AUTOMATIC ACCOMPANIMENT OF VOCAL MELODIES

IN THE CONTEXT OF POPULAR MUSIC

A Thesis
Presented to
The Academic Faculty

by

Xiang Cao

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Music Technology in the
Music Department, College of Architecture

Georgia Institute of Technology
MAY 2009

AUTOMATIC ACCOMPANIMENT OF VOCAL MELODIES

IN THE CONTEXT OF POPULAR MUSIC

Approved by:

Dr. Parag Chordia, Advisor
Music Department, College of Architecture
Georgia Institute of Technology

Dr. Jason Freeman
Music Department, College of Architecture
Georgia Institute of Technology

Dr. Gil Weinberg
Music Department, College of Architecture
Georgia Institute of Technology

Date Approved: April 2, 2009

ACKNOWLEDGEMENTS

First, I would like to thank my thesis advisor Dr. Parag Chordia. Without his feedback and guidance, this thesis work is not possible to be completed. My thank also goes to the rest of my thesis committee members for their suggestions and reminders. Additionally, I want to thank the director of the music technology program Dr. Gil Weinberg, the funding found by him supported me for two years through this master program. At last, I also would like to thank all the master students of the music technology program for their everyday help.

TABLE OF CONTENTS

		Page
ACKNOWLEDGEMENTS		iii
LIST OF TABLES		vi
LIST OF FIGURES		vii
SUMMARY		viii
CHAPTER		
1	Introduction	1
	Motivation	1
	Related Works	1
	System Description	4
	System Integration	5
2	Pitch Tracking	7
	Pitch Detection	7
	Post Processing	9
3	Key Estimation	11
	Tonic and Scale	11
	The Method	12
	Evaluation	14
4	Structure Analysis	16
	The Role of Structure Information in Automatic Accompaniment	16
	Similarity Matrix	17
	Adaptive Threshold	19
	Structure Boundaries	22

Evaluation 23

5 Chord Assignment 25

The Problem of Harmonization 25

Chord Set and Transition Probabilities 27

Output Probabilities 28

HMM Decoding 29

6 Style Player 31

Automatic Accompaniment Style Files 31

Structural Sections 32

Instrumentation and Note Transposition 32

7 Application and Conclusion 35

Applications 35

Future Works 38

Conclusion 40

REFERENCES 42

LIST OF TABLES

Page

Table 1: Key Estimation Accuracy 14

Table 2: Structure Analysis Test Results 24

Table 3: All Possible Sections in a Style file 32

Table 4: Style File Channel Arrangement 33

LIST OF FIGURES

	Page
Figure 1: System Structure	4
Figure 2: Inter-peak Distances for the First Maximum	8
Figure 3: Pitch List	9
Figure 4: Major and Minor Scale PCDs	13
Figure 5: Similarity Matrix	18
Figure 6: Time-lag Matrix	19
Figure 7: Filtered Time-lag Matrix	21
Figure 8: High Level Repetitions	22
Figure 9: Section Change implied by Repetition	22
Figure 10: Graphical Representation of Hidden Markov Model	26
Figure 11: PCDs of Major Scale Chord Set	28
Figure 12: GUI of the Prototype Application	35

SUMMARY

A piece of popular music is usually defined as a combination of vocal melody and instrumental accompaniment. People often start with the melody part when they are trying to compose or reproduce a piece of popular music. However, creating appropriate instrumental accompaniment part for a melody line can be a difficult task for non-musicians. Automation of accompaniment generation for vocal melodies thus can be very useful for those who are interested in singing for fun. Therefore, a computer software system which is capable of generating harmonic accompaniment for a given vocal melody input has been presented in this thesis. This automatic accompaniment system uses a Hidden Markov Model to assign chord to a given part of melody based on the knowledge learnt from a bank of vocal tracks of popular music. Comparing with other similar systems, our system features a high resolution key estimation algorithm which is helpful to adjust the generated accompaniment to the input vocal. Moreover, we designed a structure analysis subsystem to extract the repetition and structure boundaries from the melody. These boundaries are passed to the chord assignment and style player subsystems in order to generate more dynamic and organized accompaniment. Finally, prototype applications are discussed and the entire system is evaluated.

CHAPTER 1

INTRODUCTION

In this chapter, the motivation and related works will be discussed. Furthermore, the architecture of the automatic accompaniment system and the structure of this thesis will also be illustrated.

1.1 Motivation

A piece of popular music is usually defined as a combination of vocal melody and instrumental accompaniment. Melody is a series of notes which forms the theme of the music. Hence people often start with the melody part when they are trying to compose or reproduce a piece of popular music. On the other hand, chord progression builds up the harmony part which is the core of the accompaniment. However, creating appropriate instrumental accompaniment part for a melody line can be a difficult task for non-musicians. Automation of accompaniment generation for vocal melodies thus can be very useful for those who are interested in singing for fun, because instrumental accompaniment plays a significant part in building the tension and feeling of a piece of popular music. Under this circumstance, an automatic accompaniment software system is designed. In this system, music novices sing or hum a melody line along with the selected rhythm. Then the system generates a MIDI accompaniment which contains rhythms, chords and phrases to be played with the melody together. This generated music could be used for entertainment purposes. Besides this off-line mode, another interactive application is also developed at the last chapter to provide continuous music interactions between computer and user.

1.2 Related Works

Automatic accompaniment has been a standard feature on some professional arranger keyboards [15] since 1990 [22]. This feature allows players to change the current chord of the background music by striking and holding a chord on keyboard in real-time. But in strict sense, this is not automatic accompaniment, because players still have to figure out the chords to play for given melodies manually.

The term “automatic accompaniment” is also widely used in the automatic score following research. Typical examples of the automatic accompaniment system based on score following can be found in Roger Dannenberg’s work [1] and Barry Vercoe’s work [25]. In this type of systems, the computer has the ability to follow a soloist. It processes the input from a live performer and matches this input against the expected score. The timing information is generated to control the playback of the accompaniment. Nevertheless, the content of the accompaniment here has to be determined beforehand. This limitation makes the system inapplicable to improvisation.

As mentioned before, harmony is the core of the accompaniment. To generate accompaniment for any given melodies, the problem of automatic harmonization should be addressed. This leads to another type of automatic accompaniment system which is capable of harmonizing symbolic melodies. Ching-Hua Chuan and Elanie Chew [2] presented a hybrid system for generation of style-specific accompaniment. They constructed the chord progression list from MIDI melody using neo-Riemannian transforms. The alternate paths are represented in a tree structure. Then a Markov chain with learnt probabilities for these transforms generates the final chord progression. Similar approaches can be found in commercial products such as Band-in-a-Box [23] as well. But one major limitation of this kind of systems is that they require users to input the melody in formal musical formats like MIDI or score. This request may result in abandonment of amateur users who are more like to enjoy the pleasure of automatic accompaniment.

Recently, Ian Simon, Dan Morris and Sumit Basu [3] proposed an automatic accompaniment system for vocal melodies based on Hidden Markov Model. This work is trying to provide a fast-prototyping system for composers to record the melodies in their minds and represent these melodies with instrumental accompaniments. Users could change the mood and jazzy parameters to get inspired by different chord progression possibilities for a given melody. This idea addressed the problems of previous works but more efforts can be made to improve the quality of the generated music. For instance, the generated instrumental accompaniment from this software always repeats the same pattern again and again no matter how the melody line goes. Besides this, the system tends to generate accompaniment which is out of tune with the melody input if the user does not sing in chromatic key.

Our approach of automatic accompaniment differs from these related works by focusing on the popular music style accompaniment and aiming at non-musician users and entertainment purposes. Specifically in contrast with the system described in [3], a pitch class distribution based key estimation algorithm is performed before chord assignment in order to improve the relevance between chord and melody. This key estimation also features 10 cents resolution which is helpful to overcome the singing in non-chromatic key issue. In addition to this, a similarity matrix based structure analysis algorithm is applied to the melody input to detect repetition and structure boundaries. These boundary positions are used in style player and chord assignment subsystems to reinforce the predictability of the generated accompaniment. Moreover, the statistical models used in the chord assignment subsystem are trained on vocal audio data instead of MIDI data. This choice of training data makes the entire system more coherent and more relevant to the real input of the system. At last, we introduce a new interactive application of the system which is able to provide non-stop interactions between vocal melody and instrumental accompaniment. In this application, users sing along with the accompaniment which is generated based on previous melody input.

1.3 System Description

This section provides the brief introduction of our automatic accompaniment system. The system architecture and data flow are shown in Figure 1. As you can see, it is capable of generating MIDI accompaniment for a given vocal audio track. Basically, our system consists of five subsystems which are pitch tracking, key estimation, structure analysis, chord assignment and style player. In the following paragraphs, the overall designs and arrangements of these subsystems are explained.

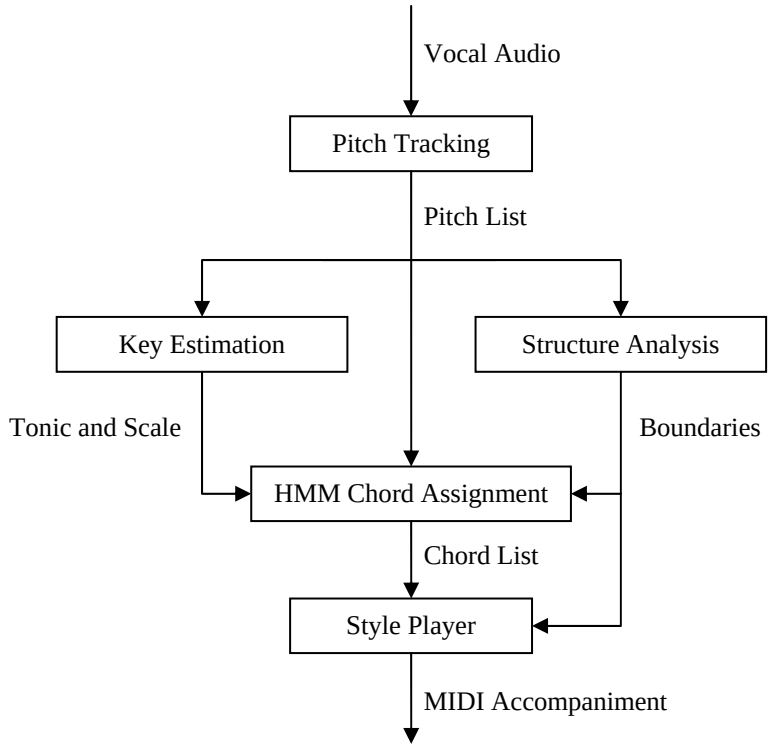

Figure 1 System Structure

Just like any pattern recognition algorithm, our analysis must rely on certain feature. The input of the system is vocal audio. It is very obvious that the most meaningful information for us in this audio track is the pitch. Therefore, a real-time pitch detection algorithm is applied to convert the vocal audio to a series of MIDI note number. We call this series of MIDI note number “pitch list”. Due to the nature of vocal audio, we

also introduce several post processing steps to eliminate the non-pitched noise and smooth the pitch transitions.

When a musician is going to play along with a vocalist, the first thing he wants to know is the key of this piece of music. In popular music, this key is usually in the mode of either major or minor. Therefore, a pitch class distribution based key estimation algorithm is provided to identify both the tonic and scale information of a given pitch list.

Music Structure is also a very important factor in popular music accompaniment. With accurate structure information, the automatic generated accompaniment can be more dynamic and organized. We developed a structure analysis algorithm based on melody similarity matrix. A section boundary list is obtained and passed to the chord assignment and style player subsystems in order to improve the quality of the generated accompaniment.

The core of the system is a chord assignment algorithm based on Hidden Markov Model (HMM). It is a statistical model which is capable of assigning a chord list to a pitch list by performing a decoding process. This harmonization process benefits from a bank of training data which are a number of vocal tracks with embedded chord labels. As a result, both melody-chord matching and chord progression problems are taken care by this model.

At last, a style player is implemented to play MIDI accompaniment template according to the chord progression list given by HMM. We found a way to render the standard automatic accompaniment style files used by Yamaha keyboard. In this way, users have a lot of different options to arrange the accompaniment and different melody sections are played with different accompaniment patterns.

1.4 System Integration

The proper design and arrangement of these subsystems which make the entire system fits best in our specific application are really what we pursue. Therefore, several interesting points of system integration are discussed below.

Different from other similar systems [3], the work flow we designed put the key estimation before chord assignment. This arrangement is very helpful to improve the chord assignment results. If there is no key information, the number of chords in the HMM could be very big, because the model has to take care all the possibilities. But this large number of chords makes the system easier to go wrong. To improve the accuracy of the automatic accompaniment, the key of the melody input must be estimated first. With a correct key, the generated chord list is less probable to be too far “away” from the melody. This arrangement also reduces the number of chords in our statistical model, so the HMM decoding process is faster.

Another interesting point in system integration is the structure analysis part. By introducing this subsystem, chord assignment and style player subsystems benefit from the detected structure boundary information. These boundaries separate the pitch list into sections. Then the chord assignment algorithm is applied to each section instead of the entire pitch list. In this way, the generated chord list is more organized. Moreover, the structure information is also helpful for the style player to change the accompaniment patterns at the boundaries, so users won’t get bored with a static accompaniment.

To clarify the underlying principle of these subsystems, all the detail will be discussed in the following chapters. The key estimation and structure analysis methods we designed are evaluated at the end of the respective chapters. Then, the prototype application will be shown. Additionally, another tentative mode of this application will also be discussed to demonstrate its interactive capability. Ultimately, a conclusion is made and user feedbacks are discussed in the last chapter.

CHAPTER 2

PITCH TRACKING

A real-time pitch tracking algorithm based on Eric Larson and Ross Maddox's research [4] is described in this chapter. The detected pitch list is the raw feature used by other subsystems including key estimation, structure analysis and chord assignment.

2.1 Pitch Detection

Pitch is the fundamental repetition period of an audio signal. Regularly, pitch detection involves Fourier Transform [5] and Autocorrelation Functions [6]. But the method we used for this automatic accompaniment system is quite unique. By calculating the mode distance of the peaks in audio waveform, this algorithm could find the pitch of a monophonic audio segment quickly in the time-domain. This performance gain plays an important role in reducing the user waiting time which is helpful to improve the user experience on a mobile platform. Furthermore, the post processing steps as you will see in the next section are specially designed for vocal input.

The incoming vocal audio is divided to *N* frames first, where

$$N = \left\lceil \frac{\text{Total Number of Samples}}{\text{Window Size}} \right\rceil$$

The default window size is 1024. This means a pitch value is obtained for every 1024 samples. For each frame, the following steps are performed sequentially:

1. Low pass filtering: We know that our input will be vocal audio, so a low pass filter with cutoff frequency of 1000Hz is used to process the audio in order to remove noise and turbulence. This filtering is also effective to eliminate the non-pitched voice at the beginning of each word.

2. Searching maximums: This could be done by tracing the zero crossing of the waveform. But not all the peaks between two zero crossings are considered to be

maximums. Only the peaks which are bigger than $0.7 \times$ absolute max of the current frame are valid. This threshold can prevent some octave errors.

3. Calculate the mode distance: The distance between two maximums is the difference between their sample indices. For each valid maximum, the distances between itself and another three maximums after it are calculated and pushed into a distance set. These inter-peak distances are illustrated in Figure 2. After this has been done for each maximum, the distances in the distance set will be clustered to groups based on their values. That is to say the distances in the same group should have similar values. Finally, the mean distance of the distance group which has most number of distances is elected as the mode distance of the current frame. If there are several groups which have the same number of distance in it, then the group with smaller mean distance will be picked.

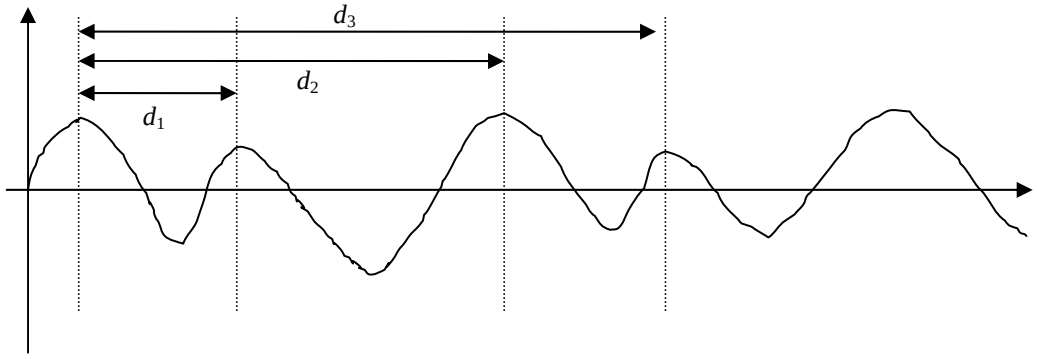

Figure 2 Inter-peak Distances for the First Maximum

4. Convert the mode distance to pitch frequency: The mode distance obtained in previous step is considered to be the fundamental repetition period in samples. That is to say:

$$\text{pitch frequency} = \frac{\text{sample rate}}{\text{mode distance}}$$

The pitch detection result could be confirmed by repeating the step 2 to 4 using minimums instead of maximums. If they are not identical to each other, then the pitch

from the previous frame should be referred. For more information about this basic procedure, please check out reference [4].

2.2 Post Processing

In order to optimize the results of pitch tracking, several post processing steps are introduced.

First, we want to remove the pitches where the corresponding audio level is very low, because these parts of audio are usually background noise. To achieve this, an amplitude threshold is determined as the global peak amplitude – 25dB for a given recording. That is to say, any frame which has average amplitude lower than this threshold will be ignored by pitch tracking algorithm. Instead, an invalid pitch symbol will be pushed into the pitch list. This threshold is also adaptive because it ensures pitch tracking's performance on different microphone level settings or for different users.

Second, in the interest of revealing the musical meaning of the pitch list, pitch values are converted to MIDI note number with formula:

$$midinote = 69 + 12 \times \log_2(\frac{f}{440})$$

Where *f* is the detected pitch frequency for a given audio frame. Please note that this *midinote* could be non-integer due to the nature of vocal.

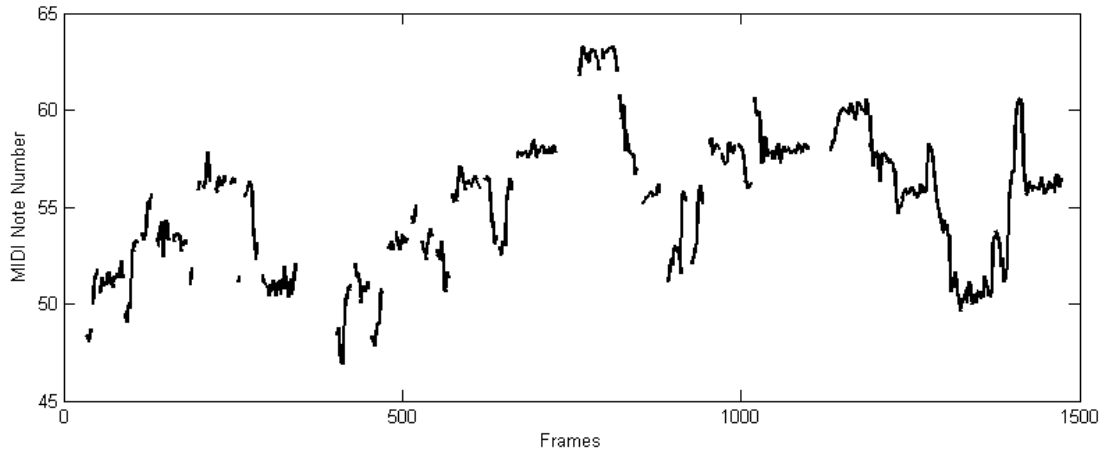

Figure 3 Pitch List

Figure 2 shows the pitch tracking result of a vocal audio track. All invalid pitches are not plotted. As you can see, unlike certain musical instruments, the pitch list of human vocal is very bumpy and rough. This is caused by singing techniques such as the pitch bend and vibrato. Instantaneous noises could also be introduced from inharmonic parts of the lyrics. These inharmonic parts are usually the beginnings of the lyric words.

To eliminate the attack noise and smooth the pitch list, third post processing is applied. For any detected pitch, if its value varies more than one midi note number to its left and right neighbors, then this pitch is considered as a noise point and its value is changed to its left neighbor. In this way, only stable pitches are maintained.

CHAPTER 3

KEY ESTIMATION

A key detection method based on pitch class distribution for monophonic vocal is developed in this chapter. About forty-five vocal tracks of popular music are collected to train the scale pitch class distributions and eighty-nine vocal tracks are used to evaluate the performance of this key estimation algorithm. With the correct key information of the input, the number of chords in the chord assignment statistical model is reduced and the overall accuracy of the automatic accompaniment can be improved.

3.1 Tonic and Scale

In western music theory, the term “key” is an abstract concept which can be challenge to describe thoroughly here. However, in our case, key defines the basic pitches for a piece of music. The music does not have to use the notes in this key exclusively, but the majority of the notes will come from this key [18]. A key can be further determined by two elements: tonic and scale. Tonic is the harmonic center of the pitch set and scale describes the intervals between these pitches. In the context of popular music, there are usually twelve possible tonics, they are C, C#, D, D#, E, F, F#, G, G#, A, A# and B. The scale is either major or minor for most of the time.

Thus, our task of key estimation is to find the most probable tonic- scale combination for a given pitch list. Our assumptions of the incoming pitch list are: 1) the key is constant in a given pitch list; 2) the scale is either major or minor. Given the assumption that the key should be constant, any accidentals in the pitch list of input melody should be considered as a potential key change in that part of music. Therefore, they are rounded to the nearest key note in the chord assignment subsystem.

However, before we start, there is one thing need to mention. Since our system allows users to sing or hum freely without asking them to align to the chromatic scale.

It’s quite possible that the tonic of melody could actually lie in between chromatic notes, for example C and C#. In order to address this issue, our key estimation should be able to produce the tonic result in a higher resolution. Thereby, we increase the number of possible tonic pitches from twelve to one hundred and twenty. That is to say, the key output is going to be in the form like “C Major +50 cents”.

3.2 The Method

First and foremost, major and minor scale pitch class distributions (PCD) [24] are constructed from training data. Forty-five vocal tracks of Chinese popular music are collected and processed by the pitch tracking algorithm. Each track is about one minute long. They are not sung by professional singers, so pitch inaccuracy of amateur users is taken into account in these distributions. The reason why we choose these vocal tracks of Chinese popular music instead of western popular music is that a recording studio in China provided these tracks to help this project. Although these tracks are singed in the language of Chinese, the melodies of the music we selected are totally in western style. That is to say, there is no significant difference in music theory between these Chinese popular music and western popular music. But the actual minor differences between them should be taken care by extending the training set or creating genre specific models in the future.

A scale PCD can be represented in a 120-bin histogram, the value in each bin indicates the percentage of the pitches which fall into that corresponding 10-cent range. The PCD of each vocal track from the training set is shifted to the tonic of C. This transformation could be done in the PCD by performing circular shift on all the bins. Because we do not care about octave information of pitches in key estimation, all the MIDI note numbers in the pitch list are folded into an octave. Similarly, histogram based methods for music key identification have been used widely in content-based music as well [7].

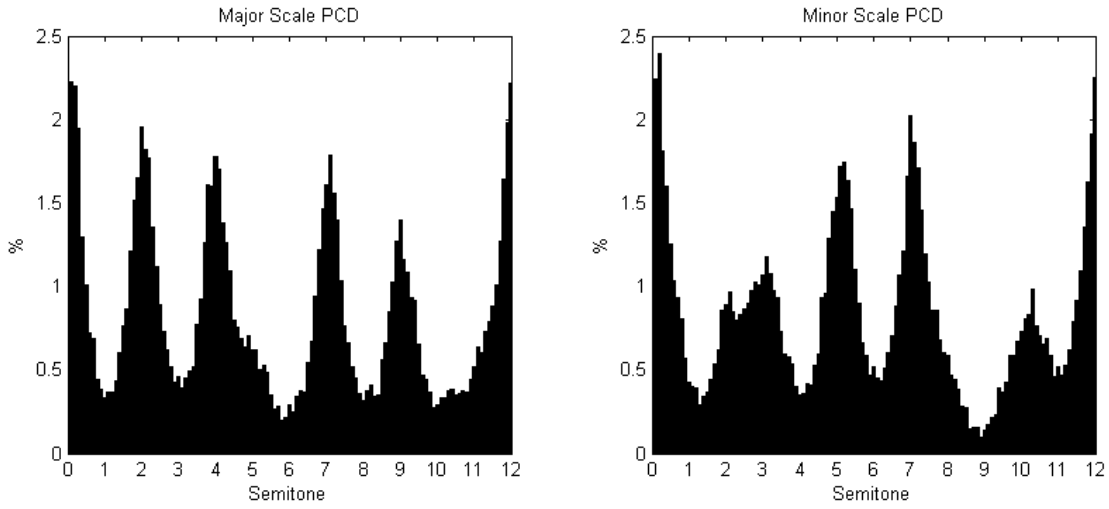

Figure 4 Major and Minor Scale PCDs

Figure 3 shows the major and minor scale pitch class distributions learnt from the training data. The X axis means how far this bin is from the tonic. Actually, each bin is 10 cents in range. As you can see, peaks in these scale PCDs reveal the interval patterns between notes. For major scale, the interval patterns are whole, whole, half, whole, whole, whole and half. For minor scale, they are whole, half, whole, whole, half, whole and whole. Some of the notes in scale are frequently used and some of them are rarely used.

When we want to identify the key information of a new pitch list, a similar pitch class distribution is constructed for this target pitch list. Then a cross correlation function $ccf(\mathrm{t})$ is derived for t = 0, 1, 2, 3 …… 119:

$$ccf(\mathrm{t}) = \sum_{\mathrm{i}=1}^{120} scalepcd(\mathrm{i}) * pcd((\mathrm{i} + \mathrm{t}) \bmod 120)$$

Where *pcd* is the pitch class distribution of the target pitch list and *scalepcd* is either the major or minor pitch class distribution we learnt from training data.

Finally, the offset which makes the largest value in the cross correlation function is considered to be the most probable tonic of the target pitch list. Written formally, it is:

$$T = \underset{t}{\operatorname{argmax}}\, ccf(t)$$

T is the tonic detection result in the dimension of 10-cent. That means the estimated tonic is $\frac{t}{10}$ semitones away from C. To identify the scale of the target pitch list, we just need to calculate two *ccf*(*t*) with both major and minor *scalepcd* and pick the tonic which makes a bigger *ccf* value.

3.3 Evaluation

In order to evaluate the performance of our key estimation method, another eighty-nine vocal tracks are collected. Similarly, they are also amateur vocal tracks of Chinese popular music and each of them is about one minute in length. They are manually labeled with key information in a text file and a program is developed to perform this evaluation automatically. Tonic precision tolerance is 50 cents which means a tonic will be treat as correct if it is 50 cents less from the ground truth. The reason why we set this relatively large tolerance is that test data set is not labeled with 10 cents precision. They are only labeled with the chromatic tonic according to the original music scores. But singers might deviate from this original tonic a little bit, so this evaluation is done with 50-cent tolerance. All results are shown in Table 1.

Table 1 Key Estimation Accuracy

Key	Tonic	Scale	Relative Key
84.3%	85.4%	88.8%	91.0%

As you can see, four different judging criteria have been set to evaluate the performance. For “key” accuracy, it means both tonic and scale have to be correct, while “tonic” accuracy only requires the tonic to be in the tolerant range. Similarly, “scale” accuracy only asks for major or minor correctness. Specially, “relative key” accuracy forgives the error cases like a major key is recognized as its relative minor or minor key is identified as its relative major. The reason why it’s hard to tell this kind of error is that relative major and minor keys share the same key signature. That is to say, the pitch class

distributions of relative keys show similar peaks. The relative key error could be reduced by introducing Pitch Class Dyad Distribution (PCDD) which is a bi-gram distribution that measures the transition intervals between notes [19]. However, this improvement requires knowing the onsets of each musical note which are not available in our pitch list.

In our automatic accompaniment system, the key information plays an important part in the HMM chord assignment subsystem. Tonic value will be used to shift the pitch list back to a standard tonic in order to match a chord. The choice of scale is also critical because two independent statistical models are built to simulate different chord progressions in major and minor music. However, a small portion of popular music is ambiguous in major and minor scales. For example, a piece of music could start in a minor scale and end in major scale to express different feelings. In this case, relative key errors are acceptable.

CHAPTER 4

STRUCTURE ANALYSIS

In this chapter, a melody structure analysis method is discussed. The structure boundaries of a melody are passed to style player subsystem to improve the dynamics of the generated accompaniment. In addition, structure information is also helpful for the chord assignment subsystem to produce more organized chord sequence.

4.1 The Role of Structure Information in Automatic Accompaniment

One major problem of the previous automatic accompaniment systems is that the generated accompaniment always has the same pattern or arrangement no matter how melody line varies. By listening to modern popular music, a piece is usually divided into different sections such as intro, verse, bridge, chorus and ending. For different sections, different instrumentations, styles or rhythms are used. Moreover, from one section to another, there is a transition effect applied. Fortunately, as you will see, the accompaniment style files which are used in style player subsystem contain different variations, intros, endings and transition effects. They can be applied to a certain part of the chord list.

Furthermore, another problem of the previous automatic accompaniment systems is that the generated chord list does not reflect the melody structure. For example, composers have a tendency to use similar chord progressions for similar melody sections. Besides this, a chord progression for a section usually starts from tonic chord and ends on tonic chord or fifth chord. This chord progression similarity and regularity lead to a listening pleasure. That is to say, popular music listeners are happy to see these underlying rules of accompaniments. However, the chord progressions generated by previous systems do not have a sense of these rules. You will notice that these systems still keep developing a chord progression instead of going to a conclusion when the

melody is repeated. In order to make the automatic accompaniment sounds more organized, we need to figure out a way to analyze the melody structure and then apply our chord assignment algorithm on each section of the melody separately.

From the perspective of music psychology, it has been shown that predictability evokes pleasure and predictable stimuli lead to positively valenced responses [20]. As one of the most important predictable elements in popular music, we need to put emphasis on the repetitions of the input melody. Therefore, based on the structure boundaries, transition effects are added between repetitions and chord list is reorganized.

Unfortunately, unlike other methods such as content based structure analysis [8] or symbol based structure analysis [9], we face a unique problem of detecting structure boundaries of a vocal audio track. A vocal audio track does not produce a note list as accurate as symbolic data such as MIDI. It also does not provide a timbre diversity as rich as music content data such as CD audio.

To find the boundaries between sections of vocal melodies, we start with a measure to measure pair wise similarity matrix. Then, an adaptive threshold is applied to the matrix and boundaries of repeated measures are found. Multiple restrictions are also developed to refine the repetitions. Finally, structure boundaries are implied from these repetitions.

4.2 Similarity Matrix

Because our automatic accompaniment system requires user to record vocal in a fixed tempo, we assume that section changes in melodies only happen between measures. Under this circumstance, a $N \times N$self similarity matrix D is computed based on the pitch list of the target melody.

To compute this similarity matrix, a distance function has to be defined. For content based structure analysis, cosine distance of MFCC is usually used [21]. But it's not very reasonable extract MFCC from vocal audio, because there are no major timbre

differences between our vocal segments. Another cosine distance function between pitch class distributions has also been considered. But it does not work very well in separating different melodies, because similar PCDs may come from very different melodies. In our case, each element in the matrix is a distance between measures. This distance is defined as:

$$d_{i,j} = \frac{1}{N}\sum_{n=0}^{N}|p(i,n) - p(j,n)|$$

Where $d_{i,j}$ is the distance between measure i and measure j, $p(i,n)$ is the nth pitch value in measure i, and N is the number of pitches in a measure of this melody. Specially, if an invalid pitch is compared with a valid pitch, a constant distance of 1 is added for that element. Figure 4 shows a similarity matrix plotted as a gray scale image.

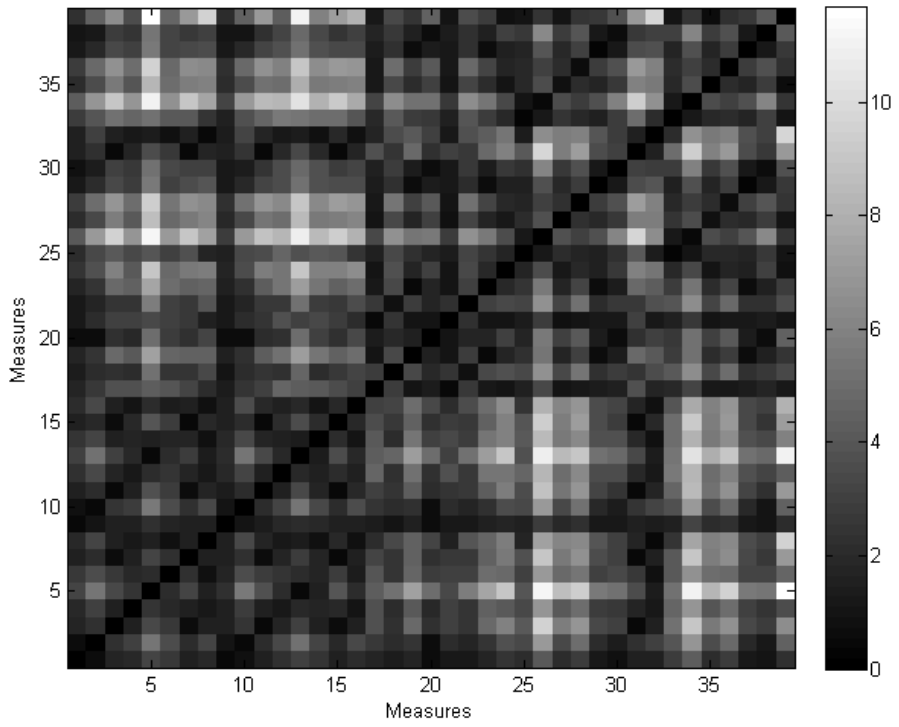

Figure 5 Similarity Matrix

In Figure 4, a pixel is darker if the two corresponding measures are more similar with each other. As you can see, the diagonal of this symmetric matrix is black because

the distance between a measure and itself is always zero. To make repetitions easier to see, a time-lag matrix [8] *D'* is introduced:

$$d'_{i,j} = d_{i,i+j}$$

Basically, this equation transposes the possible repetition measures from diagonal lines in similarity matrix to horizontal lines in time-lag matrix. Since there is no meaning if the sum of time and lag exceeds the number of measures, only the bottom-left half of the time-lag matrix is calculated. This is shown in Figure 5.

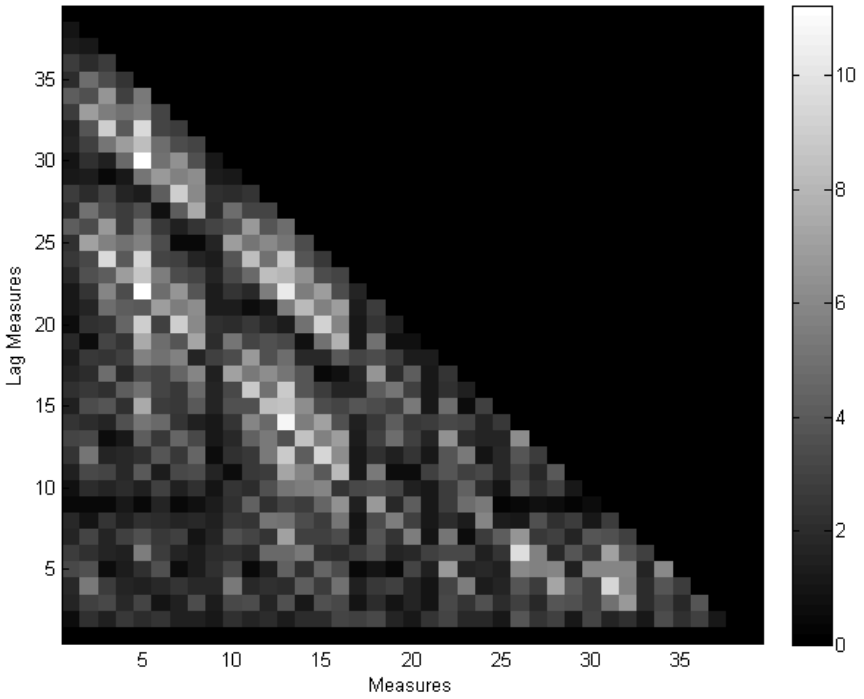

Figure 6 Time-lag Matrix

The next task is to find out all significant horizontal lines in the time-lag matrix. In other words, if an element in the matrix is small enough, it should be considered as a significant similarity. To achieve this goal, an adaptive threshold is calculated to filter the matrix.

4.3 Adaptive Threshold

Although, there are some existing algorithms of threshold selections [10] in the field of image processing, our case is different from them in terms of picture dimensions. We need to figure out a way to emphasize the horizontal lines which represent melody repetitions. Our adaptive threshold algorithm is described as the following steps:

First, we assume that any repetition must be at least four measures long. That is to say the first four rows in the time-lag matrix can be removed. If we allowed repetitions that are less than four measures long to be detected, the music generated by our style player would keep jumping from one pattern to another. This is quite disturbing.

Second, the remaining elements are treated as potential thresholds, because only these thresholds could produce a different filtered matrix. Then each of these thresholds is applied to the matrix, the mean of the remaining elements are calculated. Only threshold values which are able to produce means lower than 0.7 are reserved. This empirical mean range in obtained by looking at the distances between identical melody measures. It means the average deviation between two pitch lists must be less than 0.7 MIDI note number to be considered as a threshold candidate.

Finally, for each threshold found in the last step, a searching algorithm is performed to find all the horizontal lines on the matrix filtered by a certain threshold. Then, the threshold which produces the longest horizontal line is picked as the global optimal threshold.

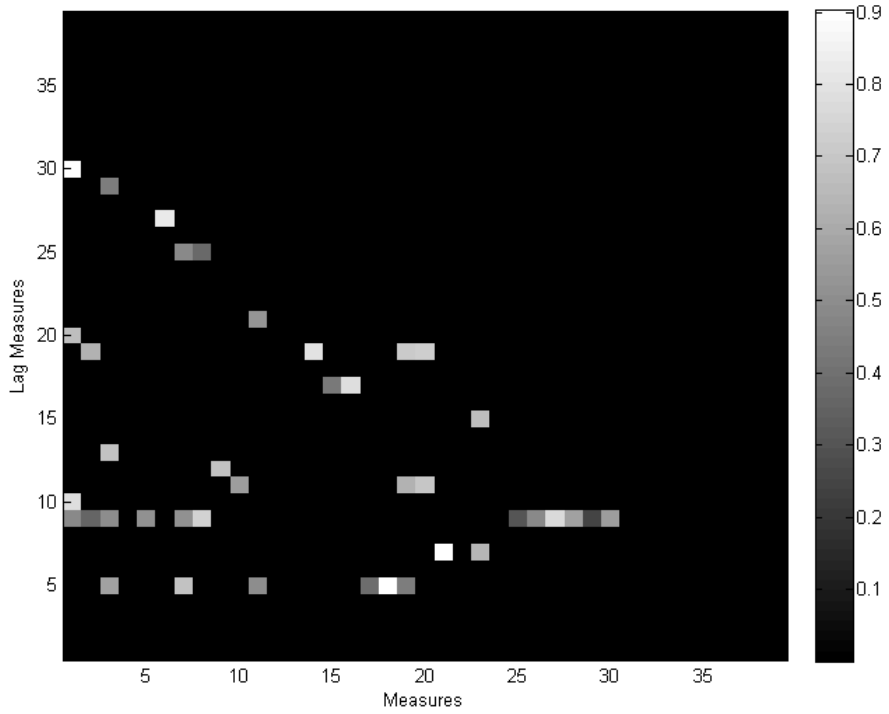

Figure 7 Filtered Time-lag Matrix

Figure 6 shows the time-lag matrix of Figure 5 filtered by the adaptive threshold. Please note that all pure black pixels are emptied by the threshold.

To emphasize the high level repetitions, several rules are applied to the filtered time-lag matrix: First, horizontal lines in the filtered matrix whose length is less than three are ignored. Second, if there are multiple horizontally overlapped lines, only the longest is reserved. The final time-lag matrix looks like Figure 7.

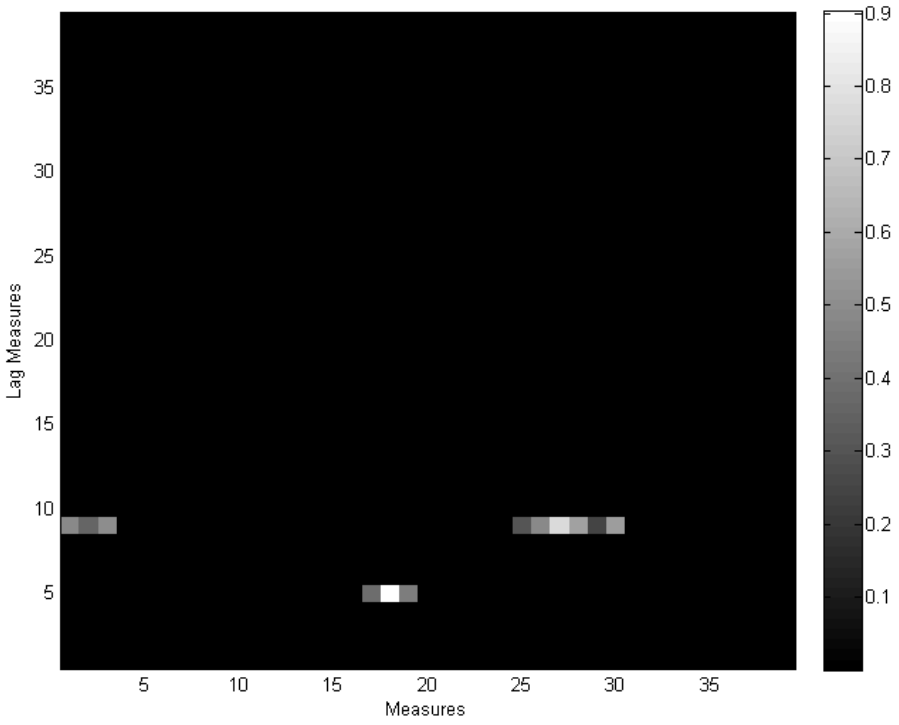

Figure 8 High Level Repetitions

4.4 Structure Boundaries

Because popular music often intends to vary its melody at the end of a repetition, only the beginning measure of a repetition line in time-lag matrix can be recognized as a boundary.

Besides repetition boundaries, boundaries between different sections can be implied from the repetitions inside a section. For example, in Figure 8, the beginning of chorus 2 implied the position of section change from verse to chorus.

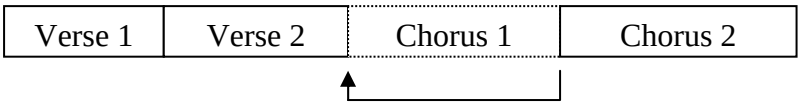

Figure 9 Section Change implied by Repetition

Mathematically speaking, given coordinate set of the starting points of each horizontal line in the filtered time-lag matrix:

$$C = \{(x_i, y_j)\} \quad i, j = 1, 2, 3, \ldots, N$$

Where N is the number of horizontal lines found in the matrix. For each coordinate, measure indices i *and* $i+j$ are pushed into the structure boundaries. At last, duplicated boundaries and boundaries which are too close (less than four measure away) are removed.

All the detected boundaries including repetition boundaries and structure boundaries are treated as section changes by the style player. In this way, different sections are played with different MIDI patterns which are available in the accompaniment style files.

4.5 Evaluation

Just like what we did in the key estimation, an evaluation is conducted in order to test the performance of this algorithm. The test data is twenty vocal tracks taken from training data of the key estimation subsystem. Similarly, they are Chinese popular music singed by amateur singers. We manually trimmed these tracks and estimated the tempo in beats per minute, so they are aligned with the dimension of measures. Then both structure boundaries (not including start and end) and repetition boundaries are labeled manually as the ground truth for each track. In this test set, there are altogether fifty-four boundaries. Recall that the fundamental element of our structure analysis is measure. Therefore the boundary result is just an array of measure numbers.

The raw pitch list with the tempo information is passed into the structure analysis subsystem. Because the algorithm only wants to know the shape of the pitch list, no key rounding and octave folding are performed on this pitch list. Next, the detected boundaries are compared with ground truth automatically.

In this evaluation, F-measure is used to reflect both false positive and false negative errors. It is defined as the following:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

In our case, precision and recall are:

$$\text{precision} = \frac{\text{the number of correct boundaries detected}}{\text{the number of all boundaries detected}}$$

$$\text{recall} = \frac{\text{the number of correct boundaries detected}}{\text{the number of boundaries in ground truth}}.$$

The test results are shown in Table 2.

Table 2 Structure Analysis Test Results

Precision	**Recall**	**F-measure**
85.7%	55.6%	67.3%

Obviously, the limitation of this structure analysis method is that it is unable to find the point of section change if there are no melody repetitions in these sections. Since incomplete structure boundaries won't cause troubles in the chord assignment subsystem, we can accept a low recall rate. However, we do want to avoid the situation that many false positives are detected because these false boundaries break the melody into small chunks. If this happened, the structure information would ruin the user experience by asking style player to change accompaniment patterns at wrong time. This explains why we have a high precision rate but a low recall rate in Table 2.

CHAPTER 5

CHORD ASSIGNMENT

In our automatic accompaniment system, appropriate chords need to be assigned to each section of the melody. That is to say, for a given pitch list from pitch tracking subsystem, a chord list should be provided. Through this chapter, a Hidden Markov Model is developed to do this job. Two independent models are trained for major and minor scales based on the knowledge from a set of training data. With the information of structure boundaries and melody key, a HMM decoding process is performed on each melody section. As a result, the optimal chord sequence is obtained under chord progression constrains for the given melody observations.

5.1 The Problem of Harmonization

Given a measure of melody, there are usually several appropriate chord options for harmonization. For example, a measure of melody contains only note C, what are the possible chords for it? The answer could be C Major, F Major, A minor, etc. There is no best answer for this question. Exaggeratedly speaking, all chords are possible for this measure of melody, because different chord-melody combination provides different tension and feeling. Hence it's very hard to decide which chord is the best option without considering the context. In fact, there are some typical chord progressions widely used in popular music [11] which build up a common sequence of tensions and feelings. However, these progressions won't work as expected without considering the corresponding melody, because the harmonic feeling of a measure relies on the combination of the chord and the melody. This leads to the conclusion that chord assignments should take both melody-chord relationship and chord progression into consideration.

Mathematically speaking, melody harmonization problem is a two dimensional stochastic process. That is, for single measure of melody, the choice of chord is constrained by melody and progression preferences. Fortunately, there is a statistical model which is designed to describe this kind of process. It is the Hidden Markov Model (HMM). HMM can be formally defined as a quintuple:

$$(\Omega_q, \Omega_o, A, B, \Pi)$$

In this definition, $\Omega_q = \{q_1, q_2, \dots, q_N\}$ is a finite set of all the states, $\Omega_o = \{o_1, o_2, \dots, o_T\}$ is a finite set of all the observations. $A = \{a_{i,j}\}$ is the transition probability matrix of the N states, that is to say $a_{i,j} = \mathrm{p}(q_j|q_i)$. $B = \{b_{j,k}\}$ is the output or emission probability matrix of an observation *k* given a current state *j*, that is $b_{j,k} = \mathrm{p}(o_k|q_j)$. Finally, $\Pi = \{\pi_i\}$ are the initial probabilities of all the states. For more information about HMM, please refer to [12].

In the context of our chord assignment algorithm, the HMM states are all the chords that could be used for accompaniment, the observations are all the measures of the input melody, the transition probability matrix stands for the chord progression probabilities and the output probability is the conditional probability of a measure of melody given a certain chord accompaniment.

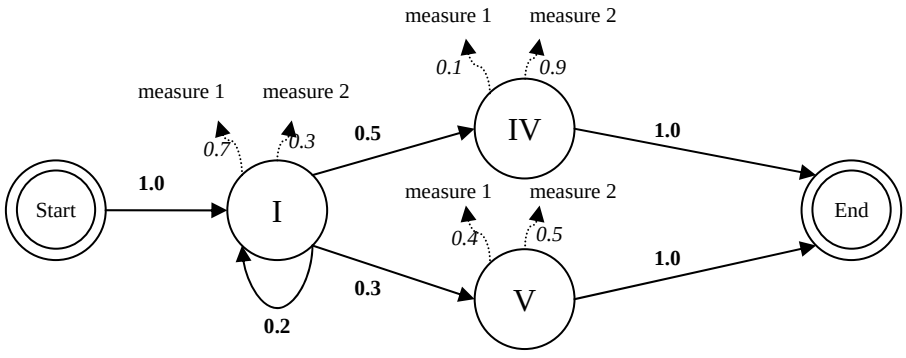

Figure 10 Graphical Representation of Hidden Markov Model

Figure 9 is an informal example of our chord assignment HMM. As you can see, this HMM can be represented in a directional graph, where each node is a chord state and

each solid arc is a probability that chord sequence could move from one to another. On each chord node, the dash arcs stand for output probabilities which indicate how probable this chord matches with every melody measures.

However, before trying to solve this statistical model, we have to determine its parameters. These parameters including state set, observation set, transition probabilities and output probabilities will be discussed in the following sections.

5.2 Chord Set and Transition Probabilities

Before chord assignment, we have already got the tonic and scale information about the input melody from the key estimation subsystem. In order to describe different characteristic of major and minor music accompaniment and simplify the chord transitions, two independent chord assignment HMMs are built. With the help of the tonic information, each model has a chord set of all the triads relative to the tonic and in the scale. Written in degree scale [11], they are "I-ii-iii-IV-V-vi-vii^{o}" for major scale and "i ii^{o} III iv v VI VII" for minor scale. Taking start state and end state into account, there are altogether nine states for each HMM.

To train the transition probability matrix of these states for both major and minor scale, the chord progressions of seventy popular songs are manually written into a text file. Some of them are in major scale and some of them are in minor scale. All chords are annotated in degree scale, so the progressions become tonic independent. Then, a training program will calculate out two 9×9 transition probability matrices by parsing this text file.

Please note we do not need the audio of seventy popular songs to start the training, all chord progressions are just written as sequences of numbers into a text file. The chord progressions can be easily obtained by referring to their performance scores. Thus, this training set does not only include Chinese popular music, but also some western popular music.

5.3 Output Probabilities

Next, the output probabilities are obtained by comparing the average melody pitch class distribution of a given chord with the target melody pitch class distribution.

In detail, fifty vocal tracks of Chinese popular songs are labeled with chord names to corresponding segments. For each chord, the average melody PCD is computed to show how melody pitches are usually distributed given the fact that a certain chord is used by musicians to harmonize them. We call these average melody distributions “chord PCDs” because they reflect the original melody-chord relationship of the training data. Additionally, in order to make this average PCD tonic independent, the pitch list of every training vocal track is shifted back to tonic of C according to its labeled key information.

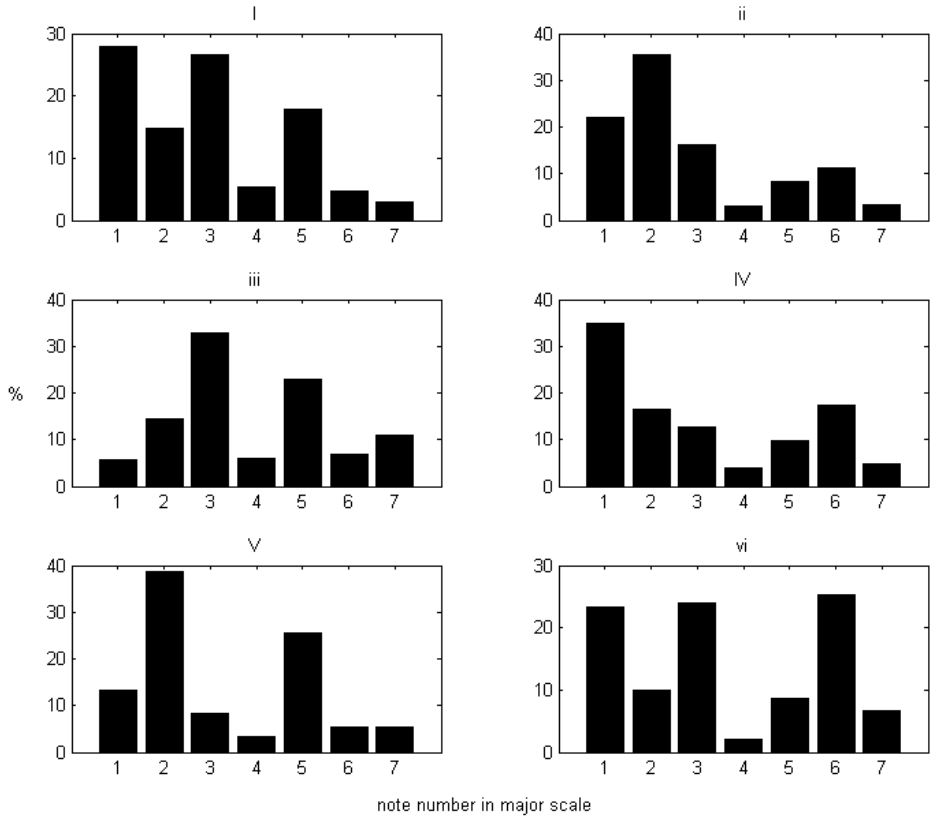

Figure 11 PCDs of Major Scale Chord Set

Figure 10 is the histogram representations of most of the major chord PCDs. You may notice that the X axis is the note number in major scale instead of chromatic scale.

This is because the pitch list is rounded to the key before processing. This approximation is helpful to eliminate the inaccurate components in vocal pitches and enhance the PCD distance measure performance.

Then, the output probability of a melody measure given a chord state is estimated by measuring the cosine distance between the chord PCD *A* and target measure PCD *B*:

$$d = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where $A \cdot B$ is the dot product of these two PCD vectors and $\|A\|\|B\|$ is the product of the vector magnitudes. The advantage of this measurement is that the cosine distance is normalized into [0, 1], so it's more like a probability. Do not forget that PCD *B* is also shifted to the tonic of C before comparison.

5.4 HMM Decoding

After the training stage, all parameters of the models ready, it's the time to ask the question: What is the most probable chord progression given a series of melody observations? Fortunately, this question is exactly same as the HMM decoding problem [12] which can be described formally as the following:

For a given HMM $\lambda = (A, B, \Pi)$ and a observation list $\Omega_o = \{o_1, o_2, \dots, o_T\}$, what is the state sequence $\Omega_q = \{q_1, q_2, \dots, q_T\}$ that could maximize the probability $p(\Omega_o | \lambda)$.

The standard solution for this problem is the Viterbi algorithm [13]. The Viterbi algorithm chooses the best state sequence that maximizes the likelihood of the state sequence for the given observation sequence.

Basically, the Viterbi algorithm constructs $\delta_t(i)$, which is the maximum probability of state sequence that ends up on state *i* for the first *t* observations. It is defined recursively as:

$$\delta_t(j) = \max\left[\delta_{t-1}(i)\, a_{i,j}\right] b_{j,t}$$

At the same time, another matrix $\psi_t(j) = \text{argmax}_i\left[\delta_{t-1}(i)\, a_{i,j}\right]$ is constructed to remember the decisions made along the path. Finally, the optimal path is back traced into q_t:

$$q_T = \underset{i}{\text{argmax}}[\delta_T(i)]$$

$$q_t = \psi_{t+1}(q_{t+1})\ ,\ t = T\text{-}1,\ T\text{-}2,\ \dots\ ,\ 1$$

And the total probability of the optimal path or called Viterbi path is $\max[\delta_T(i)]$. One thing need to mention about the implementation of the Viterbi algorithm is that instead of using probability in the computation, we used logarithm probability. The logarithm probability convert the range of probability from [0, 1] into $(-\infty, 0]$ to avoid the denormal situation [14] of floating numbers. Furthermore, logarithm probability also turns the multiplication operation between probabilities into addition, which is much faster.

As shown in Figure 9, the Hidden Markov Model we designed involves an end state. That is to say, any path found by Viterbi algorithm must end at the end state. To make sure the chord progression ends properly, a pseudo observation is added to the end of the Ω_o. This special observation has an output probability of one on the end state but it has an output probability of zero on the other states. Additionally, the transition probably matrix also described how probable a chord could be the ending. In this way, we force the Viterbi path to be ended with the end state.

As a summary, let's review the process of chord assignment. First, the incoming pitch list is divided into measures. Each measure is an observation. Second, with the help of structure boundaries obtained in the last chapter, this observation list is further divided into groups. Third, depending on which scale is the melody in, the corresponding HMM is used to find out the most probable chord list for each observation group. Finally, these chord lists are connected together as the chord assignment result.

CHAPTER 6

STYLE PLAYER

Up to now, the chord list has already been generated for a given pitch list. However, a sequence of chord is still not the music accompaniment. The next problem is how to realize or play this chord list, so the actual sound can be heard. In this chapter, a style player is developed to play standard MIDI accompaniment style files. This player follows the generated chord list and applies transition effects during section changes.

6.1 Automatic Accompaniment Style Files

For a long time, automatic accompaniment has already become a standard feature of some professional arranger workstations [15].This feature is favored by a lot of live performers, because one man is able to play the role of an entire band with the help of this feature. But this automatic accompaniment feature requires users to play the current chord on the keyboard and then the background music will change its harmonic components to this chord. Although the way of chord input is different from our automatic accompaniment system, the music realization part is highly valuable for reference.

The automatic accompaniment feature in keyboards relies on style files. A style files is actually a format 0 MIDI file [16] with special markers inside. The most popular automatic accompaniment style file format is designed by Yamaha Corporation for their professional keyboard products. Although this file format is not officially published, there are some third party documentations [17] on the basic architecture of the Yamaha style files. Based on these documents, we implemented a style player which is capable of playing these files in the specific section and chord. Although it’s feasible to meet this style file standard, we still need to design our own file content if this system is going to be published.

6.2 Structural Sections

One of the advantages to use these standard accompaniment style files is that they provide rich accompaniment variations inside. Instead of playing the same pattern over and over again, several different sections are available in the style file. These sections are separated by MIDI markers, so our style player can jump to any sections and start a loop playback in a certain section. Table 3 shows all possible sections in a style file.

Table 3 All Possible Sections in a Style file

Intro	**Main**	**Fill In**	**Ending**
Intro A	Main A	Fill In AA	Ending A
Intro B	Main B	Fill In BB	Ending B
Intro C	Main C	Fill In CC	Ending C
	Main D	Fill In DD	
		Fill In BA	

In the context of our automatic accompaniment system, an intro section will be played first to give the user a clue to start singing, then it is followed by section main A. Once a structure boundary is detected, style player will jump to the next main section, which is main B for the first time. If more than four structure boundaries are detected, the style player will loop back to section main A. Specially, a corresponding fill in section will be added to smooth the transitions before the main section change. At the end, an ending section will be played with last measure of input melody.

6.3 Instrumentation and Note Transposition

As we mentioned before, automatic accompaniment style is essentially a MIDI file. In this MIDI file, channels from 9 to 16 are used for accompaniment output. Details are shown in Table 4.

Table 4 Style File Channel Arrangement

Channel	Usage	Instrument
9	Sub Rhythm	Secondary percussions
10	Main Rhythm	Standard Drum Set
11	Bass	All kinds of bass instruments
12	Chord 1	Rhythm guitar
13	Chord 2	Piano
14	Pad	Strings or organs
15	Phrase 1	Monophonic instrument
16	Phrase 2	Monophonic instrument

Note messages in all the channels except rhythm channels are written in the CMaj7 chord as default. When the desired chord is different than this default one, transpositions need to be done. There are two different note transposition rules available for different accompaniment parts. For melodic parts such as Phrase 1 and Phrase 2, root transposition is usually used. This rule requires pitch relationship between notes to be maintained during transpositions. For chord parts, root fixed rule is often applied. This means the original CMaj7 notes should be moved to the nearest slots of the desired key, that is to say different inversion of the chord may be used. Other complicated note transposition rules are defined as a matrix at the end of the MIDI file. These note transposition matrices are usually used for special instruments like guitar.

Another advantage of these MIDI style files is that all the notes in the accompaniment can be bent to the key of vocal. Recall that the key estimation subsystem is able to provide key information in the resolution of 10 cents. Thus, by sending pitch bend messages to the MIDI accompaniment channels, the accompaniment could adjust itself to the input melody.

Nevertheless, some other points of interest in orchestrating the chord list are not dealt with here. For example, issues like voice leading and chord transition are also

important factors of an accompaniment. These inter-measure constrains could be taken care in the chord assignment module in the future.

CHAPTER 7

APPLICATION AND CONCLUSION

All components of the automatic accompaniment system have been covered in previous chapters. To demonstrate the usage of this system, a Windows application is built as a system prototype. It allows users to record vocal input in a customizable tempo and then listen to the generated accompaniment in different styles. Ultimately, the future and tentative works about this system is discussed and a conclusion is made for the entire thesis.

7.1 Applications

The automatic accompaniment system including its all components is implemented as a Windows prototype application written in C++ with Microsoft Foundation Class library.

Figure 12 GUI of the Prototype Application

Figure 11 shows the graphical user interface of the prototype application. As illustrated in this screen shot, user can tap the tempo button four times to start the recording at a desired tempo. After the intro part, the user is supposed sing or hum with

the rhythm track of the style file until the stop button is pressed. During this process, a level meter on the right will display microphone input level and the audio is recorded as a mono WAV file at sample rate of 44.1 kHz. Once the stop is pressed, the recorded vocal audio is then passed to the pitch track subsystem for feature extraction. This feature list is used by key estimation, structure analysis and chord assignment subsystems. Usually, the entire analysis and generating process only takes about one second for an input vocal audio of one minute. Finally, the estimated key information and the generated chord list are displayed in the result box. Users can hit the play button to listen to the accompaniment playing along with vocal melody. The melody and accompaniment can also be saved as a WAV file for later recall.

Another more interactive application is also developed to provide continuous experience. In this application, users choose the desired key and tempo first. Then the system will play an intro part with tonic chords inside to give users a sense of the key. Next eight measures, only percussion part is played and users are supposed to sing any melody in mind. At the last beat of these eight measures, chord assignment algorithm will be performed on the previous recorded vocal. For the next eight measures, full accompaniment music will be played with the chord list generated for the first eight measures. At this time, users can sing any melody lines that fit in the same chord progression. In this way, interactions between human and machines are repeated until the stop button is pressed. This idea is inspired by the fact that some popular music shares chord progressions between different parts of the melody. Hence, with the knowledge of the accompaniment of a melody section, people can sing or improvise on it with different melodies. The user experience is boosted by the illusion that the automatic accompaniment is working in real-time. Furthermore, the role of computer here has changed a little bit. Instead of following the order of user input, the generated accompaniment asks users to follow the computer now.

7.2 Evaluation

Unfortunately, unlike the key estimation and structure analysis, chord assignment has no ground truth. In other words, the generated accompaniment cannot be evaluated objectively. In real life, different musicians may assign different chord progressions to a same melody line due to variety of esthetic, practical and technical considerations. But the goal of this automatic accompaniment system is to provide an experience of the popular music accompaniment which most of the non-musicians are familiar with. That's also the reason why statistical models are used here to reflect the harmonization techniques of vast majority.

In this situation, an informal subjective user study is conducted. The prototype application is distributed to ten college students who do not play any musical instruments. The reason why we choose non-musicians is that we want to avoid their professional background knowledge push the evaluations deep into music theory. Instead, we just want to find out what are the obstacles are strength of this system for its overall music entertainment proposes. They are asked to try this software and answer several questions we designed. For the overall appeal of this system, all the participants showed their surprise to the intelligent responses of a computer. For the question about the best part of this system, most of the people said they are satisfied with the capability that the system allows singing in arbitrary keys. This demonstrated the strength of our key estimation subsystem. The problem that the people could sing in a key between chromatic notes is solved successfully by the 10-cent resolution in key estimation and pitch bend in style player. For the question about the weakness of this system, two major points are highlighted. One is the generated chord progression could fall into a loop such as always "I V I V …". After some investigations, we found this issue is caused by the biased transition probability. That is to say, more training data is needed to show the general trends of chord transitions. Another issue is that some participants showed that they are unable to follow the rhythm track properly when recording. As a result, the generated

accompaniment does not sound synchronized with the melody. This is a limitation of the current work which does not involve any tempo or beat detection. This is a good reminder for us to pay more attention to the difference between experienced and novice users.

Furthermore, a case study is done on a specific melody input to compare the accompaniment results of our system and “songsmith” which is a commercialized version of the system mentioned in [3]. This melody is a thirty three seconds recording of a male vocal humming a Chinese popular music in the tempo of one hundred and twenty beats per minute. It is recorded in system A with the built-in microphone of a laptop computer. Then, the vocal audio is exported from system A to our system. The original recording and generated results from both systems can be found in the following URL:

http://www.maizesoft.cn/casestudy.zip

The chord sequence generated by our system is “I vi IV V | I vi IV V I vi IV V vi IV V I” in a key of “G# major 40 cents”, where “|” is a repetition boundary. On the other hand, the chord progression generated by “songsmith” is “D#m G#m C# F# D# G#m C# F# B C# F# G#m C# F# B F#”. By listening to the generated accompaniment with the melody, one can easily find that the later one is out of tune. Although the sound quality of our accompaniment is worse than songsmith’s, the appropriate pattern transition during the section change made by our style player is really helpful to evoke listener’s pleasure.

7.3 Future Works

There are a few limitations of this automatic accompaniment system we would like to discuss here and tentative solutions for these problems are suggested as well.

First, the key estimation subsystem assumes that the key of the melody is constant. However, some of the popular music involves key changes. This is called “modulation” in music theory. One of the situations is that these key changes happen on the section boundaries, for example, when chorus section is repeated for another time. Therefore, a

possible solution is that we should apply key estimations on all melody sections respectively. Then key changes could be detected and the corresponding accompaniment can be adjusted to the new key. Another situation is that user changes the key randomly during a melody singing, this phenomenon is typical for amateur singers. To deal with this problem, a dynamic key estimation algorithm has to be developed. This dynamic algorithm must be capable of detecting possible key changes by analyzing the average pitch deviation from the assumed key.

Second, the current prototype application has to ask for a tempo before recording because the structure analysis and chord assignment need this temporal information. But this extra step is not very user friendly as users usually do not know the tempo before start singing. One tentative solution is to design a tempo or beat detection algorithm for vocal input. If the detection result is acceptable, this new component can be added to the automatic accompaniment system after pitch tracking.

Third, the chord set used in chord assignment subsystem is limited for model simplification. Obviously, only triads are not enough for some popular music accompaniment. Modern popular music also would like to use a lot of other chords such 7th chord and 9th chord. In the future, this chord set should be expanded to provide richer emotion and tension. To do this, the dimension of the chord PCDs may need to be expanded to chromatic scales in order to reflect the pitch components which are not in key.

Forth, the sound quality of the generated accompaniment is not as good as other similar systems. This is caused by the use of system default MIDI synthesizer. In the future, a sample based synthesizer should be implemented. This synthesizer will be in charge of rendering audio samples in an appropriate form so the system output sounds more natural and expressive.

At last, the training set should be expanded. In the current system, the number of songs used to train the HMM is limited. This leads to a problem that the generated chord

progression may fall into a loop. To avoid the situation, more training corpus is needed to smooth the biased transition probability matrix. In addition to this, we should also involve western popular music. Although the Chinese songs may be totally in western style, but the unique cultural and stylistic differences may tilt the results to an unintended direction. One solution could be building style specific models for different kinds of popular music.

7.4 Conclusion

An automatic accompaniment system for vocal melodies is presented in this thesis. The pitch information of the input vocal audio is extracted by a real-time pitch tracking algorithm. Then a pitch class distribution based key estimation method is used to obtain the tonic and scale of the input melody. Moreover, a structure analysis algorithm is developed to get the structure boundaries on the vocal melody. Two Hidden Markov Models are trained with real life popular music in major and minor scales. With the help of the key and structure, the most probable chord sequence is calculated out for a given melody line by performing a decode operation on the HMMs. At the end, this chord list is realized by a MIDI style player.

Comparing to other similar systems designed for accompanying vocal melodies [3], our automatic accompaniment system is better in the following aspects. With a focus on popular music accompaniment, structure boundaries are highlighted in style player and chord assignment to reflect the structural characteristic of popular music. Additionally, the statistical models and key estimation algorithm are also trained by pure vocal tracks from real life popular music instead of MIDI data. This could make the entire system more coherent, because the expected system input is vocal audio not MIDI data. To enhance the accuracy of chord assignment, the key estimation is done before the HMM decoding. In this way, only the chords that are relevant to the key are considered in the statistical models. As a result, the number of states in the HMMs is dramatically

reduced, and the generated chord list is more unlikely to be too far from the harmonic center of the melody.

REFERENCES

[1] Dannenberg, R., “An On-line Algorithm for Real-Time Accompaniment”, in Proceedings of the International Computer Music Conference, 1984

[2] Chuan, C.-H., Chew, E. “A Hybrid System for Automatic Generation of Style-Specific Accompaniment”. 4th Intl Joint Workshop on Computational Creativity, June 2007.

[3] Morris, D., Simon, I., and Basu, S. “MySong: Automatic Accompaniment Generation for Vocal Melodies”, in Proceedings of Computer-Human Interaction, Florence, 2008.

[4] Larson, E., Maddox, R., "Real-Time Time Domain Pitch Tracking Using Wavelets", http://people.bu.edu/edlarson/KZOO_05_Wavelets.pdf (Accessed August 25, 2008)

[5] Patricio de la Cuadra, Aaron Master, and Craig Sapp. “Efficient Pitch Detection Techniques for Interactive Music”, in Proceedings of the International Computer Music Conference, 2001.

[6] Alain de Cheveign´e and Hideki Kawahara. “Yin, a Fundamental Frequency Estimator for Speech and Music”. Journal of the Acoustical Society of America, 111(4), 2002.

[7] S. Pauws. “Musical Key Extraction from Audio”, in Proceedings of the 5th ISMIR, Barcelona, Spain, 2004.

[8] L. Lu, M.Wang, and H. Zhang. “Repeating Pattern Discovery and Structure Analysis from Acoustic Music Data”, in Proceedings of 6th ACM SIGMM International Workshop on Multimedia Information Retrieval, October 2004.

[9] J. L. Hsu, C. C. Liu, and L. P. Chen, “Discovering Non-Trivial Repeating Patterns in Music Data,” IEEE Trans. On Multimedia, vol. 3, No. 3, pp. 311-325, 2001

[10] A. K. C. Wong and P. K. Sahoo, “A Gray-level Threshold Selection Method Based on Maximum Entropy Principle”, IEEE Trans. Syst.Man Cybern. SMC-19, 866–871 ~1989.

[11] Benward, B., Saker, M. “Music in Theory and Practice”, 8th ed., McGraw-Hill, Boston. 2008, ISBN: 0011254945

[12] Duda, R., Hart P., Stork D. “Pattern Classification”, 2nd ed. Wiley-Interscience, New York, 2001, ISBN: 0471056693

[13] Andrew J. Viterbi. “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm”, IEEE Transactions on Information Theory 13(2):260–269, April 1967

[14] E. M. Schwarz, M. Schmookler, and S. D. Trong. “Hardware Implementations of Denormalized Numbers”, in Proceedings of the 16th IEEE Symposium on Computer Arithmetic,Washington, DC, 2003.

[15] Yamaha Corporation, “Arranger Workstations”, www.yamaha.com, (Accessed March 21, 2009)

[16] MIDI Manufacturers Association, “Standard MIDI Files (SMF) Specification”, www.midi.org/techspecs/smf.php, (Accessed March 21, 2009)

[17] Wierzba, P., Bedesem, M. “Style Files - Introduction and Details”, www.wierzba.homepage.t-online.de/stylefiles_v100.pdf, (Accessed Feburary 4, 2009)

[18] Schonbrun, M. “The Everything Music Theory Book”, Adams Media, Massachusetts, 2006, ISBN: 1593376529

[19] Chordia, P., Rae, A. “Raag Recognition Using Pitch-class and Pitch-class Dyad Distributions”, in Proceedings of International Conference on Music Information Retrieval, 2007.

[20] Huron, D. “Sweet Anticipation”, The MIT Press, Massachusetts, 2006, ISBN: 0262083450

[21] Cooper, M., Foote, J. “Summarizing Popular Music via Structural Similarity Analysis,” in Proceedings of IEEE Workshop Applications of Signal Processing to Audio and Acoustics, 2003.

[22] Ishida, T., Abe, Y. US Patent 4,939,974. July 10, 1990.

[23] PG Music Inc, “Band-in-a-Box”, http://www.pgmusic.com, (Accessed April 1, 2009)

[24] Fujishima, T., “Real-time Chord Recognition of Musical Sound”, in Proceedings of the International Computer Music Conference, 1999.

[25] B. Vercoe, “The Synthetic Performer in the Context of Live Performance,” in Proceedings of the International Computer Music Conference, 1984.