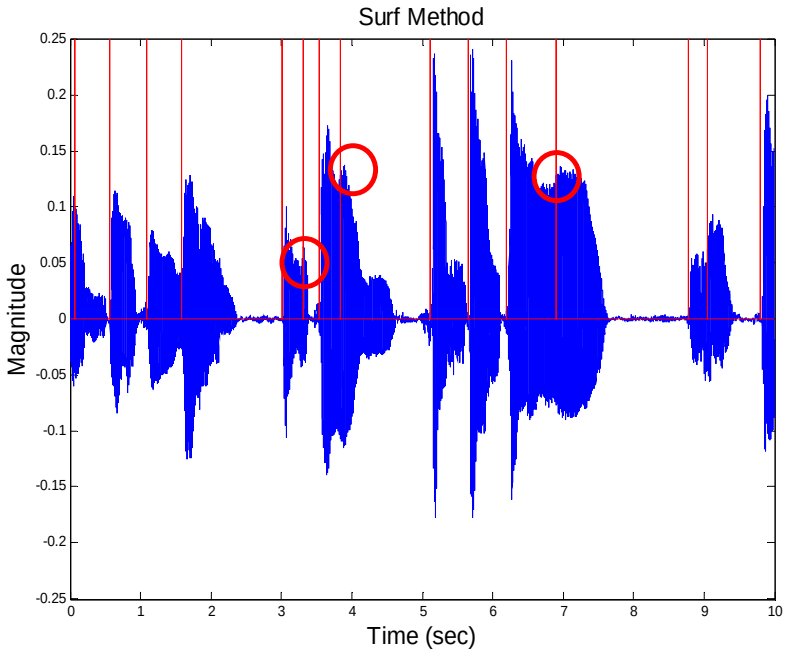


Fig. 7.4 Misjudge in Surf method

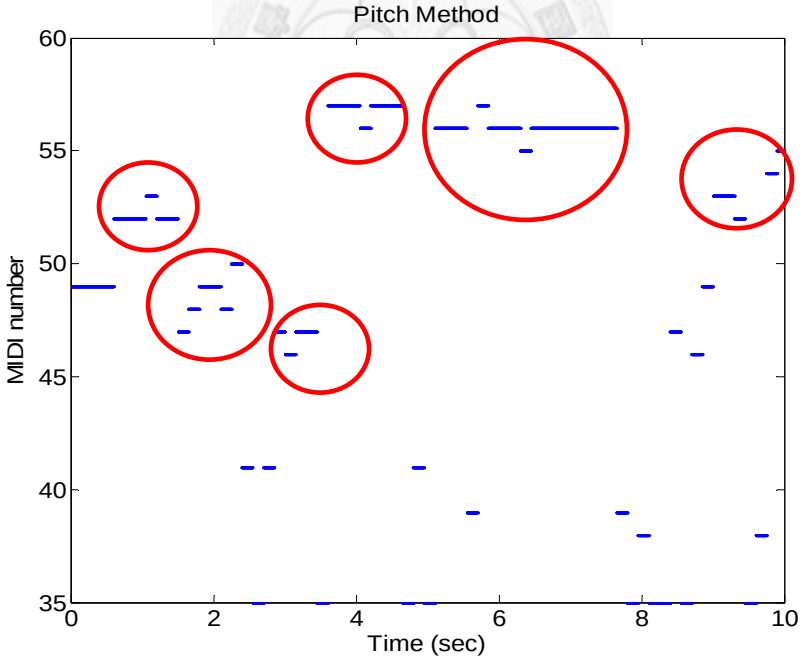


Fig. 7.5 Time-frequency distribution in pitch method

The pitch method is a kind of different than other methods due to it doesn't have to detect onset first. On the contrary, this method calculates the time-frequency distribution

of the querying signal. Then, using the time-frequency distribution result to reverse where are onsets. However, this approach does still not work. From Fig. 7.5, where are circled by red circles mean the same pitch by comparing with Fig. 3.17. How does this condition happen? It is because we use a minimal window size to calculate its frequency in each frame and then turn the frequency to pitch by equation (2.2). The difficult part is how to recover a complete pitch by assembling those pitch fragment which may be the same group in Fig. 7.5.

7.1 Proposed Onset Detection

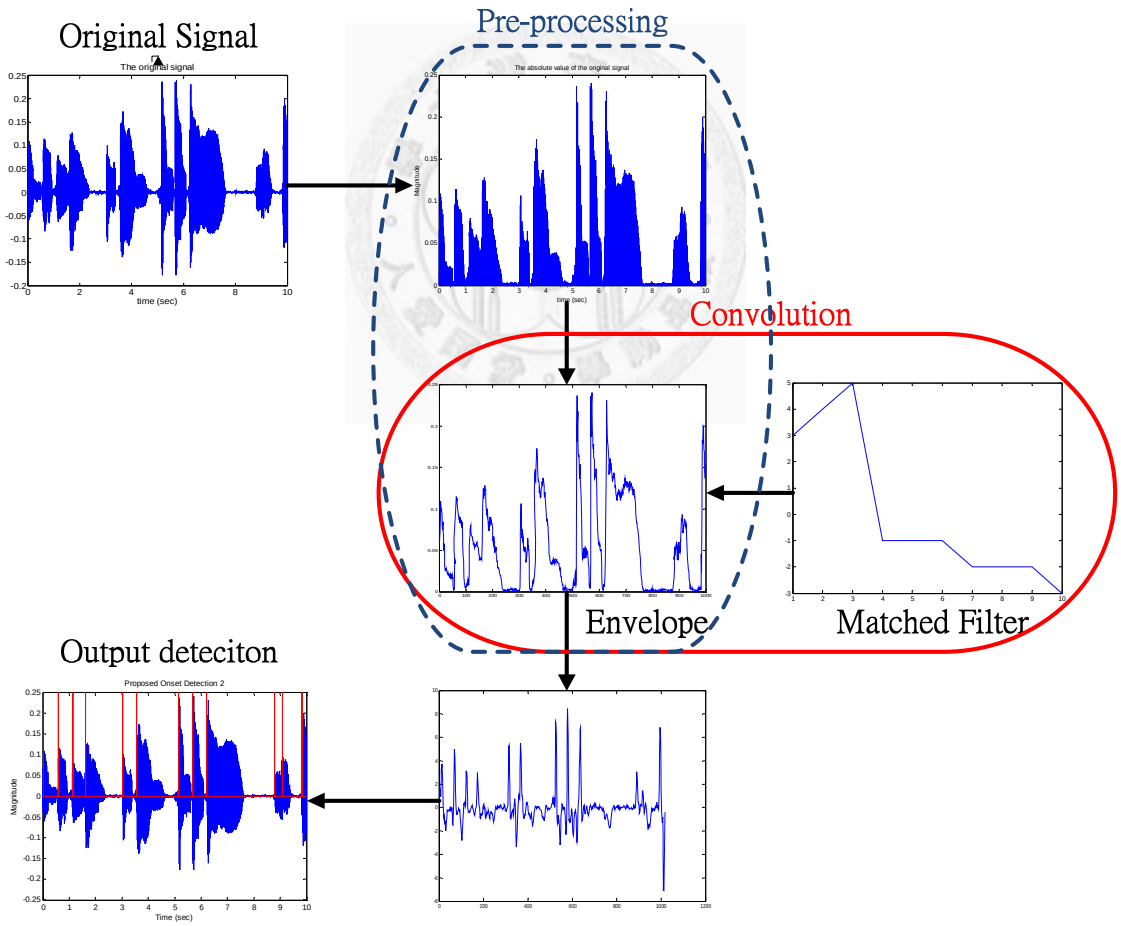


Fig. 7.6 The system block of our proposed onset detection.

In order to solve the onset detection problems mentioned above, we proposed a method which solves the problems of over-detection and under-detection. Following the step in Fig. 3.3, the system block is illustrated in Fig. 7.6. The input signal used here is the song "愛情釀的酒" as in Fig. 3.1.

First, the input signal passes a pre-process to extract its absolute value. Then, the absolute value of the signal is cut into a lot of slots. Each slot frame size is 20 ms and the sliding window slides 10 ms per motion. Sampling rates is 8000 Hz and the length of this signal is 10 seconds, so there are 240000 samples in this signals. Using frame size 20 ms, the envelope is shown in Fig. 7.7 and cut into 1000 parts.

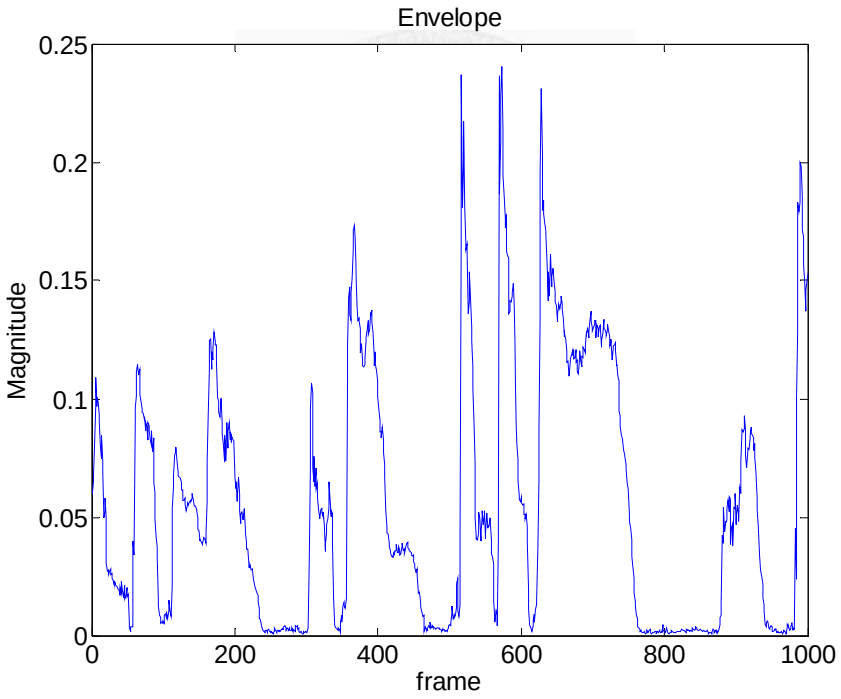


Fig. 7.7 Envelope of absolute value of input signal.

Assumed that the attacking signal is like that in Fig. 7.8, we can evaluate its matched filter as in Fig. 7.9. The signal in Fig. 7.8 is denoted as $s(t)$, then its matched filter, $h(t)$, can be evaluated as $s(T-t)$, where T denotes the length of $s(t)$.

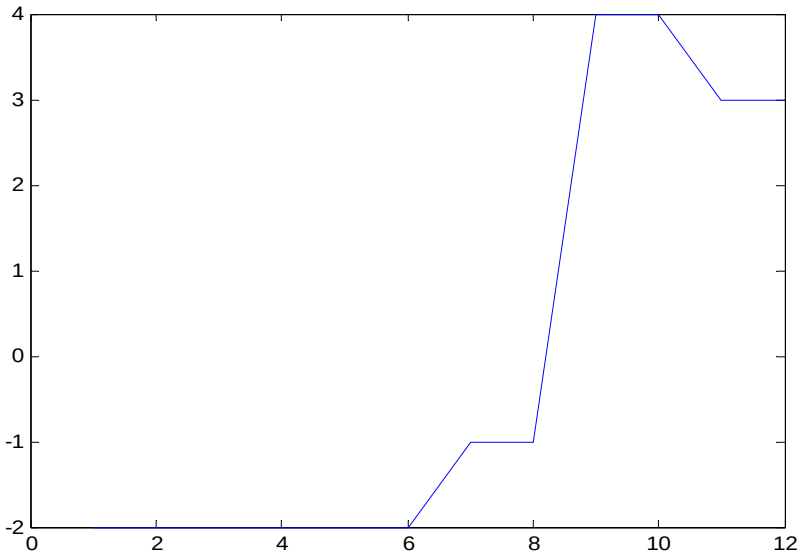


Fig. 7.8 Assumed attacking signal.

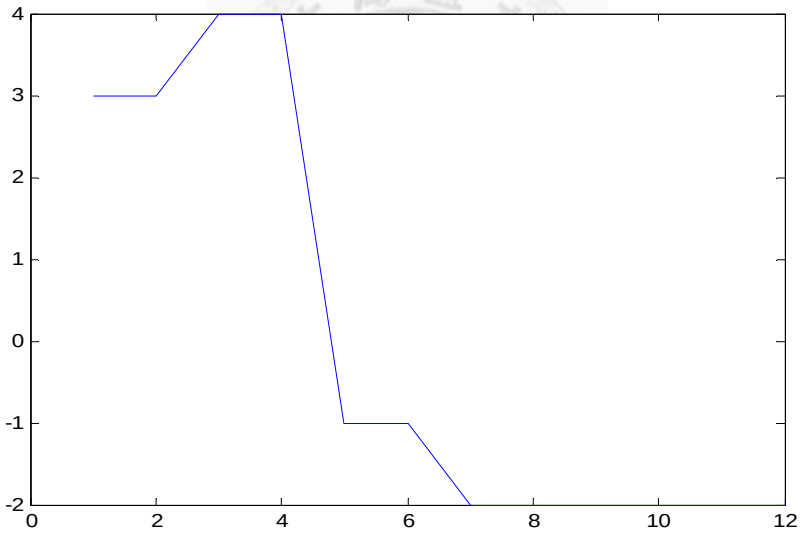


Fig. 7.9 The matched filter of signal in Fig. 7.8.

Then, we convolve the signal in Fig. 7.7 and Fig. 7.9. This step represents that the attacking will be enhanced after convolution with these two signals. Before doing the convolution, some post processes are taken as normalizing and adjusting the difference of magnitudes. The signal is normalized as (7.1).

$$\frac{x(t)}{0.2 \cdot mean(x(t))} \tag{7.1}$$

where $x(t)$ denotes the signal in Fig. 7.7 and $mean(x(t))$ represents the average of $x(t)$. The signal after normalizing is shown in Fig. 7.10.

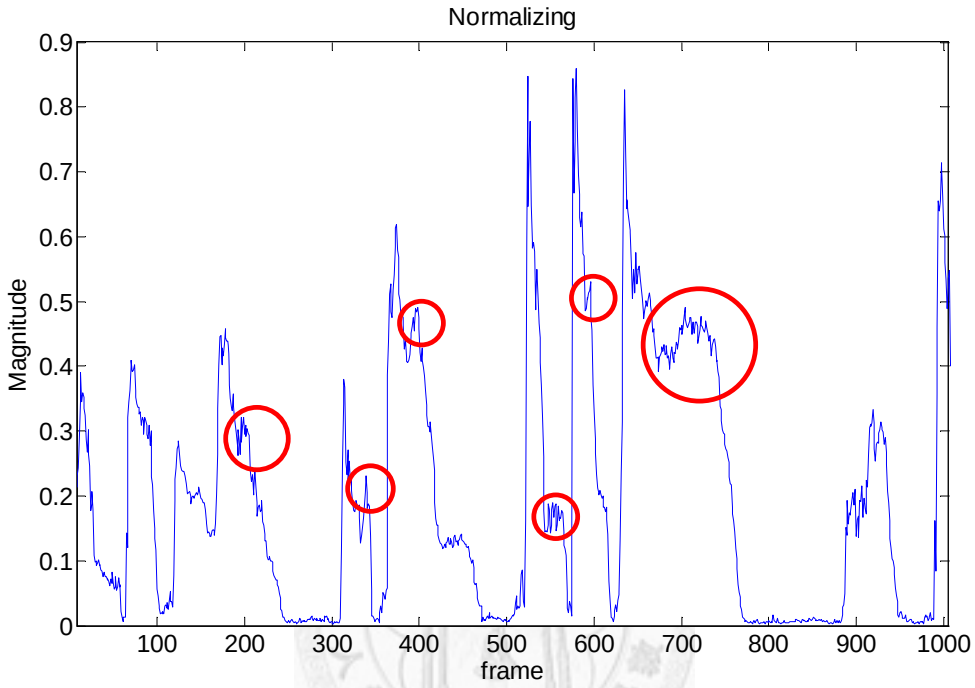


Fig. 7.10 The envelope signal after normalizing

From Fig. 7.10, we can observe that the maximum value of the signal is close to 1 unlike which in Fig. 7.7. However, some places are onset-like such as red circles in Fig. 7.10. That is just generated form an end of a singing sound and those places are not the real onsets. Therefore, the second step in post process is to compute the signal by getting its power of 0.6 and the result is shown in. From that, the magnitude difference of signal becomes smaller. The second peak in one note has less difference from the first peak in the same one note. This step is to prevent identifying two onsets in one note. We can see from the 700th frame between Fig. 7.10 and Fig. 7.11. The differences of the second peak are from about 0.7 to 0.5.

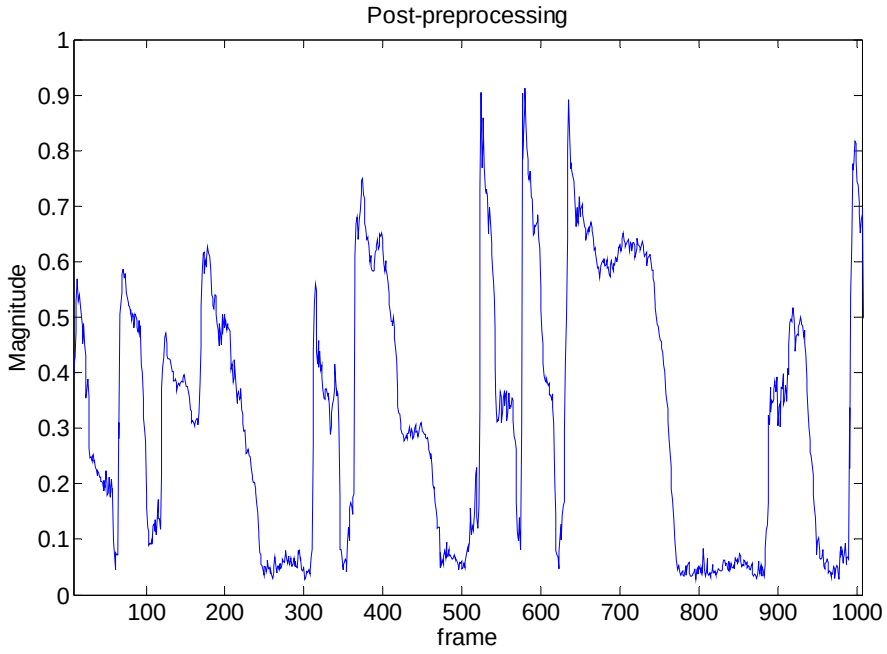


Fig. 7.11 The power of 0.6 after normalizing.

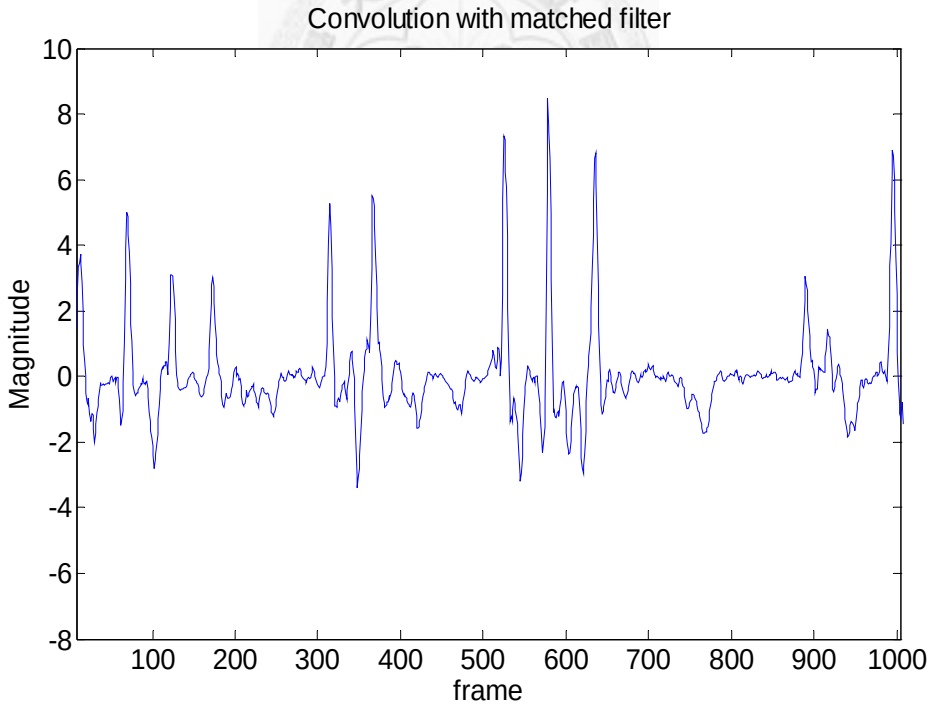


Fig. 7.12 After convolution with the matched filter.

Then, the signals in Fig. 7.9 and Fig. 7.11 are convolved and the result is shown in Fig. 7.12. After convolution, the first peak in one note is enhanced and the second peak

in one note is lowered. Then, use a threshold value as 1.2, and the onset detection is as in Fig. 7.13.

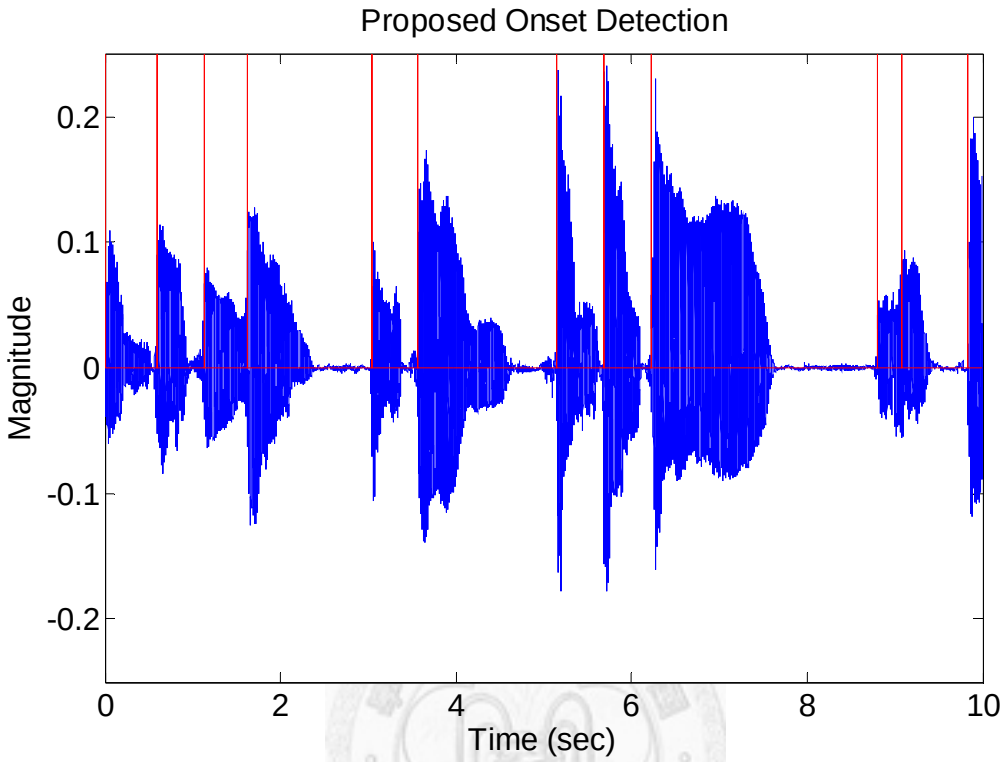


Fig. 7.13 the onset detection of proposed method.

As we know the numbers of notes are 12, the onset detection by using our proposed method is perfect. Our proposed method solves the problem of other onset detections. It solves the problem in HFC method which causes the silence part to high frequency content. It solves the problem in surf method which detects all possible high slope parts as onsets like the second rise in the end of a sound. It solves the problem in magnitudes of difference method which is over-detected and cost lots of computation time. It solves the problem in short-tern energy method which is always under-detected and highly depends on silence between two contiguous notes. To summarize, the threshold choice let other onset detection methods become unpractical. Our proposed method provides a robust method and let the problem of threshold minimizes.

7.2 Proposed Pitch Estimation

In Chapter 4, some pitch estimation methods are introduced like autocorrelation pitch tracking, modified Hibert-Huang transform, sub-harmonic summation and harmonics elimination method.

The autocorrelation pitch tracking is a simple concept for tracking pitch by computing the autocorrelation function of a sinusoid signal. Its advantage is to observe the fundamental period then reverse it to find the fundamental frequency. However, it is suitable for single-tone signal but not suit for multiple-tone signal. The reason is the autocorrelation function of a compound waveform can't indicate the real fundamental period of this signal as Fig. 7.14 shows.

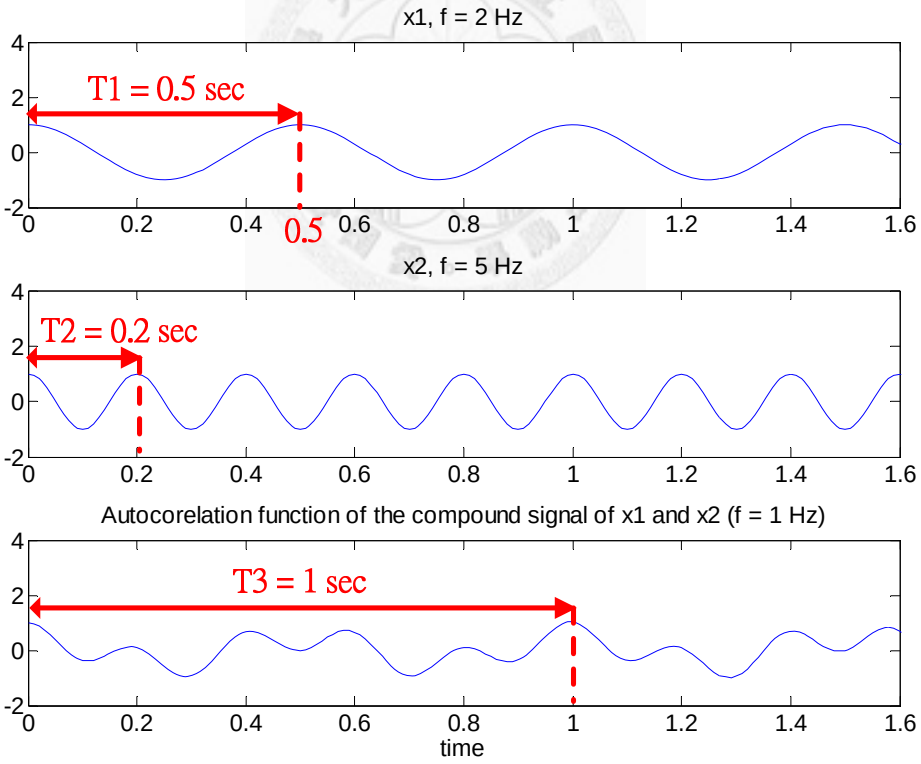


Fig. 7.14 The autocorrelation function of a compound signal composed by a 2-Hz and a 5-Hz signal.

As to other three pitch detection methods, it is usually used to detect the fundamental frequency of a compound signal. Due to humming signal produced by human is a single-tone signal. It means that whatever how fast does a person sing a piece of melody, it is also a single tone which may last a short time or long time.

Owing to above reasons, we use the FFT to extract the fundamental frequency for each detected note. Take the signal in Fig. 3.1 for example, we first do onset detection for this signal. The detection result is shown as bellow:

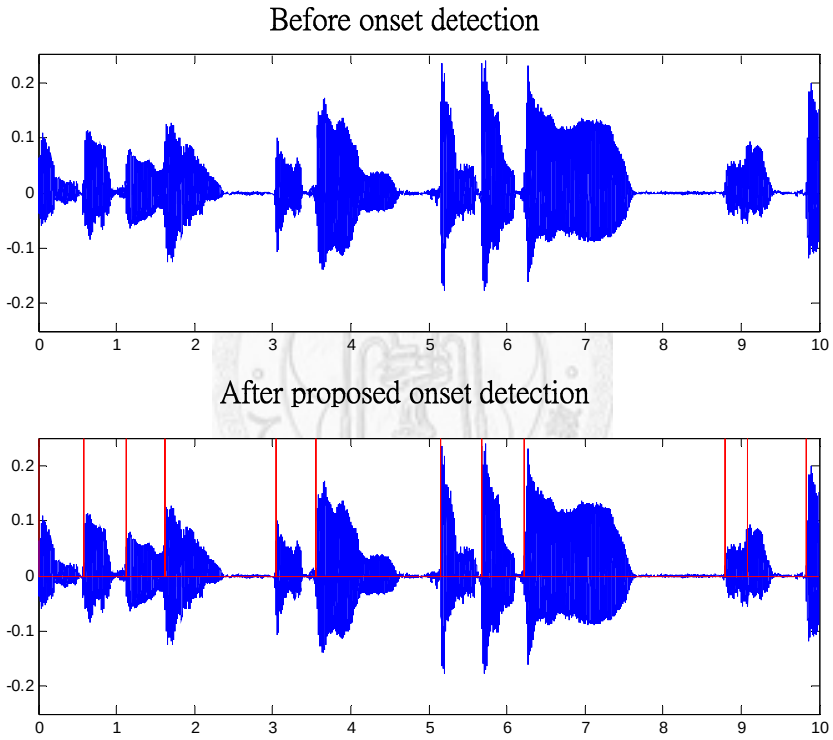


Fig. 7.15 After our proposed onset detection

After proposed onset detection, each note is computed for its fundamental frequency by using the FFT. However, the harmonics is still a problem. From Fig. 7.16, we choose frequency which has first peak in magnitude as the fundamental frequency. A simple illustration is shown in Fig. 7.16.

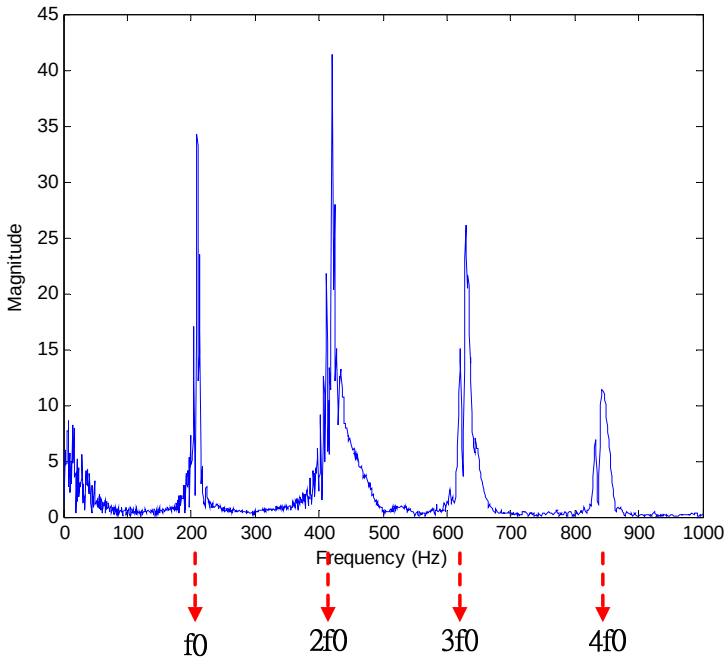


Fig. 7.16 The harmonics of a sound produced by a male.

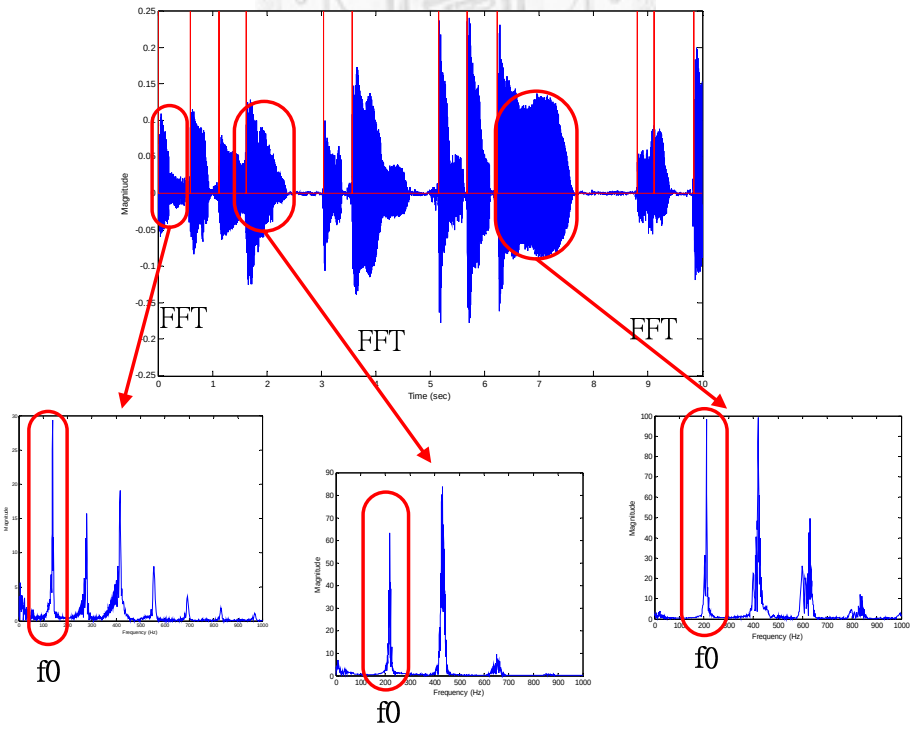


Fig. 7.17 The process of extracting the fundamental frequency.

Our method is to find the top 3 peak in the frequency domain of the signal and

choose frequency which has the first peak as the fundamental frequency. Following the rule above, the fundamental frequency in Fig. 7.16 is about 210 Hz. Observing the figure, the harmonics are about the multiples of the fundamental frequency: 2f0, 3f0 and 4f0.

7.3 Proposed Melody Matching

The two main melody matching methods are introduced in Chapter 5. One is dynamic programming and the other method is hidden Markove model. We proposed a method based on dynamic programming method to match melody. Here, we use pitch interval to be as melody information. The drawbacks of dynamic programming is it cost too much computation time and the disadvantage of HMM is its probability model is very difficult to model due to the property of music characteristic. Therefore, we need a fast algorithm due to QBH requires real time online.

7.3.1 Modified Dynamic Programming

We proposed a DP-like algorithm to match melody information. Unlike DP algorithm, we don't have to construct a alignment score matrix by using (5.1) but we have to construct a pitch interval matrix as shows. Assumed the pitch interval of target is {-4 2 2 -4 2 2 3 0 2 -9 -3} and the pitch interval of query is {-4 3 1 -3 1 3 2 0 2 -9 -2}. The pitch interval matrix is constructed by:

$$PitchInterval(i, j) = |Target(j) - Query(i)| \quad (7.2)$$

where the matrix is shown in Table 7.1, 0< j < length(target) and 0< i < length(query).

In Table 7.1, the first row is the pitch interval of target and the first column is the pitch interval of query. The second row in the table is the first row in the matrix and the

second column in the table is the first column in the matrix. The basic element 1 in this matrix represents one semitone pitch interval. In our algorithm, we start from cell (1,1) in the matrix to match the melody sequence.

Table 7.1 the pitch interval matrix of target and query.

Q \ T	-4	2	2	-4	2	2	3	0	2	-9
-4	0	6	6	0	6	6	7	4	6	5
3	7	1	1	7	1	1	0	3	1	12
1	5	1	1	5	1	1	2	1	1	10
-3	1	5	5	1	5	5	6	3	5	6
1	5	1	1	5	1	1	2	1	1	10
3	7	1	1	7	1	1	0	3	1	12
2	6	0	0	6	0	0	1	2	0	11
0	4	2	2	4	2	2	3	0	2	9
2	6	0	0	6	0	0	1	2	0	11
-9	5	11	11	5	11	11	12	9	11	0

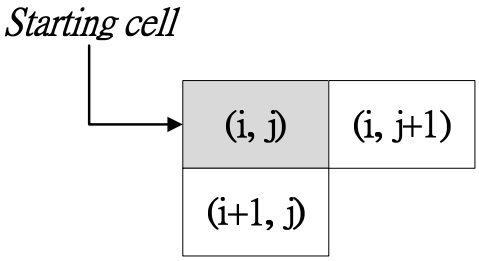


Fig. 7.18 The kernel of comparing the max value.

At the first step is to compare the size of values in a kernel as shown in Fig. 7.18. If the starting cell is at location (i, j), then the value in (i, j), (i, j+1) and (i+1, j) will be compared. Position whose value is the largest is chosen to be the score cell.

Before we continue estimating the routes, we can find that there are several yellow

parts in Table 7.1. Those are values which exceed 6, and the values in the matrix mean that the difference of semitone. As we can know, the difference of semitone of a fundamental frequency and its second harmonic is 12. However, we may make wrong estimation in pitch tracking. That is, we may judge a harmonic as its fundamental frequency. Therefore, we can not let the difference of semitone between a fundamental frequency and its harmonics that so much. We adjust this scheme by the following equation in (7.3).

$$Function: \begin{cases} a = \mathrm{mod}(A(i,j)+5,12) \\ b = (A(i,j)+5-a)/12 \\ A(i,j) = abs(a-5)+b \end{cases} \tag{7.3}$$

Then, the matrix is become as bellow:

Table 7.2 The alignment matrix after function in (7.3).

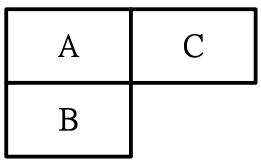
Q \ T	-4	2	2	-4	2	2	3	0	2	-9
-4	0	6	6	0	6	6	2	4	6	5
3	1	1	1	1	1	1	0	3	1	1
1	5	1	1	5	1	1	2	1	1	3
-3	1	5	5	1	5	5	6	3	5	6
1	5	1	1	5	1	1	2	1	1	3
3	1	1	1	1	1	1	0	3	1	1
2	6	0	0	6	0	0	1	2	0	2
0	4	2	2	4	2	2	3	0	2	9
2	6	0	0	6	0	0	1	2	0	2
-9	5	2	2	5	2	2	1	4	2	0

When a score cell is chosen, the next route will point to its diagonal cell. That is if

a cell (k, h) is chosen as the score cell, then the next starting cell is in the position (k+1, h+1) and the cells in next kernel are (k, h), (k, h+1) and (k+1, h). The routing is shown in Table 7.3. At the beginning, the starting cell is (1,1), then the comparing kernel has three cells: cell(1,1), cell(1,2) and cell(2,1). Position which has smallest value is chosen as the score cell. In this case, the score cell is chosen as cell(1,1) and the value in the cell is 0. If the values of three cells in one kernel are all the same, then we give the highest priority to cell A, middle priority to cell B and lowest priority to cell C in Fig. 7.19

Table 7.3 The routing of the proposed melody matching.

Q \ T	-4	2	2	-4	2	2	3	0	2	-9
-4	0	6	6	0	6	6	2	4	6	5
3	1	1	1	1	1	1	0	3	1	1
1	5	1	1	5	1	1	2	1	1	3
-3	1	5	5	1	5	5	6	3	5	6
1	5	1	1	5	1	1	2	1	1	3
3	1	1	1	1	1	1	0	3	1	1
2	6	0	0	6	0	0	1	2	0	2
0	4	2	2	4	2	2	3	0	2	9
2	6	0	0	6	0	0	1	2	0	2
-9	5	2	2	5	2	2	1	4	2	0



Priority: A > B > C

Fig. 7.19 The priority of cell A, B and C.

After routing by our proposed matching method, the difference of these two matching sequences is {0 1 1 1 1 0 0 0 0}. From Table 7.3, we can see there are yellow parts and orange parts. Yellow parts mean the kernel in every steps and the orange parts mean the candidates of every yellow kernel. Notice that, if two contiguous orange parts are not on the same diagonal direction, it means that a skip happens. If the candidate is not at its starting cell (i,j) but at the cell (i+1,j), then the difference value should be the sum of cell (i,j) + cell (i+1,j). If the candidate is not at its starting cell (i,j) but at the cell (i ,j+1), then the difference value should be the sum of cell (i,j) + cell (i ,j+1). Hence, the difference of these two matching sequences becomes {0 1 1 1 1 1+0 3+0 0 0} and it is {0 1 1 1 1 1 3 0 0}.

In Table 7.3, those which have orange parts are the possible values in the comparing kernels and values which are deep blue represent the minimal differences. The values denoted with deep blue color mean that their positions are corresponding to the target (j) and query (i). Take Table 7.3 for example, the first deep blue value is 0 and it represents target (1), -4, is matching to query (1), -4, and the difference is 0. The next step deep blue value is 1 and it represents target (2), 2, is matching to query (2), 3, and the difference is 1. By observing the positions which has deep blue values, the alignment can be written as Fig. 7.20 shows:

Target : -4 2 2 -4 2 - 2 3 0 2 -9
Query : -4 3 1 -3 1 3 2 - 0 2 -9

Fig. 7.20 Alignment of target and query.

Denote the symbol "–" as a skip. If a minimal value is not in the cell (i,j) as in Fig. 7.18, a skip will happen. We should give a skip a penalty. The scoring method in our

approach is by using exponential function. We obtain the negative values of the stored values in each kernel. If the value is 0, then we obtain its negative value is -0 and exp (-0) is equal to 1 which is the maxima score in our scoring method. If the value is 2, then we obtain its negative value is -2 and exp (-2) is equal to 0.1353. Therefore, the sum of the score of the alignment in Fig. 7.20 is sum of 4*exp(-0) and 4*exp(-1) and is equal to 5.4715.

Why to use the exponential function to calculate the matching score in our proposed method? As we know, queries are sung by people and they are always not exactly correct as the same as the recording targets in mp3. Hence, few errors like slightly off-key, missing few sounds or singing a little bit more sounds have to be tolerated in our system. Our method also provides a fast searching scheme. Regularly, the length of target is always much longer than the length of query. Therefore, query has to compare every segment in target form the first element to the end by repeating the step in Table 7.3. However, in our assumption, we only compare the elements whose difference below one in targets with the first element of query. Take the case in Table 7.1 for example, the pitch interval of target is {-4, 2, 2, -4, 2, 2, 3, 0, 2, -9} and the pitch interval of query is {-4, 3, 1, -3, 1, 3, 2, 0, 2, -9}. We only need to repeat the step in Table 7.3 from target(1), -4 and target(4), -4. By this fast scheme, the matching system reduces lots of computation time.

Owing to any high-similar melody piece of other targets, we also proposed a consecutive weighting method. The melody distance is {0 1 1 1 1 0 0 0 0} in Fig. 7.20. In our method, melody distance which is under 2 is all classified to “0”, and between 2 and 20 will be given a penalty. The calculation of the scoring scheme is shown as (7.4).

$$MelodyDis\tan ce(k) = 0, \qquad MelodyDis\tan ce(k) < 2 \tag{7.4}$$

where k is from 1 to the length of melody distance. The algorithm of melody matching

score is computed in (7.5).

$$
\begin{aligned}
&Score \quad = zeros(1, length(MelodyDis\tan ce)) \\
&Penalty = zeros(1, length(MelodyDis\tan ce)) \\
&\\
&for\ k = 1 : length(MelodyDis\tan ce) \\
&\quad Score(k) = Score(k) + e^{MelodyDis\tan ce(k)} \\
&\quad if \quad 2 <= MelodyDis\tan ce(k) <= 20 \\
&\qquad penalty(k) = penalty(k) + MelodyDis\tan ce(k)^{1.5} \\
&\quad end \\
&end \\
&\\
&Score = sum(Score) - sum(Penalty)
\end{aligned}
\tag{7.5}
$$

Our penalty scheme is extract melody distance which is between 2 and 20 and obtain its power of 1.5. That means the distance in this range have a great error, so we have give it a big penalty.

7.3.2 Modified Hidden Markov Model

We have introduced hidden Markov model (HMM) in section 5.2, and we will modified HMM more suitable for music property in QBH system in order to solve the lower hit rates of the HMM method in melody matching.

The HMM method in section 5.2 is to replace the observation where has zero probability with a minimal probability, P_m (e.g. 0.01), but the performance seems bad. Therefore, we proposed a modified HMM to improve the performance by remodeling the probability model. We consider that original pitch may become its higher pitch (e.g. more than 1 or 2 semitones) or its lower pitch (e.g. less than 1 or 2 semitones). It means that the estimated pitches may not the same as the actual ones. Hence, the model can't be established by only observing the pitch interval of targets. Here, given a target as: 2 3 2 -1 2 0 -1 3 1 2 1, the model can be built as in Fig. 7.21. Then, Fig. 7.22 shows the

modeling process of HMM and the modeling process of modified HMM.

O2 \ O1	-1	0	1	2	3
-1	0	1	0	1/3	0
0	0	0	0	0	0
1	0	0	0	1/3	1/2
2	1/2	0	1	0	1/2
3	1/2	0	0	1/3	0

Fig. 7.21 The transition probability model from target

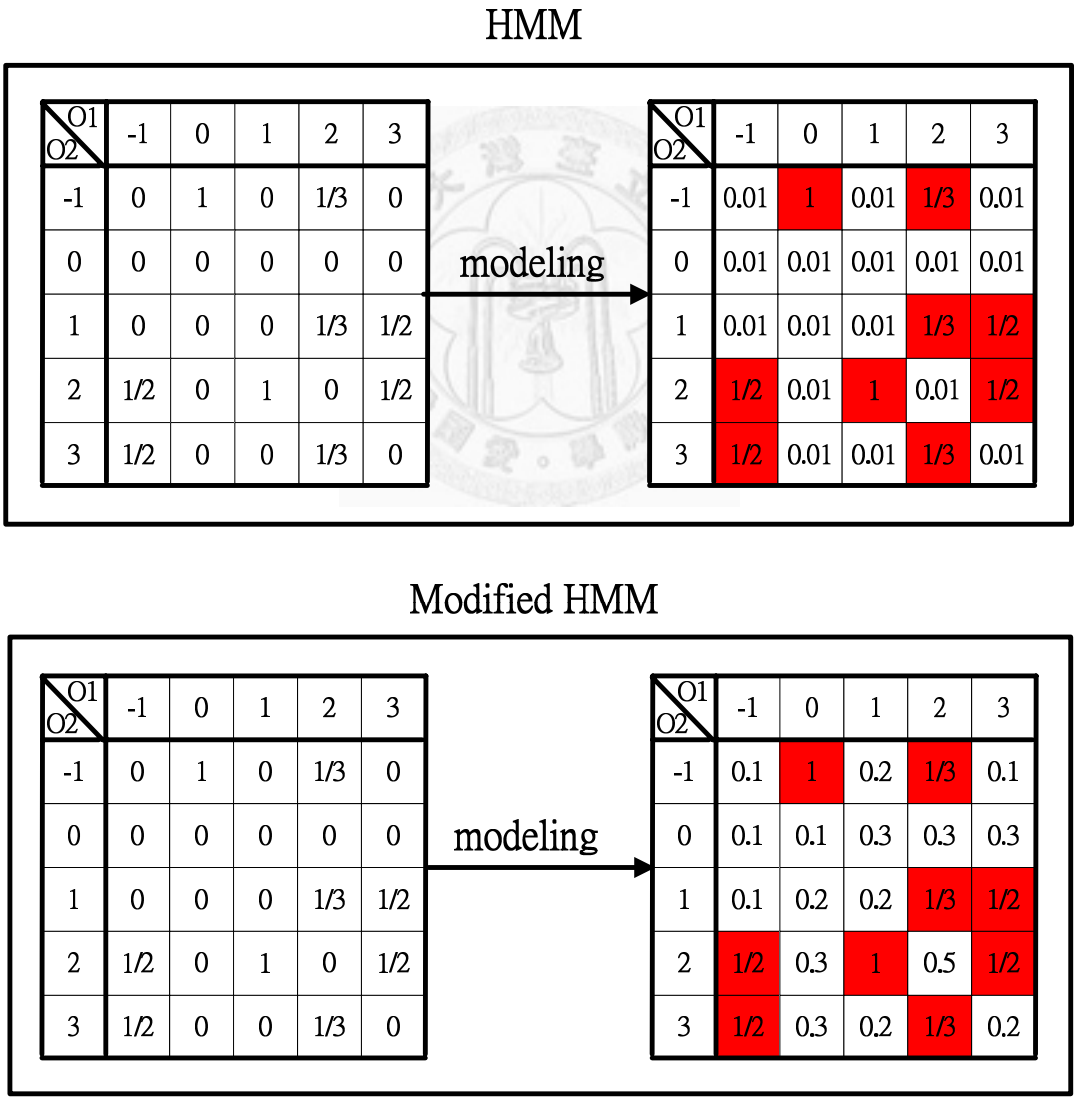


Fig. 7.22 The modeling process of HMM and MHMM

In above figures, O1 denotes the current state and O2 denotes O1's the next state. If an observation really exists, then its probability will be filled in its corresponding position in the probability table with the red background. In HMM, the zero probabilities are all replaced by the minimal probability, P_m, but in Fig. 7.22, we replace it by non-fixed value in MHMM. In our assumption, we assume one pitch may become its adjacent pitches. For example, if a pitch interval is 3, we assume it may be estimated as 2 or 4. People always sing the higher or lower pitch than the original one by one semitone. Therefore, the modified HMM assign higher probability to those observations around the real existing observation in targets.

Chapter 8 Experimental Result

In this Chapter, we improve onset detection for music signals by dividing the time slots more accurately. Several onset detection methods are compared by using computation time and recognition rate and melody matching methods are compared by using computation time, hit rates and mean reciprocal right rank (**MRRR**).

First, we will introduce some performance standards in QBH system like hit rate and MRRR. Later, we will compare the performance for different onset detection methods. Then, we compare the performance for different melody matching methods. Finally, the whole system, QBH, is compared by different combination of onset detection and melody matching methods, and we will prove our proposed method promote the QBH system performance best.

8.1 Performance Standard

In this section, we will introduce recognition rate, hit rate and mean reciprocal right rank (MRRR). Recognition rate is used to judge a onset detection good or not good. Hit rate and MRRR are used to measure the performance of a QBH system.

As mentioned in section 3.6, recognition rate is defined in (3.16). Hit rate means the numbers of positive hitting over numbers of queries and is defined in (8.1). Mean reciprocal right rank (**MRRR**) is defined in (8.2). The main idea of QBH system is to help people search what songs they want by humming a passage of a melodic piece. Therefore, the searching result shown on web is just to list some possible songs for people. The desired song is not necessarily on the first rank of searching result. It can be shown on the second rank, third rank or more. When people see the song list on web,

they can find their desired target by their impressions. We can set number of rank as 3, 5, 10 or 20. In (8.2), the method of MRRR calculation is a kind of different from (8.1). The term, $rightrank_n$, means the rank of the query shown on web list.

$$hit\ rate = \frac{numbers\ of\ \ positive\ hittings}{numbers\ of\ \ Queries} \times 100\% \tag{8.1}$$

$$MRRR = \frac{\sum_{n=1}^{numberofQueries} \frac{1}{rightrank_n}}{numberofQueries} \times 100\% \tag{8.2}$$

8.2 Comparison of Onset Detection Performance

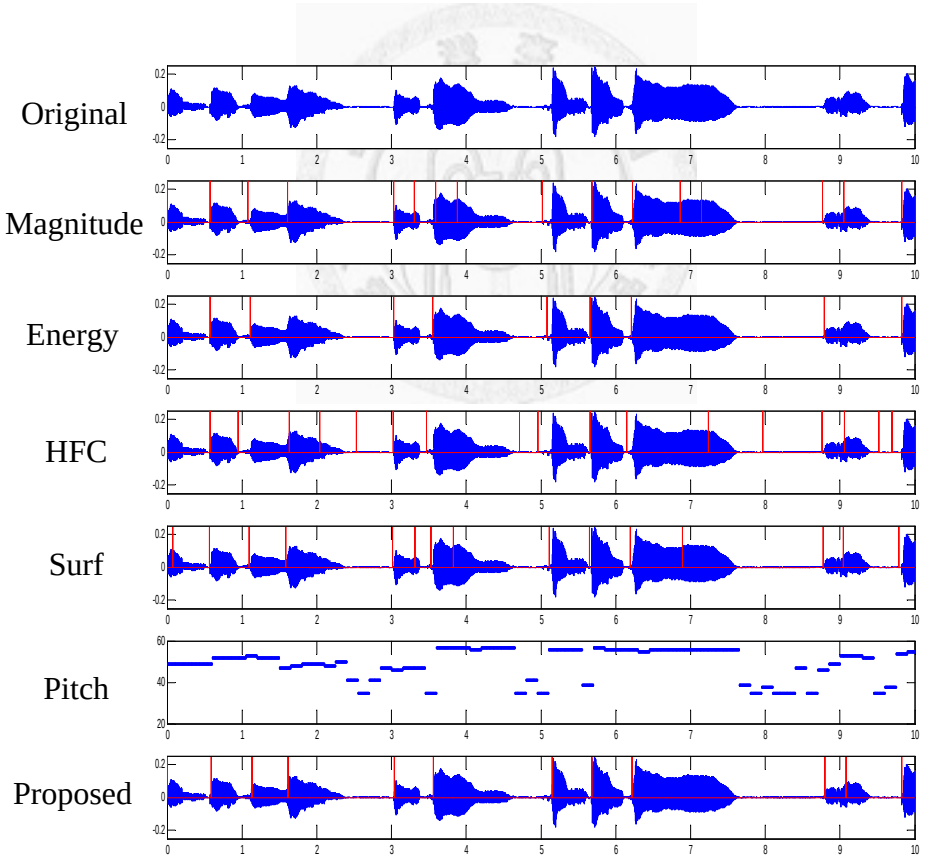


Fig. 8.1 The performance of all onset detection methods for query “愛情釀的酒” which has 12 actual notes.

Fig. 8.1 use the same query in Chapter 3, the input is singing by a male and the song is “愛情釀的酒” which has 12 actual notes. The first bin is the original signal, the second bin is the result of magnitude method, the second bin is the result of short-tern energy method, the third bin is the result of HFC method, the fourth bin is the result of surf method and the last bin is the result of proposed method.

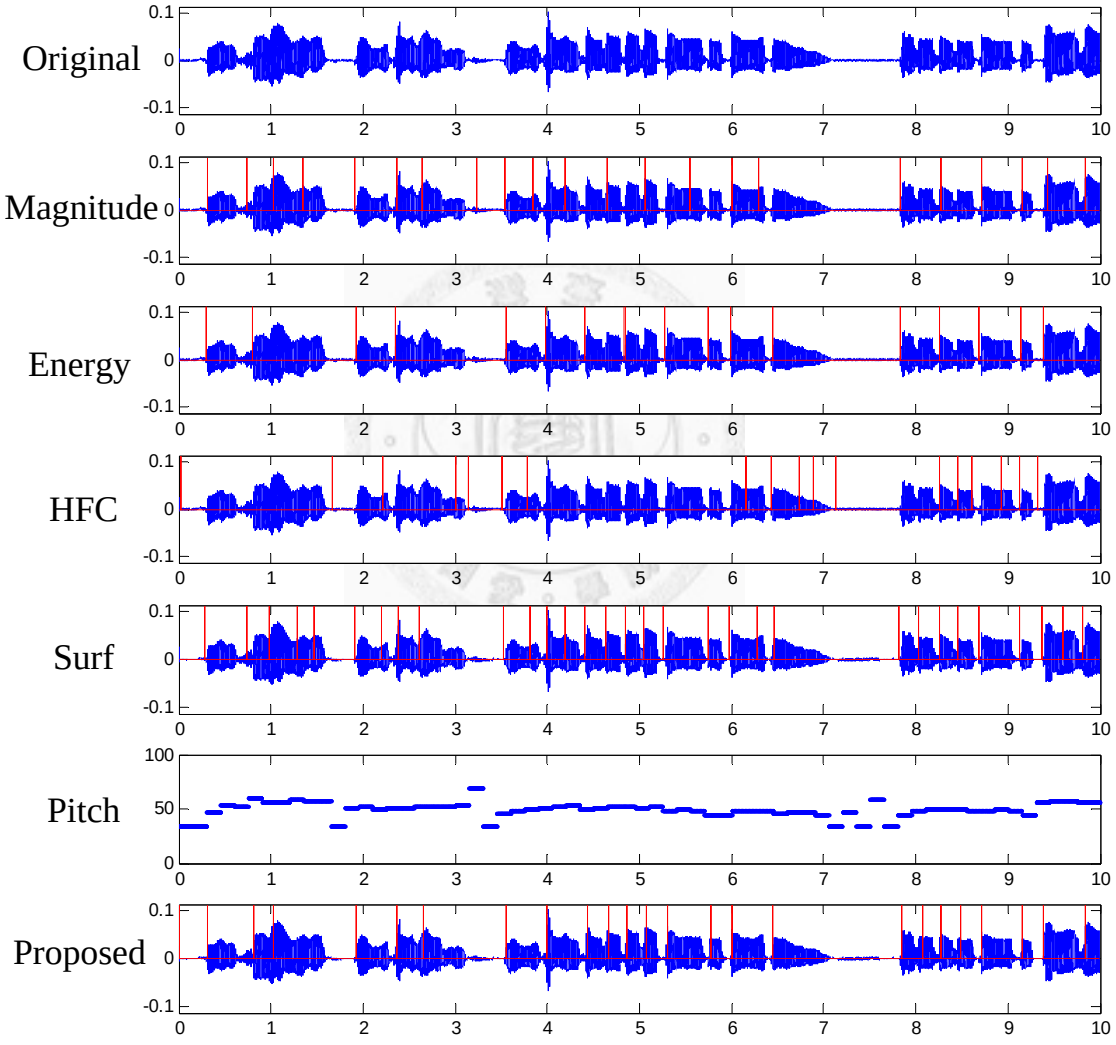


Fig. 8.2 The performance of all onset detection methods for query “挪威的森林” which has 24 actual notes.

Table 8.1 shows the TP rate and FP rate of onset detection for the known onset detection introduced in Chapter 3 and our proposed onset detection in Chapter 7. The experiments use three different lengths of queries to test: 10-second, 15-second, 20-second of length. Each version has 26 queries and the sampling rate is 8000 Hz. It is obviously that our proposed onset detection has the best TP rate and the lowest FP rate for all three cases.

Table 8.1 Performance of all kinds of onset detection methods for 3 cases and each of them has 26 queries.

Query / Onset Method	10-second version	15-second version	20-second version
Magnitude Method	TP = 75.49% FP = 27.30%	TP = 87.03% FP = 16.04%	TP = 89.49% FP = 15.82%
Short-Term Energy Method	TP = 94.20% FP = 9.15%	TP = 78.63% FP = 23.56%	TP = 74.30% FP = 27.16%
HFC Method	TP = 77.32% FP = 26.00%	TP = 86.58% FP = 15.39%	TP = 87.59% FP = 14.29%
Surf Method	TP = 86.22% FP = 18.16%	TP = 94.70% FP = 10.76%	TP = 97.05% FP = 6.14%
Proposed Onset Detection	TP = 98.72% FP = 7.23%	TP = 98.09% FP = 6.31%	TP = 98.16% FP = 4.54%

8.3 Comparison of Melody Matching Performance

In this section, we fix the onset detection by using our proposed onset detection method. As section 8.2 proved, our onset detection performs best in all mentioned onset detections. Therefore, we use 4 melody matching methods to compare which has the best performance for 3 versions of recordings (10-second, 15-second and 20-second of length). MRRR whose rank size is 5 and hit rate are discussed in following tables.

Our experimental settings use 26 queries in 50 targets of our music database. We discuss computation time for searching each query, hit rate and MRRR for each query version as follows:

Table 8.2 MRRR and HR of 26 queries for 3 versions of length from 50 targets by using DP algorithm.

Query / Method	10-second		15-second		20-second	
	HR	MRRR	HR	MRRR	HR	MRRR
DP	28%	31.33%	40%	44%	64%	68%
MDP	64%	66%	96%	96%	100%	100%
HMM	32%	40.67%	28%	39.33%	44%	53.33%
MHMM	40%	51%	68%	74.33%	84%	87%

In Table 8.2, it is obvious that MDP improve the performance of DP and MHMM improve the performance of HMM. Observing each of the melody matching method, we can find that its hit rate (HR) and MRRR (in top 5) is rising due to the length of query. It represents the query is recorded for a longer time can rise the correct rate. If we fix the

query type, the modified melody matching method performs best of each query type.

From Table 8.2 the conclusion is our modified melody matching melody based on dynamic programming algorithm has the best performance. Modified HMM method improves HMM's performance but still not better than modified melody matching method. In next section, we will compare the performance with all combination of onset detections and melody matching methods we have mentioned before.

Table 8.3 The computation time of these four melody matching methods by using our proposed onset detection.

Query / Method	10-second	15-second	20-second
	Running time / Query	Running time / Query	Running time / Query
DP	14.54 sec	25.17 sec	35.07 sec
MDP	2.1 sec	3.09 sec	3.8 sec
HMM	1.06 sec	1.1 sec	1.14 sec
MHMM	1.42 sec	1.47 sec	1.52 sec

8.4 Comparison of Whole QBH System Performance

Table 8.4 indicates all combination of each onset detection and melody matching method for 10-second querying version. Denote that **MHMM** is modified HMM and **MDP** is modified melody matching method. MRRR used here is to calculate the right answer in top 3. The combination of proposed onset detection and modified melody matching method has the best performance for 10-second queries in Table 8.4. The combination of proposed onset detection and modified melody matching method also

has the best performance for 10-second queries in Table 8.5.

Table 8.4 Performance of 26 queries which are 10-second long from 50 targets.

Melody Matching / Onset Detection		HMM	MHMM	MDP
Magnitude Method	Time	3.04 sec	3.5 sec	4.6 sec
	Hit Rate	16 %	28 %	48 %
	MRRR	22 %	34 %	57.47 %
Short-term Energy Method	Time	1.08 sec	1.43 sec	2.05 sec
	Hit Rate	16 %	24 %	60 %
	MRRR	24.33 %	39.47 %	63.33 %
HFC Method	Time	2.13 sec	2.55 sec	2.7 sec
	Hit Rate	24 %	28 %	44 %
	MRRR	31.8 %	39.93 %	46 %
Surf Method	Time	1.12 sec	1.51 sec	2.75 sec
	Hit Rate	24 %	32 %	60 %
	MRRR	29.33 %	41 %	63.33 %
Proposed Onset Detection	Time	1.06 sec	1.42 sec	2.1 sec
	Hit Rate	28 %	40 %	64 %
	MRRR	35 %	51 %	66 %

Table 8.5 Performance of 26 queries which are 15-second long from 50 targets.

Melody Matching / Onset Detection		HMM	MHMM	MDP
Magnitude Method	Time	4.08 sec	4.4 sec	6.01 sec
	Hit Rate	24 %	28 %	88 %
	MRRR	30.47 %	36.93 %	91 %
Short-term Energy Method	Time	1.1 sec	1.51 sec	2.31 sec
	Hit Rate	20%	24 %	56 %
	MRRR	29.8 %	39.33 %	60 %
HFC Method	Time	2.76 sec	3.13 sec	4.53 sec
	Hit Rate	20 %	52 %	92 %
	MRRR	27.33 %	59 %	93.33 %
Surf Method	Time	1.19 sec	1.58 sec	3.26 sec
	Hit Rate	28%	48 %	68 %
	MRRR	35.6 %	53.33 %	68 %
Proposed Onset Detection	Time	1.1 sec	1.47 sec	3.09 sec
	Hit Rate	28 %	68 %	96 %
	MRRR	33.33 %	74.33 %	96 %

Although the best performance is not the combination of proposed onset detection and modified melody matching method in Table 8.6 but the combination of surf method and modified melody matching method. However, the performance of combination of

surf method and modified melody matching method is not stable.

Table 8.6 Performance of 26 queries which are 20-second long from 50 targets.

Melody Matching / Onset Detection		HMM	MHMM	MDP
Magnitude Method	Time	4.96 sec	5.31 sec	7.36 sec
	Hit Rate	28 %	52 %	80 %
	MRRR	39.73 %	61.93 %	83.33 %
Short-term Energy Method	Time	1.15 sec	1.56 sec	3.17 sec
	Hit Rate	32 %	40 %	52 %
	MRRR	40.73 %	51.47 %	54.13 %
HFC Method	Time	3.3 sec	3.75 sec	6 sec
	Hit Rate	36 %	60 %	72 %
	MRRR	48.67 %	72.67 %	74 %
Surf Method	Time	1.29 sec	1.68 sec	4 sec
	Hit Rate	44 %	68 %	84 %
	MRRR	50.67 %	75.33 %	87 %
Proposed Onset Detection	Time	1.14 sec	1.52 sec	3.8 sec
	Hit Rate	44 %	84 %	100 %
	MRRR	54.27 %	87 %	100 %

Here, we don’t discuss dynamic programming algorithm because it had already

been proved to take the longest computation time and performs bad in section 8.3. On the whole, the combination of the proposed onset detection and modified melody matching method is still the best for QBH system due to its stability. The trend is that long-length query has higher hit rate than short-length query due to it provides more information for melody matching method to compare the similarity and calculate the best similar score.

In aspect of onset detection, the magnitude method and HEC method take more than 1 second even 2 seconds while other three onset detection methods take less than 1 second. Considering the computation time, short-term energy method, surf method and our proposed method reach the basic requirement for searching time online for each query.

Chapter 9 Conclusion and Future Work

9.1 Conclusions

Querying-By-Humming system makes people search their desired songs by using content-based method instead of using traditional text-based method in music information retrieval (MIR). Before designing a QBH system, we have to understand some music property and characteristics for us to more easily implement it.

We try to improve the onset detection performance of some known onset detection methods mentioned in Chapter 3. The proposed onset detection solves the problems in these known onset detections, it provides higher hit rates, MRRR and is more robust to any type of singing waveform from free singing styles depending on people. Our proposed onset detection reaches 98% TP rate and beats all known onset detection introduced in Chapter 3.

On pitch estimation, we don't use the complex methods mentioned in Chapter 4. Instead, we hold the tip of the relationship between the fundamental frequency and its harmonics. We extract the fundamental frequency due to its harmonics are the multiple of itself and we find the first peak shown in its frequency spectrum. In the process of extracting the fundamental frequency, some note segmentation may have large low frequency content, this problem are also solved for our pitch estimation method.

The last significant step is melody matching for similarity comparison. We proposed two methods to improve the traditional dynamic programming algorithm and hidden Markov model: modified melody matching method and modified hidden Markov method. These two methods provide better performance than the traditional two methods in Chapter 8.

On the aspect to construction of our music database, it is also a big work. Because there are no free open sources for database of pitch interval, we have to construct one by ourselves. Our construction is introduced in section 6.3 in detail by transferring the numbered notes into pitch interval.

Finally, our proposed onset detection shows better TP rates than any other onset detections in section 8.2. Our proposed melody matching method shows better HR and MRRR than any other melody matching method in section 8.3. Our system shows the best performance of the combination of our proposed onset detection and MDP in section 8.4 and the HR reaches from 96 to 100 % in the versions of 15-second and 20-second.

9.2 Future Work

Although we improve the performance of the known onset detections and melody matching methods, the performance is still not good enough due to the database size is too small. We want create a QBH system whose performance can reach 90 % up for hit rate and 95 % up for MRRR (right answer in top 5) in ideal case.

The onset detection is still a problem that we have to design a more robust onset detection to deal with any kind of singing style and singing error. Although our onset detection has good detection rate, we still require higher TP rate on onset detection. Our next goal is to design a 99% up onset detection. Pitch estimation seems not a problem by using our method but we can combine both onset detection and pitch estimation together to detect more accurate onset times in future work.

We need more fast speed in matching the similarity due to QBH system requires real time response online. Our MDP and MHMM method perform better than DP

algorithm and HMM method though, the future work needs to design a more robust and fast algorithm. We want to reach 90 % up for HR in 1000 targets of database for the next goal.

In this paper, we only use one music feature: pitch interval. We can add another common feature, ***IOI,*** to the database. Not only calculating the similarity of pitch interval but also comparing the similarity of ***IOI*** between query and targets is our next goal.

Finally, we will compare the performance for more situations like different query length or same song sung by different individuals. Our goal is to make a fast, high hit rate and robust QBH system for future work.

REFERENCE

A. Query-by-Humming System

[1] S Pauws, “CubyHum: a fully Operational Query by Humming System,” *in Proc. of ISMIR*, pp. 187-196, Citeseer, 2002.

[2] A. Ghias, J. Logan, D. Chamberlain, and B.C. Smith,, “Query By Humming: Musical Information Retrieval in an Audio Database,” *in Proc. of the ACM international Multimedia conference and exhibition*, pp. 231-236, *San Francisco, California*, Nov. 1995.

[3] N. Kosugi, Y. Nishihara, S. Kon'ya, M. Yamamuro, and K. Kushima, “Music Retrieval by Humming,” *in Proc. of PACRIM'99. IEEE*, Aug. 1999.

B. Onset Detection

[4] A. Klapuri, “Sound Onset Detection by Applying Psychoacoustic Knowledge,” in *Proc. of IEEE International Conference on Acoustics, Speech and Signal*, 1999.

[5] A. M. Barbancho, A. Jurado, I. Barbancho, “Identification of Rhythm and Sound in Polyphonic Piano Recordings,” *in Proc. of Forum on Acousticum*, Sevilla, 2002.

[6] C. Duxbury, M. Sandler, and M. Davies, “A hybrid approach to musical note onset detection,” *in Proc.* of *the 5th International Conference on Digital Audio Effects*, pp. 33–38, 2002.

[7] J. Foote, “Automatic audio segmentation using a measure of audio novelty,” *in Proc. of IEEE International Conference on Multimedia and Expo*, issue 1, pp. 452–455, 1999

[8] P. Masri, and A. Bateman, ” Improved modelling of attack transients in music

analysis-resynthesis," *in Proc. of International Computer Music Conference (ICMC 96), Hong-Kong, Aug 1996,*

[9] P. Bello J, G. Monti, M. Sandler, "Techniques for Automatic Music Transcription," *in Proc. of International Symposium on MIR,* Polymouth, Massachusetts, 2000.

C. Pitch Estimation

[10] A. de Cheveigne and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *in Proc. of Acoust. Soc. Am.,* vol, 111, Issue, 4 pp. 1917-1930, April 2002.

[11] A. Klapuri, "Multipitch analysis of polyphonic music and speech signals using an auditory model," *in Proc. of IEEE Trans. Audio, Speech and Language*, vol. 16, pp.255–266, February 2008.

[12] D. Hermes, "Measurement of pitch by subharmonic summation," *in Proc.* of *JASA*, vol. 83, no. 1, pp. 257–273, 1988.

[13] D.J. Levitin, "Absolute memory for musical pitch: Evidence from the production of learned melodies," *Perception & Psychophysics,* vol. *58*, pp. 927-935. 1994

[14] E. S. Tsau, N Cho, CCJ Kuo, "Fundamental Frequency Estimation for Music Signals with Modified Hilbert-Huang Transform," *in Proc. of ICME'09*, June, 2009.

[15] E. Pollastri, "Melody-retrieval based on pitch-tracking and string-matching methods," *in Proc. of the 12th Colloquium on Musical Informatics,* 1999.

[16] H. Hajimolahoseini, M.R. Taban, Abutalebi, H.R. "Automatic Transcription of Music Signal Using Harmonic Elimination Method," *Telecommunications,* 2008.

[17] L.R. Rabiner, and B. Juang, *"Fundamentals of Speech Recognition*," Prentice-Hall Inc. 1993.

[18] N.E. Huang, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," et al. (1998).

D. Melody Matching

[19] Z.K. Peynirçioglu, A.I. Tekcan, J.L. Wagner, T.L. Baxter, and S.D. Shaffer, "Name or hum that tune: Feeling of knowing for music," *Memory & Cognition, vol. 26, issue. 6*, pp. 1131-1137, 1998

[20] B. Pardo, WP Birmingham and J. Shifrin, "Name that Tune: A Pilot Study in Finding a Melody from a Sung Query", *in JASA. on Information Science and Technology*, vol. 55, pp. 283-300, 2000

[21] E. Ukkonen, "Approximate string matching with qgrams and maximal matches. *Theoretical Computer Science,"* vol. *92, issue 1*, pp. 191-211, 1992

[22] G. Myers "A fast bit-vector algorithm for approximate string matching based on dynamic programming*," Journal of the ACM, vol. 46, issue. 4*, pp. 395-415, *1999*

[23] J.S. R. Jang and H.R. Lee, "Hierarchical Filtering method for content-based music retrieval via acoustic input,", *in Proc. of the 9th ACM international conference on Multimedia*, pp. 401-410, ACM Press, 2001.

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

[25] P.H. Sellers, "The theory and computation of evolutionary distances: Pattern recognition," *Journal of Algorithms,* issue 1, pp. 359-373. 1980

[26] R. B. Dannenberg, "Understanding Search Performance in Query by Humming

Systems," *in Proc. of ISMIR*, Citeseer, 2004.

[27] R.A. Wagner, and M.J. Fischer, " The string-to-string correction problem, "*Journal of the Association of Computing Machinery*, vol. 21, issue 1, pp.168-173. 1974

[28] R. Baeza-Yates, and G. Navarro," Faster approximate string matching," *Algorithmica*, vol. 23, issue 2, pp. 127-158, 1999.

[29] R.J. McNab, L.A. Smith, I.H. Witten, and C.L. Henderson, "Tune retrieval in the multimedia library, *Multimedia Tools and Applications," vol. 10*, pp. 113-132, 2000

[30] T. Nishimura, H. Hashiguchi, J. Takita, J.X. Zhang, M. Goto, and R. Oka, "Music Signal Spotting Retrieval by a Humming Query Using Start Frame Feature Dependent Continuous Dynamic Programming," *in proc. of ISMIR*, pp. 211-218, Bloomington, Indiana, 2001.

[31] W. Chang, and E. Lawler "Sublinear approximate string matching and biological applications," *Algorithmica*, vol. 12,issue. 4/5, pp. 327-344. 1994

[32] W.L. Dowling, "Scale and Contour: Two components of a theory of memory for melodies.," *Psychological Review*, vol. 85, issue. 4, pp. 341-354. 1978

[33] W.J. Dowling, and D.L. Harwood,"*Music cognition*.," *New York: Academic Press*, 1986

[34] W. Schloss, "*On the Automatic Transcription of Percussive Music: From Acoustic Signal to High Level Analysis*," *PhD Thesis, Department of Music*, Report No. STAN-M-27, Stanford University, CCRMA. 1985

[35] Y. Zhu, and M. Kankanhalli, "Music Scale Modeling for Melody Matching," *In Proc. of 11th ACM international conference on Multimedia*, pp. 359~362, Nov. 2003.

[36] Y. Zhu and D. Shasha, ”Query by humming: a time series database approach,” *In Proc. of SIGMOD*, 2003.

[37] Y.W. Zhu, M. Kankanhalli, “A Robust Music Retrieval Method for Query-by-Humming”, *In International Conference on Information Technology: Research and Technology,* New Jersey, USA, Aug. pp. 10-13, 2003

E. Others

[38] E.D. Scheirer , “Tempo and Beat Analysis of Musical Signals,” *In J. Acoust. Soc. Am*, vol. 103, Issue 1, pp. 588-601, January 1998.

[39] J.C. Brown, and J.C. Vaughn, ”Pitch center of stringed instrument vibrato tones,” *In J. Acoust. Soc. Am, , vol. 100*, pp. 1728-1735, 1996