Development of Scoring Algorithm for Karaoke Computer Games

DONGPING QIU

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden April 2012

XR-EE-SB 2012:004

Development of Scoring Algorithm for Karaoke Computer Games

Master Thesis

Dongping Qiu
dqiu@kth.se

Supervisor Per Zetterberg
Royal Institute of Technology
Oscar Norlander
Pop Atom AB

Examiner Magnus Jansson
Royal Institute of Technology

Royal Institute of Technology, KTH
Stockholm, Sweden
April, 2012

Those who wish to sing always find a song. — Swedish proverb

Abstract

In a Karaoke computer game, the users receive a score as a measure of their performance. A music recognition system estimates the underlying music notes the users have performed. Many developed approaches use deterministic signal processing.

This thesis builds statistical hidden markov models (HMM) to be used in scoring. The HMM-based note models are based on the musical features pitch, accent, zero crossing rate and power. Gaussian mixture models (GMM) are used to determine the probability distributions of the features.

A singing database is constructed by an amateur singer who is well-trained in the area of music. During the training stage, an intra-note model is trained using the HTK toolbox. The intra-note model describes the statistical behavior of model states inside a note.

Then, a group of note models, covering a wide range of pitch, is built upon the trained intra-note model. Also a recognition network is constructed. During the recognition stage, the test data are decoded with the aid of the note models and the recognition network.

An experiment compares the proposed approach with the framed-based and note-based approaches. The results show that our approach provides promising results over deterministic approaches.

Acknowledgements

This thesis is provided by Pop Atom AB, and carried out in the School of Electrical Engineering at the Royal Institute of Technology (KTH). It is the final part of my education to a Master degree in Wireless Systems.

I would like to express my sincere gratitude to my supervisor Per Zetterberg for his insightful advice and patience as well as his consistent encouragement throughout this project.

I am deeply grateful to my supervisor Oscar Norlander for introducing me to this interesting work and for all valuable help, instruction and discussions.

I would like to make a solute to my friend and an amateur singer Siyang Dai for his aid of building the acoustic database used in this thesis, and for sharing his knowledge of music theory.

I want to appreciate Marcus Lippert, the CEO of Pop Atom AB, for his support and kindness toward me. I wish to thank my examiner Magnus Jansson and thesis opponent Xiaohua Lu for reviewing this report and for their useful feedback.

I extend my gratitude to Matti Ryynänen from Tampere University of Technology and Olivier Lartillot from University of Jyväskylä, Finland for introducing me to their work.

Finally, I dedicate special thanks to my parents for their endless support and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In a Karaoke computer game, users solo along with musical accompaniment and lyrics rolling on the screen. According to the degree of precision a song is performed, users receive a grade in the end. Some games also afford users on-line feedback.

Figure 1.1 shows the screen-shot of a Karaoke computer game. The light

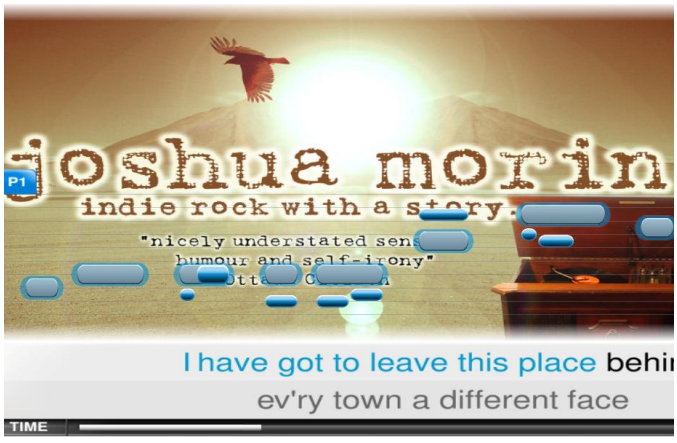

Figure 1.1: Screen-shot of a Karaoke computer game

blue bar indicates the reference notes. The dark blue bar indicates the sung notes. On the bottom of screen are lines of lyrics, where the performed portion is marked by blue color.

Karaoke computer game is adorable entertainment for music fans who want to interact with popular songs. More importantly, it provides an opportunity

for people lack of music education to improve their singing skills.

The motivation of this work is to develop a tool for the automatic evaluation of single-voice singing for a Karaoke computer game. The idea is to design a music recognition system. Traditionally, music recognition means handwriting a music sheet for a piece of tune. Automatic recognition system accepts acoustic input and generates music notes.

This project requires to take the pitch and duration of each note into account. To examine the users' performance, a quantitative evaluation is employed to calculate the error rate between recognized notes and reference notes. The outcome is presented in percentage, which works as a *score* assigned to the users.

Figure 1.2 describes the flow chart of Karaoke computer game.

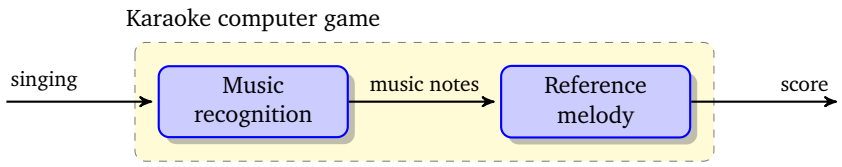

Figure 1.2: Flow chart of Karaoke computer game

Human voice is considered to be the most expressive musical instrument [40]. However, accurate conversion from singing waveform to notes has turned out to be very difficult [43]. Many developed music recognition systems use deterministic signal processing.

This thesis builds note models in order to statistically capture the most common behavior of sung notes. The note models are used to further predict the most potential sequence of notes by calculating the likelihood.

During the training stage, an intra-note model is trained with data performed by an amateur singer who is well-trained in the area of music. The intra-note model describes the statistical behavior of model states inside a note.

Then, a group of note models, covering a wide range of pitch, are built upon the trained intra-note model. Also a recognition network is constructed. During the recognition stage, the test data are decoded with the aid of the note models and the recognition network.

1.2 Outline

The rest of this report is organized as follows.

Chapter 2 reviews the elementary music knowledge necessary to comprehend the thesis. Chapter 3 briefly describes diverse methods of signal processing in the field of deterministic recognition.

Chapter 4 focuses on the knowledge of statistical recognition. Chapter 5 explores the implementation of a HMM-based recognizer.

Chapter 6 compares the proposed recognizer with two deterministic approaches. Eventually, conclusions and future works are presented in Chapter 7.

Chapter 2

Music Theory Fundamentals

The purpose of this chapter is to introduce several concepts related to music theory and notations used throughout the report. Section 2.1 deals with our human's perception to music signals, including the basic music element — tone, and its attributes. Section 2.2 discusses the music unit to denote tone, i.e. note, the relation between distinct notes, a widely-used notation system, and finally the representation of music in digital world.

2.1 Perception of Music

Human being can easily distinguish music and noise without any formal musical education. Music is ordered and patterned sound waves. The sound source (e.g., musical instruments) produces various harmonic vibrations and spreads through the surrounding air in the form of sound wave that is regular and periodic, which is called tone.

Consequently, it is the presence of tones that distinguishes the music from noise. A larger part of music is build upon the combination of tones. To better understand music, it is important to be familiar with its major attributes. These attributes include pitch, intensity, duration and timbre [28, pp. 37-38], which are shown in Figure 2.1.

2.1.1 Pitch

Musical tones all have certain pitch values. The pitch of a tone refers to how high or how low a tone is. Pure tones have periodic, sinusoid waveforms. Pitch is the ear's perception of the wavelength of a tone, shown in Figure 2.2.

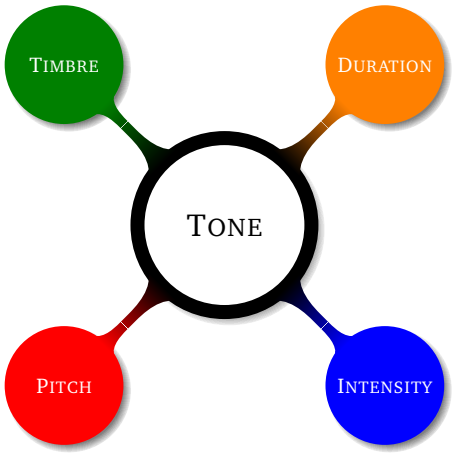

Figure 2.1: Attributes of musical tone

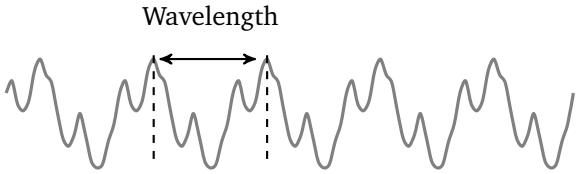

Figure 2.2: Wavelength of pure tone

High-pitched tones have relatively short wavelengths, while low-pitched tones have relatively long wavelengths [19, p. 3]. Since the tones are results of oscillation of sound source [29], high pitch means very rapid oscillation, and low pitch means slow oscillation.

Complex and aperiodic sound waves can be decomposed into a combination of pure tones [15, p. 381]. In general, the pitch of complex sound like sung voice corresponds to the repetition rate of similar patterns in the waveform.

A musical instrument produces many sorts of vibration simultaneously. The listeners hear various frequencies. The vibration that has the slowest rate is called fundamental frequency (F_0); any frequency higher than the fundamental is called an overtone [22]. An important class of overtones are harmonics, which have frequencies in integer multiples of fundamental.

Pitch is usually quantified as fundamental frequency, but these two terms are inequivalent. Frequency is a purely objective physical concept; whereas pitch is a subjective psychoacoustic attribute of sound [20, p. 68]. Sound waves themselves do not have pitch, and their oscillations can be measured to obtain a frequency. It takes a human brain to map the internal quality of pitch.

Frequency is usually measured in Hertz (Hz) or kilo Hertz (kHz), which

is a measurement of the number of sound waves per second. The general range of human hearing spans from 200 Hz to 20 kHz. Beyond this range the sounds are too low or too high for our hearing system.

Not all music sound make a clear pitch. A sound of definite pitch is one of which it is possible to discern the pitch [26]. Sounds with definite pitch have harmonic frequency spectra. On the other side, a sound of indefinite pitch is one of which it is very difficult to discern a pitch. These sounds do not have harmonic spectra, for instance, all the consonant phonemes in English speaking system.

2.1.2 Intensity

Another important property of musical tone is intensity, also known as loudness. The intensity of sound refers to the magnitude of acoustic waves. High-magnitude sound carries more energy than low-magnitude sound. Consequently, an increase of wave magnitude results in an increase of volume perceived by ears, and vice versa. Figure 2.3 depicts a sound that is getting louder. A-weighting decibel, abbreviated as dBA, is in common use for assessing the loudness. 0 dBA is considered as the quietest level of sound the human ears can pick up.

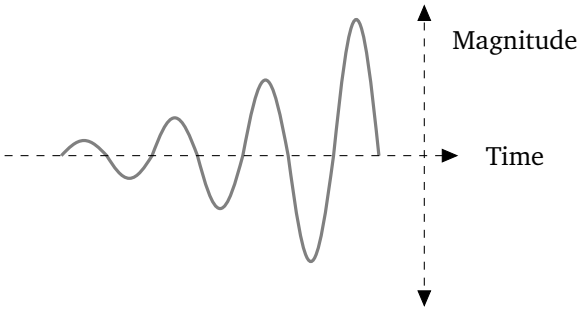

Figure 2.3: Increase of magnitude in a sinusoid waveform

2.1.3 Duration and Timbre

Musical tone can sustain for a specific amount of time, which is called duration. Timbre, also called tone quality, is the attribute of tone that makes ear to identify the difference between, say a piano and a guitar that playing the same note [19, p. 8]. Even both instruments produce the same fundamental frequency, they create different harmonics, which are tightly related with timbre, through vibration.

2.2 Music Terminology

Like any other principles, music world has its own rules and laws. It is necessary to understand the basic music theory for any research in this area. The musical terms introduced in this section is helpful to understand the entire thesis work. Even though many readers may have encountered the following musical terms before, it is still useful to review them once again.

2.2.1 Note

In music theory, the term note is used to represent the pitch and duration of a tone. The term octave is used to indicate the range of two notes having a frequency ratio of any power of two. For example, if one note has a frequency of 400 Hz, the note one octave above is at 800 Hz, and the note one octave below is at 200 Hz. Human ears tends to hear notes with 'octave' relations as being very similar, due to the closely related harmonics. For this reason, all notes with one or multiple octaves apart are grouped in the same pitch class.

Traditionally pitch class is denoted through the use of the first seven Latin letters, i.e. A, B, C, D, E, F and G. Letter names are suffixed by the accidental, sharp (♯) and flat (♭). Sharp raises a note by one semitone, and flat lowers a note by one semitone. For instance, note C♯ is one semitone higher than note C, and note B♭ is one semitone lower than note B.

A note also can be classified with regard to its duration. In order of duration, it generally has: whole note, half note, quarter note. A whole note has time length of four beats in 4/4 time. A half note has half the duration of a whole note and twice the duration of a quarter note. Thus, a quarter note has time length of one beat. It is also helpful to point out that the onset of a note means the start instant, and the offset means the end instant of a note.

2.2.2 Chromatic Scale

A music scale is a sequence of notes in ascending or descending order. Most scales are octave-repeating, which means their pattern of notes is same in every octave. For instance, the C major scale is C-D-E-F-G-A-B-C, where the last note C is one octave higher than the first note C.

One widely used scale is chromatic scale, with twelve notes, and each a semitone apart. Chromatic scale corresponds to the white-black piano keyboard, i.e. C-C♯-D-E♭-E-F-F♯-G-G♯-A-B♭-B-C. Again the last note C is one octave above the first note C. Figure 2.4 shows one octave scale in the piano keyboard.

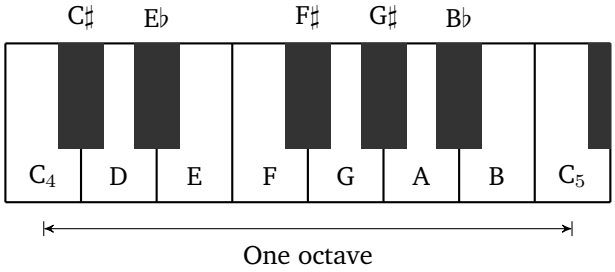

Figure 2.4: One octave of chromatic scale on piano keyboard

2.2.3 Scientific Pitch Notation

Diverse music notations systems are used to denote notes. One formal way named 'scientific pitch notation' is extensively applied in the music world. It suffixes a note of chromatic scale with a number to denote octave. For example, C_3 means the note C one octave below C_4 (the middle C).

Technically, each note corresponds to a fundamental frequency F_0. These frequencies are defined around the central note, A_4 ($440\,\text{Hz}$). Assume the semitone distance of a note away from A_4 is s. If the note is above A_4, s is positive; otherwise s is negative. The F_0 is calculated as

$$F_0 = 2^{\frac{s}{12}} \cdot 440. \tag{2.1}$$

Conversely, one can obtain s from F_0

$$s = 12 \cdot \log_2 \left\{ \frac{F_0}{440} \right\}. \tag{2.2}$$

Table 2.1 summaries the frequencies of some common music notes. In each cell, the number in the left hand-side is the frequency of a note, and the second number is the semitone distance s above or below A_4.

2.2.4 Digital Music and MIDI Format

With the development of digital technology, digital audio is widely used in the recording, manipulation, and storage of sound. Modern storage of music across the Internet depends on digital recording and digital compression techniques.

In an analog audio system, sound begins as physical waveform in the air, and is stored via magnetic materials. Analog signal is susceptible to distortion, unavoidable due to the influence of environmental factors. When an analog signal is digitized, it is first sampled, and then converted into binary signals. Digital audio is discrete-time discrete-level signal.

Fundamental Frequency F_0 in Hz — Semitone distance s away from A_4										
Note \ Octave	2		3		4		5		6	
C	65.41	-33	130.81	-21	**261.63**	**-9**	523.25	+3	1046.5	+15
C♯	69.30	-32	138.59	-20	277.18	-8	554.37	+4	1108.7	+16
D	73.42	-31	146.83	-19	293.66	-7	587.33	+5	1174.7	+17
E♭	77.78	-30	155.56	-18	311.13	-6	622.25	+6	1244.5	+18
E	82.41	-29	164.81	-17	329.63	-5	659.26	+7	1318.5	+19
F	87.31	-28	174.61	-16	349.23	-4	698.46	+8	1396.9	+20
F♯	92.50	-27	185.00	-15	369.99	-3	739.99	+9	1480.0	+21
G	98.00	-26	196.00	-14	392.00	-2	783.99	+10	1568.0	+22
G♯	103.83	-25	207.65	-13	415.30	-1	830.61	+11	1661.2	+23
A	110.00	-24	220.00	-12	**440.00**	**+0**	880.00	+12	1760.0	+24
B♭	116.54	-23	233.08	-11	466.16	+1	932.33	+13	1864.7	+25
B	123.47	-22	246.94	-10	493.88	+2	987.77	+14	1975.5	+26

Table 2.1: Note frequencies of scientific pitch notation

Digital music has diverse formats, e.g. mp3 and wma. One format named 'musical instrument digital interface' (MIDI) is special. Three benefits make MIDI very popular. First, instead of recording a waveform of music, MIDI format stores the instructions on how to recreate the music. MIDI synthesizer could recover the original music from those instructions.

Second, MIDI file costs much less storage space than other digital formats. For example, a MIDI file of 5 kB can reproduce a full minute of music. Small file size is beneficial for the applications with limited storage space. Third, MIDI helps electronic music composers enjoy a significant degree of creation.

To further understand MIDI, a note matrix is used to represents all the notes[13]. The note matrix contains r rows and 7 columns for each channel, where r is the number of notes. Table 2.2 gives the row format of note matrix.

Onset (beat)	Duration (beat)	MIDI Channel	MIDI Pitch	Velocity	Onset (sec)	Duration (sec)

Table 2.2: Row format of note matrix for MIDI files

The first column indicates the onset instant of a note. The second column is the duration of a note. They are measured in unit of beat numbers. Beats per minute (BPM) is typically used to decide the tempo of music. For instance, 120 BPM indicates that there is 120 beats per minute. As mentioned in

Section 2.2.1, the duration of a quarter note is one beat. In other words, a quarter note of 120 BPM lasts for 0.5 seconds. Briefly, large BPM means short duration.

The third column is the channel index, with value of 1 – 16. The fourth column is the pitch, with A_4 defined as 69. One semitone higher increases the pitch by 1, e.g. $A\sharp_4$ is 70. In a similar manner, one semitone lower reduces the pitch by 1. Velocity in the fifth column defines the intensity (see Section 2.1.2) of a note, with valued of 0 – 127.

Column 6 and 7 reiterate column 1 and 2 in unit of second. In a word, MIDI includes three attributes of musical tone: pitch, intensity and duration. Table 2.2 emphasizes the fact that MIDI tells synthesizer how to reproduce music sound rather than storing a waveform.

Figure 2.5 summarizes the relationship between notes and MIDI numbers.

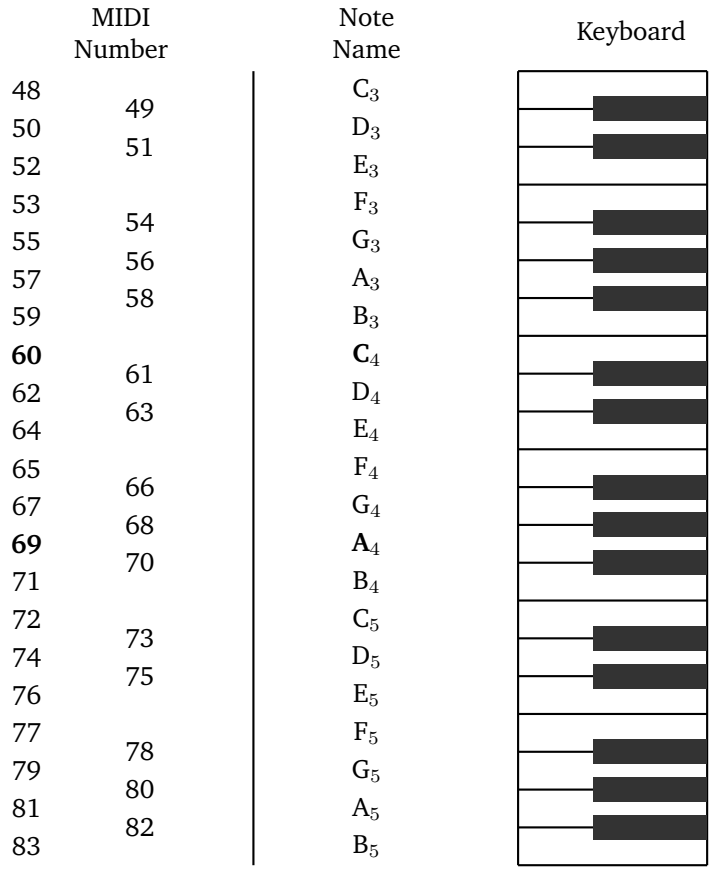

Figure 2.5: Mapping from notes to MIDI numbers, C_3 – B_5

Throughout this thesis, pitch is represented as a MIDI number. Equation 2.2

infers how to calculate MIDI number p from the F_0:

$$p = 69 + 12 \cdot \log_2 \left\{ \frac{F_0}{440} \right\} \tag{2.3}$$

where 69 is the MIDI number of note A_4, 440 is its frequency in Hz, and 12 indicates one octave distance.

Chapter 3

Deterministic Music Recognition

The deterministic recognition of single-voice melodies consists of two steps. First, a trajectory of pitch is detected from an acoustic waveform. This step is largely a solved problem. Second, the pitch trajectory is transcribed to notes.

Figure 3.1 shows the flow chart of deterministic recognition, consisting of three modules. The pitch detection module realizes the first step, and post-

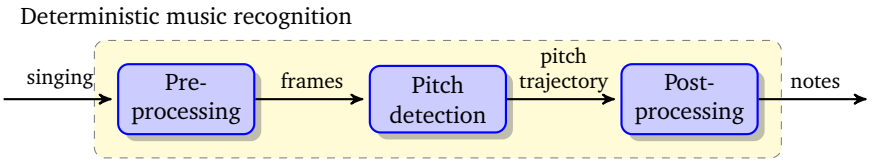

Figure 3.1: Flow chart of deterministic music recognition

processing module realizes the second step. In this chapter, Section 3.1, 3.2 and 3.3 discusses these modules in a serial order.

3.1 Pre-Processing of Audio Signals

Splitting an audio signal into a sequence of frames (i.e. segmentation) is a basic necessity before pitch detection. The motivation is that the characteristics of audio signal are continuously changing. For example, pitch is stable only within a short period. It makes sense to segment an audio signal into frames, in which the characteristics are stationary. Frame lengths from 20 ms to 60 ms are common options.

The most simple idea of segmentation is to split the signal into disjointed frames sequentially. The feature of a frame refers to information that determines the characteristic of the frame. One condition might happen that the feature is chopped in half and falls in two frames: the first half is in the preceding frame, and the other half is in the succeeding frame [25, p. 18]. In this manner, any vital information carried in this feature will be missed.

A common solution is overlapping frames. Overlapping means the head of frame $i+1$ starts inside the body of frame i, instead of following after the tail of frame i [25, p. 18]. Each new frame contains part of the preceding frame and part of the succeeding frame. The degree of overlapping (ρ) refers to the ratio of the frames $i+1$ repeated in the frame i

$$\rho = \frac{\text{The length of overlapped portion}}{\text{The length of frame}} . \tag{3.1}$$

One advantage of overlapping is that it sufficiently utilizes the limited resource, and results in more frames. For instance, a 50% overlapping doubles the number of frames. Figure 3.2 illustrates the process of overlapped segmentation. Panel (a) presents an audio signal. Panel (b) splits the signal into seven 50% overlapped frames.

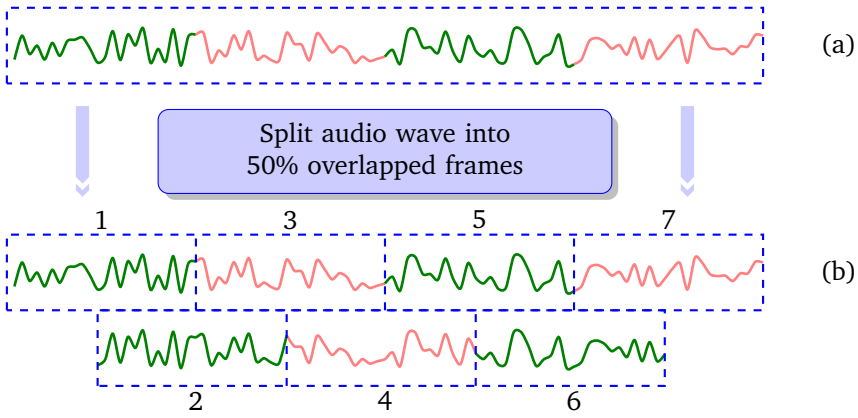

Figure 3.2: Overlapped segmentation of an audio signal: (a) The original waveform; (b) The signal is split into seven 50% overlapped frames.

3.2 Pitch Detection Algorithm

Pitch detection algorithm (PDA) is an algorithm to select the best candidate for the pitch or fundamental frequency (F_0). Tracking pitch of the human voice is a popular subject of study. In the early 1940's, the idea of designing

a pitch-frequency estimator was developed. Precise estimation of voice pitch appears to be necessary in the fields of speech synthesis, music retrieval, and recognition of singing.

Hundreds of pitch estimation schemes have been proposed and tested. For instance, B. Gold and L. Rabiner proposed a parallel estimator in 1969, which consisted of six individual peak detectors [14]. In their scheme, the distance between adjacent peaks was computed and used to determine the final pitch period, whose inverse was the pitch frequency.

Instead of obtaining the pitch in time domain, Puckette utilized frequency spectrum to discover the underlying music notes of singing [30]. In his first step, constant-Q spectrum [8] was adopted to get a pitch trajectory. With regard to the fact that the power at onset and offset of a music note was much lower than the average level, a power threshold was used to recognize the boundaries of potential notes. Finally, the maximum likelihood function was used to determine the most matching note trajectory.

PDAs are modified and used in various contexts to meet different demands placed upon the algorithms. Thus there is no single ideal algorithm, and a variety of algorithms exist. However most PDAs fall into two groups: the time domain and the frequency domain. Section 3.2.1 describes the basic autocorrelation algorithm and its modifications.

Section 3.2.2 discusses the frequency-domain PDAs through two specific algorithms. Finally, Section 3.2.3 analyzes the weakness and limitations of PDA algorithms.

3.2.1 Time Domain Approach

In the time domain, the PDA primarily intends to estimate the fundamental period T, and then inverts that value to acquire the fundamental frequency F_0 [20, p. 71].

One of the most reliable and widely-used approach is autocorrelation function (ACF) [34]. It is the cross-correlation of a signal with itself. It is a mathematical tool to find out the F_0 of a periodic signal buried under harmonics. Many PDAs are build upon the ACF. Average squared difference function (ASDF) is an ACF-like algorithm. Normalized cross-correlation function (NCCF) was used in Talkin's RAPT pitch tracking algorithm [41].

Autocorrelation Function (ACF)

ACF compares a segment with other segment offset by a trial distance to find a similarity. The distance of the most match is viewed as the fundamental

period. The computation is made directly on the waveform, and is fairly straightforward [32].

Given a sampled periodic signal $x(t)$, and analysis window size W, the ACF can be defined as

$$r_t(\tau) = \sum_{j=t+1}^{t+W} x(j) \cdot x(j+\tau) \tag{3.2}$$

where t is the start time index, and τ is the lag [33].

The minimal non-zero lag in the summit of $r_t(\tau)$ is picked up as fundamental period $\hat{T}$

$$\hat{T} = \arg\max_{\tau>0}\{r_t(\tau)\}\,. \tag{3.3}$$

The associated fundamental frequency is

$$\hat{F}_0 = 1/\hat{T}\,. \tag{3.4}$$

ACF produces quite accurate results for highly periodic signals, whereas it copes badly with octave errors. The term 'octave error' is explained in the beginning of Section 3.3.1. Another flaw is the relative large window size in order to cover adequate spectrum range. Because the lag T increases as F_0 wanes, a large window is required for a small F_0. Noise mixed in the signal also can degrade the algorithm's accuracy to measure periodicity.

Average Squared Difference Function (ASDF)

The definition of ASDF is:

$$d_t(\tau) = \frac{1}{W} \sum_{j=t+1}^{t+W} [x(j) - x(j+\tau)]^2. \tag{3.5}$$

Equation 3.6 reveals its resemblance with ACF:

$$\begin{aligned} d_t(\tau) &= \frac{1}{W}\left[r_t(0) + r_{t+\tau}(0) - 2r_t(\tau)\right] \\ &= \frac{2}{W}\left[r_t(0) - r_t(\tau)\right] \end{aligned} \tag{3.6}$$

where $d_t(\tau)$ is an inverted, time-shifted and then scaled version of $r_t(\tau)$ [9, p. 43]. ASDF seeks minima for fundamental period instead of maxima like ACF does:

$$\hat{T} = \arg\min_{\tau>0}\{d_t(\tau)\}\,. \tag{3.7}$$

The ACF is sensitive to amplitude variation, that is — an increase in signal amplitude causes the ACF peak of amplitude to grow rather than to remain constant [18, p. 355]. In contrast, ASDF is less sensitive to amplitude changes.

Next, an example is considered to graphically illustrate ACF and ASDF. A signal of 50 ms consists of the fundamental frequency at 160 Hz, and the 3rth, 5th and 8th harmonics with the amplitude of 0.2, 0.4 and 0.1, and additive Gaussian noise. The sampling frequency is 16 kHz.

Figure 3.3 shows the waveform of this signal. The associated parameters are summarized in Table 3.1.

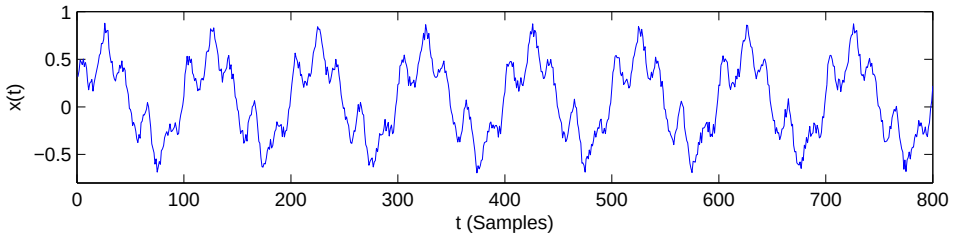

Figure 3.3: Waveform of a periodic signal with F_0 at 160 Hz and F_s at 16 kHz

Parameter	Setting
Signal duration	50 ms
Fundamental frequency, F_0	160 Hz
Number of harmonics	3
Standard deviation of noise, σ_N	0.15
Sampling frequency, F_s	16 kHz

Table 3.1: Parameters of the signal shown in Figure 3.3

ACF and ASDF are employed to analyze this signal in turn. The results are presented in Figure 3.4, indicating that detected pitches are close to the true F_0. It is claimed in [9, p. 43] that since ACF and AMDF are strongly related, none of them offer better results than the other. Nevertheless, modifications can lead to significant improvement, e.g. YIN algorithm.

YIN Algorithm

The YIN algorithm, proposed by de Cheveigné and Kawahara [11], is based on ACF and ASDF. YIN algorithm is chosen as the default PDA in this thesis. The YIN algorithm includes 5 steps stated as follows:

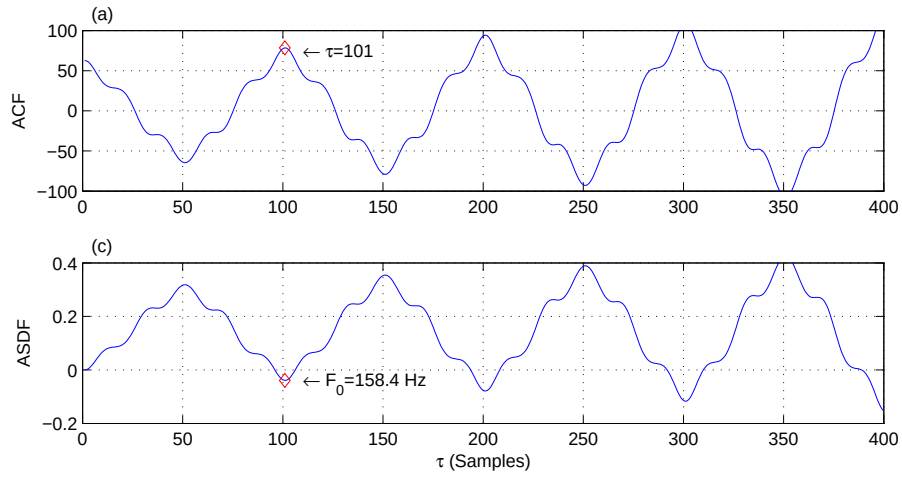

Figure 3.4: Comparison of ACF and ASDF: (a) ACF has its first local maxima at $\tau = 101$ samples, which is equivalent to a pitch of 158.4 Hz; (b) ASDF has its first local minima in the same position.

1. Compute the ACF of $x(t)$, i.e. $r_t(\tau)$.

2. Compute the ASDF of $x(t)$ as

$$d_t(\tau) = r_t(0) - r_t(\tau)\,. \tag{3.8}$$

3. Compute cumulative mean normalized ASDF:

$$\tilde{d}_t(\tau) = \begin{cases} 1 & \tau = 0 \\ d_t(\tau) / \left[(1/\tau) \sum_{j=1}^{\tau} d_t(j) \right] & \text{Otherwise} \end{cases} \tag{3.9}$$

4. Set a positive threshold κ, and search the first τ which corresponds to a local minimum of $\tilde{d}_t(\tau)$ below κ. If none such τ is found, the global minimum is chosen instead. Define the found lag as $\bar{\tau}$.

5. Interpolate the $\tilde{d}_t(\tau)$ using cubic spline across $\{\bar{\tau} - 1, \bar{\tau}, \bar{\tau} + 1\}$. Find the minimum between $[\bar{\tau} - 1, \bar{\tau} + 1]$ as period estimate $\hat{T}$.

Step 1 and 2 present ACF and ASDF separately. In Step 3, $\tilde{d}_t(\tau)$ starts from 1 instead of 0, and remains large for low lags. The algorithm avoids to choose a too small lag as period. Step 4 uses a threshold to determine a list of period

candidates. Step 5 improves the lag resolution via parabolic interpolation, otherwise the time period would be restricted to integer numbers.

Applying YIN algorithm to the periodic signal shown in Figure 3.3, the intermediate results are presented in Figure 3.5. Panel (a) – (c) present $r_t(\tau)$,

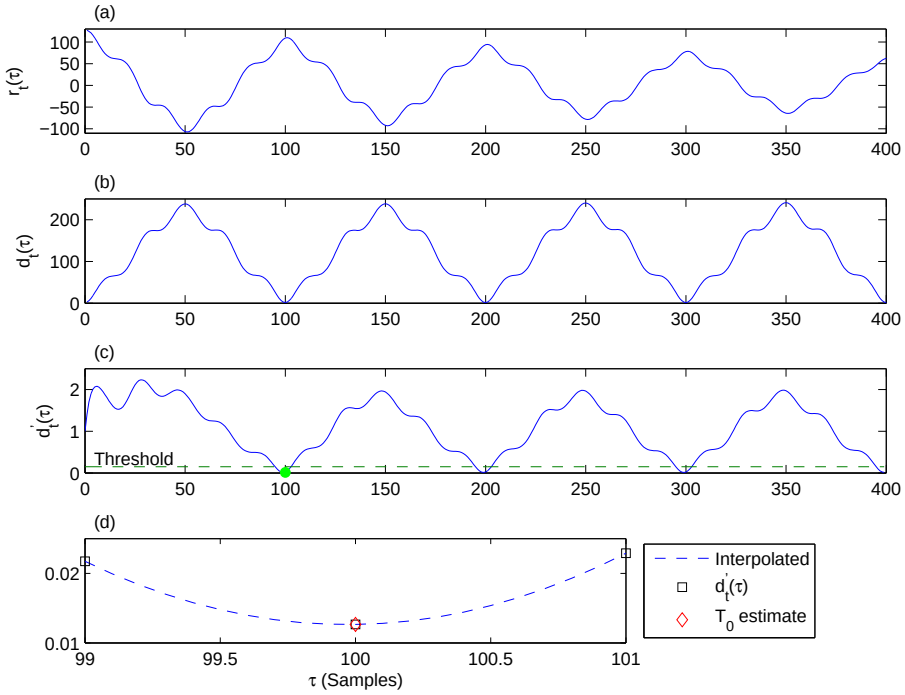

Figure 3.5: Illustration of YIN algorithm: (a) the ACF $r_t(\tau)$, (b) the ASDF $d_t(\tau)$, (c) the cumulative mean normalized ASDF $\tilde{d}_t(\tau)$, and (d) interpolated $\tilde{d}_t(\tau)$ around $\bar{\tau}$. The period estimate $\hat{T} = 99.9$ corresponds to a frequency $\hat{F}_0 = 160.1\,\text{Hz}$. A threshold $\kappa = 0.15$ is used.

$d_t(\tau)$ and $\tilde{d}_t(\tau)$ respectively. The first minima of $\tilde{d}_t(\tau)$ below threshold κ is found at the lag of $\bar{\tau} = 100$ samples. Panel (d) interpolates $\tilde{d}_t(\tau)$ with cubic spline between $[\bar{\tau} - 1, \bar{\tau} + 1]$ and results in an estimate of F_0 at $160.16\,\text{Hz}$.

3.2.2 Frequency Domain Approach

In frequency domain approach, signals are transformed (e.g. via FFT) to inspect their spectrum. This section discusses two common methods: harmonic product spectrum (HPS) and cepstrum (CEP).

Harmonic Product Spectrum (HPS)

HPS multiples the spectrum at the harmonics of F_0 and seeks the maximum as pitch [38]. Given $X(f)$ as the Fourier spectrum of signal $x(t)$, the definition is

$$\hat{F}_0 = \arg\max_f \prod_{r=1}^{R} |X(rf)| \tag{3.10}$$

where R is the number of harmonics to be considered. Usually R is limited between 5 to 11 in order to reduce the computational cost. Since the logarithm of a product is equal to the sum of logarithms, HPS can also be written as

$$\hat{F}_0 = \arg\max_f \sum_{r=1}^{R} \log |X(rf)| \tag{3.11}$$

A drawback of HPS is that, if any harmonic component has low energy, the product is close to zero, then the true pitch will be missed. Another problem is the need to enhance low frequency resolution with zero padding to the signal before Fourier transforming [12].

Cepstrum (CEP)

Cepstrum provides another solution to determine pitch. An audio signal $x(t)$ is assumed to be

$$x(t) = e(t) \circledast h(t) \tag{3.12}$$

where $e(t)$ is the excitation source, $h(t)$ is the impulse function of vocal tract, and $\circledast$ denotes convolution. To produce the indefinite pitch (see Section 2.1.1), a noise-like excitation signal is needed. To produce the definite pitch, a sequence of pulses is needed [42, p. 25]. Apply Fourier transform $\mathcal{F}\{\cdot\}$ to both sides of Equation 3.12, the convolution relationship becomes a multiplication:

$$X(f) = E(f) \cdot H(f) \tag{3.13}$$

where

$$\begin{aligned} X(f) &= \mathcal{F}\{x(t)\} \\ E(f) &= \mathcal{F}\{e(t)\} \\ H(f) &= \mathcal{F}\{h(t)\} \end{aligned}$$

First apply logarithm $\log\{\cdot\}$ then inverse Fourier transform $\mathcal{F}^{-1}\{\cdot\}$ to the magnitude of Equation 3.13, the cepstrum becomes

$$\begin{aligned} C(f) &= \mathcal{F}^{-1}\{\log\{|X(f)|\}\} \\ &= \mathcal{F}^{-1}\{\log\{|E(f)|\}\} + \mathcal{F}^{-1}\{\log\{|H(f)|\}\}\,. \end{aligned} \tag{3.14}$$

The multiplicative relationship in Equation 3.13 is transformed into an additive relationship. The effects of the excitation source and vocal tract are nearly independent or separable, whereas in ACF $e(t)$ and $h(t)$ are convolved together [18]. This explains why CEP is generally more accurate than ACF. The pitch is the frequency of the cepstrum's peak

$$\hat{F}_0 = \arg\max_f C(f)\,. \tag{3.15}$$

Applying HPS and CEP methods to the periodic signal shown in Figure 3.3, the resulting spectrum are shown in Figure 3.6. It indicates that both estimated pitches slightly deviate away from the true F_0.

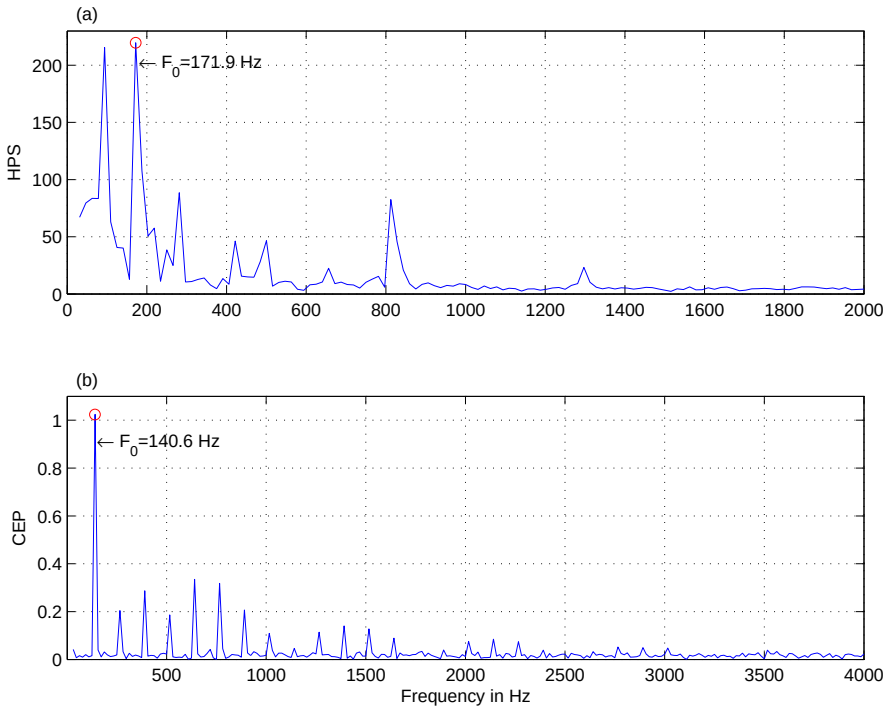

Figure 3.6: Comparison of HPS and CEP: (a) HPS has its highest peak at 172 Hz; (b) CEP has its global maxima at 141 Hz.

3.2.3 Shortcomings

Despite the fact pitch detection is an essential step in musical recognition, the direct results of PDAs are not quite reliable. The phenomena stated in below make F_0 difficult to estimate [41],

- F_0 changes rapidly and constantly with time, often with a short period.
- The estimates are easy to be identified at double or half of the true F_0.
- F_0 is often very unstable at voice onset and offset. The onset of a note can have an instantaneous pitch far away from the value eventually stabilized.
- Even during the "steady state" of a sung note, singing skills can cause deviation of several semitone away from the true pitch. For example, *vibrato* is a musical effect, using pitch vibration to add expression to vocal performance.

The pitch of sung voice does not keep steady during a note time. This problem even occurs for musical instruments, such as piano and violin. Thus PDAs result in an fluctuating pitch trajectory. Our human's perception system can ignore the small vibration of pitch, whereas PDAs magnify this variation.

Professional singers make use of skills to enhance the expressive effect of their performance. The singing skills can bring audience more euphonic perception of music. Unfortunately pitch can not reflect this improvement. Additionally, determining the boundaries between music notes is not easy due to the transition effect.

3.3 Post-Processing of Audio Signals

Having segmented the singing wave into overlapped frames, and calculated the pitch of each frame, Section 3.3.1 illustrates the pitch trajectory generated from raw estimates. To detect the implicit music notes of singing, Section 3.3.2 further transforms the pitch trajectory to note trajectory.

3.3.1 Frame-Based Pitch Estimation

In this thesis, the known reference file provides the correct pitch p_{ref} for each frame. The YIN algorithm is used to detect the raw estimate p_{raw} for each frame. Strong harmonics may lead p_{raw} to be a multiple value of the true F_0, which is called 'octave error'. If this case happens, p_{ref} is used to compensate octave errors by shifting p_{raw} by integer octaves to keep the distance of p_{raw} and p_{ref} no more than six semitones.

The compensated estimate is written as p_{est}. For example, p_{ref} is C_4 (MIDI 60), and p_{raw} is D_5 (MIDI 74), then p_{est} becomes D_4 (MIDI 62) after compensation. In addition, the concept of pitch error

$$p_{\Delta} = p_{est} - p_{ref} \tag{3.16}$$

is introduced to measure the gaps between p_{est} and p_{ref}. In this example, $p_{\Delta} = 62 - 60 = 2$ semitones.

In an attempt to absorb the variation of pitch trajectory, which is caused by the fluctuation of human singing voice, the term 'semitone tolerance' (φ) is introduced in this thesis. $\varphi = n$ means that any p_{est} within the distance of n semitones apart the p_{ref} is assumed to be correct. In the application of Karaoke computer game, semitone tolerance could be used to determine the difficulty levels of the game. Small φ allows an easy level, and large φ allows a hard level.

Figure 3.7 exemplifies the benefits of semitone tolerance. The data comes from the ninth song of an acoustic database introduced in Appendix A.1. The red dashed line denotes the pitch trajectory, and the blue solid line denotes the boundaries of $p_{ref} \pm \varphi$. The vertical dashed line indicates the onset of a note.

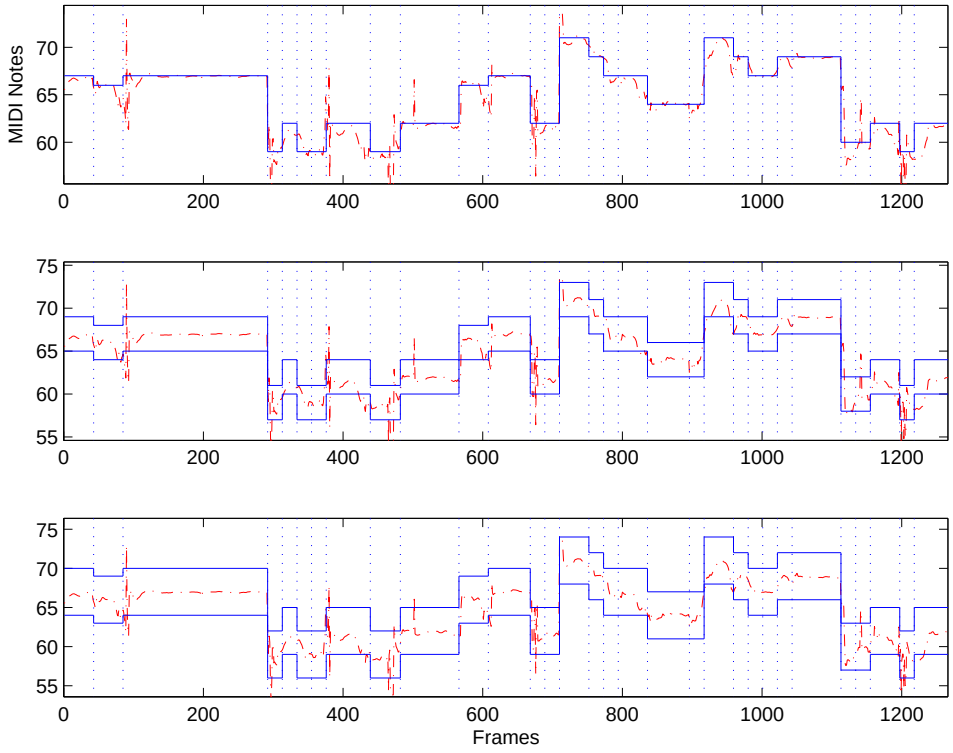

Figure 3.7: Detected pitch trajectory time aligned with reference boundaries under $\varphi = 0, 1, 2$: case I. The blue solid line denotes the boundaries $p_{ref} \pm \varphi$, and the red dashed line denotes p_{est}.

.

The top panel shows the initial pitch trajectory time aligned with reference. The spiking of pitch estimates in the onset stage is evident, since frames in

this period have low periodicity. Thus YIN algorithm fails to detect the true pitch, and leads to very high inaccuracy.

The middle panel adds one-semitone tolerance to the reference, in which the pitch trajectory generally falls in the area between the reference boundaries. The bottom panel adds two-semitone tolerance, and further reduces the pitch errors.

Figure 3.8 illustrates another example. In the top panel, the pitch errors are evidently large. In the onset and offset stages of many notes, the pitch errors even exceed three semitones. Though the singing sounds right and proper, the top panel implies that the performance is out of tones. The remaining panels add one and two-semitone tolerance respectively.

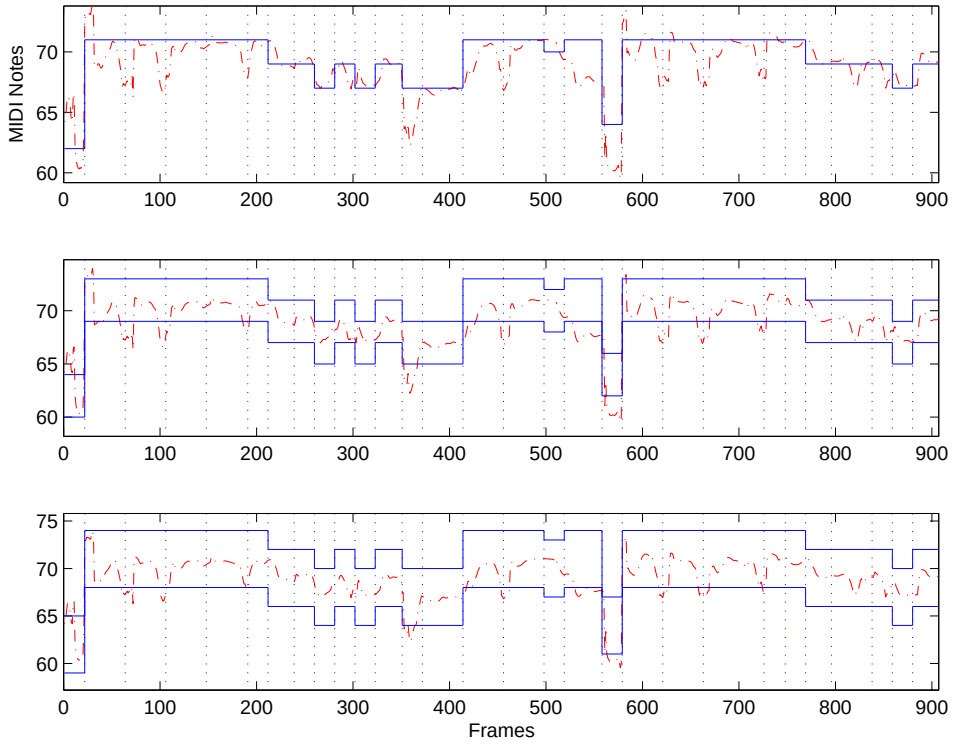

Figure 3.8: Detected pitch trajectory time aligned with reference boundaries under $\varphi = 0, 1, 2$: case II. The blue solid line denotes the boundaries $p_{ref} \pm \varphi$, and the red dashed line denotes p_{est}.

3.3.2 Note-Based Pitch Estimation

To further transform the pitch trajectory to note trajectory, the p_{est} across all the frames of a note is averaged. It is an open problem to determine the true boundaries of sung notes. To simplify this problem, this thesis assumes an ideal condition, that is — the singing exactly has the same timing with the reference melody. Any sung note begins in the right instant, also ends in the right instant.

The pseudo-code in below presents the algorithm used in this thesis.

Algorithm 1 Calculation of note pitch from frame pitches

Require: $\mathcal{I} \leftarrow \{i | k_1 \leq i \leq k_2\}$ ▷ The frame index set

1: $p_{mdn} \leftarrow \text{median}(p_{est})$
2: **for all** $i \in \mathcal{I}$ **do**
3: **if** $|p_{est}(i) - p_{mdn}| \geq 4$ **then**
4: $p_{est}(i) \leftarrow p_{mdn}$ ▷ Exclude highly biased estimates
5: **end if**
6: **end for**
7: $p_{avg} \leftarrow \text{mean}(p_{est})$

The top line defines a note whose frame index i spans from k_1 to k_2. Line 1 calculates the median p_{mdn} of all the frames within this note is calculated. Median is a statistical concept used to pick up the middle value of a list of numbers. It is recommended to use median value instead of mean value in skewed distribution, since median minimizes the risk of bias errors.

Consider a list of numbers {1, 3, 5, 7, 9, 11, 100}, the median value is 7 and the mean value is 20.83. The median represents the central tendency of this number set. Outlier 100 dramatically impacts the mean, whereas the median is less affected. Recall Figure 3.7, the estimates at the onset stage often drastically deviate from the correct pitch, leading to large outliers. Such deviation makes the distribution of estimates skewed. Thus median value is more feasible than mean value in our situation.

It will be useful if the outliers are further excluded from the data set. Line 4 assumes outliers as any p_{est} outside the boundaries of $p_{ref} \pm 4$ semitones. The outliers are further substituted with p_{mdn}. In the end of this algorithm, an averaged value p_{avg} is regarded as the pitch of a sung note.

Figure 3.9 transforms the pitch trajectory shown in Figure 3.7 into note trajectory by using the algorithm discussed in above. From the top panel to the bottom panel, φ posses the value of 0, 1 and 2 semitones separately. When $\varphi = 0$, the detected notes barely overlap with the reference. When $\varphi = 1$, it turns out that the note trajectory generally stays between the upper and lower reference bounds.

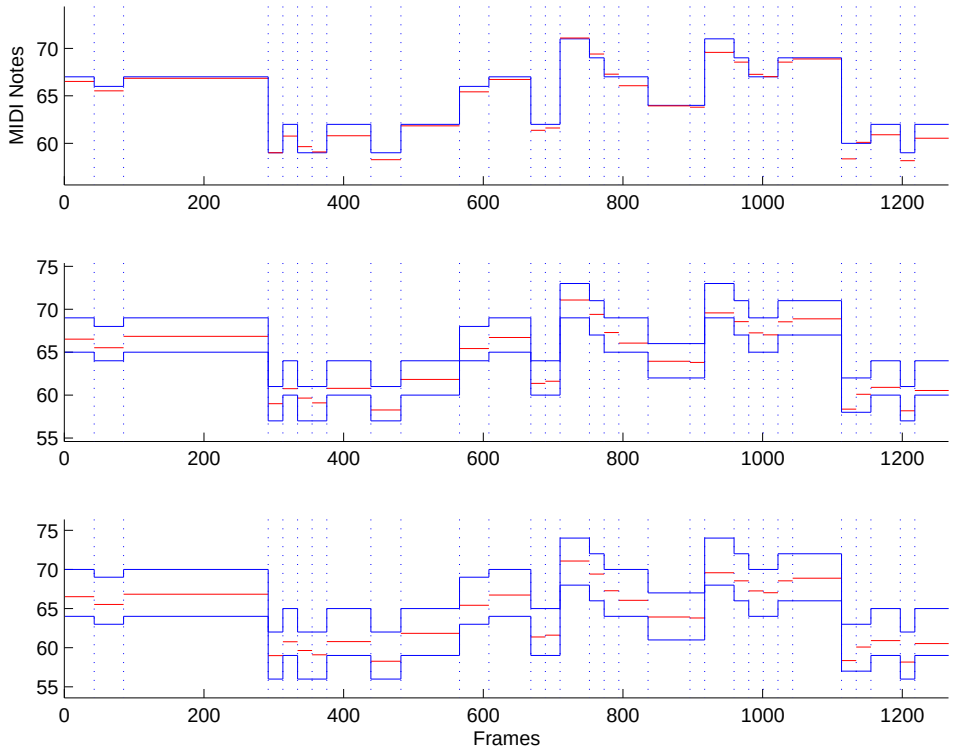

Figure 3.9: Detected notes time aligned with reference boundaries with $\varphi = 0, 1, 2$: case I. The blue line denotes $p_{ref} \pm \varphi$; the red line denotes p_{avg}.

Figure 3.10 adjusts the pitch trajectory in Figure 3.8. In the top panel of Figure 3.10, the pitch errors are considerably evident, owing to a rapid drop of pitch in the start time and end time of notes, which are shown in the top panel of Figure 3.8.

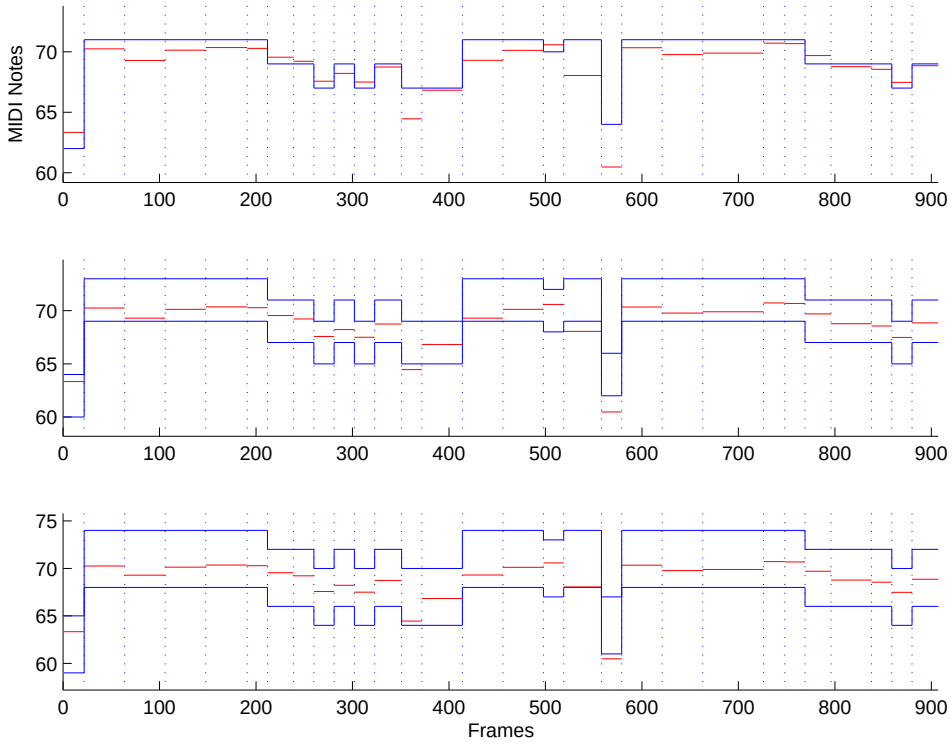

Figure 3.10: Detected notes time aligned with reference boundaries with $\varphi = 0, 1, 2$: case II. The blue line denotes $p_{ref} \pm \varphi$; the red line denotes p_{avg}.

Chapter 4

Statistical Music Recognition

In fact, deterministic recognition is not suffice to work with singing voice. PDAs suffer from well-known problems of accuracy, particularly on the boundaries of music notes, and very large errors are common. Statistical methods have turned out to be more successful for transcribing music signals [43]. Since the late 1990s, many people have been working on the statistical modeling for various applications, e.g. automatic segmentation, music tracking.

Grubb and Dannenberg proposed a stochastic model for location and duration of sung notes, in order to trace the tempo changes in singing [16]. Raphael utilized a hidden Markov model (HMM) to identify the time boundaries of sung notes, in an attempt to segment the data into music notes [35].

Moreover, Orio and Déchelle designed a two-level HMM to align the singing with a music score-sheet [27]. In their scheme, the higher level classified the notes correctly performed and the errors that the singer made, while the lower level described the amplitude changes of notes. Viitaniemi, Klapuri and Eronen [43] proposed a probabilistic model on a purpose similar to that of this work. Their system consisted of three parts operating in parallel: a HMM pitch model, a musicological model, and a duration model.

Generally, statistical music recognition consists of two modules: feature extraction and statistical model. The associated flow chart is shown in Figure 4.1. Section 4.1 explains the preliminary concepts of HMM. Section 4.2 is concerned with the training process. Section 4.3 is concerned with recognizing process. Section 4.4 addresses the issue of feature extraction.

4.1 Introduction to Hidden Markov Models

In this section, one well-known and widely used statistical method, namely the HMM approach, are discussed. The goal is to build a signal model to

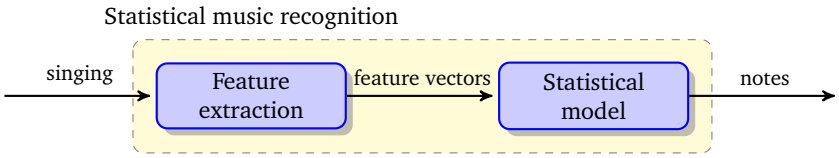

Figure 4.1: Flow chart of statistical music recognition

characterize a set of symbols produced in a real world process. If such model is acquired, it can be used to recognize other homogeneous symbols. The basic theory of HMM was published in a series of papers by Baum and his colleagues in the late 1960s [7, 6]. It was first applied in speech processing in the early 1970s. Continuous refinements in the Markov modeling techniques have greatly enhance this method. HMM is most familiar in the area of speech recognition, and is also adaptable to a wide variety of other applications.

4.1.1 Definition of HMM

To set idea, we assume that weather operates as a discrete Markov chain. The atmospheric pressure has two 'states': 'low' and 'high'. But they cannot be observed directly, that is, they are *hidden* from us. On each day, one of the following weather is observed: 'rainy' and 'sunny', which are called 'observations'. The atmospheric pressure affects the weather: low pressure is prone to cause rainy weather, and high pressure is prone to cause sunny weather. The entire system is a hidden Markov model. Figure 4.2 describes the relationship of atmospheric pressure and weather.

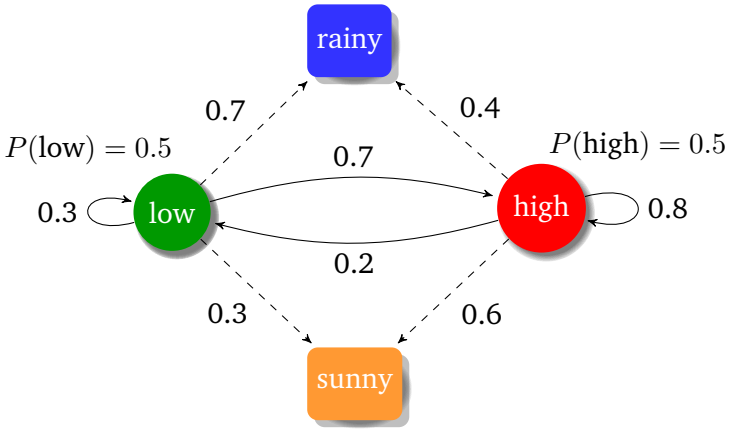

Figure 4.2: Relationship of atmospheric pressure and weather

The above example introduces the concept of HMM model. The elements of

HMM are listed out in below [31, p. 329]:

1. N, the number of model states. In Figure 4.2, there are two states, 'low' and 'high'. For the purpose of simplicity, states are labeled as $\{1, 2, \ldots, N\}$ in this thesis. The state at time t is written as q_t.

2. $\boldsymbol{\pi} = \{\pi_i\}$, the initial state probability

$$\pi_i = P(q_1 = i), \qquad 1 \leq i \leq N \tag{4.1}$$

   Figure 4.2 indicates that the first atmospheric pressure could equally be 'high' or 'low' on average.

3. $A = \{a_{ij}\}$, the state transition probability

$$a_{ij} = P(q_{t+1} = j | q_t = i), \qquad 1 \leq i, j \leq N \tag{4.2}$$

   with the following properties,

$$a_{ij} \geq 0, \qquad \forall i, j \tag{4.3}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall i \tag{4.4}$$

   It is necessary to mention that, this thesis only considers the first-order discrete-time Markov process. Hence, the state transition probability is dependent on the current state and the preceding state

$$P\left(q_t = j | q_{t-1} = i, q_{t-2} = k, \ldots\right) = P\left(q_t = j | q_{t-1} = i\right) \tag{4.5}$$

   In Figure 4.2, state transition represents the change of the atmospheric pressure in the underlying Markov chain. For instance, there is only a 20% chance that tomorrow will have a low atmospheric pressure if today has a high pressure.

4. K, the number of observation events per state. An observation event corresponds to a single output of HMM model. It is labeled as one of the observation symbols $\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_K\}$. In Figure 4.2, the observation events are simply 'rainy' and 'sunny'. Thus $K = 2$ and $\boldsymbol{V} = \{\text{rainy, sunny}\}$.

5. $B = \{b_i(\boldsymbol{o}_t)\}$, the observation probability

$$b_i(\boldsymbol{o}_t) = P(\boldsymbol{o}_t = \boldsymbol{v}_k | q_t = i), \qquad 1 \leq k \leq K,\ 1 \leq i \leq N \tag{4.6}$$

   which defines the probability distribution of observations in state i. In Figure 4.2, $b_i(\boldsymbol{o}_t)$ represents how likely the atmospheric pressure results in a certain weather. If the pressure is low, there is a 70% chance that the weather is rainy; if the pressure is high, there is a 40% chance that the weather is rainy.

Hidden states	Log probability
{'high', 'high'}	-2.34
{'high', 'low'}	-4.43
{'low', 'high'}	**-1.92**
{'low', 'low'}	-3.46

Table 4.1: Hidden states of observations {'rainy', 'sunny'} and their log probabilities

A full specification of HMM requires the model parameters N and K, observation symbols $\boldsymbol{V}$, and the probability distributions A, B and π. For convenience, this report uses a compact notation

$$\lambda = (A, B, \pi) \tag{4.7}$$

to indicate the complete parameter set of HMM.

Suppose the observations {'rainy', 'sunny'} is seen in the weather example, one possible case of model states is {'high', 'low'}. The probability of this case is

$$\begin{aligned} & P(\{\text{'rainy', 'sunny'}\}|\{\text{'high', 'low'}\}) \\ = \; & P(\text{'hight'}) \cdot P(\text{'rainy'}|\text{'high'}) \cdot P(\text{'low'}|\text{'high'}) \cdot P(\text{'sunny'}|\text{'low'}) \\ = \; & 0.5 \times 0.4 \times 0.2 \times 0.3 \\ = \; & 0.012 \end{aligned}$$

Table 4.1 lists out all the available states for the observations {'high', 'low'}, and their log probabilities. It can be deduced that {'low', 'hight'} is the most likely states generating the observations, since it has the maximal probability.

4.1.2 Types of HMM

One way to classify the types of HMM is using the structure of the transition matrix $\{a_{ij}\}$. Generally, the model is full interconnected, in which each state can be reached from any other state. Such a model is called an ergodic model.

For many applications, other types of HMM have been found better to account for the properties of the signal being modeled. Figure 4.3 shows one type, namely a left-to-right model or a Bakis model [5]. The state index increases as the time index increases, that is — the state proceeds from left to right. The left-to-right model is desirable to model an audio signal, because audio signal changes over time in a successive manner.

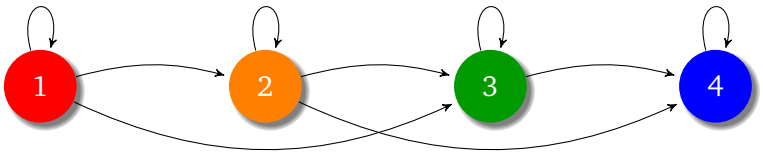

Figure 4.3: A four-state left-to-right model

First, the fundamental property of a left-right model is — no transition is allowed from state of large index to state of small index. In other words, transitions from the future states to the previous states make no sense. The state transition probabilities satisfy the constraint

$$a_{ij} = 0, \qquad j < i \tag{4.8}$$

Second, any sequence of observations starts in state 1. The initial state distribution has the property

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{4.9}$$

Third, the constraint

$$a_{ij} = 0, \qquad j > i + \Delta_i \tag{4.10}$$

ensures that large jumps across states will not occur. In Figure 4.3, Δ_i posses a value of 2, i.e. no jumps across 2 states are allowed. Its state transition matrix can be written as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{4.11}$$

4.1.3 Continuous Observation Probability Distribution

In the weather example, observations are discrete symbols. In this thesis, observations are in form of continuous feature vectors. The most general approach to model the probability distribution of a continuous signal is by using a weighted sum of Gaussian probability density functions (pdf), which is called Gaussian mixture model (GMM). The formula for computing $b_i(\boldsymbol{o}_t)$ is then [31, p. 350]

$$b_i(\boldsymbol{o}_t) = \sum_{k=1}^{M} w_{ik} \cdot \mathcal{N}(\boldsymbol{o}_t, \boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}), \qquad 1 \leq i \leq N \tag{4.12}$$

where M is the number of Gaussian components, $\boldsymbol{o}_t$ is the feature vector, and w_{ik} is the mixture weight of the kth component in state i, which satisfies the constraints

$$\begin{cases} \sum_{k=1}^{M} w_{ik} = 1 \\ w_{ik} \geq 0 \end{cases} \qquad 1 \leq i \leq N,\ 1 \leq k \leq M \tag{4.13}$$

The distribution function $\mathcal{N}(\cdot)$ is a Gaussian pdf with mean vector $\boldsymbol{\mu}_{ik}$ and covariance matrix $\boldsymbol{\Sigma}_{ik}$. Suppose the dimension of feature vector is D, the Gaussian pdf has the form of

$$\mathcal{N}(\boldsymbol{o}_t, \boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) = \frac{1}{(2\pi)^{D/2} \left|\boldsymbol{\Sigma}_{ik}\right|^{1/2}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{o}_t - \boldsymbol{\mu}_{ik}\right)' \boldsymbol{\Sigma}_{ik}^{-1} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{ik}\right)\right\} \tag{4.14}$$

where the prime denotes vector transpose.

The complete GMM model is parameterized by the mean vectors, covariance matrices, and mixture weights of all components. These parameters are collectively represented by the notation

$$\theta = (w, \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{4.15}$$

GMM is widely used due to its capacity of smoothly approximating an arbitrarily shaped pdf. Figure 4.4 compares the pdf obtained by using a normal distribution model and a GMM model.

Panel (a) shows the histgram of 2000 samples; panel (b) applies normal distribution to model this data set; panel (c) shows a GMM model and four Gausian components. Their means μ and standard deviations σ are listed in the legend. Clearly, GMM provides a smooth fit to the data.

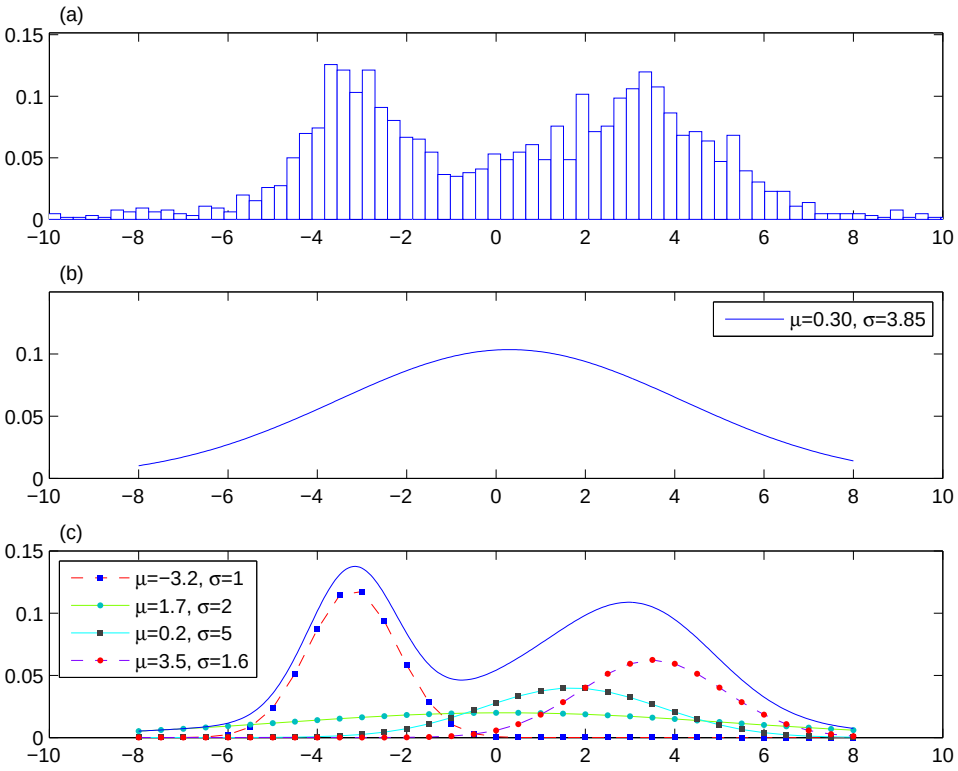

Figure 4.4: Comparison of distribution modeling: (a) histgram of 2000 samples, (b) normal distribution model, and (c) GMM and its components.

4.2 HMM Training

The training process in HMM aims to estimate the model parameters that can best describe how a given set of observations come about. A mathematical interpretation of HMM training is to find the optimal model λ which maximizes the likelihood $P(\boldsymbol{O}|\lambda)$.

The flow chart of the training procedure shown in Figure 4.5 consists of two modules, 'initialization' and 're-estimation', which are explored in Section 4.2.1 and 4.2.2 separately.

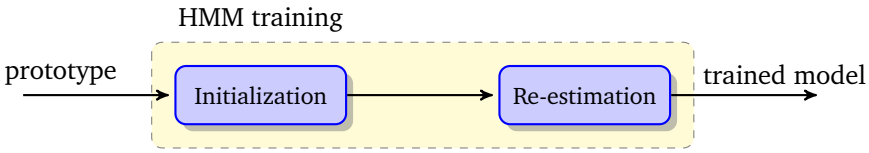

Figure 4.5: Flow chart of training procedure

4.2.1 Initialization of Parameters

A set of R training sequences $\boldsymbol{O}^r$ $(1 \leq r \leq R)$ and a prototype are used to initialize a new HMM. A training sequence is a stream of T_r observations, i.e. $\boldsymbol{O}^r = \{\boldsymbol{o}_1^r, \boldsymbol{o}_2^r, \ldots, \boldsymbol{o}_{T_r}^r\}$. In this thesis, $\boldsymbol{O}^r$ refers to feature vectors acquired from T_r frames. The prototype defines the form of a HMM model, i.e. $\hat{\lambda}^{(0)}$ and $\hat{\theta}^{(0)}$.

Figure 4.6 shows the flow chart of the entire initialization procedure, where the symbol $\hat{\theta}^{(0)}$ is equivalent to the parameter set $\{\hat{\boldsymbol{\mu}}_{ik}^{(0)}, \hat{\boldsymbol{\Sigma}}_{ik}^{(0)}, \hat{w}_{ik}^{(0)}\}$, the symbol $\hat{\lambda}^{(0)}$ is equivalent to the parameter set $\{\hat{a}_{ij}^{(0)}, \hat{b}_i^{(0)}(\boldsymbol{o}_t^r)\}$, and `hmm(0)` is the resulting model after initialization.

The initialization process starts with a uniform partition, that is — allocating the training data equally amongst the model states. For each sequence $\boldsymbol{O}^r$, the data are assigned into N disjoint sets $\{S_i^r | 1 \leq i \leq N\}$.

For a model state i, the k-means clustering algorithm is used to allocate the training data in S_i^r into M disjoint sub-sets $\{S_{ik}^r | 1 \leq k \leq M\}$. This thesis assumes that all the model states have the same amount of GMM components. Each sub-set corresponds to a GMM component. The unknown means, covariance matrices and mixture weights could be estimated by averaging across the associated data sets.

Consider a data set S_i for model state i, the k-means clustering aims to divide the data into M sub-sets $\{S_{i1}, S_{i2}, \ldots, S_{iM}\}$ with the constraint that $\sum_{k=1}^{M} \sum_{\boldsymbol{o}_t \in S_{ik}} \|\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{ik}\|^2$ is minimized, where $\hat{\boldsymbol{\mu}}_{ik}$ is the mean in S_{ik}.

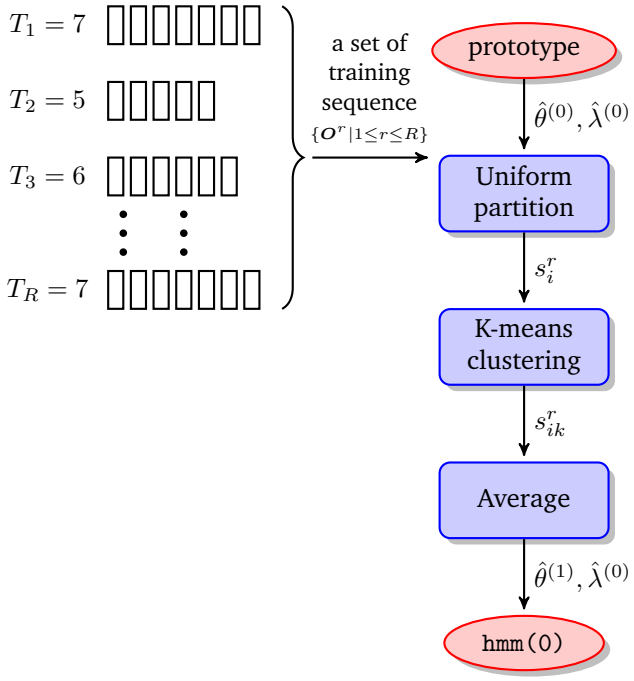

Figure 4.6: Flow chart of initialization

The most common algorithm is Lloyd's algorithm [23]. Given an initial group of M means $\{\hat{\boldsymbol{\mu}}_{i1}(1), \hat{\boldsymbol{\mu}}_{i2}(1), \ldots, \hat{\boldsymbol{\mu}}_{iM}(1)\}$, e.g. randomly choosing M observations from the data set [17], the algorithm proceeds iteratively between two steps [24]:

Allocation allocate each observations to the sub-set with the closest mean

$$S_{ik}^{(n)} = \{\|\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{ik}(n)\| \le \|\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{ik^*}(n)\|, \ \forall k^* = 1, 2, \ldots, M\} \quad (4.16)$$

Update compute the new means to be the centroid of the observations in the sub-set.

$$\hat{\boldsymbol{\mu}}_{ik}(n+1) = \frac{\sum_{\boldsymbol{o}_t \in S_{ik}^{(n)}} \boldsymbol{o}_t}{|S_{ik}^{(n)}|} \quad (4.17)$$

where $|S_{ik}^{(n)}|$ denotes the size of kth sub-set at nth iteration, and $\hat{\boldsymbol{\mu}}_{ik}(n)$ denotes the mean of set $S_{ik}^{(n)}$. The algorithm is deemed to be converged when the allocations no longer change.

Figure 4.7 exemplifies the process of initialization. Having segmenting the waveform into frames, the feature vectors are first calculated frame by frame, and are evenly assigned to the model state 1 and 2. In state 1, vectors are allocated amongst two GMM components via the k-means clustering.

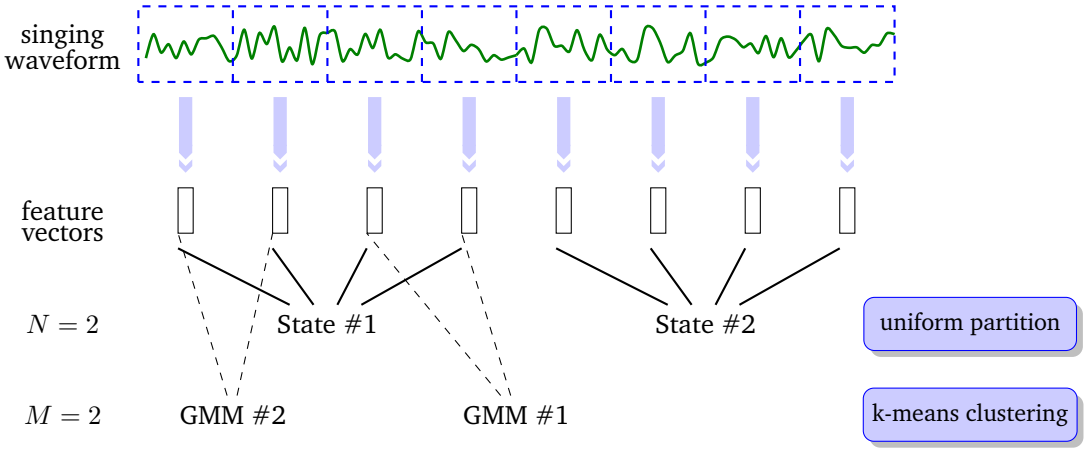

Figure 4.7: Assignment of training data

Next, the model parameters are calculated from the training data. Let $\phi_{ik}^r(t)$ be an indicator function, which is 1 if $\boldsymbol{o}_t^r$ is associated with the kth GMM component of model state i, and is 0 otherwise:

$$\phi_{ik}^r(t) = \begin{cases} 1 & \boldsymbol{o}_t^r \in S_{ik}^r \\ 0 & \text{Otherwise} \end{cases} \tag{4.18}$$

The parameter set $\hat{\theta}^{(1)}$ is calculated via simple 'average' [3, p. 143]

Means

$$\hat{\boldsymbol{\mu}}_{ik}^{(1)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \phi_{ik}^r(t) \cdot \boldsymbol{o}_t^r}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \phi_{ik}^r(t)} \tag{4.19}$$

Covariance Matrices

$$\hat{\boldsymbol{\Sigma}}_{ik}^{(1)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \phi_{ik}^r(t) \cdot (\boldsymbol{o}_t^r - \hat{\boldsymbol{\mu}}_{ik}^{(1)})\,(\boldsymbol{o}_t^r - \hat{\boldsymbol{\mu}}_{ik}^{(1)})'}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \phi_{ik}^r(t)} \tag{4.20}$$

Mixture Weights

$$\hat{w}_{ik}^{(1)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \phi_{ik}^r(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\sum_{m=1}^{M} \phi_{im}^r(t)} \tag{4.21}$$

4.2.2 Re-Estimation of Parameters

The re-estimation is similar to the initialization described in the previous section, except that the hard boundary implied by the function ϕ is replaced by a soft boundary function L which represents the probability of an observation being associated with a given model state.

In other words, instead of assigning each training vector to a specific state as in the initialization, each vector is assigned to every state in proportion to the probability of the model being in that state when the vector was observed [3, p. 7]. This probability is called occupation probability.

The occupation probability is computed from the forward and backward probabilities. The forward probability is defined as

$$\alpha_i(t) = P(\boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_t, q_t = i|\lambda) \tag{4.22}$$

and $\alpha_i(t)$ is calculated by the forward recursion

$$\alpha_j(t+1) = \left(\sum_{i=1}^{N} \alpha_i(t) \cdot a_{ij}\right) \cdot b_j(\boldsymbol{o}_{t+1}) \tag{4.23}$$

with initial condition $\alpha_i(1) = \pi_i \cdot b_i(\boldsymbol{o}_1)$.

Similarly, the backward probability is defined as

$$\beta_i(t) = P(\boldsymbol{o}_{t+1}\, \boldsymbol{o}_{t+2} \ldots \boldsymbol{o}_T|q_t = i, \lambda) \tag{4.24}$$

and $\beta_i(t)$ is computed by the backward recursion

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} \cdot b_j(\boldsymbol{o}_{t+1}) \cdot \beta_j(t+1) \tag{4.25}$$

with the initial condition $\beta_i(T) = a_{iN}$.

More details of the forward-backward algorithm are explained in Appendix B. The definitions of forward and backward probabilities infer that

$$\alpha_i(t) \cdot \beta_i(t) = P(\boldsymbol{O}_T, q_t = i|\lambda) \tag{4.26}$$

Thus the total probability of $\boldsymbol{O}_T$ is

$$\begin{aligned} P(\boldsymbol{O}_T|\lambda) &= \sum_{i=1}^{N} P(\boldsymbol{O}_T, q_t = i|\lambda) \\ &= \sum_{i=1}^{N} \alpha_i(t) \cdot \beta_i(t) \end{aligned} \tag{4.27}$$

And the probability of $\boldsymbol{o}_t$ being 'generated' by state i is

$$\begin{aligned} L_i(t) &= P(q_t = i|\boldsymbol{O}_t, \lambda) \\ &= \frac{P(\boldsymbol{O}_T, q_t = i|\lambda)}{P(\boldsymbol{O}_T|\lambda)} \\ &= \frac{\alpha_i(t) \cdot \beta_i(t)}{P(\boldsymbol{O}_T|\lambda)} \end{aligned} \tag{4.28}$$

Then the proportion of $\boldsymbol{o}_t$ being 'generated' by the kth mixture of state i is

$$\begin{aligned} L_{ik}(t) &= L_i(t) \cdot \frac{\hat{w}_{ik} \cdot \mathcal{N}(\boldsymbol{o}_t, \hat{\boldsymbol{\mu}}_{ik}, \hat{\boldsymbol{\Sigma}}_{ik})}{\sum_{m=1}^{M} \hat{w}_{im} \cdot \mathcal{N}(\boldsymbol{o}_t, \hat{\boldsymbol{\mu}}_{ik}, \hat{\boldsymbol{\Sigma}}_{im})} \\ &= \frac{\alpha_i(t) \cdot \beta_i(t)}{P(\boldsymbol{O}_T|\lambda)} \cdot \frac{\hat{w}_{ik} \cdot \mathcal{N}(\boldsymbol{o}_t, \hat{\boldsymbol{\mu}}_{ik}, \hat{\boldsymbol{\Sigma}}_{ik})}{\sum_{m=1}^{M} \hat{w}_{im} \cdot \mathcal{N}(\boldsymbol{o}_t, \hat{\boldsymbol{\mu}}_{ik}, \hat{\boldsymbol{\Sigma}}_{im})} \end{aligned} \tag{4.29}$$

The steps of the re-estimation are stated as follows [3, pp. 8-9]

- For all the training sequences $\boldsymbol{O}^r$ $(1 \leq r \leq R)$, and time t $(1 \leq t \leq T_r)$
  - For all the model states i $(1 \leq i \leq N)$
    - Update $\hat{b}_i^{(n)}(\boldsymbol{o}_t^r)$ from the current observation $\boldsymbol{o}_t^r$ and $\theta^{(n)}$ by using Equation 4.12.
    - Calculate the forward probability $\alpha_i^r(t)$ and backward probability $\beta_i^r(t)$ by using Equation 4.23 and 4.25 respectively.
  - Calculate $P(\boldsymbol{O}_{T_r}|\lambda)$ by using Equation 4.27.
  - For all the model states i, and GMM components k $(1 \leq k \leq M)$
    - Calculate $L_{ik}^r(t)$ by using Equation 4.29.
- Terminate the re-estimation procedure, if any of the following conditions is satisfied
  - Calculate the average log probability per frame of the training data

$$P_{avg}^{(n)}(\boldsymbol{O}|\lambda) = \frac{\sum_{r=1}^{R} \log\{P(\boldsymbol{O}_{T_r}|\lambda)\}}{\sum_{r=1}^{R} T_r} \tag{4.30}$$

    where $P(\boldsymbol{O}_{T_r}|\lambda) = \sum_{i=1}^{N} \alpha_i(T_r) \cdot \beta_i(T_r)$.

    The re-estimation procedure is converged, if the relative change between successive values of $P_{avg}^{(n)}(\boldsymbol{O}|\lambda)$ and $P_{avg}^{(n-1)}(\boldsymbol{O}|\lambda)$ is smaller than the convergence factor (default value 0.0001),

- The maximum number of estimation cycles (default value 300) is reached.

- Otherwise, for all the model states i, and GMM components k
  - Update $\hat{\theta}^{(n+1)}$

$$\hat{\boldsymbol{\mu}}_{ik}^{(n+1)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} L_{ik}^{r}(t)\cdot \boldsymbol{o}_t^r}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} L_{ik}^{r}(t)} \tag{4.31}$$

$$\hat{\boldsymbol{\Sigma}}_{ik}^{(n+1)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} L_{ik}^{r}(t)\cdot (\boldsymbol{o}_t^r - \hat{\boldsymbol{\mu}}_{ik}^{(n+1)})\,(\boldsymbol{o}_t^r - \hat{\boldsymbol{\mu}}_{ik}^{(n+1)})'}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} L_{ik}^{r}(t)} \tag{4.32}$$

$$\hat{w}_{ik}^{(n+1)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} L_{ik}^{r}(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\sum_{m=1}^{M} L_{im}^{r}(t)} \tag{4.33}$$

  - Update the state transition probability $\hat{a}_{ij}^{(n)}$

$$\hat{a}_{ij}^{(n)} = \frac{\sum_{r=1}^{R} \frac{1}{P(\boldsymbol{O}_{T_r}|\lambda)} \sum_{t=1}^{T_r-1} \alpha_i^r(t)\cdot \hat{a}_{ij}^{(n-1)}\cdot \hat{b}_j(\boldsymbol{o}_{t+1}^r)\cdot \beta_j^r(t+1)}{\sum_{r=1}^{R} \frac{1}{P(\boldsymbol{O}_{T_r}|\lambda)} \sum_{t=1}^{T_r} \alpha_i^r(t)\cdot \beta_i^r(t)} \tag{4.34}$$

The above procedure is called Baum-Welch algorithm. Figure 4.8 shows this algorithm, where `hmm(n)` is the temporary model after n cycles of re-estimation, the symbol $\hat{\theta}^{(n+1)}$ is equivalent to the parameter set $\{\hat{\boldsymbol{\mu}}_{ik}^{(n+1)}$, $\hat{\boldsymbol{\Sigma}}_{ik}^{(n+1)}$, $\hat{w}_{ik}^{(n+1)}\}$, and the symbol $\hat{\lambda}^{(n)}$ is equivalent to the parameter set $\{\hat{a}_{ij}^{(n)}$, $\hat{b}_i^{(n)}(\boldsymbol{o}_t^r)\}$.

4.3 HMM Recognition

Given the observation sequence $\boldsymbol{O}_T = \{\boldsymbol{o}_1\ \boldsymbol{o}_2\ \ldots\ \boldsymbol{o}_T\}$, and a HMM model λ, the recognition procedure aims to find the *optimal* state sequence $\boldsymbol{q}_T = (q_1 q_2 \ldots q_T)$, which can best explain the observations. It attempts to uncover the 'hidden' part of the model, that is — to find the underlying sequence of states.

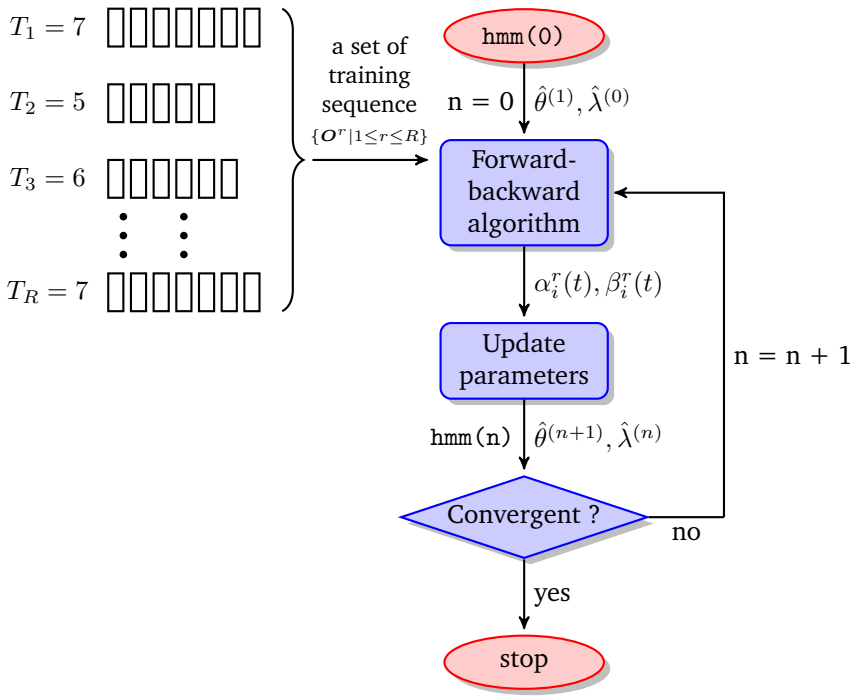

Figure 4.8: Flow chart of re-estimation procedure

Usually an optimality criteria is needed to solve this recognition problem. One typical criteria is to find the state sequence that maximizes $P(\boldsymbol{q}_T|\boldsymbol{O}_T, \lambda)$, which is equivalent to maximize $P(\boldsymbol{q}_T, \boldsymbol{O}_T|\lambda)$, i.e. the joint probability of $\boldsymbol{q}_T$ and $\boldsymbol{O}_T$ given the model λ.

$$P(\boldsymbol{q}_T|\boldsymbol{O}_T, \lambda) = \frac{P(\boldsymbol{q}_T, \boldsymbol{O}_T|\lambda)}{P(\boldsymbol{O}_T|\lambda)} \tag{4.35}$$

Assuming that the observations are mutually independent, then the probability $P(\boldsymbol{O}_T|\boldsymbol{q}_T, \lambda)$ is

$$\begin{aligned} P(\boldsymbol{O}_T|\boldsymbol{q}_T, \lambda) &= \prod_{t=1}^{T} P(\boldsymbol{O}_t|q_t, \lambda) \\ &= b_{q_1}(\boldsymbol{o}_1) \cdot b_{q_2}(\boldsymbol{o}_2) \cdot \ldots \cdot b_{q_T}(\boldsymbol{o}_T) \end{aligned} \tag{4.36}$$

In addition, the probability of $\boldsymbol{q}_T$ given $lambda$ is

$$P(\boldsymbol{q}_T|\lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdot \ldots \cdot a_{q_{T-1} q_T} \tag{4.37}$$

The joint probability of $\boldsymbol{O}_T$ and $\boldsymbol{q}_T$ is simply the product of $P(\boldsymbol{O}_T|\boldsymbol{q}_T, \lambda)$

and $P(\boldsymbol{q}_T|\lambda)$

$$\begin{aligned} & P(\boldsymbol{q}_T, \boldsymbol{O}_T|\lambda) \\ = \; & P(\boldsymbol{O}_T|\boldsymbol{q}_T, \lambda) \cdot P(\boldsymbol{q}_T|\lambda) \\ = \; & \pi_{q_1} \cdot b_{q_1}(\boldsymbol{o}_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(\boldsymbol{o}_2) \cdot \ldots \cdot a_{q_{T-1} q_T} \cdot b_{q_T}(\boldsymbol{o}_T) \end{aligned} \tag{4.38}$$

The Equation 4.38 is interpreted as following. At time $t = 1$, state q_1 occurs with probability π_{q_1}, and $\boldsymbol{o}_1$ is observed with probability $b_{q_1}(\boldsymbol{o}_1)$. At time $t = 2$, a transition to state q_2 occurs with probability $a_{q_1 q_2}$ and $\boldsymbol{o}_2$ is observed with probability $b_{q_2}(\boldsymbol{o}_2)$. The process continues in this manner until it transits from state q_{T-1} to state q_T with probability $a_{q_{T-1} q_{T-2}}$, and ends in observation $\boldsymbol{o}_T$ with probability $b_{q_T}(\boldsymbol{o}_T)$.

Let $\delta_i(t)$ be the highest probability of $P(\boldsymbol{q}_t, \boldsymbol{O}_t|\lambda)$ with state i observed at time t (i.e. $q_t = i$)

$$\delta_i(t) = \max_{q_1 \; q_2 \cdots q_{t-1}} P(q_1 \; q_2 \; \ldots \; q_{t-1}, \; q_t = i, \; \boldsymbol{o}_1 \; \boldsymbol{o}_2 \; \ldots \; \boldsymbol{o}_t|\lambda) \tag{4.39}$$

Assume that state j is observed at time $t + 1$, the highest probability of $P(\boldsymbol{q}_{t+1}, \boldsymbol{O}_{t+1}|\lambda)$ is

$$\begin{aligned} & \delta_j(t+1) \\ = \; & \max_{q_1 \; q_2 \cdots q_t} P(q_1 \; q_2 \; \ldots \; q_t = i, \; q_{t+1} = j, \; \boldsymbol{o}_1 \; \boldsymbol{o}_2 \; \ldots \; \boldsymbol{o}_{t+1}|\lambda) \\ = \; & \max_i \{\delta_i(t) \, a_{ij}\} \cdot b_j(\boldsymbol{o}_{t+1}) \end{aligned} \tag{4.40}$$

The recognition procedure starts to compute $\delta(1)$ at time $t = 1$ for all the model states, and continues to compute $\delta(t + 1)$ till time instant T. After acquiring the state i^* that maximizes $\delta_i(T)$ $(1 \leq i \leq N)$, it tracks back to retrieve the optimal sequence of states.

Let $\psi_j(t)$ records the preceding state at time $t - 1$ that maximizes $\delta_j(t)$ at time t. The entire procedure to find the best state sequence is described in below [31, p. 340]:

Preprocessing

$$\tilde{\pi}_i = \log\{\pi_i\} \tag{4.41}$$

$$\tilde{b}_i(\boldsymbol{o}_t) = \log\{b_i(\boldsymbol{o}_t)\} \tag{4.42}$$

$$\tilde{a}_{ij} = \log\{a_{ij}\} \tag{4.43}$$

Initialization

$$\begin{aligned} \tilde{\delta}_i(1) &= \log\{\delta_i(1)\} \\ &= \log\{\pi_i \cdot b_i(\boldsymbol{o}_1)\} \\ &= \pi_i \cdot b_i(\boldsymbol{o}_1) \\ &= \tilde{\pi}_i + \tilde{b}_i(\boldsymbol{o}_1) \end{aligned} \tag{4.44}$$

$$\psi_i(1) = 0 \tag{4.45}$$

Recursion

$$\begin{aligned}
\tilde{\delta}_j(t+1) &= \log\{\delta_j(t+1)\} \\
&= \log\left\{\max_i\{\delta_i(t)\cdot a_{ij}\}\cdot b_j(\boldsymbol{o}_{t+1})\right\} \\
&= \max_i\left\{\tilde{\delta}_i(t)+\tilde{a}_{ij}\right\}+\tilde{b}_j(\boldsymbol{o}_{t+1}) \qquad (4.46)\\
\psi_j(t+1) &= \arg\max_i\left\{\tilde{\delta}_i(t)+\tilde{a}_{ij}\right\} \qquad (4.47)
\end{aligned}$$

Termination

$$\begin{aligned}
\tilde{P}^* &= \max_i \tilde{\delta}_i(T) \qquad (4.48)\\
q_T^* &= \arg\max_i \tilde{\delta}_i(T) \qquad (4.49)
\end{aligned}$$

Backtracking

$$q_t^* = \psi_{q_{t+1}^*}(t+1) \qquad 1 \le t \le T-1 \qquad (4.50)$$

This recognition procedure is called Viterbi algorithm [44]. The purpose of taking logarithms is to avoid the need of multiplications.

Figure 4.9 visualizes the Viterbi algorithm via a trellis diagram. The vertical dimension represents the model states. The horizontal dimension represents the time. The red line indicates the optimal states sequence. The log probability of any path is computed by summing the log transition probabilities and log observation probabilities along the path.

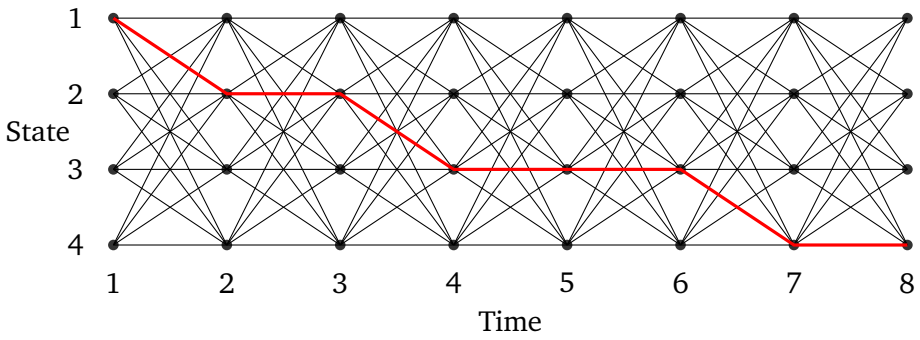

Figure 4.9: Trellis diagram of the Viterbi algorithm: the optimal path is marked by the red line.

4.4 The Extraction of Musical Features

In the statistical music recognition, the choice of good musical features is key to a good performance. Various features have been considered and tested.

Cano, Loscos and Bonada chose six features to build a note model, including energy measure, energy difference between neighboring frames, zero crossings, pitch, and pitch error [10]. In their left-to-right model, three states were selected to mimic the note behavior of attack, steady and release state. Also, they used an extra HMM to model the silence.

Shih, Narayanan and Kuo proposed a fourteen-element feature vector, which contained twelve mel-frequency cepstral coefficients (MFCC), one energy measure, and one pitch error [39].

Ryynänen and Klapuri combined pitch error, voicing, accent and meter in their HMM model. Pitch and voicing are musical attributes in frequency domain; accent and meter are musical attributes in time domain [36].

This section discusses four potential features used to render observation vectors. These features include pitch, pitch error, accent, zero-crossing rate and root-mean-squared energy.

4.4.1 Pitch & Pitch Error

In the training stage, pitch error $p_{\triangle}$ introduced in Section 3.3.1 is chosen as a feature in the process of building an intra-note model. The pitch error captures the deviation of the sung pitch around the correct pitch p_{ref}. The calculation of $p_{\triangle}$ is defined in Equation 3.16.

The pitch estimate p_{est} could alternatively be used as a feature instead of the pitch error. However, this means that we need to train a note model for each pitch (e.g. C_4, $C\sharp_4$, D_4, etc.). The main reason for using the $p_{\triangle}$ is that only one set of HMM parameters need to be trained in order to represent all different pitches.

Let's take an example: 20 sung notes of C_4 and 10 sung notes of D_4 have been recorded. If p_{est} is used as a feature, the note models of C_4 and D_4 will be trained separately. Now, if $p_{\triangle}$ is used, only one intra-note model will be trained with 30 sung notes (20 from C_4 and 10 from D_4).

The construction of note models through the use of the trained intra-note model will be discussed in Section 5.4.1. In the recognition stage, p_{est} is used as a feature to detect the hidden sung notes for test data.

4.4.2 Accent

Accent describes the degree of changes in signal intensity [21]. The concept of intensity was introduced in Section 2.1.2. Accent can be used to indicate the onset of notes, because it has higher value at the onset stage. The algorithm used in this thesis was proposed by Klapuri in 2003 [21]. It consists of four steps:

1. At first, the discrete Fourier transform (DFT) is calculated for each frame $x(k)$, where k is the frame index. A bank of B triangular-shape bandpass filters with equal bandwidth is generated in the range from $50\,\text{Hz}$ to $20\,\text{kHz}$. The power at each sub-band is calculated as $X_b(k)$, where b is the sub-band index, namely $1 \leq b \leq B$.

$$X_b(k) = \text{DFB}\left\{\mathcal{F}\{x(k)\}\right\} \tag{4.51}$$

   where DFB$\{\cdot\}$ means digital filter bank, $\mathcal{F}\{\cdot\}$ means DFT, and $B = 32$ in this thesis.

2. Given the power intensity envelopes $X_b(k)$ of spectrum, the μ-law compression is applied to measure the degree of changes in intensity

$$y_b(k) = \frac{\ln\{1 + \gamma \cdot X_b(k)\}}{\ln\{1 + \gamma\}} \tag{4.52}$$

   where a close-to-log value $\gamma = 100$ is employed, but any value in range $[10, 10^6]$ is valid.

3. Differentiation of $y_b(k)$ is performed as follows. First, a half-wave rectification (HWR) of $y_b(k)$ is calculated

$$y_b'(n) = \text{HWR}\left\{y_b(n) - y_b(n-1)\right\} \tag{4.53}$$

   where

$$\text{HWR}\{x\} = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.54}$$

   Then a weighted average of $y_b(k)$ and $y_b'(k)$ is calculated

$$u_b(k) = (1 - w) \cdot y_b(n) + w \cdot y_b'(n) \tag{4.55}$$

   where $w = 0.9$ is preferred.

4. Finally, all sub-bands are summed up to get accent signal $a(k)$

$$a(k) = \sum_{b=1}^{B} u_b(k)\,. \tag{4.56}$$

4.4.3 Zero-Crossing Rate

The idea of zero-crossing rate (ZCR) is to count how many times the waveform crosses the zero-axis in a certain time [25, p. 136]. Given a signal $x(n)$ with frame length L, the ZCR of kth frame is

$$z_k = \frac{1}{L}\sum_{n=1}^{L} |\text{sign}\{x_k(n)\} - \text{sign}\{x_k(n-1)\}| \tag{4.57}$$

where function "sign$\{\cdot\}$" returns 1 when $x_k(n)$ is positive, and returns 0 otherwise. The coefficient $1/L$ ensures the result to be a ratio.

The ZCR measures the periodicity of a signal. As mentioned in Section 2.1.1, definite pitch has stronger harmonic spectra and higher periodicity than indefinite pitch. Indefinite pitch tends to result in a close-to-one ZCR value; definite pitch tends to result in a close-to-zero value [25, p. 137]. At the onset or offset stage of a note, the ZCR easily have an abrupt increase, due to the rapid eruption or decline of voice.

4.4.4 Root-Mean-Squared Energy

The root-mean-squared (RMS) energy e_k describes the intensity of sound (see Section 2.1.2). Consider the kth frame of length L, the RMS energy is defined as

$$e_k = \sqrt{\frac{1}{L}\sum_{i=1}^{L} |x_k(i)|^2}\,. \tag{4.58}$$

Figure 4.10 illustrates these features. The top panel shows the MIDI reference. In the second panel, the pitch error is chiefly smaller than two semitones. The middle panel shows that the accent often has higher value in the beginning of notes. The fourth panel shows the ZCR. Peaks located on the boundaries of notes can be used for coarse segmentation. Bottom panel shows the envelop of RMS energy.

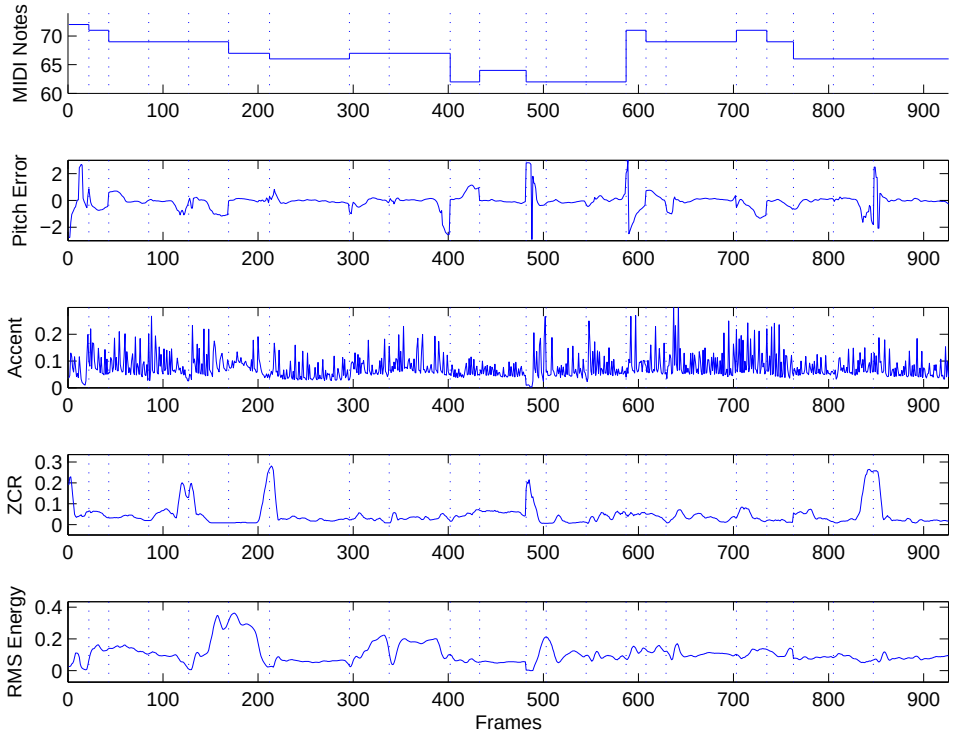

Figure 4.10: The extraction of features

Chapter 5

Experiment

This chapter describes the construction of a HMM-based music recognizer for Karaoke computer games. The recognizer is designed to detect the underlying notes of singing. The entire process is described in a step-by-step manner in order to give the readers a clear view of the principles discussed in Chapter 4. A toolbox named hidden Markov model toolkit (HTK)[2] is chosen to build and manipulate HMM models.

Section 5.1 is concerned with the data preparation. Section 5.2 is concerned with the training of an intra-note model. Section 5.3 is concerned with the probability distributions of features. Section 5.4 is concerned with the recognition of music notes.

5.1 Data Preparation

The first stage of any recognizer development is data preparation. An acoustic database of singing (see Section 5.1.1) and an associated transcription file (see Section 5.1.2) are needed for training and for testing. In addition, the singing waveform will be further transformed to feature vectors (see Section 5.1.3).

5.1.1 Step 1 – Recording the Data

Due to the fact that no public database of singing is available at present, the initial step of the data preparation is to create an acoustic database for enabling the HMM training and recognition.

The recording was done by the aid of a high-quality microphone and an easy-to-use audio editor 'Audacity' [1] in a quiet indoor environment. The

software automatically executes clipping during recording. Clipping removes the spiking of waveforms, which is usually a pulsar noise.

The analog audio were sampled at 44.1 kHz with a depth of 16 bits. All the songs in this database were performed by a male amateur singer, who had a solid foundation of music performance. During the recording, the singer wore a pair of earphones in which the original CD record was played. Hence only a single-voice was recorded.

The database contains ten pieces of songs in total. Appendix A.1 affords the details of the database. The training set incorporates 5 songs (i.e. 2315 music notes). The test set incorporates another 5 songs (i.e. 1657 music notes). In spite of the fact that this database posses a relative small size, it is still sufficient in the sense of statistics.

5.1.2 Step 2 – Labeling the Data

Every song needs an associated note level transcription as reference. The transcription file assigns labels (e.g. pitch, lyrics) to notes. The transcription files used in this thesis were downloaded from the user forum of a Karaoke video game 'UltraStar' [4]. The transcription files were inspected carefully, verified, then saved in standard text format.

Table 5.1 affords an example of the transcription text. Each line shows the associated information of one note. The first column is the note index. The

note	type	onset	duration	pitch	lyrics
1	:	0	3	69	Sum
2	:	3	2	71	mer
3	:	5	3	71	has
4	:	8	2	69	come
5	:	10	1	71	and
6	:	11	2	71	past,
7	-	13	2		
8	:	15	1	64	The
9	:	16	2	69	in
10	:	18	1	71	no
11	:	19	1	71	cent
12	-	20	1		

Table 5.1: Example of a transcription text

second column is the type of notes. The symbol ':' indicates note, and the symbol '-' indicates silence. The third column is the onset of a note in unit of beat number. The fourth column is the offset. The fifth column is the pitch in unit of MIDI number. The last column is the lyrics associated with the

note. It is useful to point out that the columns 'onset', 'duration' and 'pitch' respectively correspond to the first, second and fourth column of the MIDI note matrix shown in Table 2.2.

Next, a MIDI file and a lyrics file are created from the transcription file. The following text shows an example of the lyrics file.

```
0.0000 0.8571 Sum
0.8571 1.4286 mer
1.4286 2.2857 has
2.2857 2.8571 come
2.8571 3.1429 and
3.1429 3.7143 past,
4.3333 4.5714 The
4.5714 5.1429 in
5.1429 5.4286 no
5.4286 5.7143 cent
```

Each line expresses a note. The first two columns are the onset and offset time of a note (in unit of second). The third column is the lyrics.

Then the two files are loaded together with the singing record by software Audacity. Having been repeatedly played, listened and manually aligned, the singing record is synchronized with MIDI and lyrics. In a word, the singing record is precise in terms of timing.

Figure 5.1 displays the interface of Audacity. The top channel loads a singing

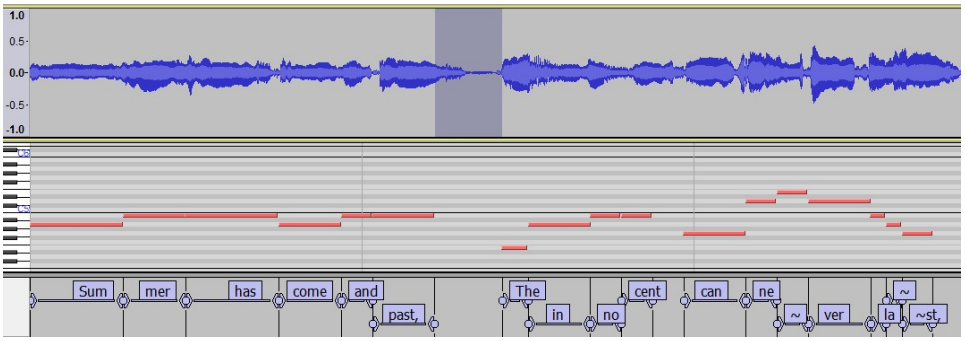

Figure 5.1: The interface of audio editor Audacity: top channel – singing waveform, middle channel – MIDI reference, bottom channel – lyrics.

record. The middle channel loads a MIDI trajectory. The bottom channel loads lyrics. In this thesis, the silence regions are ignored. Silence regions merely contain background noise. The removal of silent part of the audio signal leads to a substantial reduction in computational complexity at later stages [37].

5.1.3 Step 3 – Extracting the Features

This step parameterizes the raw singing waveform into sequences of feature vectors.

The singing is split into sung notes according to the transcription file shown in Table 5.1. The waveform of a sung note is first segmented into 50% overlapped frames with length of 50 ms. Then those frames are converted into feature vectors by using the signal processing operations discussed in Section 4.4. Feature vector is the unit of observations processed in the HMM training and recognition.

In the training stage, an intra-note model is built to describe the statistical behavior of model states inside a note. The feature vector $\{p_{\Delta}, a, z, e\}$ is took into account in this chapter. The trained intra-note model is further used to set up a group of note models, covering a wide range of pitch. In the recognition stage, those note models use the feature vector $\{p_{est}, a, z, e\}$ to transcribe the test data.

5.2 Creating an Intra-Note Model

In this section, the creation of an intra-note model is explored. The starting point is a prototype (see Section 5.2.1) in which every mean and variance is identical. The HMM model is then initialized (see Section 5.2.2) and is iteratively retrained (see Section 5.2.3).

5.2.1 Step 4 – Defining a Prototype

The first stage of training HMM model is to define a prototype. The purpose is to define the model topology. The total information needed to specify the prototype is summarized as follows

- dimension of feature vector
- number of model states
- for each 'active' state
  - number of mixture components
  - mixture weight
  - means
  - covariance matrices
- state transition probabilities

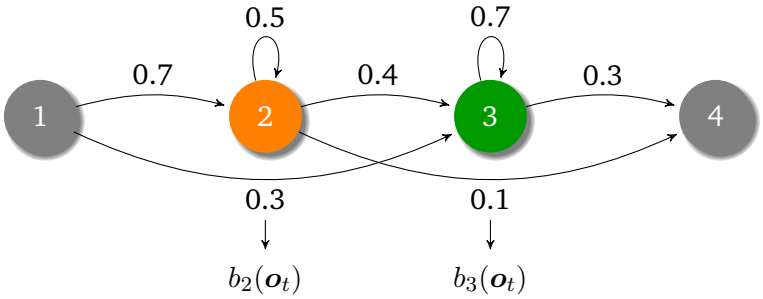

Figure 5.2: Prototype of intra-note model

Figure 5.2 shows a prototype of the intra-note model, which is a left-to-right model with four states in total. State 2 and 3 are 'active' states, and state 1 and 4 are 'null' states. In HTK toolkit, the first and last states are reserved as null states [3, p. 109]. Only active states have an associated observation probability distribution $b_i(\boldsymbol{o}_t)$.

The following text shows the description file of HMM prototype illustrated in Figure 5.2.

```
~o <VecSize> 4 <USER>
~h "intra-node"
<BeginHMM>
    <NumStates> 4
    <State> 2 <NumMixes> 4
        <Mixture> 1 0.45
            <Mean> 4
                0.0 0.0 0.0 0.0
            <Variance> 4
                1.0 1.0 1.0 1.0
        <Mixture> 2 0.30
                    ....
        <Mixture> 3 0.20
                    ....
        <Mixture> 4 0.05
                    ....
    <State> 3 <NumMixes> 4
        <Mixture> 1 0.45
            <Mean> 4
                0.0 0.0 0.0 0.0
            <Variance> 4
                1.0 1.0 1.0 1.0
        <Mixture> 2 0.30
                     ....
        <Mixture> 3 0.20
                    ....
        <Mixture> 4 0.05
```

```
                    ....
    <TransP> 4
        0.0 0.7 0.3 0.0
        0.0 0.5 0.4 0.1
        0.0 0.0 0.7 0.3
        0.0 0.0 0.0 0.0
<EndHMM>
```

The symbols '<Mixture> 1 0.45' indicates that first mixture of state 2 possess a mixture weight of 0.45 (i.e. $w_{12} = 0.45$). The keyword '<Mean>' indicates the mean vectors $\hat{\boldsymbol{\mu}}_{ik}^{(k)}$. In this thesis, the covariance matrices are assumed to be diagonal, that is — the features are uncorrelated with each other. The keyword '<Variance>' indicates the diagonal variance vectors of $\hat{\boldsymbol{\Sigma}}_{ik}$.

It is additionally assumed that all the model states share the same configurations, that is — every mean is equal to zero mean [0, 0, 0, 0], and every covariance matrix is equal to unit matrix diag{1, 1, 1, 1}. Finally, <TransP> indicates the state transition probability matrix A.

The symbols '<VecSize> 4 <USER>' specifies that the feature vector has a dimension of 4. The symbols ~h "intra-node" specifies the model's name. The symbols '<BeginHMM>(...)<EndHMM>' encloses the description of HMM. The symbols '<NumStates> 4' gives the total number of model states, including 2 null states and 2 active states. The symbols '<State> 2 <NumMixes> 4' indicates that state 2 has four mixtures.

5.2.2 Step 5 – Initializing the Model

The principles of model initialization are discussed in Section 4.2.1. The following text shows the description file of the initialized model, namely hmm(0).

```
~o <VecSize> 4 <USER>
~h "intra-node"
<BEGINHMM>
    <NUMSTATES> 4
    <STATE> 2 <NUMMIXES> 4
        <MIXTURE> 1  0.1332
            <MEAN> 4
                2.9406    0.1157    0.0604    0.1742
            <VARIANCE> 4
                2.7419    0.0100    0.0100    0.0100
        <MIXTURE> 2 0.2523
            <MEAN> 4
               -1.1217    0.1217    0.0462    0.2055
            <VARIANCE> 4
```

```
                0.0961    0.0100    0.0100    0.0107
        <MIXTURE> 3 0.2525
            <MEAN> 4
               -3.1137    0.1141    0.0528    0.1793
            <VARIANCE> 4
                1.4206    0.0100    0.0100    0.0104
        <MIXTURE> 4 0.3620
            <MEAN> 4
               -0.3062    0.1212    0.0420    0.2170
            <VARIANCE> 4
                0.0463    0.0100    0.0100    0.0100
    <STATE> 3   <NUMMIXES> 4
        <MIXTURE> 1 0.1890
            <MEAN> 4
                1.0481    0.1200    0.0434    0.2123
            <VARIANCE> 4
                0.3978    0.0100    0.0100    0.0104
        <MIXTURE> 2 0.3292
            <MEAN> 4
                0.1679    0.1150    0.0325    0.2264
            <VARIANCE> 4
                0.0100    0.0100    0.0100    0.0118
        <MIXTURE> 3 0.3857
            <MEAN> 4
               -0.1167    0.1147    0.0382    0.1974
            <VARIANCE> 4
                0.0100    0.0100    0.0100    0.0100
        <MIXTURE> 4 0.0961
            <MEAN> 4
               -1.0703    0.0930    0.0455    0.1292
            <VARIANCE> 4
                0.6448    0.0100    0.0100    0.0100
    <TRANSP> 4
        0    0.7228    0.2772         0
        0    0.9660    0.0157         0.0183
        0    0         0.9719         0.0281
        0    0         0              0
<ENDHMM>
```

Each state has different mixture parameters. The mixture weights of state 1 becomes `[0.1332, 0.2523, 0.2525, 0.3620]` from `[0.45, 0.3, 0.2, 0.05]` of the prototype. The mean $\hat{\mu}_{11}$ becomes `[2.9406, 0.1157, 0.0604, 0.1742]`. The covariance matrix $\hat{\Sigma}_{11}$ becomes `diag{2.7419, 0.0100, 0.0100, 0.0100}`.

5.2.3 Step 6 – Re-Estimating the Model

The initialized intra-note model `hmm(0)` is re-trained by using the Baum-Welch algorithm discussed in Section 4.2.2. The following text traces the

progress of re-estimation.

```
Re-estimating HMM intra-node . . .
 States   :   2  3 (width)
 Mixes  s1:   4  4 (  4  )
 MaxIter  :  100
 Epsilon  :  0.000100
 Updating :  Transitions Means Variances

 - system is PLAIN
1010 Examples loaded, Max length = 296, Min length = 13
Ave LogProb at iter 1 =  -893.3666 change =   29.41020
Ave LogProb at iter 2 =  -886.9374 change =    6.42924
Ave LogProb at iter 3 =  -883.1830 change =    3.75444
Ave LogProb at iter 4 =  -880.9557 change =    2.22723
Ave LogProb at iter 5 =  -879.4280 change =    1.52776
        ...
        ...
        ...
Ave LogProb at iter 44 =  -859.2107 change =    0.00025
Ave LogProb at iter 45 =  -859.2106 change =    0.00011
Ave LogProb at iter 46 =  -859.2106 change =    0.00007
Estimation converged at iteration 46
```

The line of 'States' indicates that the model has two active states. i.e. state 2 and 3. The line of 'Mixies' indicates that each state has four mixtures, and that the feature vector has a depth of four. The line of 'MaxIter' sets the maximum number of re-estimation cycles to 100.

The line of 'Epsilon' sets the convergence factor to 0.0001. As defined in Equation 4.30, the convergence factor is the relative change between successive values of $P_{avg}(\boldsymbol{O}|\lambda)$, where $P_{avg}(\boldsymbol{O}|\lambda)$ refers to the average log probability per frame of the training data.

Next, 1010 examples are loaded, where an example means one sequence of vectors. The length of example spans from 13 to 296 frames. The probability $P_{avg}(\boldsymbol{O}|\lambda)$ and its relative change of each iteration are displayed. The convergence is reached after 46 iterations with the last change of $P_{avg}(\boldsymbol{O}|\lambda)$ equal to 0.00007.

When training the model from limited data, a variance will becomes quite small if the variance being underestimated through lack of data. A variance floor (default value 0.0) is often set to prevent any variance getting too small.

The following text shows the description file of the trained model, namely hmm(46). Compared with the model hmm(0), the mixture parameters and state transition probabilities have a dramatic change.

```
~o <VECSIZE> 4 <USER>
~h "intra-node"
<BEGINHMM>
    <NUMSTATES> 4
    <STATE> 2 <NUMMIXES> 4
        <MIXTURE> 1 0.2621
            <MEAN> 4
               -0.7627    0.0688    0.0188    0.2041
            <VARIANCE> 4
                7.0855    0.0008    0.0001    0.0126
        <MIXTURE> 2 0.2959
            <MEAN> 4
               -0.5094    0.2003    0.0344    0.2373
            <VARIANCE> 4
                4.9200    0.0076    0.0003    0.0081
        <MIXTURE> 3 0.2064
            <MEAN> 4
               -0.5391    0.1058    0.1374    0.0946
            <VARIANCE> 4
                4.6999    0.0030    0.0055    0.0017
        <MIXTURE> 4 0.2357
            <MEAN> 4
               -0.4126    0.0774    0.0495    0.1881
            <VARIANCE> 4
                2.5412    0.0002    0.0004    0.0047
    <STATE> 3 <NUMMIXES> 4
        <MIXTURE> 1 0.3374
            <MEAN> 4
               -0.0701    0.1895    0.0324    0.2465
            <VARIANCE> 4
                0.0880    0.0078    0.0003    0.0089
        <MIXTURE> 2 0.2056
            <MEAN> 4
               -0.1513    0.0777    0.0105    0.2489
            <VARIANCE> 4
                0.2248    0.0015    0.0000    0.0173
        <MIXTURE> 3 0.3803
            <MEAN> 4
               -0.0750    0.0697    0.0379    0.1797
            <VARIANCE> 4
                0.0604    0.0004    0.0004    0.0051
        <MIXTURE> 4 0.0768
            <MEAN> 4
               -0.8049    0.1150    0.0665    0.1314
            <VARIANCE> 4
                1.8084    0.0049    0.0032    0.0039
    <TRANSP> 4
         0    0.7933    0.2067         0
         0    0.9602    0.0234         0.0164
```

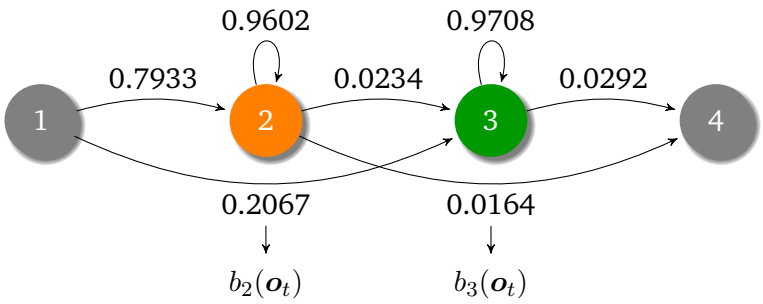

Figure 5.3: The trained intra-note model

```
          0     0          0.9708          0.0292
          0     0          0               0
<ENDHMM>
```

Figure 5.3 graphically presents the trained intra-note model. The probability distributions of $\{p_\Delta, a, z, e\}$ are shown in Section 5.3.

5.3 The Probability Distributions of Features

This section explores the probability distributions of $\{p_\Delta, a, z, e\}$. Since the features are mutually uncorrelated (i.e. diagonal covariance matrices), their probability distributions are presented separately.

The performance of the intra-note model is affected by the number of model states N and the number of mixtures M in each state. For an illustrative purpose, this section chooses N from 2 to 5, and chooses M from 2 to 8. Figure 5.4 – 5.10 explore the influence of variable N and M on the probability distributions $\hat{b}_i(\boldsymbol{o}_t)$.

Figure 5.4 – 5.7 show the $\hat{b}_i(\boldsymbol{o}_t)$ of $M = 4$ and $N = 2, 3, 4, 5$ separately. Figure 5.4 presents a two-state model of four GMM components. The first state models the onset phase. p_Δ and z are widely-spread due to the low periodicity of voice in the initial instant. In the second state, p_Δ shrinks into one pulse, and z becomes smaller. The variance of a and e only have a slight reduction.

Figure 5.5 shows a three-state model. The probability distributions of the first two states are close to the $\hat{b}_i(\boldsymbol{o}_t)$ in Figure 5.4. The third state models the offset phase, in which the variance of p_Δ has a tiny increase.

Figure 5.6 presents a four-state model. The variance of p_Δ is relatively large in state 1. The mean value of a gradually shifts to the close-to-zero side. z spreads the density in state 1 and 4, indicating the instability of pitch. The

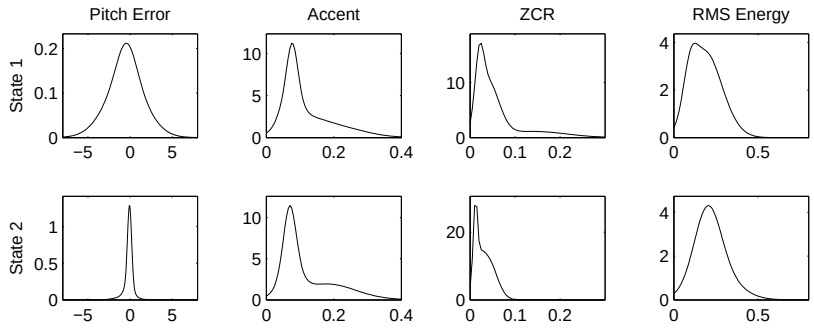

Figure 5.4: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 2$ and $M = 4$

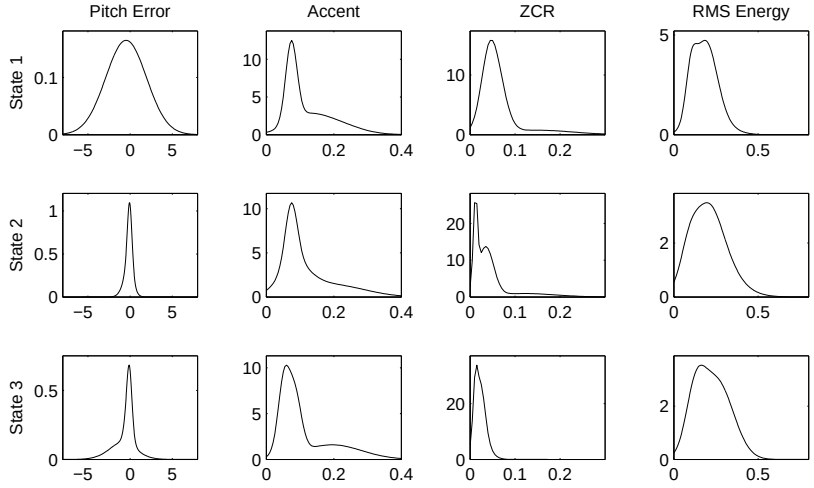

Figure 5.5: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 3$ and $M = 4$

mean value of e has an obvious increase in state 2, and slowly fades in state 3 and 4.

Figure 5.7 introduces a five-state model. State 4 and 5 have similar distributions. Albeit more states are able to improve the accuracy of estimation, excessive states will swiftly increase the computational complexity.

Figure 5.8 – 5.10 illustrate the $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 4$ and $M = 2, 6, 8$ respectively.

Figure 5.8 shows a four-state model with two GMM mixtures. Compared with the case of $N = 4$ and $M = 4$ in Figure 5.6, it is apparent that two mixtures are insufficient to express the probability distributions of features. For example, p_Δ in state 2, a in state 1 and 3, and e in state 3 are widely dispersed, due to the fact that two mxitures are not capable of providing an accurate description of $\hat{b}_i(\boldsymbol{o}_t)$.

Figure 5.9 shows the condition of six GMM mixtures, which provides finer curves than Figure 5.6. The a in state 4 and e in state 1 turn to be bimodal.

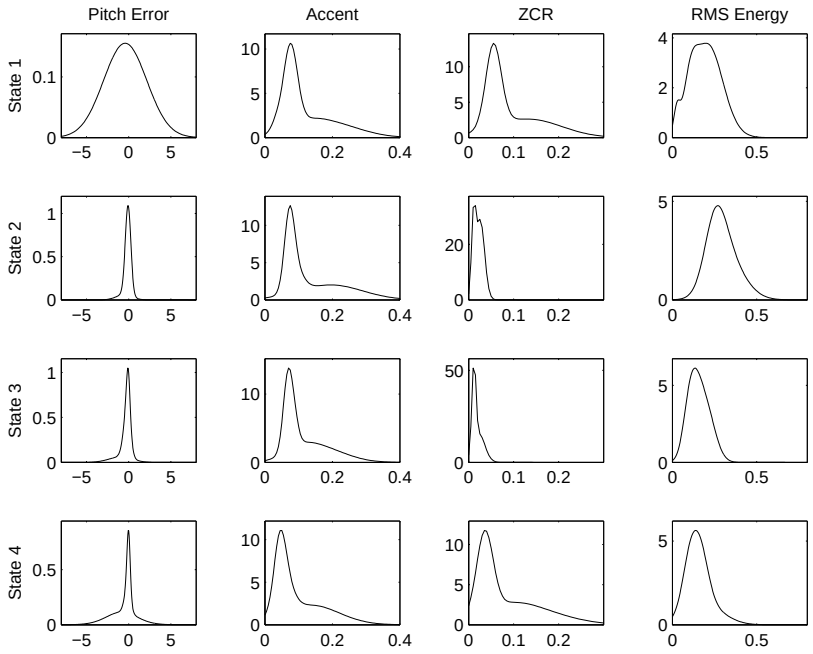

Figure 5.6: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 4$ and $M = 4$

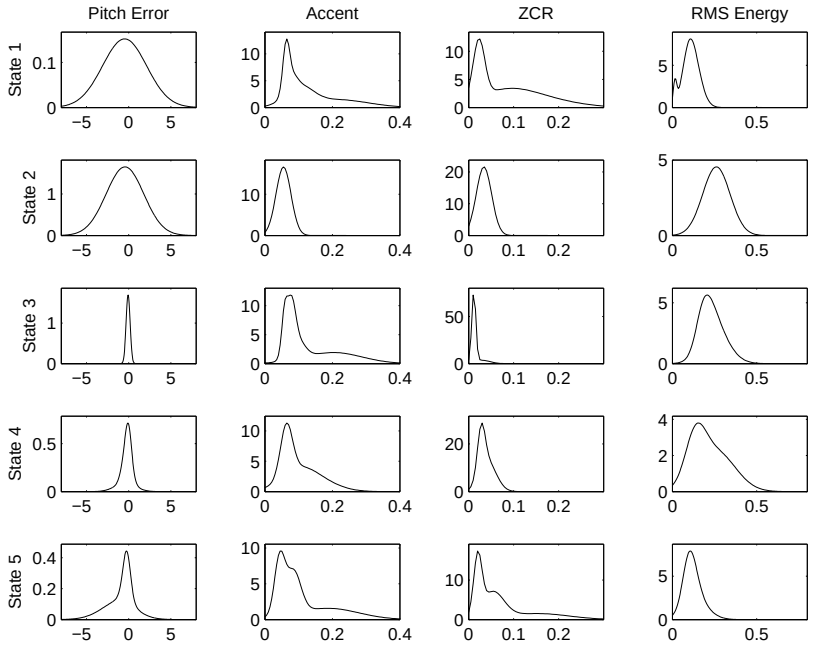

Figure 5.7: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 5$ and $M = 4$

But we can not simply assert that six mixtures excel four mixtures because

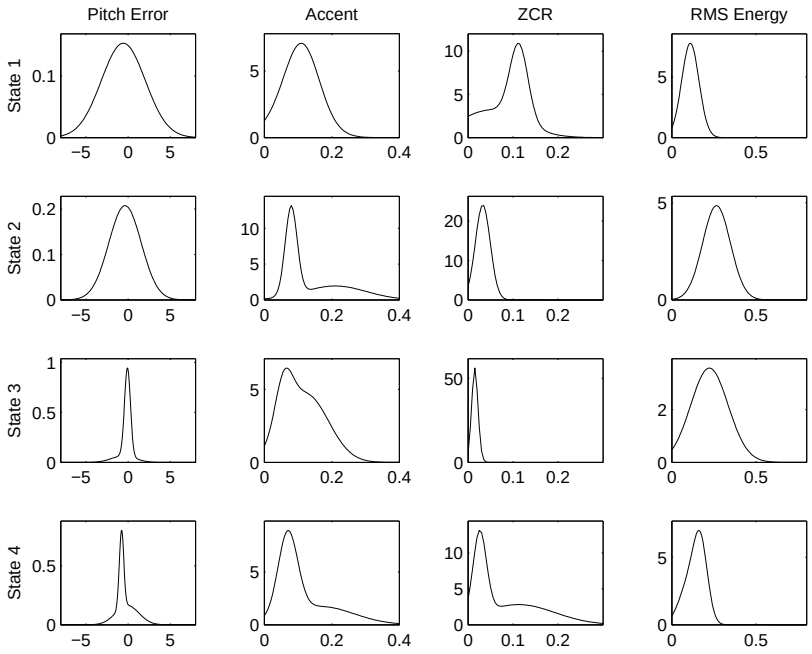

Figure 5.8: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 4$ and $M = 2$

more components may over-fit the $b_i(\boldsymbol{o}_t)$.

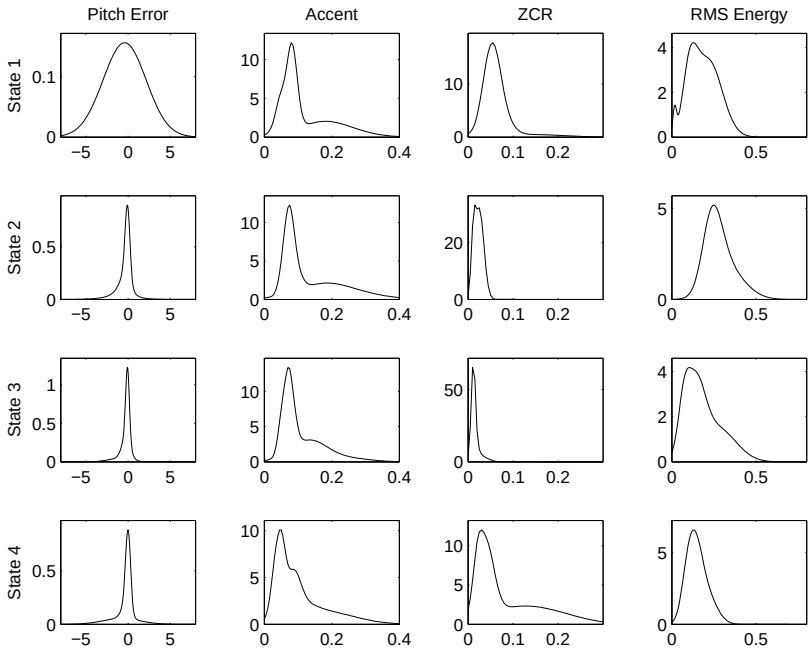

Figure 5.9: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 4$ and $M = 6$

Figure 5.10 shows the model with eight GMM mixtures. Figure 5.10 resembles Figure 5.9 too much. High similarity reflects redundance. It might be a waste of resource to select eight mixtures in each state.

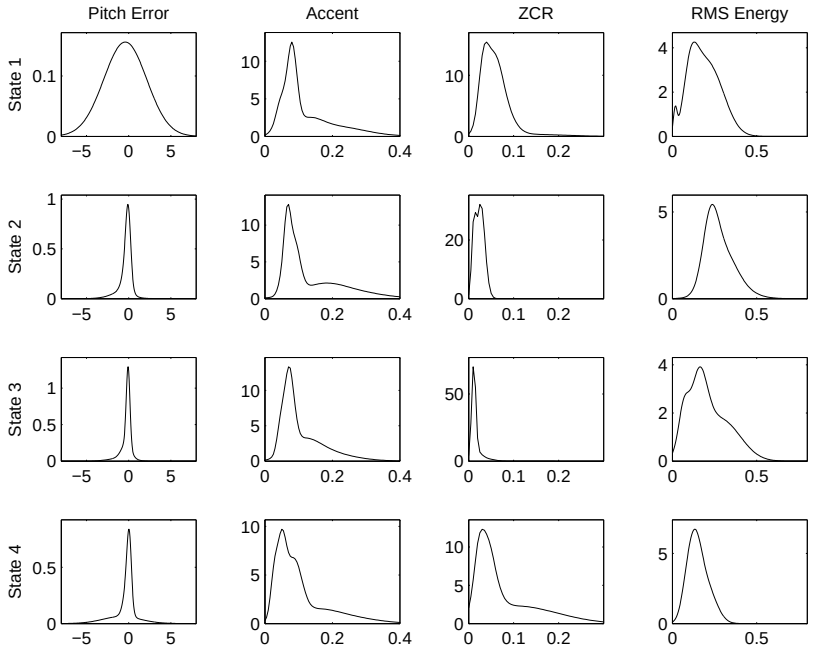

Figure 5.10: The probability distribution $\hat{b}_i(\boldsymbol{o}_t)$ of $N = 4$ and $M = 8$

5.4 Performing Test

Figure 5.11 illustrates the overall recognition procedure. Section 5.4.1 creates a group of HMM note models from the trained intra-note model. Section 5.4.2 constructs a note network. The group of HMM models and note network are input to generate a recognition network. Section 5.4.3 uses the recognition network and Viterbi algorithm to decode the input test data.

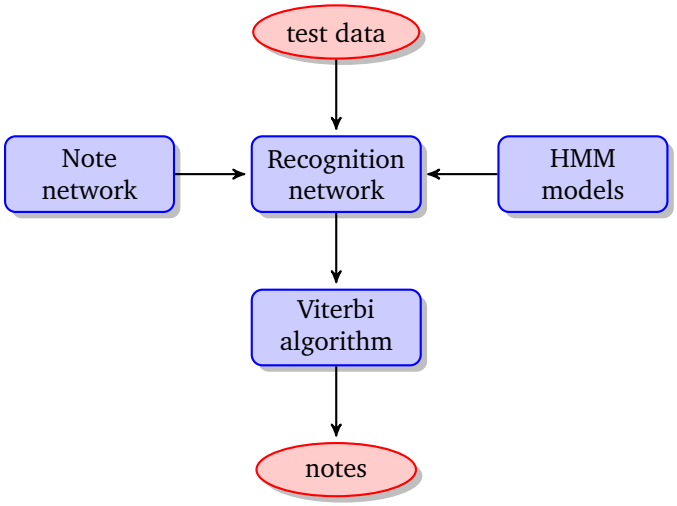

Figure 5.11: Flow chart of recognition procedure

5.4.1 Step 7 – Creating a Group of Note Models

The trained intra-note is used to construct music note models by adding the mean of p_Δ with a MIDI number. The following text shows the description file of note model C_3 (MIDI 48). The symbol 'N48' indicates the note name. The $\hat{\boldsymbol{\mu}}_{11}$ of the intra-note model is `[-0.7627, 0.0688, 0.0188, 0.2041]`. We add 48 to p_Δ, then get the $\hat{\boldsymbol{\mu}}_{11}$ of C_3 as `[47.2373, 0.0688, 0.0188, 0.2041]`.

```
~o <VECSIZE> 4 <USER>
~h "N48"
<BEGINHMM>
    <NUMSTATES> 4
    <STATE> 2 <NUMMIXES> 4
        <MIXTURE> 1 0.2621
            <MEAN> 4
                47.2373    0.0688    0.0188    0.2041
            <VARIANCE> 4
                7.0855      0.0008    0.0001    0.0126
        <MIXTURE> 2 0.2959
```

```
            <MEAN> 4
                47.4906    0.2003    0.0344    0.2373
            <VARIANCE> 4
                4.9200      0.0076    0.0003    0.0081
    ...
    ...
    ...
<ENDHMM>
```

In a similar manner, a group of note models are built up, with note pitch spanning from C_3 to C_6 (i.e. MIDI 48 – 84). This range covers three octaves, and is generally sufficient to encompass all the possible sung pitches of both male and female.

5.4.2 Step 8 – Constructing a Note Network

The note network is generated from a text containing high-level grammar notations [3, pp. 186-188]. This grammar text is read-in directly by the recognizer and compiled into a finite state note network at run-time. The following text illustrates the use of grammar in this experiment.

```
$NOTE = N48|N49|N50|N51|N52|N53|N54|N55
        |N56|N57|N58|N59|N60|N61|N62|N63
        |N64|N65|N66|N67|N68|N69|N70|N71
        |N72|N73|N74|N75|N76|N77|N78|N79
        |N80|N81|N82|N83|N84;
(
    <$NOTE>
)
```

The first part defines a variable '`Note`'. The symbol '`|`' denotes alternatives. The symbol '`N48`' denotes the note of MIDI number 48 (i.e. C_3). Hence, the '`Note`' can represent any note model from C_3 (MIDI 48) to C_6 (MIDI 84). The angle bracket '<>' denotes repetitions, that is — the grammar text expresses note loops. This translates into the note network shown in Figure 5.12.

5.4.3 Step 9 – Running the Recognition

As illustrated by Figure 5.11, decoding of test data is controlled by a recognition network compiled from a note-level network, and a group of note models. The recognition network consists of a set of nodes connected by arcs. Each node is a note model. Each note model is itself a sub-network consisting of HMM states. Thus, once fully compiled, a recognition network ultimately consists of HMM states connected by transitions. Briefly, it can be

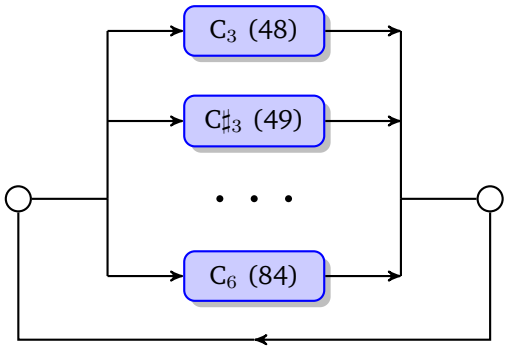

Figure 5.12: Music note network

viewed at two different levels: note and HMM state. Figure 5.13 illustrates this hierarchy.

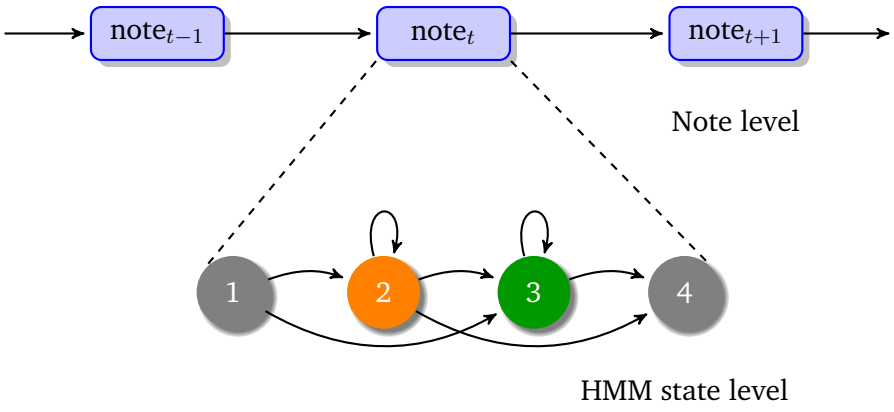

Figure 5.13: Recognition network

The decoding process is performed by using the Viterbi algorithm discussed in Section 4.3. For an unknown input with T frames, every path from the start node to the exit node of the network, which passes through exactly T active HMM states is a potential recognition hypothesis. Each of these paths has a log probability which is computed by summing the log probability of each transition in the path and the log probability of each active state generating the associated observation.

Within-note transitions are determined from the HMM parameters. Between-note transitions are assumed to be constant in this thesis. The task of recognizer is to find the most likely path which has the largest summed log probability. For readers interested with the details of implementation, please read the materials in [3, pp. 199-202].

The following text shows an example of the output printed by recognizer.

```
16600000 18300000 N73 C#5
18300000 19600000 N71 B4
19600000 21700000 N70 A#4
21700000 22200000 N69 A4
22200000 23500000 N70 A4#
```

The first two columns show the start and end time of each note (in unit of $0.1\,\mu$s or 10^{-7}s). For example, the first line indicates that the note $C\sharp_5$ start at 1.66 s, and ends at 1.83 s.

Figure 5.14 and 5.14, which correspond to the Figure 3.7 and Figure 3.8 respectively, compare the output of HMM-based recognizer and the output of deterministic recognizer described in Figure 3.1 of Chapter 3.

The green curve is the pitch trajectory p_{est}. The blue line is the 'averaged' note trajectory p_{avg}. The red line is the HMM-based note trajectory. The dashed vertical line denotes the onset time of the reference notes.

The HMM-based note trajectory better follows the movement of the pitch trajectory. In contrast to the note-based approach discussed in Section 3.3.2, the HMM-based recognizer determines the note boundaries by itself rather than by the reference notes.

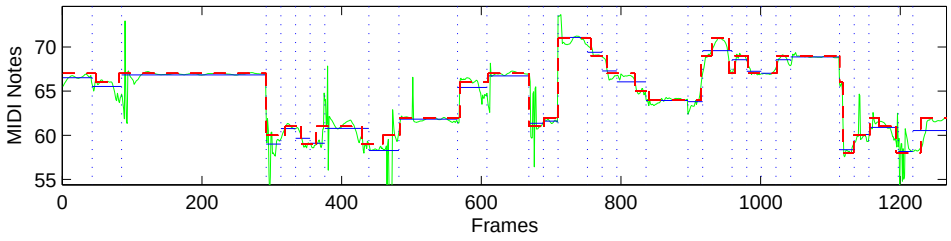

Figure 5.14: Comparison of recognized note trajectories: case I. The green line denotes the pitch p_{est}; the blue line denotes p_{avg}; the red line denotes the output of the HMM-based recognizer.

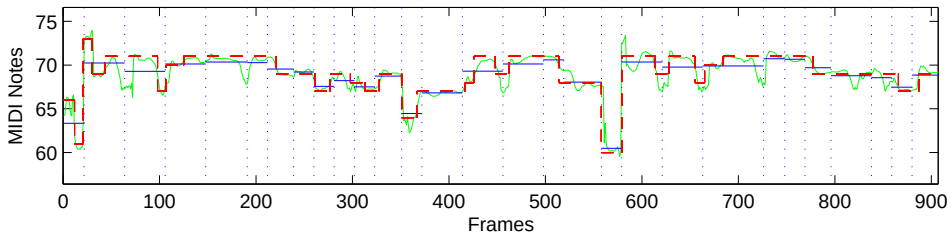

Figure 5.15: Comparison of recognized note trajectories: case II. The green line denotes the pitch p_{est}; the blue line denotes p_{avg}; the red line denotes the output of the HMM-based recognizer.

Chapter 6

Results

This chapter presents the evaluation of the proposed music recognizer. Section 6.1 shows the flow chart of the proposed system. Section 6.2 compares the proposed approach with two deterministic approaches by employing quantitative evaluations. Finally, Section 6.3 reports the results.

6.1 The Proposed Music Recognizer

Figure 6.1 displays the flow chart of the proposed system, which consists of three modules: pre-processing, feature extraction and statistical model.

6.2 Evaluation

The recognizer uses an error rate as a *score* to measure the accuracy of singing. This error rate is defined being a ratio between the number of erroneously detected frames η_{err} and the number of entire frames η_s.

In this thesis, a desirable recognizer is expected to produce small error rates, due to the fact that the amateur singer participated is well-trained in the area of music.

The error rate of ith song $\xi_s(i)$ is defined as

$$\xi_s(i) = \frac{\eta_{err}(i)}{\eta_s(i)} \times 100\% \,. \tag{6.1}$$

Then $\xi_s(i)$ are averaged across all the test songs, in order to obtain the overall error rate ξ_{tot}

$$\xi_{tot} = \frac{\sum_i \eta_s(i) \cdot \xi_s(i)}{\eta_{tot}} \times 100\% \tag{6.2}$$

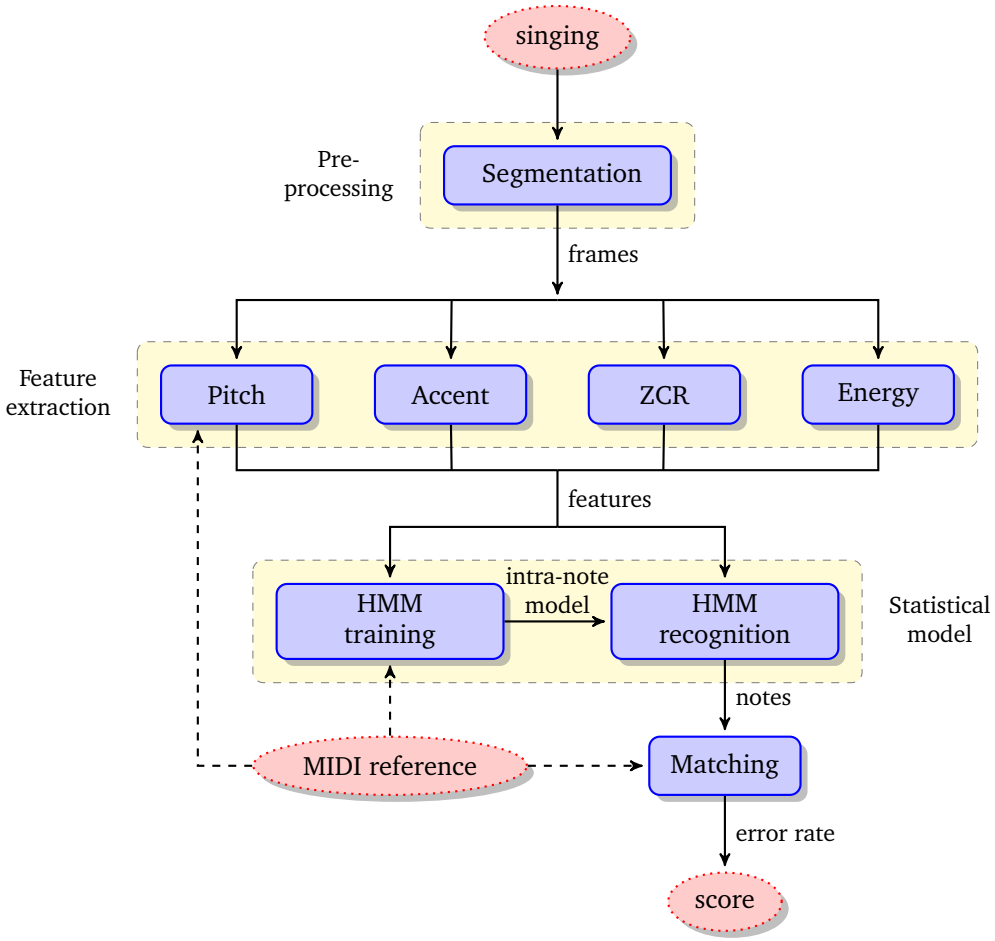

Figure 6.1: Flow chart of the proposed music recognizer

where η_{tot} refers to the amount of the entire frames in the test data set.

In an attempt to compare the proposed approach with the deterministic approaches, error rate ξ_{tot} is examined through three evaluations, i.e. frame-based, note-based and HMM-based evaluation.

Frame-Based Evaluation

The principles of frame-based pitch estimation are discussed in Section 3.3.1. Frame-based evaluation aims to test the performance of pure PDA algorithm. The pitch trajectory will not be further converted into note trajectory. The symbol ξ_f is used to denote the error rate of frame-based approach.

Note-Based Evaluation

As discussed in Section 3.3.2, the note-based estimation further obtains a note trajectory from the results of frame-based estimation. The symbol ξ_n denotes the error rate of note-based approach.

HMM-Based Evaluation

HMM-based approach depends on the proposed recognizer. The symbol ξ_h denotes the resulting error rate. The number of model states, the number of GMM mixtures and the choice of feature vector shed an influence on the performance of HMM models.

In order to find the optimal intra-note model in terms of minimal ξ_h, various conditions of parameter settings, which are shown in Table 6.1, are evaluated.

Parameters	Values
Number of HMM states, N	2, 3, 4, 5, 6
Number of GMM components, M	2, 3, 4, 5, 6
Feature vector, Ω	$\{p_\Delta, a, z\}$ $\{p_\Delta, a, e\}$ $\{p_\Delta, z, e\}$ $\{p_\Delta, a, z, e\}$
Semitone tolerance, φ	0, 1, 2, 3

Table 6.1: Experimental parameter settings

Figure 6.2 – 6.5 show the curves of ξ_h with $\varphi = 0, 1, 2, 3$ respectively. Appendix A.2 reserves the numbers of ξ_h. Each figure encompasses four panels corresponding to distinct Ω. Each panel displays the influence of N. The

x-axis denotes the number of GMM mixtures. The y-axis denotes the error rate in percentage.

Influence of the parameters is easy to be identified. Firstly, the curves of error rate have minima at $M = 5$ or 6. Smaller M (e.g. 2) affords insufficient components to describe the probability distributions of features (see Figure 5.8), and leads to large errors. Higher M (e.g. 7) can over-fits the probability distribution of features (see Figure 5.10), and also results in higher error rates.

Secondly, the curves of $N = 5, 6, 7$ are closely aligned together. This phenomena is quite evident in the two bottom panels, which correspond to the features $\{p_\Delta, z, e\}$ and $\{p_\Delta, a, z, e\}$. Large N may not greatly reduce the error rates, whereas large N costs much more computational resources than small N.

Thirdly, $\{p_\Delta, a, z\}$ does not performs as well as the rest feature vectors. This fact supports the assumption that energy contour e is competent for note segmentation.

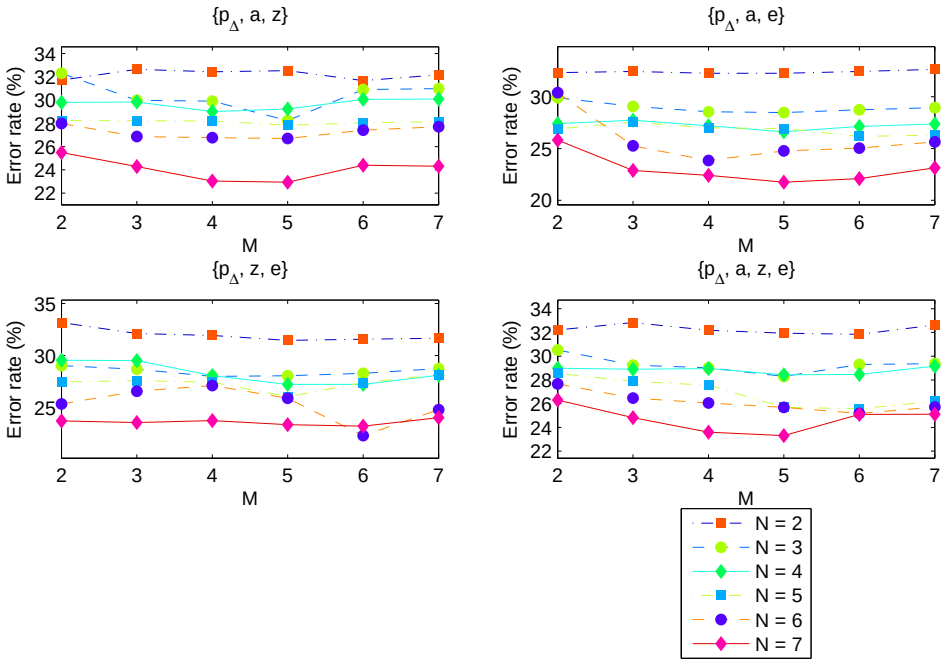

Figure 6.2: Comparison of distinct ξ_h with $\varphi = 0$

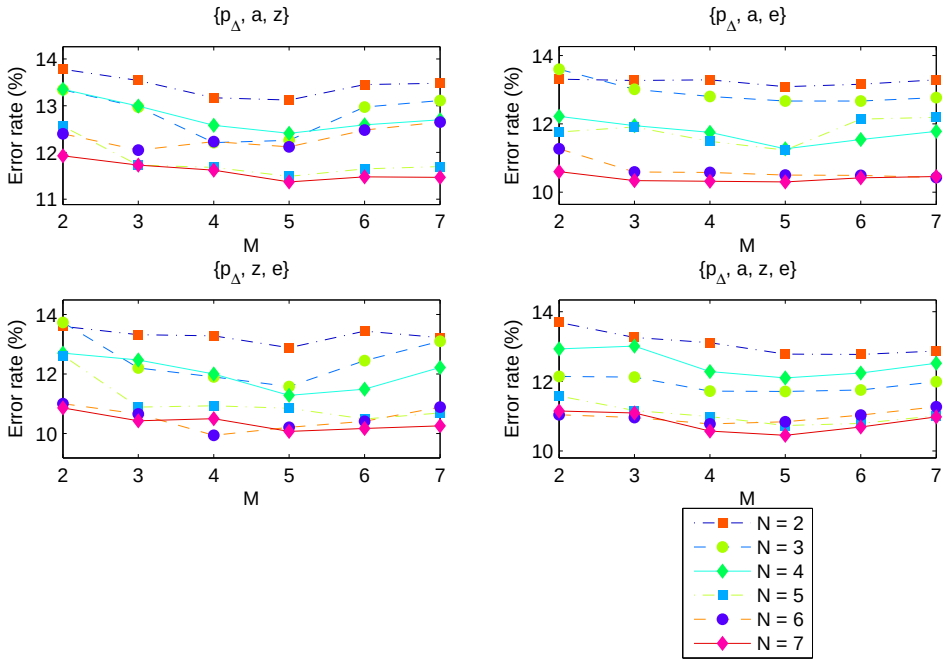

Figure 6.3: Comparison of distinct ξ_h with $\varphi = 1$

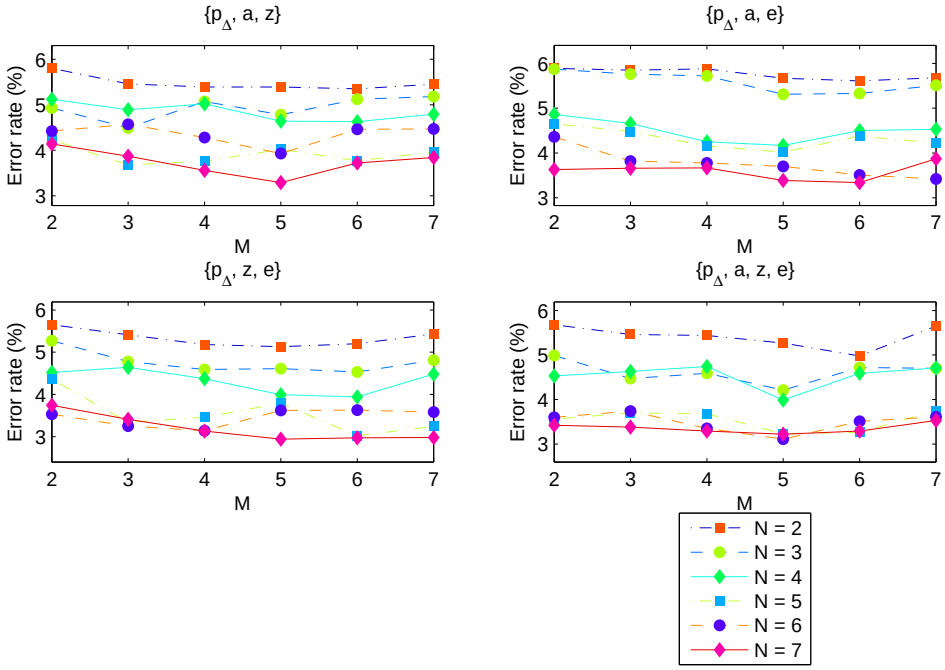

Figure 6.4: Comparison of distinct ξ_h with $\varphi = 2$

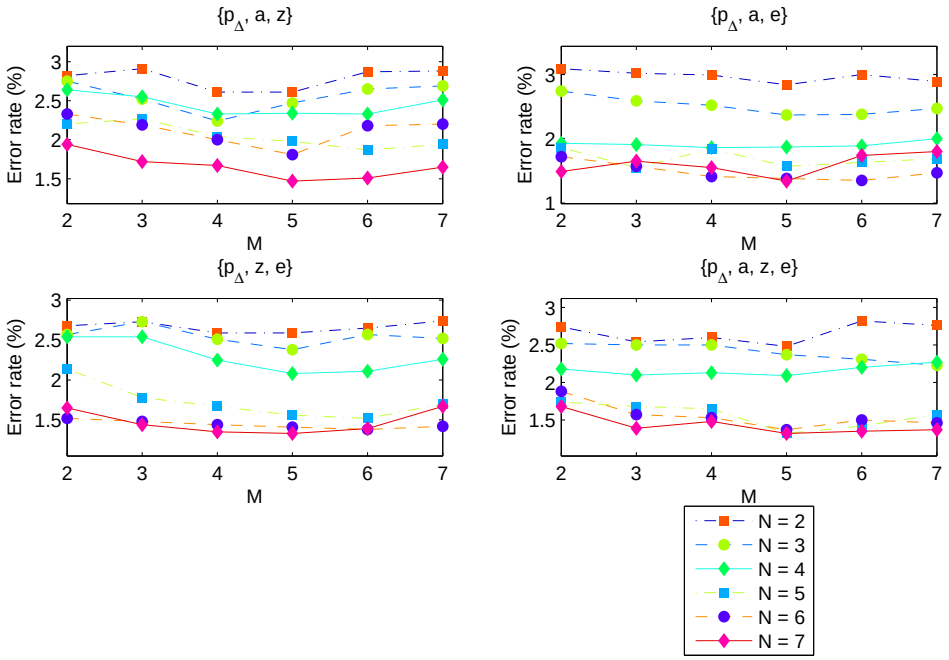

Figure 6.5: Comparison of distinct ξ_h with $\varphi = 3$

Table 6.2 lists the local minima of ξ_h for different Ω. Under tolerance $\varphi = 0$,

φ	Ω	ξ_h	M	N	φ	Ω	ξ_h	M	N
0	$\{p_\Delta, a, z\}$	22.94	7	5	1	$\{p_\Delta, a, z\}$	11.37	7	5
	$\{p_\Delta, a, e\}$	**21.75**	7	5		$\{p_\Delta, a, e\}$	10.30	7	5
	$\{p_\Delta, z, e\}$	22.30	6	6		$\{p_\Delta, z, e\}$	**9.94**	6	4
	$\{p_\Delta, a, z, e\}$	23.31	7	5		$\{p_\Delta, a, z, e\}$	10.45	7	5
2	$\{p_\Delta, a, z\}$	3.29	7	5	3	$\{p_\Delta, a, z\}$	1.47	7	5
	$\{p_\Delta, a, e\}$	3.34	7	6		$\{p_\Delta, a, e\}$	1.34	7	5
	$\{p_\Delta, z, e)$	**2.94**	7	4		$\{p_\Delta, z, e\}$	1.33	7	5
	$\{p_\Delta, a, z, e\}$	3.11	6	5		$\{p_\Delta, a, z, e\}$	**1.32**	7	5

Table 6.2: Local minimums of ξ_h for different features

the features $\{p_\Delta, a, e\}$ with $N = 7$ and $M = 5$ produces the minimal error.

Under tolerance $\varphi = 1$, the features $\{p_\Delta, z, e\}$ with $N = 6$ and $M = 4$ produces the minimal error.

Under tolerance $\varphi = 2$, the feature $\{p_\Delta, z, e\}$ with $N = 7$ and $M = 4$ produces the minimal error.

Under tolerance $\varphi = 3$, the features (p_Δ, a, z, e) with $N = 7$ and $M = 5$ produces the minimal error. And the features $\{p_\Delta, z, e\}$ with $N = 7$ and $M = 5$ produces a slightly higher error, which is regarded as a sub-optimal choice.

Briefly the features $\{p_\Delta, z, e\}$ with $N = 7$ and $M = 5$ is chosen to determine the optimal intra-note model. Then the recognition procedure is conducted as discussed in Section 5.4.

The following two figures illustrate the recognition results by using seven HMM states and five GMM mixtures. Figure 6.6 detects the underlying notes of the pitch trajectory shown in Figure 3.7 by using the proposed approach.

Figure 6.7 corresponds to the pitch trajectory shown in Figure 3.8. From the top panel to the bottom panel, φ posses the value of 0, 1 and 2 semitones separately. The red line denotes the recognized notes. The blue line denotes the reference boundaries.

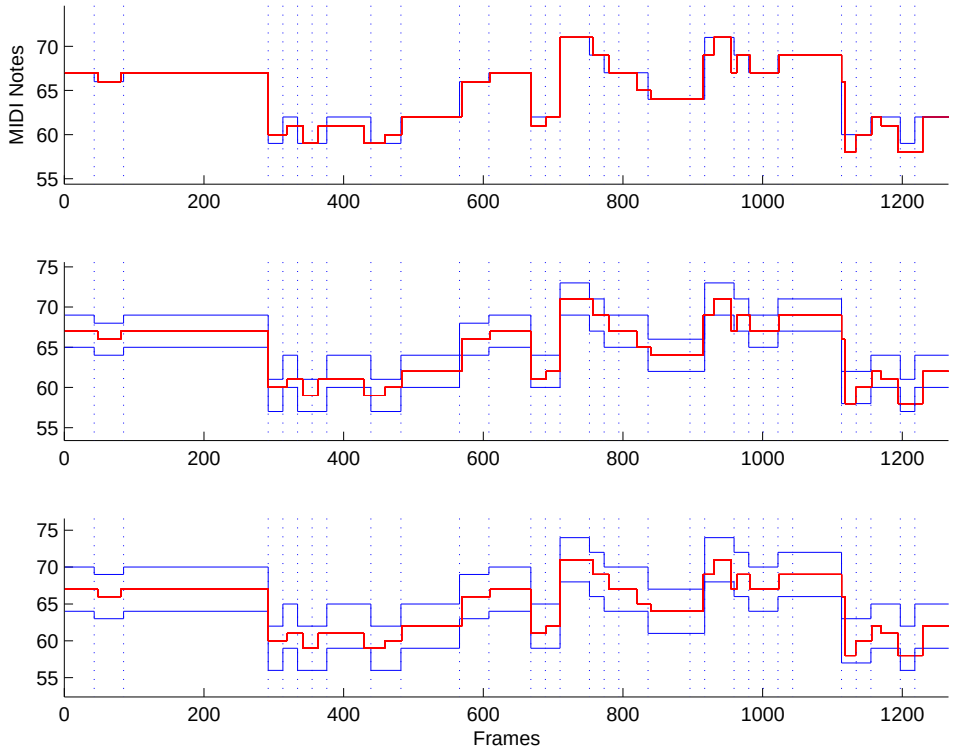

Figure 6.6: Recognized notes time aligned with reference boundaries under $\varphi = 0, 1, 2$: case I. The blue line denotes $p_{ref} \pm \varphi$, the red line denotes the recognized notes.

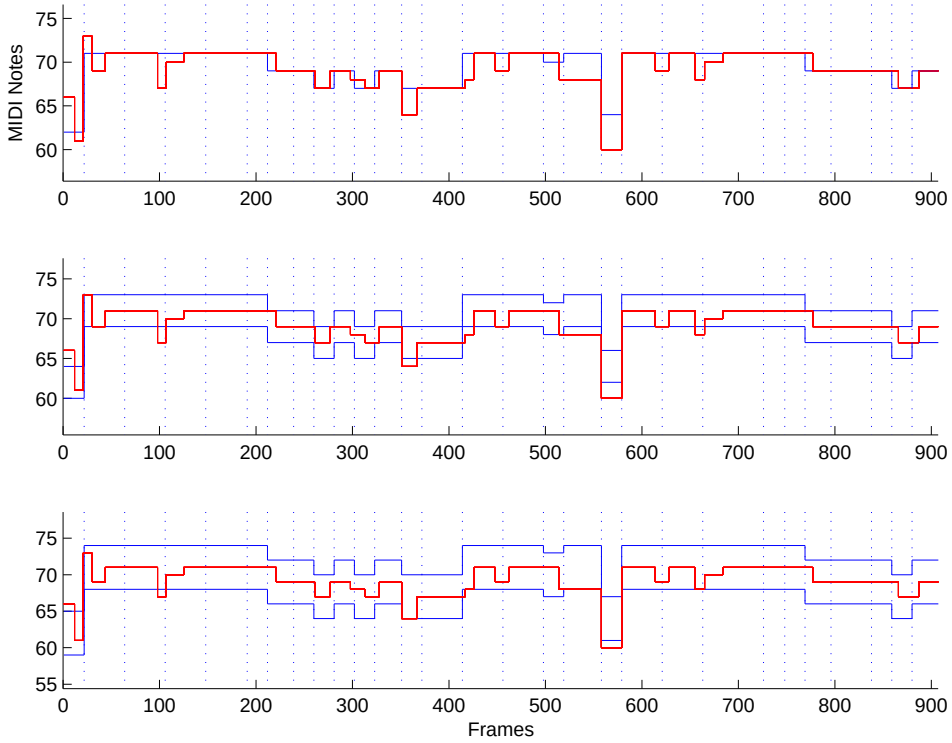

Figure 6.7: Recognized notes time aligned with reference boundaries under $\varphi = 0, 1, 2$: case II. The blue line denotes $p_{ref} \pm \varphi$, the red line denotes the recognized notes.

6.3 Comparison

Table 6.3 compares the error rates of these three evaluations. It indicates

Schemes \ φ	0.5	1	2	3
Frame-based, ξ_f	36.88	22.93	10.15	5.15
Note-based, ξ_n	28.94	15.57	5.31	2.82
HMM-based, ξ_h	23.35	10.07	2.94	1.33

Table 6.3: Comparison of error rates ξ_f, ξ_n and ξ_h

that the HMM-based approach outperforms the rest when $\varphi \leq 3$ semitones. The results support the assumption that statistical method is more feasible for singing recognition.

On the other hand, it makes sense to weight each note with a coefficient. Difficult notes demand large coefficients; easy notes demand small coefficients.

Further more, a long term goal is to assign a sensible score (grade) to the users. Musical tutors are expected to join in the scoring process. A proper score can be obtained by adjusting the error rate through linear transformation. Further work is needed to fulfil this claim.

Chapter 7

Summary

7.1 Conclusions

A statistical approach to singing recognition is studied in this paper. The proposed recognizer accepts acoustic input, typically singing. It tracks the musical features of the input and segments the feature stream into music notes. Note models are defined by HMM models with the probability distributions of features measured by GMM models.

A singing database is constructed by an amateur singer who is well-trained in the area of music. During the training stage, an intra-note model is trained using the toolbox HTK. The intra-note model describes the statistical behavior of model states inside a note.

Then, a group of note models, covering a wide range of pitch, are built upon the trained intra-note model. Also a recognition network is constructed. During the recognition stage, the test data are decoded with the aid of the note models and the recognition network.

This report discusses the signal processing and statistical modeling operations involved in the recognition. It further compares the proposed approach with two deterministic approaches, i.e. frame-based and note-based. Experimental results show that the proposed approach wins.

One advantages of this system is the recognition of note boundaries depends on the trained note models instead of the reference. Another advantage is the capability of changing the models by just adding and changing features and without requiring the design of new system.

We believe that the approach presented here can be extended to broader applications. Songwriters could compose tunes easily by humming, and the system automatically identifies the associated music notes. This system could work as a computer tutor and aid users to practice and master those elusive melodies.

7.2 Future Work

The research results, however, are preliminary. First, the database used in this work posses a small size. To enable detailed modeling, a more comprehensive collection of songs from a larger set of subjects needs to be gathered. It is helpful to classify the collection songs into groups with respect to their stylistic characteristics, e.g. artist or genre. Each group posses unique and specific note models in the training and recognition process.

Second, the note models are trained by data from a single singer alone. Such systems are commonly known as singer dependent system. Consequently, recognition accuracy for other users would be insufficient. To overcome this limitation, a singer independent system for a variety of singers could be constructed.

Usually this would require large amounts of training data to enable a specific modeling for each singer. HTK toolkit provides an alternative solution to address this issue by using a small amount of training data [3, pp. 153-164]. Adaptation techniques are applied to current singer dependent models to characterize a new singer through the use of singers' differences.

Third, the HTK toolkit is implemented through C language though. Wrappers in other common languages, including Java, Python and Matlab, afford the interfaces to use HTK in diverse environments. In the near future, embedded devices (e.g. smart phone) will become a hot-spot in our daily life. It is an interesting topic to develop a Karaoke game for the mobile internet.

Finally, it is vital to compare the proposed recognizer with other existing Karaoke computer games (e.g. Singing Star, Karaoke Revolution), via testing across a comprehensive database of singing record.

Appendix A

Details of Experiment

A.1 Information of Acoustic Database

The database contains ten pieces of songs in total for the training and test of HMM models. Table A.1 lists out the details of all the songs, i.e. title, original singer, tempo, duration and the number of notes in each song. The training set incorporates 5 songs (i.e. 2315 notes). The test set incorporates another 5 songs (i.e. 1657 notes).

	Counter	Title	Original singer	Tempo	Duration	Music notes
				BPM	min:second	
Training	1	Quiet	Jay Chou	120	5:34	508
	2	Wake Me Up When September Ends	Green Day	315	4:45	552
	3	I Want It That Way	Backstreet Boys	198	3:37	331
	4	When You're Gone	Avril Lavigne	142	4:00	414
	5	When You Say Nothing At All	Ronan Keating	261	4:20	510
						Sum 2315
Test	6	Wonderful Tonight	Eric Clapton	95	3:37	290
	7	Big City, Small Love	Leehom Wang	65	3:45	304
	8	River South	Jun-jie Lin	60	4:28	314
	9	Love Me Then Don't Leave	Chen-yue Chang	92	4:44	393
	10	Fairy Tale	Michael Wong	136	4:05	356
						Sum 1657

Table A.1: Details of the songs

A.2 Error rates of HMM-based Evaluation

This section presents the outcomes produced by HMM-based evaluation for various parameter settings shown in Table 6.1.

Table A.2 – A.5 provide the ε_h with $\varphi = 0, 1, 2, 3$ respectively. The local minima are marked by bold typeface.

$\Omega = \{p_\Delta, a, z\}$						
N \ M	2	3	4	5	6	7
2	31.71	32.64	32.43	32.52	31.67	32.16
3	32.30	29.98	29.90	28.24	30.90	30.98
4	29.80	29.83	29.01	29.23	30.06	30.09
5	28.24	28.23	28.20	27.84	28.04	28.14
6	27.99	26.86	26.77	26.69	27.41	27.71
7	25.49	24.28	23.04	**22.94**	24.40	24.31

$\Omega = \{p_\Delta, a, e\}$						
N \ M	2	3	4	5	6	7
2	32.33	32.45	32.27	32.27	32.44	32.65
3	29.91	29.07	28.56	28.47	28.75	28.95
4	27.40	27.74	27.20	26.62	27.14	27.37
5	26.90	27.54	26.99	26.94	26.18	26.28
6	30.40	25.26	23.85	24.77	25.04	25.64
7	25.82	22.88	22.41	**21.75**	22.10	23.14

$\Omega = \{p_\Delta, z, e\}$						
N \ M	2	3	4	5	6	7
2	33.17	32.12	31.93	31.46	31.57	31.66
3	29.03	28.69	27.99	28.05	28.29	28.74
4	29.54	29.52	28.08	27.22	27.22	28.10
5	27.46	27.59	27.43	26.05	27.40	28.15
6	25.35	26.56	27.11	25.89	**22.30**	24.79
7	23.71	23.56	23.75	23.35	23.21	24.02

$\Omega = \{p_\Delta, a, z, e\}$						
N \ M	2	3	4	5	6	7
2	32.21	32.83	32.19	31.94	31.84	32.62
3	30.52	29.24	29.03	28.29	29.30	29.37
4	28.99	28.90	28.95	28.44	28.46	29.16
5	28.54	27.88	27.56	25.68	25.58	26.20
6	27.67	26.47	26.06	25.69	25.18	25.71
7	26.30	24.82	23.60	**23.31**	25.09	25.11

Table A.2: Error rate ε_h with $\varphi = 0$

$\Omega = \{p_\Delta, a, z\}$						
N \ M	2	3	4	5	6	7
2	13.78	13.54	13.17	13.12	13.45	13.48
3	13.34	12.97	12.21	12.26	12.97	13.11
4	13.35	12.99	12.58	12.41	12.59	12.70
5	12.56	11.73	11.68	11.49	11.65	11.70
6	12.40	12.05	12.23	12.12	12.48	12.65
7	11.93	11.73	11.62	**11.37**	11.48	11.47

$\Omega = \{p_\Delta, a, e\}$						
N \ M	2	3	4	5	6	7
2	13.31	13.27	13.29	13.09	13.16	13.29
3	13.60	13.01	12.80	12.67	12.67	12.77
4	12.22	11.95	11.75	11.27	11.54	11.78
5	11.76	11.91	11.49	11.23	12.14	12.19
6	11.27	10.59	10.58	10.50	10.49	10.43
7	10.60	10.34	10.32	**10.30**	10.42	10.46

$\Omega = \{p_\Delta, z, e\}$						
N \ M	2	3	4	5	6	7
2	13.60	13.32	13.28	12.88	13.44	13.22
3	13.73	12.20	11.90	11.58	12.45	13.10
4	12.70	12.47	12.00	11.28	11.49	12.22
5	12.62	10.89	10.93	10.85	10.49	10.69
6	11.00	10.66	**9.94**	10.21	10.41	10.89
7	10.86	10.43	10.50	10.07	10.17	10.26

$\Omega = \{p_\Delta, a, z, e\}$						
N \ M	2	3	4	5	6	7
2	13.69	13.26	13.11	12.78	12.77	12.87
3	12.14	12.12	11.72	11.71	11.75	11.99
4	12.93	13.01	12.28	12.10	12.24	12.52
5	11.58	11.16	10.99	10.73	10.79	11.01
6	11.04	10.96	10.78	10.84	11.03	11.27
7	11.15	11.09	10.57	**10.45**	10.69	10.98

Table A.3: Error rate ε_h with $\varphi = 1$

$\Omega = \{p_\Delta, a, z\}$						
N \ M	2	3	4	5	6	7
2	5.80	5.46	5.39	5.39	5.35	5.45
3	4.93	4.49	5.07	4.78	5.12	5.18
4	5.12	4.89	5.02	4.64	4.63	4.79
5	4.23	3.68	3.76	4.02	3.77	3.96
6	4.42	4.57	4.28	3.93	4.46	4.47
7	4.14	3.87	3.56	**3.29**	3.72	3.84

$\Omega = \{p_\Delta, a, e\}$						
N \ M	2	3	4	5	6	7
2	5.89	5.85	5.88	5.67	5.61	5.68
3	5.87	5.76	5.72	5.31	5.33	5.51
4	4.86	4.66	4.25	4.17	4.50	4.53
5	4.65	4.48	4.16	4.02	4.38	4.22
6	4.36	3.82	3.78	3.70	3.51	3.42
7	3.63	3.66	3.67	3.39	**3.34**	3.87

$\Omega = \{p_\Delta, z, e\}$						
N \ M	2	3	4	5	6	7
2	5.65	5.41	5.18	5.13	5.20	5.43
3	5.27	4.78	4.59	4.61	4.53	4.81
4	4.52	4.64	4.37	3.99	3.94	4.48
5	4.36	3.34	3.46	3.79	3.02	3.24
6	3.53	3.25	3.14	3.62	3.63	3.58
7	3.74	3.41	3.13	**2.94**	2.97	2.98

$\Omega = \{p_\Delta, a, z, e\}$						
N \ M	2	3	4	5	6	7
2	5.68	5.46	5.44	5.27	4.98	5.65
3	4.99	4.47	4.59	4.21	4.72	4.70
4	4.53	4.63	4.74	3.99	4.59	4.71
5	3.56	3.70	3.68	3.23	3.26	3.75
6	3.60	3.74	3.35	**3.11**	3.51	3.61
7	3.42	3.38	3.29	3.22	3.29	3.53

Table A.4: Error rate ε_h with $\varphi = 2$

$\Omega = \{p_\Delta, a, z\}$						
N \ M	2	3	4	5	6	7
2	2.82	2.91	2.61	2.61	2.87	2.88
3	2.75	2.52	2.24	2.47	2.65	2.69
4	2.64	2.55	2.33	2.34	2.33	2.51
5	2.20	2.27	2.04	1.98	1.87	1.94
6	2.33	2.19	2.00	1.81	2.18	2.20
7	1.94	1.72	1.67	**1.47**	1.51	1.65

$\Omega = \{p_\Delta, a, e\}$						
N \ M	2	3	4	5	6	7
2	3.09	3.02	2.99	2.84	3.00	2.89
3	2.74	2.59	2.52	2.37	2.38	2.47
4	1.93	1.91	1.86	1.87	1.89	2.00
5	1.86	1.55	1.84	1.57	1.63	1.69
6	1.72	1.57	1.41	1.38	1.35	1.47
7	1.49	1.65	1.55	**1.34**	1.74	1.80

$\Omega = \{p_\Delta, z, e\}$						
N \ M	2	3	4	5	6	7
2	2.68	2.73	2.59	2.59	2.65	2.74
3	2.57	2.73	2.51	2.38	2.57	2.52
4	2.54	2.54	2.25	2.08	2.11	2.26
5	2.14	1.78	1.67	1.56	1.52	1.70
6	1.52	1.48	1.44	1.41	1.38	1.42
7	1.65	1.44	1.35	**1.33**	1.39	1.67

$\Omega = \{p_\Delta, a, z, e\}$						
N \ M	2	3	4	5	6	7
2	2.74	2.54	2.60	2.48	2.82	2.76
3	2.52	2.50	2.50	2.37	2.31	2.23
4	2.18	2.10	2.13	2.09	2.20	2.27
5	1.74	1.68	1.65	**1.32**	1.42	1.57
6	1.88	1.57	1.53	1.37	1.50	1.46
7	1.68	1.39	1.48	**1.32**	1.35	1.37

Table A.5: Error rate ε_h with $\varphi = 3$

Appendix B

Forward-Backward Algorithm

Section B.1 and B.2 explain the forward and backward procedure respectively.

B.1 Forward Procedure

The definition of the forward probability $\alpha_i(t)$ is

$$\begin{aligned} \alpha_i(t) &= P(\boldsymbol{O}_t, q_t = i|\lambda) \\ &= P(\boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_t, q_t = i|\lambda) \end{aligned} \tag{B.1}$$

that is, the joint probability of seeing the observations $\boldsymbol{O}_t$ and being in state i at time t, given the model λ.

The forward probability is calculated by the forward recursion [31, p. 335]

Initialization

$$\begin{aligned} \alpha_i(1) &= P(\boldsymbol{o}_1, q_1 = i|\lambda) \\ &= \pi_i \cdot b_i(\boldsymbol{o}_1) \end{aligned} \tag{B.2}$$

Recursion

$$\begin{aligned} \alpha_j(t+1) &= P(\boldsymbol{O}_{t+1}, q_{t+1} = j|\lambda) \\ &= P(\boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_{t+1}, q_{t+1} = j|\lambda) \\ &= P(\boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_t, q_{t+1} = j|\lambda) \cdot P(\boldsymbol{o}_{t+1}|q_{t+1} = j) \\ &= \sum_{i=1}^{N} P(\boldsymbol{o}_1\, \boldsymbol{o}_2 \ldots \boldsymbol{o}_t, q_t = i, q_{t+1} = j|\lambda) \cdot P(\boldsymbol{o}_{t+1}|q_{t+1} = j) \\ &= \left(\sum_{i=1}^{N} \alpha_i(t) \cdot a_{ij} \right) \cdot b_j(\boldsymbol{o}_{t+1}) \end{aligned} \tag{B.3}$$

Equation B.2 initializes the forward procedure as the joint probability of observing $\boldsymbol{o}_1$ and being in state i. Equation B.3 induces α at time $t+1$ from time t. The product $\alpha_i(t) \cdot a_{ij}$ is the joint probability of observing $\boldsymbol{O}_t$ and arriving state j at time $t+1$ from state i. Once the summation of this product over all possible i is done, we then consider the observation $\boldsymbol{o}_{t+1}$, by multiplying the summation with probability $b_j(\boldsymbol{o}_{t+1})$. Figure B.1 shows the forward procedure.

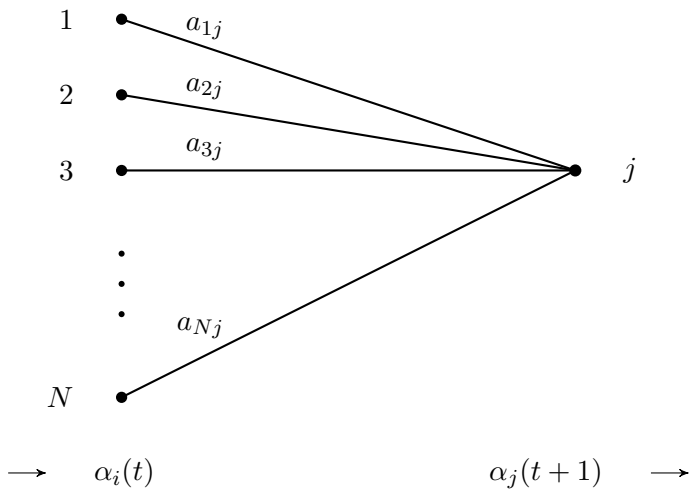

Figure B.1: Computation of $\alpha_j(t+1)$ in the forward procedure

B.2 Backward Procedure

The definition of the backward probability $\beta_i(t)$ is

$$\beta_i(t) = P(\boldsymbol{o}_{t+1}\, \boldsymbol{o}_{t+2} \ldots \boldsymbol{o}_T | q_t = i, \lambda) \tag{B.4}$$

that is, the probability of the partial observations from time $t+1$ to time T, given state i at time t and the model λ.

The backward probability is calculated by the backward recursion [31, p. 337]

Initiation

$$\beta_i(T) = a_{iN} \tag{B.5}$$

Recursion

$$\begin{aligned}\beta_i(t) &= \sum_{j=1}^{N} P(\boldsymbol{o}_{t+1}\,\boldsymbol{o}_{t+2}\ldots\boldsymbol{o}_T, q_{t+1}=j|q_t=i,\lambda)\\ &= \sum_{j=1}^{N} P(q_{t+1}=j|q_t=i)\cdot P(\boldsymbol{o}_{t+1}|q_{t+1}=j)\\ &\qquad \cdot P(\boldsymbol{o}_{t+2}\ldots\boldsymbol{o}_T|q_{t+1}=j,\lambda)\\ &= \sum_{j=1}^{N} a_{ij}\cdot b_j(\boldsymbol{o}_{t+1})\cdot\beta_j(t+1) \qquad \text{(B.6)}\end{aligned}$$

Equation B.5 initializes the backward procedure and assigns $\beta_i(T)$ with 1 for all i. Equation B.6 indicates that in order to obtain $\beta_i(t)$, it is necessary to consider the transition from i to j (i.e. the a_{ij} term), the probability of observing $\boldsymbol{o}_{t+1}$ in state j (i.e. the $b_j(\boldsymbol{o}_{t+1})$ term), and the remaining observations from time $t+2$ on (i.e. the $\beta_j(t+1)$ term). Figure B.2 shows the backward procedure.

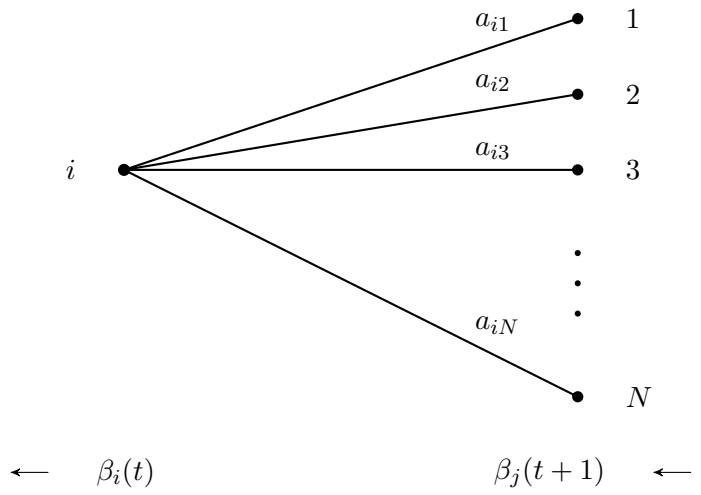

Figure B.2: Computation of $\beta_i(t)$ in backward procedure

Bibliography

[1] Digital Audio Editor Audacity, http://audacity.sourceforge.net/, 2012.

[2] Hidden Markov Model Toolkit (HTK), http://htk.eng.cam.ac.uk/, 2012.

[3] The HTK Book, http://htk.eng.cam.ac.uk/docs/docs.shtml, 2012.

[4] UltraStar Deluxe Forums, http://usdb.animux.de/, 2012.

[5] R. Bakis. Continuous speech word recognition via centisecond acoustic states. *Proc. ASA Meeting*, April 1976.

[6] L.E. Baum and J.A. Egon. An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. *Bull. Amer. Meteorol. Soc.*, 73:360–363, 1967.

[7] L.E. Baum and T. Petrie. Statistical interference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.*, 37:1554–1563, 1966.

[8] J.C. Brown and M.S. Puckette. An Efficient Algorithm for the Calculation of a Constant Q Transform. *J. Acoust. Soc. Am*, pages 2698–2701, 1992.

[9] Arturoa Camacho. Swipe: A Sawtooth Waveform Inspired Pitch Estimator for Speech and Music. 2007. PhD Thesis.

[10] Pedro Cano, Alex Loscos, and Jordi Bonada. Score-Performance Matching using HMMs. pages 441–444. International Computer Music Association, 1999.

[11] A. De Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111:1917, 2002.

[12] P. de la Cuadra, A. Master, and C. Sapp. Efficient pitch detection techniques for interactive music. *In Proceedings of the 2001 International Computer Music Conference, La Habana, Cuba*, September 2001.

[13] Tuomas Eerola and Petri Toiviainen. *MIDI toolbox: MATLAB Tools for Music Research*. Department of Music, University of Jyväskylä, Finland, 2004.

[14] B. Gold and L. Rabiner. Parallel Processing Techoniqes for Estimating Pitch Periods of Speech in the Time Domain. *The Journal of The Acoustical Society of America*, 46(2):442–228, August 1969.

[15] E. Bruce Goldstein. *Blackwell handbook of perception*. Wiley-Blackwell, fourth edition, 2001.

[16] Lorin Grubb and Roger B. Dannenberg. A Stochastic Method of Tracking a Vocal Performer. *Proceedings of the International Computer Music Conference*, pages 301–308, 1997.

[17] G. Hamerly and C Elkan. Alternatives to the k-means algorithm that find better clusterings. *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM)*, 2002.

[18] W. J. Hess. *Pitch Determination of Speech Signals*. Springer, New York, 1983.

[19] Michael Hewitt. *Music Theory for Computer Musicians*. Course Technology, 2008.

[20] Ian Howard. *Speech Fundamental Period Estimation Using Pattern Classification*. PhD Thesis, University of London, 1992.

[21] Anssi P. Klapuri. Musical Meter Estimation and Music Transcription. *In Proceedings of Cambridge Music Processing Colloquium*, March 2003.

[22] Daniel Levitin. *This is Your Brain on Music*, page 40. Penguin Group, 2007.

[23] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):29137, 1982.

[24] David MacKay. *Information Theory, Inference and Learning Algorithms*, chapter 20 An Example Inference Task: Clustering, page 284292. Cambridge University Press, 2003.

[25] Ian McLoughlin. *Applied Speech and Audio Processing: With MATLAB Examples*. Cambridge University Press, 2009.

[26] Harry F Olson. *Music, Physics and Engineering*, pages 171, 248–251. Dover Publications, 1967.

[27] N. Orio and F. Déchelle. Score following using spectral analysis and hidden Markov models. pages 151–154. International Computer Music Association, 2001.

[28] Roy D. Patterson. *Music Perception*. Springer handbook of auditory research. Springer, 2010.

[29] Christopher J. Plack. *Pitch: Neural Coding and Perception*. Springer, 2005.

[30] Miller Puckette. Score following using the sung voice. *Proceedings of the 1995 International Computer Music Conference*, December 1995.

[31] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall Signal Processing. Prentice-Hall International Inc., 1993.

[32] Lawrence R. Rabiner. On the Use of Autocorrelation Analysis for Pitch Detection. *IEEE Transaction on Acoustics, Speech, and Signal Processing*, ASSP-25(1), February 1977.

[33] Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*. Prentice-Hall, New Jersey, USA, 1978.

[34] L.R. Rabiner, M.J. Cheng, A.E. Rosenberg, and C.A. McGonegal. A comparative performance study of several pitch detection algorithm. *IEEE Ttrans. Acoust., Speech, and Signal Processing*, ASSP-24:399–418, October 1976.

[35] Christopher Raphael. Automatic Segmentation of Acoustic musical Signals Using Hidden Markov Models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(4):360–370, April 1999.

[36] M. Ryynänen and A. P. Klapuri. Modelling of note events for singing transcription. *Proc. ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Proc.*, October 2004.

[37] G. Saha, S. Chakroborty, and S. Senapati. A new Silence Removal and End Point Detection Algorithm for Speech and Speaker Recognition Applications. *Proc. of Eleventh National Conference on Communications (NCC),*, pages 291–295, January 2005.

[38] M. R. Schroeder. Period histogram and product spectrum: new methods for fundamental frequency measurement. *Journal of the Acoustical Society of America*, 43(4):829–834, 1968.

[39] H.-H. Shih, S. S. Narayanan, and C.-C. J. Kuo. An hmm-based approach to humming transcription. *IEEE International Conference on Multimedia and Expo (ICME2002)*, August 2002.

[40] J. Sundberg. Expression in music: Comparing vocal and instrumental performance. *In I Karevold, H Jörgensen, IM Hanken, E Nesheim, eds, Flerstemmige innspill 2000*, MH-publikasjoner 2000(1):5–19, 2001.

[41] David Talkin. A Robust Algorithm for Pitch Tracking (RAPT). In W.B. Kleijn and K.K. Paliwal, editors, *Speech Coding and Synthesis*, chapter 14, pages 497–516. Elsevier Science, December 1995.

[42] Peter Vary and Rainer Martin. *Digital Speech Transmission (Enhancement, Coding and Error Concealment)*. John Willy & Sons, LTD, 2006.

[43] T. Viitaniemi, A. Klapuri, and A. Eronen. A probabilistic model for the transcription of single-voice melodies. *Proc. 2003 Finnish Signal Processing Symposium*, pages 59–63, May 2003.

[44] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, IT-13:260–269, April 1967.

Index