Tempo Extraction using the Discrete Wavelet Transform

by

Tsang Kei Man

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science and Engineering

July 2006, Hong Kong

IMPORTANT NOTE: After this thesis was written some of the work described was greatly extended and improved, and was published in the following journal paper:

David Rossiter, Raymond Tsang, and Richard H. Y. So, 'Beat Deviation for Tempo Estimation Algorithms', Journal of the Audio Engineering Society, pp. 967-980, Vol. 55, No. 11, November 2007

Authorization Page

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Tsang Kei Man

Tempo Extraction using the Discrete Wavelet Transform

by

Tsang Kei Man

This is to certify that I have examined the above MPhil thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Dr. David Rossiter, Supervisor

Prof. Lionel M, Ni, Head of Department

Computer Science and Engineering
July 2006, Hong Kong

To my parents

Acknowledgments

A special thank you goes to my thesis supervisor Dr. David Rossiter for his guidance and great patience during my master's studies.

Thanks also go to the committee members of my thesis, Dr. Brian Mak and Dr. Oscar Au who took the time to read my thesis and offer valuable comments.

Thanks also go to the office staff in the Computer Science and Engineering department. They are always helpful, friendly and patient especially Ms. Connie Lau and Mr. Isaac Ma.

Finally, I would like to thank my parents for their love and support over these many years.

Tsang Kei Man

The Hong Kong University of Science and Technology

July 2006

Table of Contents

Title Page i

Authorization Page ii

Signature Page iii

Dedication iv

Acknowledgments v

Table of Contents vi

List of Figures x

List of Tables xvii

Abstract xix

Chapter 1 - Introduction 1

1.1 Thesis Objective and Motivation 1

1.2 Thesis Organization 2

Chapter 2 - Previous Work 3

2.1 Previous research on tempo extraction 3

2.2 Two important parameters 3

2.3 Tempo extraction using an off-line algorithm 5

2.4 Tempo extraction using an on-line basis 6

2.5 Tempo extraction using the discrete wavelet transform 13

Chapter 3 - Analysis of input data 15

3.1 The importance of the analysis of the input data 15

3.2 Our song set 17

3.3 The experiment 19

3.4 Data analysis 22

3.4.1 Skewness and kurtosis 22

3.4.2 Standard deviation 24

3.5 Most appropriate IOI tolerance 27

3.6 A comparison with previously published tolerance values 31

3.6.1 A survey of previously published values 31

3.6.2 Permitted IOI tolerance and our study 32

3.6.3 Permitted BPM tolerance and our study 35

3.7 Conclusion 37

Chapter 4 - Discrete wavelet transform 38

4.1 Introduction of discrete wavelet transform 38

4.2 Why the discrete wavelet transform is used..... 42

Chapter 5 - Tempo extraction algorithm 47

5.1 Overview of the system..... 47

5.2 Data Input..... 48

5.2.1 Input file format 49

5.2.2 Our input data sets..... 50

5.3 Applying the discrete wavelet transform 51

5.4 Peak detection 55

5.4.1 Full wave rectification 56

5.4.2 Moving window 57

5.5 Beat interval estimation 59

5.6 Histogram of beat interval..... 59

5.7 Improvement of the histogram..... 60

5.7.1 Introducing weight to the occurrence of beat intervals..... 60

5.7.2 Smoothing out the histogram 63

5.7.3 Analyzing the left and right channels..... 66

Chapter 6 - Experimentation and results..... 70

6.1 Implementation 70

6.2 Using our song set 73

6.3 Using the set from a previous contest – ISMIR 2004 80

6.3.1 Using other BPM deviation values 82

6.4 Using songs with karaoke version 84

Chapter 7 - Conclusion 86

7.1 Summary 86

7.2 Future Work 88

Bibliography 90

Appendix A 95

List of Figures

Figure 2.2-1: An illustration of the permitted deviation of inter-onset interval (IOI). If the second beat lies within +/-d of the target beat time, then the beat is regarded as valid. 4

Figure 2.2-2: An illustration of the permitted deviation of algorithmically derived tempo. If the BPM value determined by an algorithm lies within the permitted range of the ground truth value then the algorithm is regarded as having correctly identified the BPM value of the musical recording. 5

Figure 2.4-1: Overview of Goto's beat tracking system 7

Figure 2.4-2: Five agents are predicting the next beat in real time for Goto's beat tracking system 9

Figure 2.4-3: Examples of pre-registered drum patterns used by Goto's beat tracking system for popular music 10

Figure 2.4-4: A block-diagram of the whole system for Jensen's algorithm 11

Figure 2.4-5: A selection of beats in the beat induction histogram from Kristoffer's algorithm. The maximum histogram position of the beat induction histogram gives the current beat interval. 12

Figure 2.5-1: Block-diagram of George's beat detection algorithm based on the

DWT. WT – Wavelet Transform, LPF – Low Pass Filter, FWR – Full Wave Rectification, ↓ – Down-sampling, Norm – Normalization, ACR – Autocorrelation, PKP – Peak Periodicity, Hist – Histogram..... 14

Figure 3.3-1: A histogram of BPM values for the 100 sets of tapping data. 21

Figure 3.4-1: A graph of skewness across BPM for the 100 sets of tapping data. 23

Figure 3.4-2: A graph of excess kurtosis across BPM for the 100 sets of tapping data. 24

Figure 3.4-3: A plot of standard deviation of IOI values across BPM for all 100 sets of tapping data, using the first 96 IOI values in each set. The best fit power curve for subject A is the longer of the two curves. Each data point plotted in the graph represents the average deviation of the 96 intervals in one of the 100 sets of tapping data. The graph in total involves the analysis of 9700 individual taps. 25

Figure 3.4-4: Minimum, mean and maximum deviations of the tapping data plotted across BPM for all 100 sets of tapping data. The values of IOI mean (that is, standard) deviation lie in the range of 16.7-58.3ms. 26

Figure 3.4-5: Measures of standard deviation and the corresponding percentage of data included, for a perfect normal distribution..... 27

Figure 3.5-1: Recall measures for the 100 sets of tapping data at varying levels

of standard deviation. The maximum and minimum values of the candlestick line indicate the best and worst recall measures encountered..... 28

Figure 3.5-2: A partly heuristic illustration of valid beats not detected, valid beats detected, and invalid beats detected for an arbitrary beat detection system, across a range of IOI tolerance values expressed as a percentage of the IOI value. The portion shown assigned to B is based on the results of our analysis discussed later, shown in Figure 3.6-2..... 30

Figure 3.6-1: Permitted IOI deviation for papers included in Table 3.3, with our suggested curve "$y = 320.67x^{-0.3388}$" shown superimposed. Each of the 100 plotted data points represents the 2SD point for one of the sets of tapping data in our study. 33

Figure 3.6-2: Recall measures across a range of permitted IOI deviation values. Each data point represents the result of processing 100 sets of tapping data. The candle bar for each data point indicates the lowest and highest recall measures encountered when processing one of the tapping sets. The five circles indicate values that have previously been used in published research, shown in Table 3.3. 34

Figure 3.6-3: Accepted tolerance of the final BPM result for papers included in Table 3.3, with our suggested curve "$y = 320.67x^{-0.3388}$" shown

superimposed. Each of the 100 plotted data points represents the 2SD point for one of the sets of tapping data in our study 36

Figure 4.1-1: A continuous wavelet transform breaks down a signal into constituent wavelets of different scales and positions 39

Figure 4.1-2: Low scale (left) and high scale (right) of the wavelet 39

Figure 4.1-3: Original wavelet function (left) and shifted wavelet function (right) 40

Figure 4.1-4: Calculate a coefficient, C, to show how similar the wavelet and the signal are. 41

Figure 4.1-5: Shift the wavelet to another position 41

Figure 4.1-6: Scale the wavelet by stretching it 42

Figure 4.2-1: The original signal in the time domain (left). After the Fourier transform, the signal is transformed into the frequency domain (right) 43

Figure 4.2-2: A Fourier transform breaks down a signal into constituent sinusoids of different frequencies with different amplitudes 43

Figure 4.2-3: The original signal in time domain (left). By using the short time Fourier Transform with windowing technique, the output would become frequency against time (right) 44

Figure 4.2-4: The original signal in time domain (left). By using the Wavelet

Transform, the output would become scale against time (right). Low scale means high frequency while high scale means low frequency 45

Figure 4.2-5: Time domain, frequency domain, short time Fourier transform and wavelet analysis views of a signal..... 45

Figure 5.1-1: Overview of the tempo extraction system..... 48

Figure 5.2-1: A sinusoidal signal of frequency *f* sampled using a sampling rate *s* where $s \leq 2f$..... 50

Figure 5.3-1: Signal S passes through two filters to get signal A and signal D 52

Figure 5.3-2: Signal A and D get 1000 samples (left). Signal cD and cA get 500 coefficients by downsampling (right). ($\downarrow$) in the right figure means downsampling. 53

Figure 5.3-3: A DWT is applied to signal S to obtain cD and cA..... 53

Figure 5.3-4: A wavelet decomposition tree with four iterations..... 54

Figure 5.3-5: The frequency band of cD_1 to cD_4 if the input data is in format 44100Hz 55

Figure 5.4-1: The original signal is a sine wave (upper graph). After applying full wave rectification, the rectified signal is obtained (lower graph) 56

Figure 5.4-2: An example showing the moving window with each number representing the full wave rectified DWT coefficient. The width of the

moving window is 0.25 of a second. The local maximum within the window is shown in gray. Point A is a peak as it is a local maximum over 90% of the assessed length of time while point B is not. 58

Figure 5.5-1: Three beat intervals are calculated from four peaks 59

Figure 5.6-1: Histogram of beat interval 60

Figure 5.7-1: Eight beat intervals are used for comparison. 62

Figure 5.7-2: Similar beat (left) and not similar beat (right) 63

Figure 5.7-3: A histogram of beat interval which gets multiple maximum occurrences 64

Figure 5.7-4: A histogram of beat intervals in which the maximum occurs at the lowest density region of beat intervals 64

Figure 5.7-5: The smoothed histogram of the first example which is shown in Figure 5.7-3 65

Figure 5.7-6: The smoothed histogram of the second example which is shown in Figure 5.7-4 66

Figure 5.7-7: Overview of our algorithm if the input is in stereo format 68

Figure 5.7-8: The confidence value is defined as the area under the peak with 0.1 second width (gray in color) over the entire area under the curve 69

Figure 6.1-1: An overview of the tempo extraction system 71

Figure 6.2-1: BPM deviation percentage sorted in ascending order 76

Figure 6.2-2: A graph of average percentage error across a range of constant IOI tolerance 78

Figure 6.2-3: A graph of average percentage error across a range of percentage IOI tolerance 79

Figure 6.3-1: Accuracies 1 (light) and 2 (dark) on the dance music data set 81

Figure 6.3-2: Accuracies 1 (light) and 2 (dark) on the songs data set 82

Figure 6.3-3: A graph of accuracy 1 & 2 using a range of percentage of BPM deviation on the dance music data set 83

Figure 6.3-4: A graph of accuracy 1 & 2 using a range of percentage of BPM deviation on the songs data set 84

List of Tables

Table 3.1: Categories of the 50 songs in our set. 18

Table 3.2: A complete list of the 50 songs with associated information 19

Table 3.3: IOI and BPM tolerances stated in previously published papers. An empty cell indicates that the corresponding value was not stated in the paper. ISMIR and MIREX refers to the two contests concerning tempo estimation described in the main text. When assessing the results of tempo estimation algorithms tolerance values of 4% and 8% respectively were used in those competitions. 32

Table 6.1: Result of our song set. The boxes with bold borders are chosen by the system as the preferred tempo ‘answer’ because the confidence value (not shown) is the maximum among the mono / left / right signals..... 75

Table 6.2: Using a range of evaluation percentage of BPM deviation on our song set..... 76

Table 6.3: Using a range of constant IOI tolerance in our algorithm 77

Table 6.4: Using a range of percentage IOI tolerance in our system 78

Table 6.5: Average BPM deviation for the 50 songs in out set achieved by adding successive improvements 79

Table 6.6: Using a range of evaluation percentage of BPM deviation on the

dance music data set 83

Table 6.7: Using a range of evaluation percentage of BPM deviation on the songs data set 83

Table 6.8: Comparison of tempo extraction for two normal songs with their karaoke (i.e. non-vocal) version 85

Tempo Extraction using the Discrete Wavelet Transform

by

Tsang Kei Man

Computer Science and Engineering
The Hong Kong of Science and Technology

Abstract

This thesis presents a method to extract the tempo from an audio file. First of all, we study the audio file for the beats; the interval between two successive beats is called the inter-onset interval (IOI). In order to investigate the inter-onset interval, two musicians were invited to conduct some experiments on the inter-onset intervals for a data set. This data set consists of 50 musical recordings which were extracted from audio CDs.

For our tempo extraction system, an audio file is read into memory and then a discrete wavelet transform (DWT) is applied. The input signal is then decomposed into four levels of DWT coefficients and a peak detection algorithm is performed to extract all peaks from these DWT coefficients. All the peaks are used to calculate the

IOI. Some of them are more important for the IOI than others. So, a weight is introduced to each IOI in order to increase the accuracy of our system. We define the weight according to how many of the IOI's neighbors give similar values. All the weighted IOIs will form a histogram. The histogram is then smoothed out using a Gaussian function in order to better estimate the tempo.

For an input which is in stereo format, we treat it as three different inputs; the left channel, the right channel and the mono channel. The mono channel is the average of the left and right channels. We pass these three inputs into our system. Then, we can select the best one to be our final result.

The entire system was implemented using Matlab. We test our system using one data set of 50 musical recordings and one data set which had been used in a tempo extraction contest during the International Conference on Music Information Retrieval (ISMIR 2004). We obtained the correct tempo for 47 out of the 50 songs in our data set, achieving high accuracy. For the contest, there are in total two sets of data we can test with. Our ranking for one set is 2nd out of 12 and the other set is 3rd out of 12. This result shows that our system is competitive with the other algorithms used in the contest.

CHAPTER 1

INTRODUCTION

1.1 Thesis Objective and Motivation

Accurate and efficient search techniques for digital multimedia are one of the most important research areas in the field of computer engineering. An accurate measure of the tempo of audio recordings would greatly enhance the power of music search and retrieval systems. For example, when we want to determine what the genre of a piece of music is, tempo is one of the most important factors.

The purpose of tempo extraction is to extract the tempo value of any song or music. The tempo value is a number which represents the speed of a song or music measured by beats per minute (BPM). This work seeks to develop an advanced algorithm for the determination of the BPM metric for digital audio recordings.

For any song or music, we can always count the number of beats within 1 minute to get the tempo value, or for example, we can count for 15 seconds and then multiply the answer by 4. It is easy to do it manually, but how can a computer manage this? This thesis proposes an algorithm to extract the tempo using the discrete wavelet transform, so that a computer is manage to do it.

1.2 Thesis Organization

Chapter 2 gives a general review on the previous work done on both on-line and off-line tempo extraction algorithms. On-line algorithms process in real time while off-line algorithms process in batches.

In Chapter 3, we perform an analysis of a set of 50 songs to propose an appropriate IOI tolerance value to the system and compare the values published by other papers.

An introduction to the discrete wavelet transform is presented in Chapter 4. The reason we chose a discrete wavelet transform to perform tempo extraction is also discussed.

Chapter 5 describes the tempo extraction algorithm, which uses a series of discrete wavelet transforms.

Chapter 6 presents the implementation of our system and the analysis of the results. The system uses our 50 song set as input to demonstrate the performance of the system. It also uses a data set used in a tempo induction contest as input and shows the results.

Finally, the summary of the thesis as well as some future directions are discussed in Chapter 7.

CHAPTER 2

PREVIOUS WORK

2.1 Previous research on tempo extraction

In this chapter, we first introduce two important parameters. Then we discuss various types of tempo extraction. There are two main approaches, on-line and off-line. On-line algorithms are also called real time systems. They get the tempo value while the music is playing and keep updating the tempo value until it remains constant. Off-line algorithms involve inputing audio files into a system. After processing the audio file data, the system outputs the tempo value.

2.2 Two important parameters

The two parameters which are most pertinent to this thesis are now considered.

1. **Inter-onset interval (IOI) tolerance**. The interval between two successive beats is called the inter-onset interval (IOI). As discussed, there must be some degree of permitted deviation in a system measuring musical beats. The concept is illustrated in Figure 2.2-1. In some of the published research the value of the

permitted IOI deviation is a fixed time value. In other research a percentage value in terms of IOI has been used.

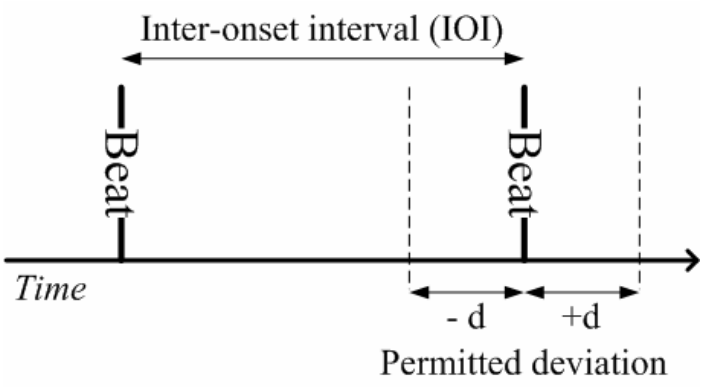

Figure 2.2-1: An illustration of the permitted deviation of inter-onset interval (IOI). If the second beat lies within +/-d of the target beat time, then the beat is regarded as valid.

2. **BPM tolerance**. Figure 2.2-2 illustrates how an algorithm which provides a measure of BPM for a musical recording is regarded as 'correct' if the algorithmically derived BPM value is within a certain range of a previously determined ground truth value.

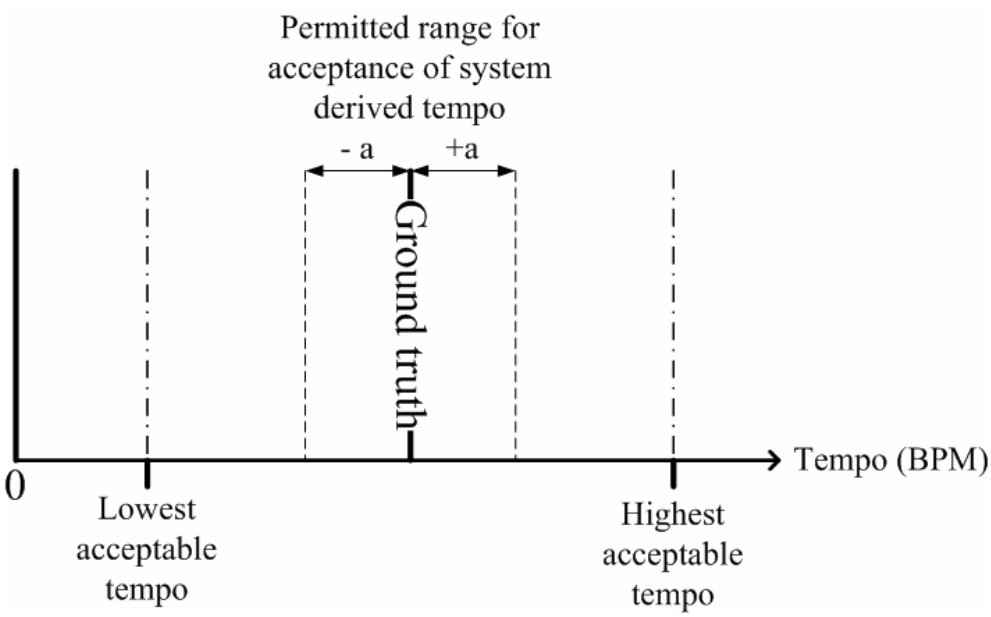

Figure 2.2-2: An illustration of the permitted deviation of algorithmically derived tempo. If the BPM value determined by an algorithm lies within the permitted range of the ground truth value then the algorithm is regarded as having correctly identified the BPM value of the musical recording.

2.3 Tempo extraction using an off-line algorithm

There are several off-line algorithms which perform tempo extraction. Alonso *et al.* [6] introduced a system that works off-line with a large database containing 489 songs from several musical genres. The algorithm involves three stages:

I. A front-end analysis that efficiently extracts onsets.

II. A periodicity detection block.

III. The temporal estimation of beat locations.

This uses an off-line method and the BPM is calculated from the estimated beat locations. For the 489 songs in the database, the author reported that the success rate of estimating the tempo was 89.7%.

2.4 Tempo extraction using an on-line basis

For the algorithms which are using an on-line method, Curtis *et al*. [4] describes that a history mechanism with a decaying memory of past beats can predict the next beat. The tempo can be calculated by all the past beats inside the memory. The predicted beat can be used for checking if the tempo is changing or not. However, a short memory cannot retain a lot of past events. Although it allows rapid tempo fluctuations, it is not stable. A long memory steadies the tempo at the expense of ignoring fast tempo changes.

Another algorithm proposed by Mont-Reynaud [7] demonstrates a tempo tracker that pursues two strategies in parallel. One extracts the “important events”. These events serve as structural anchors in the music. The important points such as strong beats can sometimes be recognized in the rhythmic or melodic parts. The other one use an independent method for tracking tempo fluctuations. This method searches for repeating patterns in successive durations and keeps running statistics on the most common durations. Combining these two methods can solve the memory

problem because of two reasons. One of the reasons is that this system does not only rely on the memory. The other one is that the flexibility is increased as shown in a parallel system.

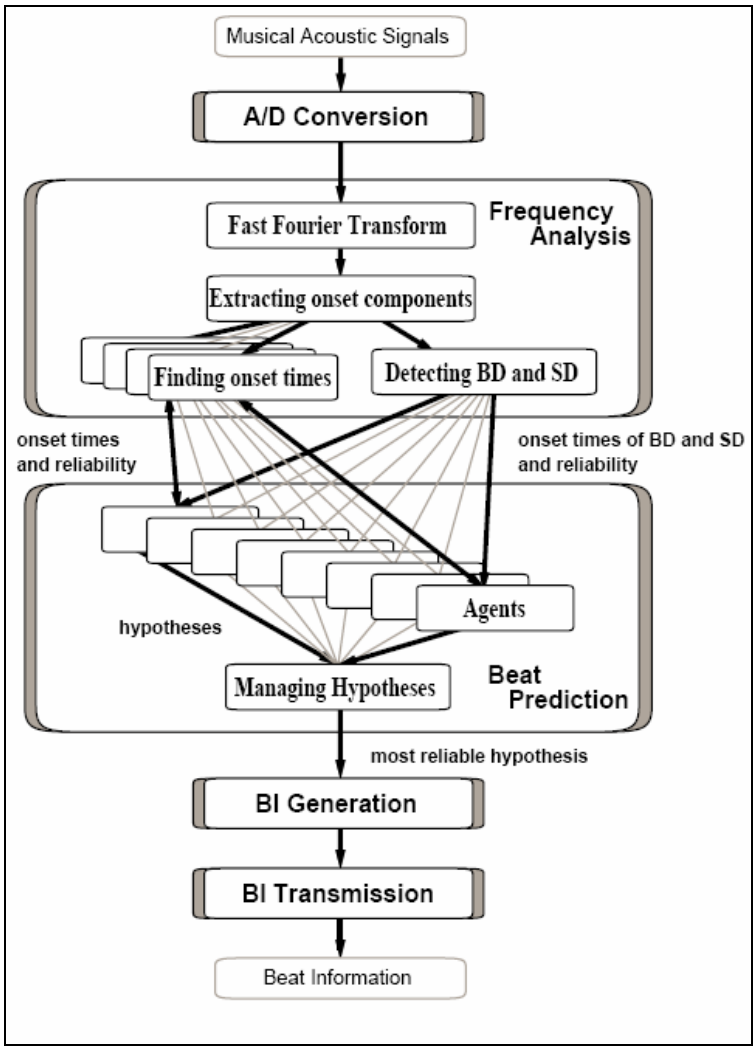

Figure 2.4-1: Overview of Goto's beat tracking system

A beat tracking system (see Figure 2.4-1) is introduced by Goto *et al*. [1]. It is a system that processes the acoustic signals of music in real time. First, the system performs an analog to digital conversion from musical acoustic signals. Then, it performs a Fast Fourier Transform (FFT) to calculate the frequency spectrum. From

the frequency spectrum, the system extracts the onset components and then finds the onset times from the onset components. It also detects the base drum and snare drum at the same time. Afterward, the system can perform beat prediction. As it is performing in real time, it has to predict the next beat given the previous beats.

The author thinks it would not be accurate enough if there was only one agent to predict the beat. In order to increase the accuracy, the author suggested using a total of thirty agents to predict the next beat (Figure 2.4-2 shows five agents only as an example). Each agent determines the next beat independently. Finally, the system chooses the agent which obtained the highest reliability for the result. The reliability is obtained as the ratio of its peak value to a recent maximum peak value. The reliability of each agent will be increased if an onset time coincides with the beat time predicted previously.

Besides getting the beat, it can determine whether a beat is a strong beat or a weak beat, which is based on the base drum sound and the snare drum to determine the strong beat and weak beat respectively. The author reported that the system correctly tracked beats in 27 out of 30 popular songs under the assumption that the time-signature is 4/4 and the beats per minute are between 70 and 180.

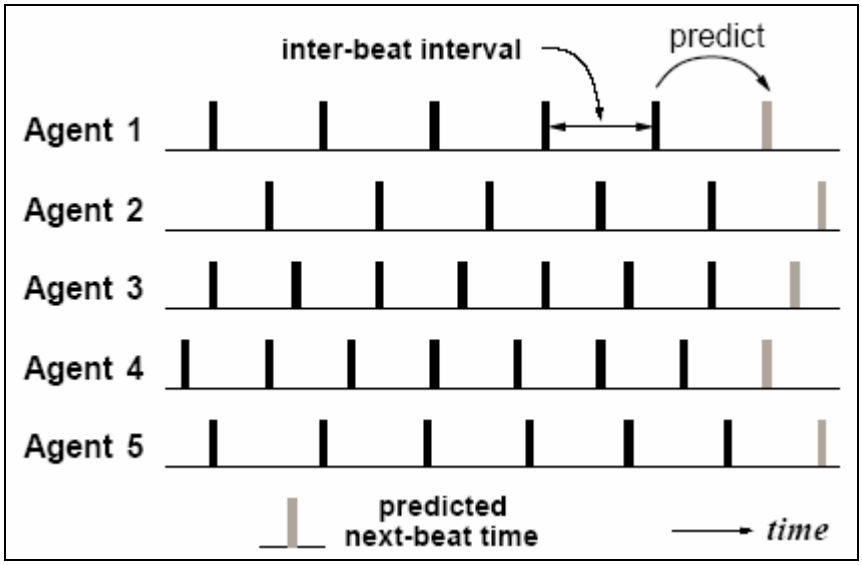

Figure 2.4-2: Five agents are predicting the next beat in real time for Goto's beat tracking system

Another real time beat tracking system is presented by Goto *et al*. [2]. This time, the system deals with popular music, particular rock and pop music. This type of music contains drum sounds that maintain a beat. The main features are similar to the previous system developed by the same author [1] except for the frequency-analysis. The frequency-analysis parameters are dynamically adjusted by interaction between low-level and high-level processing. The low-level process contains frequency analysis while the high-level process contains musical knowledge to determine a strong or weak beat. As all the songs contain drum sounds, the system contains eight pre-registered drum patterns (see Figure 2.4-3) for comparison. The author reported that this system correctly tracked beats in 40 out of 42 popular songs sampled from compact discs.

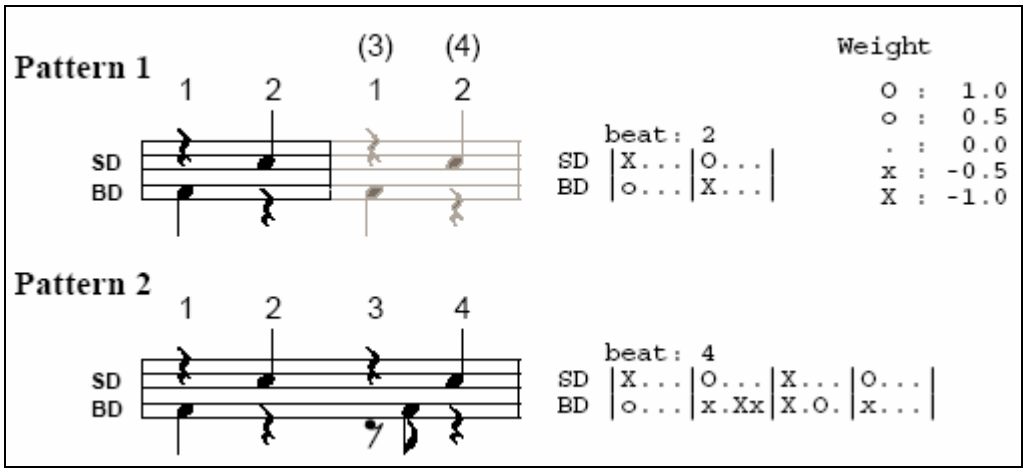

Figure 2.4-3: Examples of pre-registered drum patterns used by Goto's beat tracking system for popular music

A new system was modified by Goto *et al*. [3] and attained a better performance compared with their previously described system [2]. The new system improved the method for detecting the snare drum. First, the noise components are quantized. Second, the system calculates how widely it is distributed along the frequency axis. Another improvement is that the agents themselves are not totally independent of each other. They track beats according to different strategies utilizing auto- and cross-correlation of detected onset times to predict the next beat. Also, agents are grouped into pairs. Each agent evaluates the reliability of its own hypothesis and two agents in a pair try to track beats with different ranges of tempi. This enables one agent to track the correct beats even if the other agent tracks beats with double or half the correct tempo. In this example, the system used twenty eight agents and the author reported that it correctly tracked beats in 42 out of 44 songs.

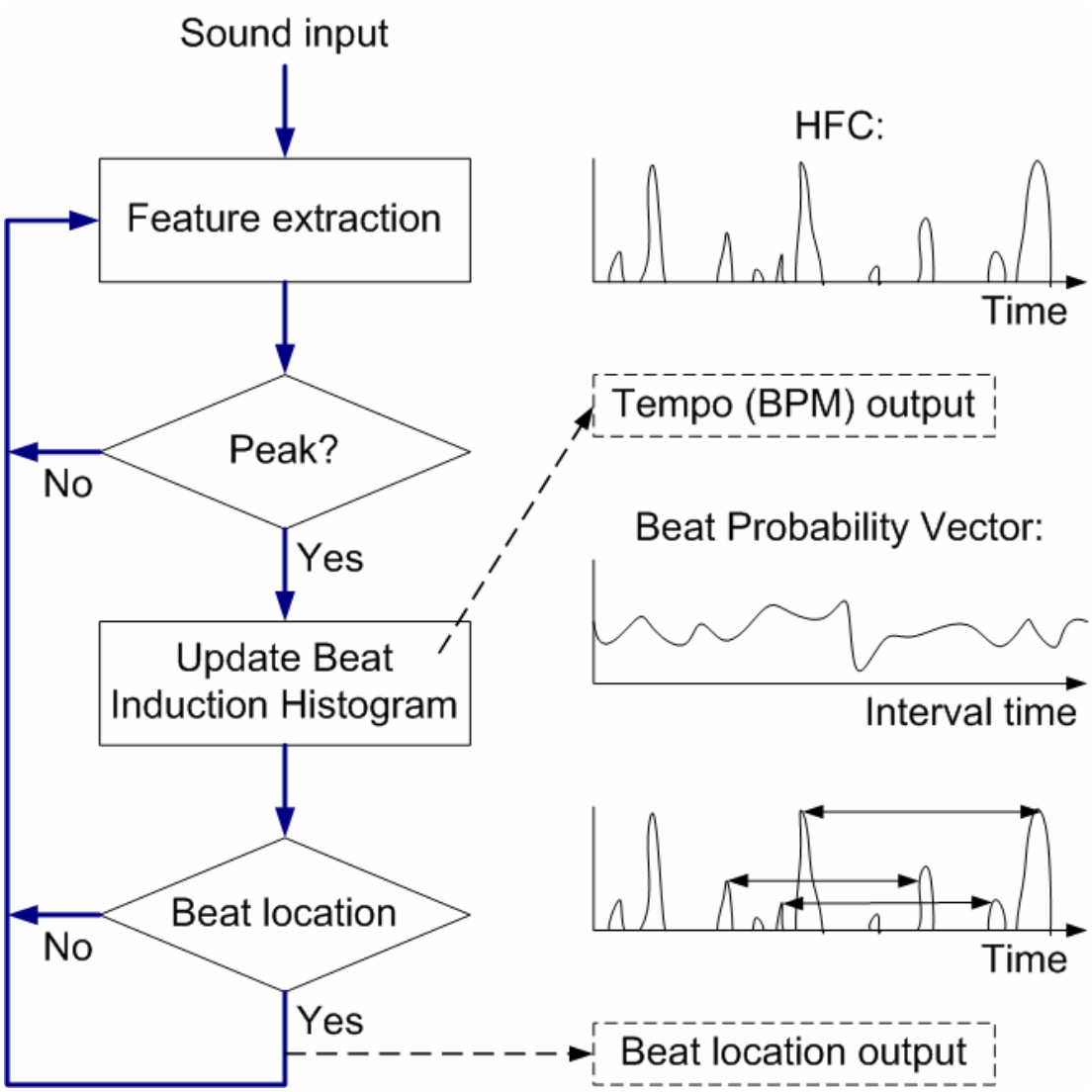

Figure 2.4-4: A block-diagram of the whole system for Jensen's algorithm

Jensen et al. [9] presented an algorithm for finding the beat location and interval (see Figure 2.4-4). This algorithm consists of four parts. First, it performs feature extraction for the input, followed by peak detection. It then updates the beat induction histogram. The beat induction histogram is a histogram that collects all the beat intervals and plots the weight of each interval. The system then estimates the beat location by using the weight of the peaks. The first two parts of this algorithm are similar to the others. The core part of the method is the beat induction histogram

(see Figure 2.4-5). It is a histogram of note onset intervals. For every new note onset occurance, the histogram is updated by adding a Gaussian shape, of which the center is placed at the intervals corresponding to the distance to the previous note onset. The maximum of the beat induction histogram gives the current beat interval.

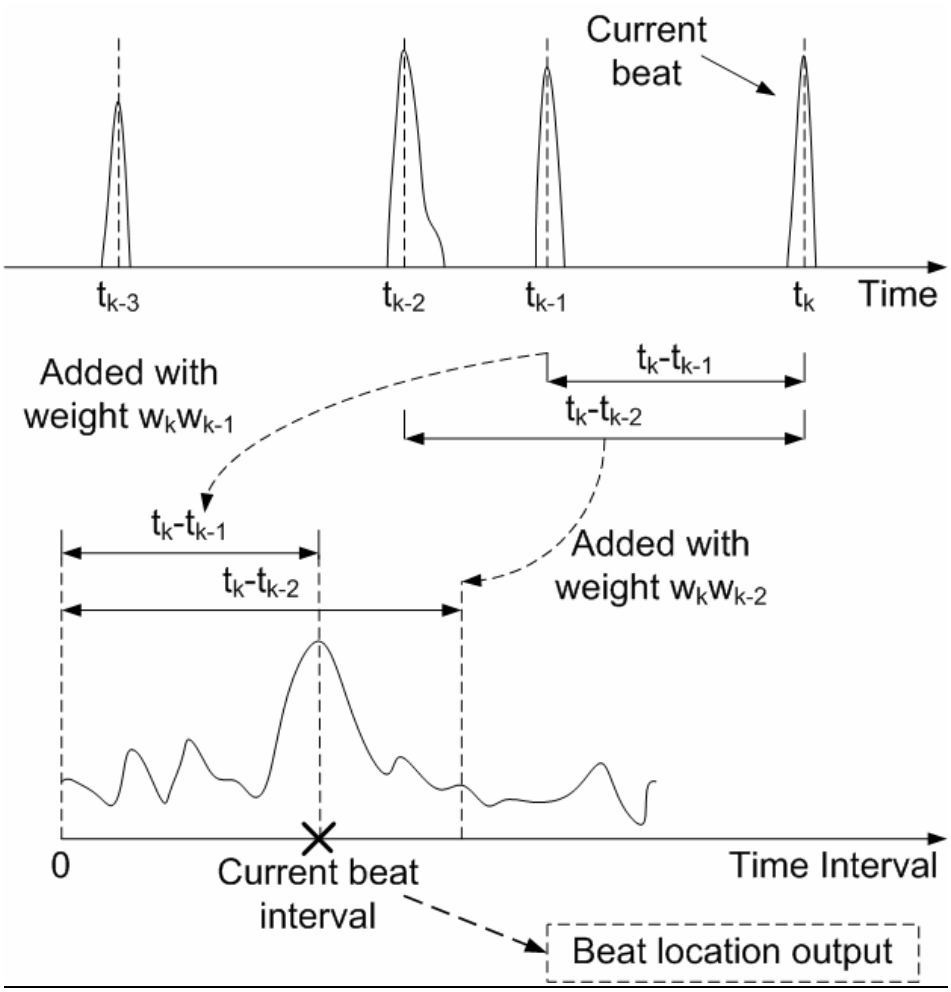

Figure 2.4-5: A selection of beats in the beat induction histogram from Kristoffer's algorithm. The maximum histogram position of the beat induction histogram gives the current beat interval.

A new system was developed by Goto *et al.* [5] that can perform beat tracking with the audio signals that have no drum sound. As there is no drum sound, the system

cannot use the old method previously described in [1], [2], [3] to detect the drum patterns. So, the system utilizes the following musical knowledge:

I. Sounds are likely to occur on beats.

II. Chords are more likely to change at the beginning of measures that at other positions.

III. Chords are more likely to change on beats than in other position between two successive correct beats.

As there is no drum, it is more difficult to track the beat correctly. The author reported that the system correctly tracked beats in 34 out of 40 popular songs that did not include drum sounds. The accuracy of this system decreases when there is no drum sounds.

2.5 Tempo extraction using the discrete wavelet transform

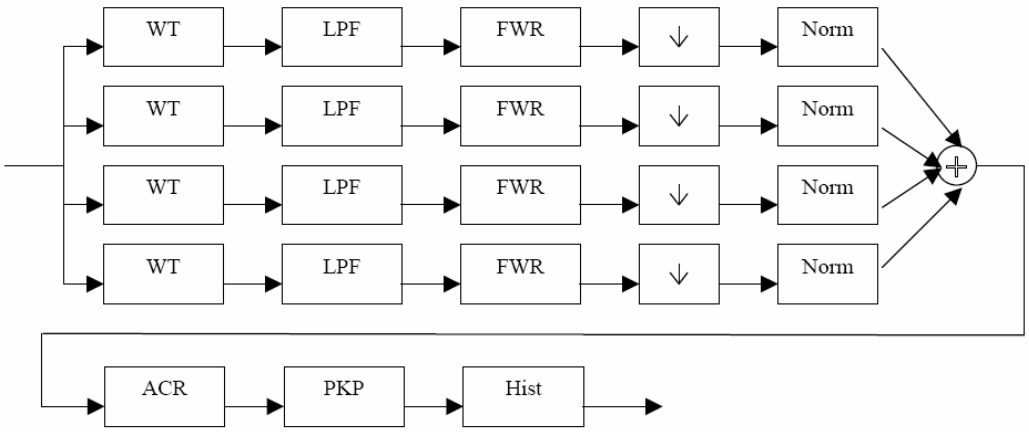

Figure 2.5-1: Block-diagram of George's beat detection algorithm based on the DWT. WT – Wavelet Transform, LPF – Low Pass Filter, FWR – Full Wave Rectification, ↓ – Down-sampling, Norm – Normalization, ACR – Autocorrelation, PKP – Peak Periodicity, Hist – Histogram.

Some algorithms use discrete wavelet transform (DWT) to perform tempo extraction. George *et al*. [8] described a method using DWT to perform this. This algorithm (see Figure 2.5-1) detects the most salient periodicities of the signal. First the signal is decomposed using the DWT. By applying a low pass filter, full wave rectification and down-sampling for each band, we can extract the time domain amplitude envelope of each band separately. Then, we sum up the envelopes of each band and compute an autocorrelation function. The first five peaks of the autocorrelation function are detected and their corresponding periodicities in beats per minute (BPM) are calculated and added in a histogram. By repeating this process over the signal, the final histogram is the estimated tempo of the song in BPM.

From the above work on on-line and off-line algorithms, a general picture of tempo determination algorithms has been given. In the next chapter the nature of musical beat information is analysed, prior to determining an appropriate tempo extraction system.

CHAPTER 3

ANALYSIS OF INPUT DATA

3.1 The importance of the analysis of the input data

As discussed in Chapter 1, one of the primary parameters of musical recordings is the tempo of the recording, commonly expressed as a BPM (beats per minute) value. The BPM measure is a key parameter for music search and retrieval systems and related fields such as genre categorization, beat tracking and musical transcription. However, musicians do not play music using precise time intervals between each beat. For example, time deviations between successive beats are a consequence of musical expression. Therefore, an algorithm which attempts to determine the tempo of a musical recording must appreciate that a ‘normal’ beat involves some time deviation.

There are two motivations for analyzing the input data. Firstly, although the range of techniques employed by BPM estimation algorithms varies considerably, the majority expressly allow for some form of beat deviation in their methodology. However, there is no known measure of what ‘normal’ deviation is. As a result the range of permitted beat deviation varies greatly from algorithm to algorithm. If an

expression for normal beat deviation was derived then such algorithms could employ the expression during beat analysis, which should result in significantly greater accuracy.

The second issue arises when the performance of a tempo estimation algorithm is being assessed. BPM results are typically compared to a ‘ground truth’ measure obtained by human subjects, with the algorithmically determined BPM value regarded as correct if it falls within a percentage range of the ground truth value. However, because of the lack of understanding of normal deviation from the ground truth BPM measure, the BPM tolerance values used by published studies vary considerably. For an illustration of this consider two recent research competitions in which tempo estimation was a major task. A BPM tolerance value of 4% was used by one competition (the International Conference on Music Information Retrieval (ISMIR 2004) Audio Description Contest [10]) and the significantly different value of 8% by another (the Music Information Retrieval Evaluation eXchange (MIREX 2005) – Audio Tempo Extraction [18]). A measure of normal deviation could be used to more accurately determine the range of values for which a BPM result should be considered correct.

The issue of beat deviation is a critical one for both these parameters. Indeed, there is a general case to be made where algorithmic methods for tempo estimation are

gradually reaching a nadir of performance. As an indication of this the top four best scores for the MIREX 2005 contest were very closely clustered at 0.670, 0.675, 0.675, and 0.689 [18], although a discussion of the scoring method used by that contest is outside the scope of this paper. An appropriate measure of beat deviation may give rise to increased performance of the tempo estimation field as a whole.

To address these issues we assembled a set of musical recordings. We then analysed the performance of two subjects in which they followed the beat of the recordings via tapping tests. The beat deviation of the two subjects was considered in the light of deviation values used by previous research. To address the issue of individual beat deviation we propose a power curve measure of deviation across BPM which can be used by algorithms which perform beat detection.

3.2 Our song set

A set of 50 musical recordings was collected for our study. The recordings were extracted from standard commercially available audio CDs and were stored in non-compressed format. During the process of accumulating the set any song which was found to have a varying tempo was discarded. The set of recordings were selected to include a wide range of styles and musical structures. A summary of the different styles is shown in Table 3.1.

6 Alt/Indie	3 Ballad	3 Blues	2 Classical
2 Country	2 Electronica	1 Folk Rock	1 Heavy Metal
1 Instrumental	6 Jazz	3 Latin	5 Pop
5 Reggae	7 Rock	2 Soul	1 Synth Pop
50 Total			

Table 3.1: Categories of the 50 songs in our set.

To ensure that the songs could be located by future researchers all albums were checked as being available for purchase over the Internet. A complete list of the songs and associated information is available in Table 3.2.

Song	**Album**	**Artist**	**Genre**
Addicted to love	Now that's what I call music 1986	Robert Palmer	Rock
Baby please don't go	They call me muddy waters	Muddy Waters	Blues
Bachelorette	Homogenic	Bjork	Alt/Indie
Besito pa ti	Mongo introduces la lupe	Mongo Santamaria	Latin
Breakfast in bed	UB40	UB40	Reggae
Buenas noches from a lonely room	Buenas noches from a lonely room	Dwight Yoakam	Country
Chariots of fire	Chariots of fire	Vangelis	Electronica
Country feedback	Out of time	R.E.M	Alt/Indie
Do you want to	You could have it so much better	Franz Ferdinand	Rock
Domingo	Stella	Yello	Synth pop
Firestarter	The fat of the land	Prodigy	Pop
Five circles	Chariots of fire	Vangelis	Electronica
Freeddie freeloader	Kind of blue	Miles Davis	Jazz
Friendly fire	Guns in the ghetto	UB40	Reggae
Guns in the ghetto	Guns in the ghetto	UB40	Reggae
Hunter	Homogenic	Bjork	Alt/Indie
I am a rock	Sounds of silence	Simon & Garfunkel	Folk Rock
I cried for you	After hours	Sarah Vaughan	Jazz
I got you	Buenas noches from a lonely room	Dwight Yoakam	Country
I wish I know	Pure jazz chillout	Billy Taylor Trio	Jazz
Knock on wood	Knock on wood	Eddie Floyd	Soul
Layla	Now that's what I call music 1982	Derek & The Dominoes	Rock
Lie to me	Rei momo	David Byrne	Alt/Indie
Loco de amor	Rei momo	David Byrne	Reggae
Maggic McGill	Morrison hotel	The Doors	Blues
Maybe someday	Bloodflowers	Cure	Alt/Indie
Mi tonada montuna	La coleccion cubana	Ibrahim Ferrer	Latin
Mofo	Pop	U2	Alt/Indie
My heart will go on	Titanic music from the motion picture	Celine Dion	Ballad
Nerve centre	The city	Vangelis	Instrumental
No expectations	The first 10 years	Joan Baez	Ballad
No surprises	OK computer	Radiohead	Pop

Paranoid android	OK computer	Radiohead	Pop
Play dead	Debut	Bjork	Alt/Indie
Pride	Rattle and hum	U2	Rock
Right here, right now	You re come a long way, baby	Fat Boy Slim	Pop
Roadhouse blues	Morrison hotel	The Doors	Blues
Song for Bob Dylan	Hunky dory	David Bowie	Rock
Summertime	After hours	Sarah Vaughan	Jazz
The pan piper	Sketches of spain	Miles Davis	Jazz
The ride of the valkyries	Classical music for dummies	Wagner	Classical
The soul cages	Soul cages	Sting	Rock
The thing that should not be	Master of puppets	Metallica	Heavy metal
Una fuerza inmensa	La coleccion cubana	Ibrahim Ferrer	Latin
Wednesday morning 3am	Wednesday morning 3am	Simon & Garfunkel	Pop
Who's crying now	Who's crying now	Joyce Sims	Soul
Will O the wisp	Sketches of spain	Miles Davis	Jazz
With god on our side	The first 10 years	Joan Baez	Ballad
Word up	Word up	Gun	Rock
Work	Uprising	Bob Marley	Reggae

Table 3.2: A complete list of the 50 songs with associated information

3.3 The experiment

Two people were employed for this study. Both were female, aged approximately 25 years old, musically trained to western exam grade 8 standard, and regularly active in musical accompaniment and teaching. In this section they are referred to as subject A and subject B. The two subjects were not known to each other and did not confer in any way during the assessment. Both subjects were required to process each of the 50 songs in our set. For each recording the procedure was as follows. The subjects first familiarized themselves with the music by listening to it as many times as they desired. They were then required to formally identify the time signature and therefore the main beat. They then listened to the song and tapped a key on a computer keyboard each time that a beat occurred. For each tap the time

interval relative to the previous tap was recorded by a software program. The subjects were required to record a continuous tap sequence from the start of the music onwards, for a minimum duration of 65 seconds. After the tap sequence for a song was recorded the time interval data was transferred to a spreadsheet for further analysis. Each individual time interval was checked. If any missing or double taps were identified the subjects were required to re-do the complete tap sequence for that recording. The average IOI value was used to determine a BPM value for each tapping sequence.

For 10 of the 50 songs the two subjects disagreed on the time signature. That is, one subject typically identified the main beat as being double or half that identified by the other subject. This is not an error as such, it is common that even professional musicians perceive the primary beat of a song differently to other musicians. This is particularly the case for songs which have a relatively more complex rhythmic structure, which is true for a number of songs in our set of 50. As any up- or down-scaling of the deviation data might falsely skew subsequent analysis no processing took place to resolve such conflicts. Consequently, the two subjects exhibited a different range of BPM values for the same 50 songs. The results for subject A cover a range of 49.5 BPM to 185.6 BPM, and the results for subject B

cover a range of 65.5 BMP to 174.9 BPM. A histogram of the 100 BPM values is shown in Figure 3.3-1.

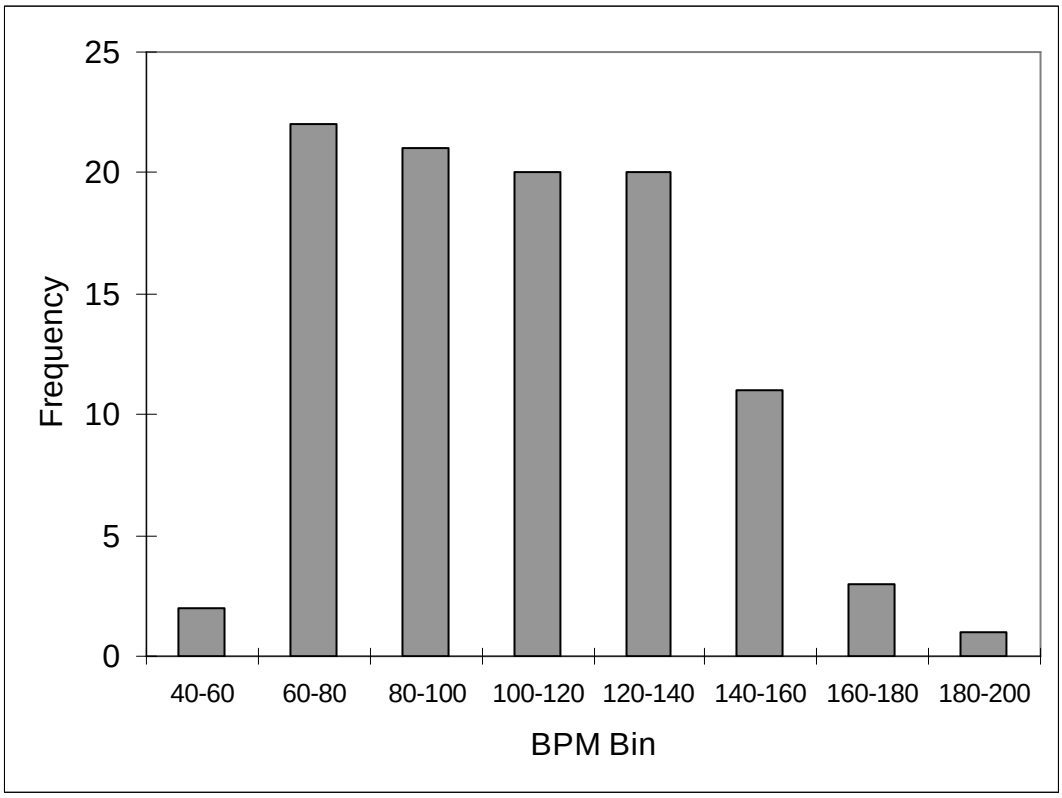

Figure 3.3-1: A histogram of BPM values for the 100 sets of tapping data.

The minimum, maximum and average tapping durations for subject A were 65.6, 234.1, and 104.9 seconds respectively, and for subject B it was 85.2, 202.1 and 120.7 seconds respectively. However, in our subsequent analysis we do not use every recorded tap interval. Relatively more taps were recorded for relatively faster tempo recordings, as the interval between beats is shorter. However, an arbitrary data set with relatively more data tends to demonstrate relatively less deviation. In order to avoid this potential influence our analysis uses the first 96 IOI values for

each song. The value of 96 comes from the minimum quantity of tapping intervals recorded for any song in our study.

3.4 Data analysis

Our study is concerned with identifying a measure of the normal deviation of beats across a range of BPM. Before determining the measure of beat deviation across BPM it is appropriate to first consider whether the tapping data is approximately normal in shape, so that standard deviation analysis can be applied.

3.4.1 Skewness and kurtosis

For this reason skewness and kurtosis values of the 100 sets of tapping data were derived. Skewness provides a measure of the symmetry of a data set about the mean; kurtosis provides a measure of how peaked or flat the data is relative to a normal distribution.

The skewness values for each of the 100 tapping sequences are shown in Figure 3.4-1. A value of zero indicates a perfectly normal distribution. Although there are many individual songs which do not exhibit a negligible value, the average skew value is 0.065 and the standard deviation is 0.38. We suggest that the songs which exhibit non-negligible values do so primarily because they have relatively more varied rhythmic patterns.

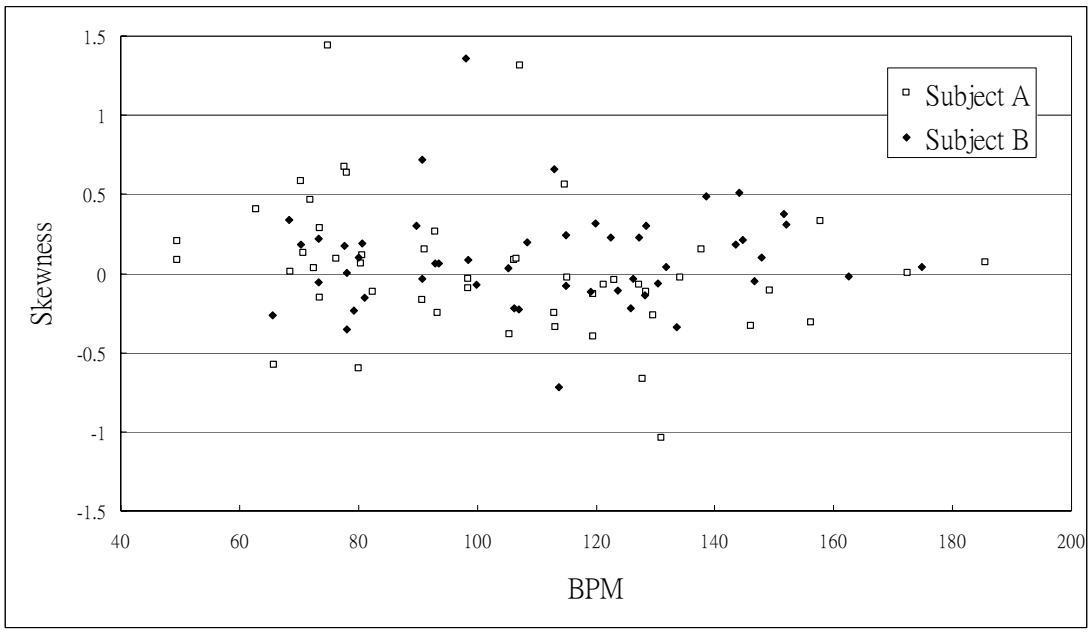

Figure 3.4-1: A graph of skewness across BPM for the 100 sets of tapping data.

The 100 sets of tapping data were then analysed for excessive kurtosis. Figure 3.4-2 shows a plot of the excessive kurtosis values across BPM. The average excess kurtosis is 0.805, and the standard deviation is 2.36. There is therefore, on average, a mild excess of kurtosis. That is, the distribution of tapping data tends to have a slightly longer tail on the right hand side compared to the left. However, Figure 3.4-2 shows that just a few sets of tapping data exhibit greatly excessive kurtosis, with the great majority of tapping data exhibiting relatively minor levels of excess kurtosis. We propose that the few songs exhibiting greatly excessive kurtosis are likely to be the result of complex rhythmic structures which have influenced the tapping measures of the subject.

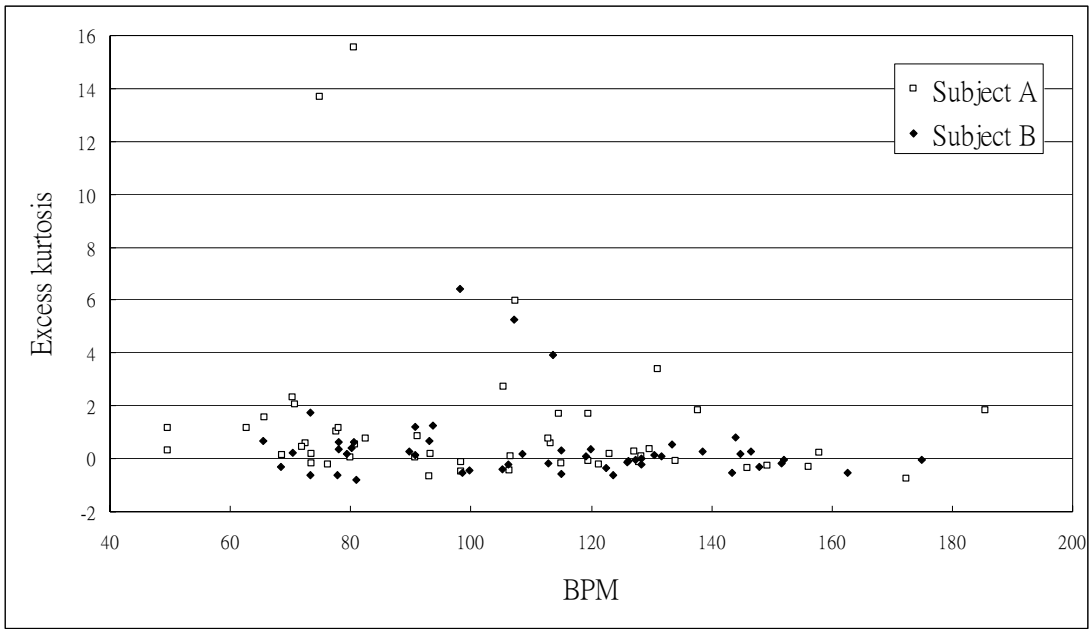

Figure 3.4-2: A graph of excess kurtosis across BPM for the 100 sets of tapping data.

We conclude that, based on the limited data available, it is sufficient to regard the sets of tapping data as having a normal distribution.

3.4.2 Standard deviation

The standard deviation was calculated for each of the 100 sets of tapping data, and is shown in Figure 3.4-3. A best fit power curve is shown for each subject, for easier comparison. A basic observation is that IOI deviation is inversely proportional to the song BPM. The general pattern of deviation across BPM is almost identical for both subjects, although subject B exhibits a minor but consistently higher level of deviation across BPM. The correlation coefficient between the two subjects is 0.9984. It is well known that different musicians playing the same piece of music

express different levels of timing deviation. The difference between subjects may be due to slight differences in musical interpretation and/or the way they perform a piece. As the difference in patterns of deviation between the subjects is minor, for subsequent analysis we aggregate both sets of 50 tapping data into a superset of 100.

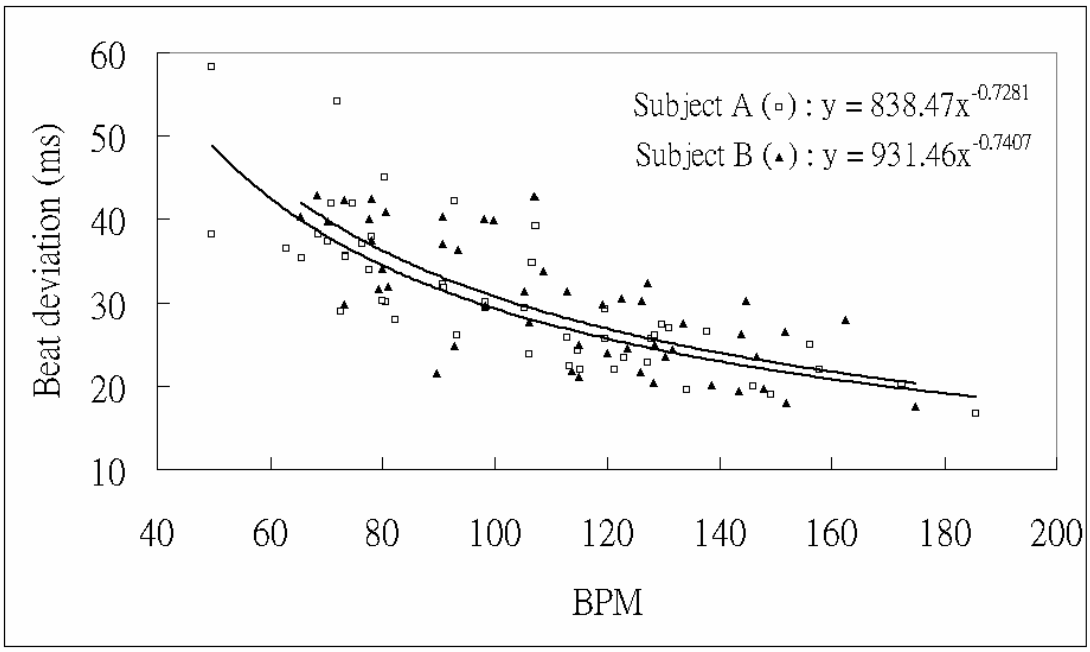

Figure 3.4-3: A plot of standard deviation of IOI values across BPM for all 100 sets of tapping data, using the first 96 IOI values in each set. The best fit power curve for subject A is the longer of the two curves. Each data point plotted in the graph represents the average deviation of the 96 intervals in one of the 100 sets of tapping data. The graph in total involves the analysis of 9700 individual taps.

The minimum, maximum and standard deviation for the 100 sets of tapping data are shown in Figure 3.4-4, together with power curves of best fit for each. The minimum deviation of IOI values from their mean (that is, the difference between the closest IOI value to the mean IOI value and the mean IOI value itself) has an average value of 3.31ms, with a minimum and maximum of 0.08ms and 14.39ms

respectively. The low values are to be expected for a normal distribution of data containing 96 values. The standard deviation of IOI values has an average value of 30.38ms, with a minimum and maximum of 16.69ms and 58.29ms respectively. The maximum deviation of IOI values from the mean has an average value of 86.71ms, with a minimum and maximum of 41.96ms and 245.22ms respectively.

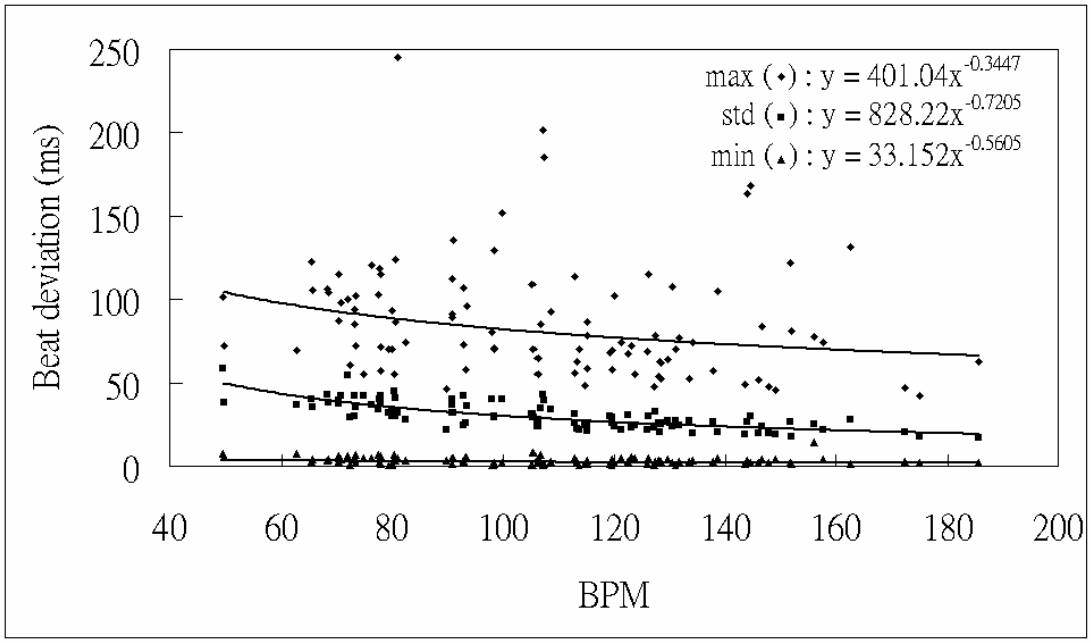

Figure 3.4-4: Minimum, mean and maximum deviations of the tapping data plotted across BPM for all 100 sets of tapping data. The values of IOI mean (that is, standard) deviation lie in the range of 16.7-58.3ms.

The question we wish to address is this: which level is the most appropriate for a beat estimation algorithm? Figure 3.4-5 illustrates a perfect normal distribution for an arbitrary data set and the quantity of data included at varying levels of standard deviation. At the most commonly considered level of deviation, that of 1 SD, only 68.3% of the data is included. With 31.7% of the data 'lost', it is probable that this

level does not involve enough of the data for our purposes. Higher levels of deviation, which would include greater amounts of the data, are more likely to be appropriate. In the following section we describe how we processed our tapping data at varying levels of standard deviation in order to help identify the most appropriate one.

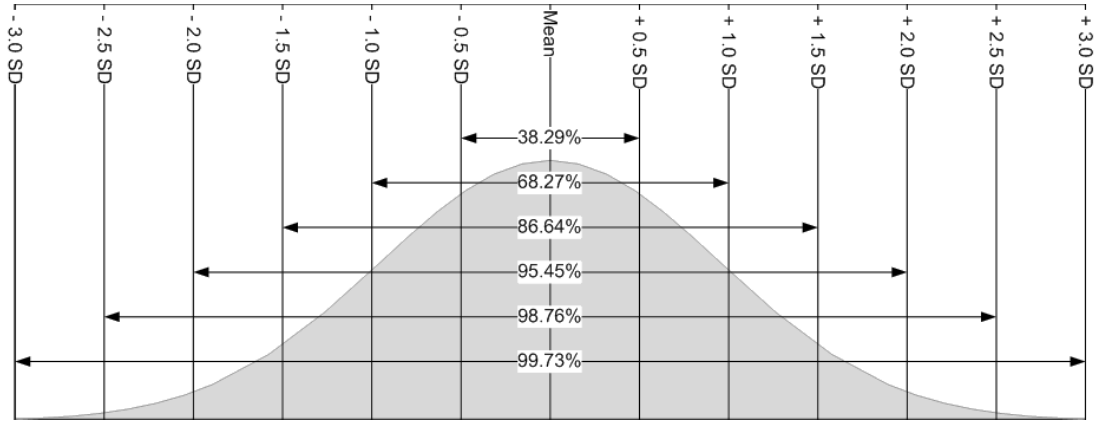

Figure 3.4-5: Measures of standard deviation and the corresponding percentage of data included, for a perfect normal distribution.

3.5 Most appropriate IOI tolerance

Prior to considering an appropriate measure of IOI deviation we must first determine an appropriate measure of success. If we define valid beats that are detected by an arbitrary beat analysis system as A, and valid events which are not detected as B, then a percentage measure of recall can be defined as follows:

$$\text{Equation 1: Recall} = \frac{A}{(A+B)} \times 100$$

We processed the 100 sets of tapping data to determine values of recall according to

different IOI tolerances. The six different tolerance values we used are multiples of standard deviation (SD), starting from a value of 0.5SD and proceeding to 3SD in half SD increments. The results are shown in Figure 3.5-1. At the lowest tolerance level of 0.5SD, on average only 68.7% of the valid beats in the 100 sets of tapping data are detected, with the lowest level of recall for one song being 47.9%, and the highest level of recall being 94.8%. In contrast, the highest tolerance level of 3SD results in an average of 98.9% of valid beats being detected, with the lowest level of recall for one song being 94.8%, and the highest level of recall being 100%.

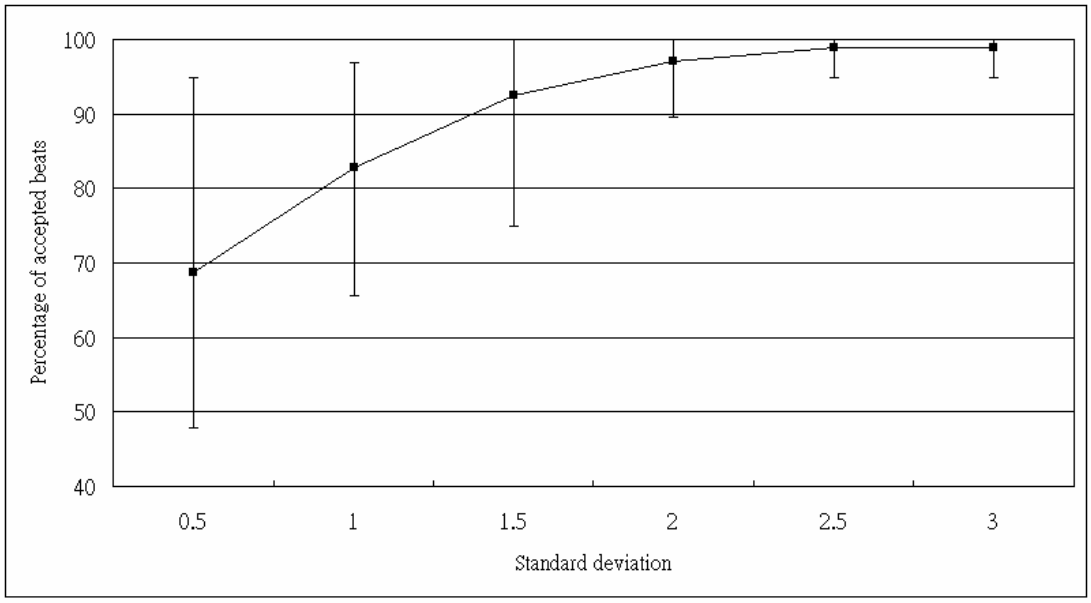

Figure 3.5-1: Recall measures for the 100 sets of tapping data at varying levels of standard deviation. The maximum and minimum values of the candlestick line indicate the best and worst recall measures encountered.

Acoustic events may be falsely detected as valid beat events by a beat detection algorithm. If we use C to represent acoustic events falsely identified as valid beats by the system then a measure of precision can be defined as follows:

$$\text{Equation 2: Precision} = \frac{A}{(A+C)} \times 100$$

An accurate understanding of precision would be of great assistance in designing beat analysis systems. However, there are significant problems in measuring precision in the context of this work. For example, in order to assess precision we could artificially introduce noise into our tapping data, and then determine the value of precision, but the result would be dependant on the quantity and distribution of such artificially introduced beats. Furthermore, the quantity of false events accepted as true by a system depends greatly upon the algorithm being used. Therefore it is not possible to measure precision in the context of our study.

However, it is possible to estimate the value of C across a range of values for IOI deviation, and hence to suggest the balance between A, B, and C. In Figure 3.5-2, the three measures are shown contrasted across a range of IOI tolerance values, expressed as a percentage. The section of the graph concerning valid beats not detected by a system is based upon our experience in processing the 100 sets of tapping data (see Figure 3.5-2, to be discussed later). The balance between valid beats and invalid beats accepted by the system is derived from heuristic experience. Clearly, the goal is to choose a level of tolerance which maximizes A and minimizes B and C. With reference to Figure 3.5-2, it can be seen that this level of tolerance would lie somewhere in the range 10-20%.

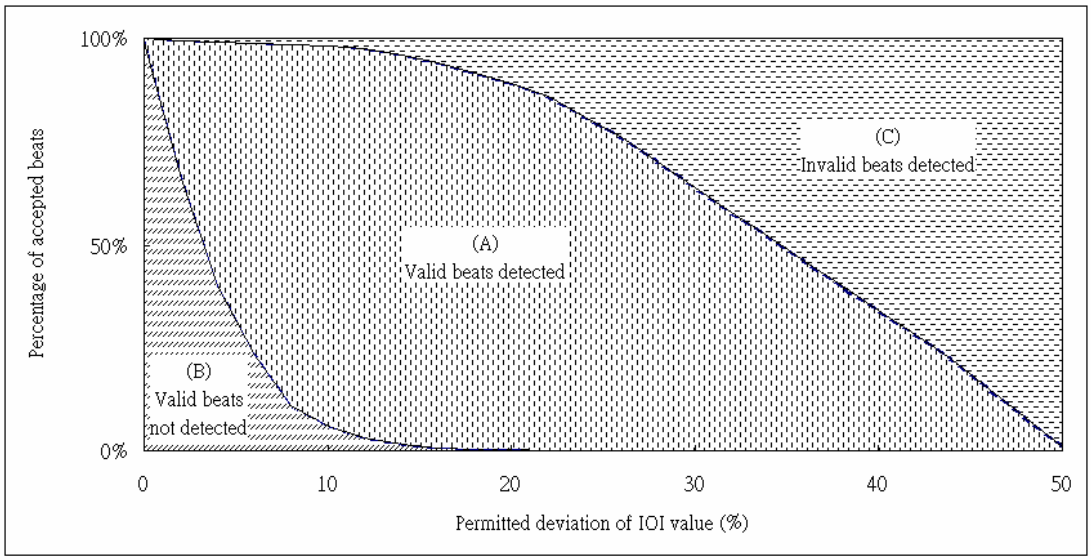

Figure 3.5-2: A partly heuristic illustration of valid beats not detected, valid beats detected, and invalid beats detected for an arbitrary beat detection system, across a range of IOI tolerance values expressed as a percentage of the IOI value. The portion shown assigned to B is based on the results of our analysis discussed later, shown in Figure 3.6-2.

After considering the results of our recall analysis we suggest that an acceptance level of 2SD is an appropriate compromise between valid and false beat acceptance. At this level the majority of the valid beat data is accepted (97.1% on average, as illustrated in Figure 3.5-1). The worst recall for any single set of tapping data in our set of 100 was 89.6%, and the best result was 100%. Tolerance levels higher than 2SD invite the possibility that too many false events become accepted as valid beats; tolerance levels lower than that reduces the number of accepted valid events to an unacceptably low level.

The level of 2SD can be approximated by the following expression of a power curve of best fit:

Equation 3: Permitted IOI deviation = $320.67x^{-0.3388}$, where x is the tempo value.

3.6 A comparison with previously published tolerance values

3.6.1 A survey of previously published values

We carried out a survey in order to consider our suggested expression for IOI tolerance in the context of previous research. We found many papers did not state either the permitted IOI tolerance or the permitted BPM tolerance. Some beat estimate research does not employ an IOI tolerance at all (for example, the beat tracking algorithm described by Sethares [27]). As such techniques cannot easily be considered within the context of this thesis they have not been included.

Our survey included every paper submitted for two recent research competitions in which tempo estimation was a primary element. Those conferences are the International Conference on Music Information Retrieval (ISMIR 2004) Audio Description Contest [10] and the Music Information Retrieval Evaluation eXchange (MIREX 2005) [18] competition. In total, we identified 11 papers that stated an accepted tolerance for the final BPM result, 7 papers that stated the permitted tolerance values for individual IOI values, and only 3 papers that stated both. These are summarized in Table 3.3. The values in column two are those for d shown in Figure 2.2-1, and the values for column three are those of a shown in Figure 2.2-2.

Paper	IOI tolerance	BPM tolerance (%)
Klapuri [10]	17.5%	10
Gouyon [11], Peeters [12], Uhle & Herre [13], Gouyon & Dixon [14]; *ISMIR 2004*	-	4
Alonso [15], Alonso [16], Alonso [17]	-	5
Davies & Brossier [19]; *MIREX 2005*	-	8
Davies & Plumbley [20]	15%	10
Klapuri [21]	-	10
Hainsworth & Macleod [22]	15%	20
Dixon [24]	70ms	-
Jensen & Andersen [25]	50ms	-
Dixon [23]	40ms	-
Jensen & Andersen [26]	20ms	-

Table 3.3: IOI and BPM tolerances stated in previously published papers. An empty cell indicates that the corresponding value was not stated in the paper. ISMIR and MIREX refers to the two contests concerning tempo estimation described in the main text. When assessing the results of tempo estimation algorithms tolerance values of 4% and 8% respectively were used in those competitions.

3.6.2 Permitted IOI tolerance and our study

As demonstrated previously, there is a negative relationship between beat deviation and BPM. It therefore makes sense if this relationship is employed when determining a value for permitted IOI time deviation. That is, relatively longer IOI

values should have a relatively longer permitted deviation. One way to achieve this is by using an IOI tolerance value which takes the form of a percentage measure of the IOI value. In our review we found several papers that did this. As summarized in Table 3.3, two different values of 15% and 17.5% were encountered. Other papers that stated tolerance values for IOI used fixed time values, meaning that their method did not exploit the relationship between deviation and BPM. We encountered values of 20, 40, 50 and 70ms. Both the percentage and fixed values we encountered have been plotted in Figure 3.6-1, together with our suggested curve previously labeled Equation 3.

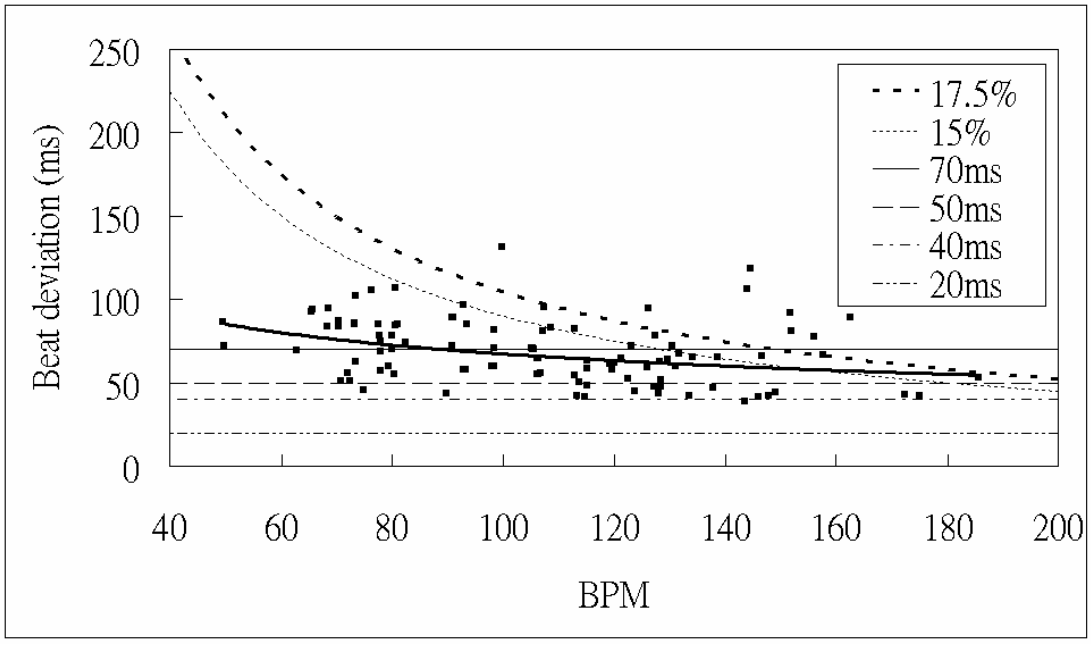

Figure 3.6-1: Permitted IOI deviation for papers included in Table 3.3, with our suggested curve "$y = 320.67x^{-0.3388}$" shown superimposed. Each of the 100 plotted data points represents the 2SD point for one of the sets of tapping data in our study.

One question of interest is the performance of previously published tolerance values. We took the permitted IOI tolerance values from the published papers listed in Table 3.3 and applied them to our 100 sets of tapping data and determined the corresponding recall measures. The results are shown in Figure 3.6-2.

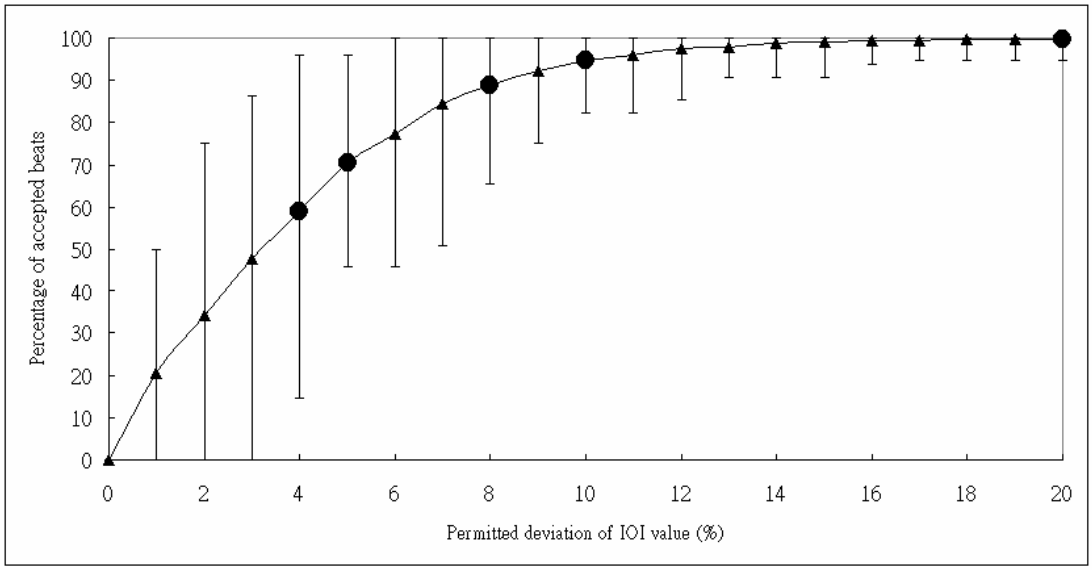

Figure 3.6-2: Recall measures across a range of permitted IOI deviation values. Each data point represents the result of processing 100 sets of tapping data. The candle bar for each data point indicates the lowest and highest recall measures encountered when processing one of the tapping sets. The five circles indicate values that have previously been used in published research, shown in Table 3.3.

As would be expected, levels of recall increase as IOI tolerance increases. The lowest published IOI tolerance we encountered was 4%. Figure 3.6-2 reveals that at that level of tolerance, only 59.1% of valid beats would have been detected if that tolerance value was applied to our 100 sets of tapping data. That is, the tolerance value would have resulted in a failure to detect approximately 40.9% of the beats. The next smallest tolerance we encountered, that of 5%, would have resulted in

29.4% of all beat events not being detected if it had been used on our song set. The largest tolerance value we encountered, that of 20%, is shown at the rightmost point. It would have resulted in 99.75% of all beat events in our 100 sets of tapping data being detected, with only 0.25% of beat events missed. However, as discussed before, this data should be interpreted in conjunction with measures of precision, and that is not within the scope of this thesis.

3.6.3 Permitted BPM tolerance and our study

When assessing the performance of a tempo estimation algorithm, all previous research that we encountered used a percentage measure to express the range within which the BPM result was regarded as correct. The resulting values are shown in Figure 3.6-3 together with our curve.

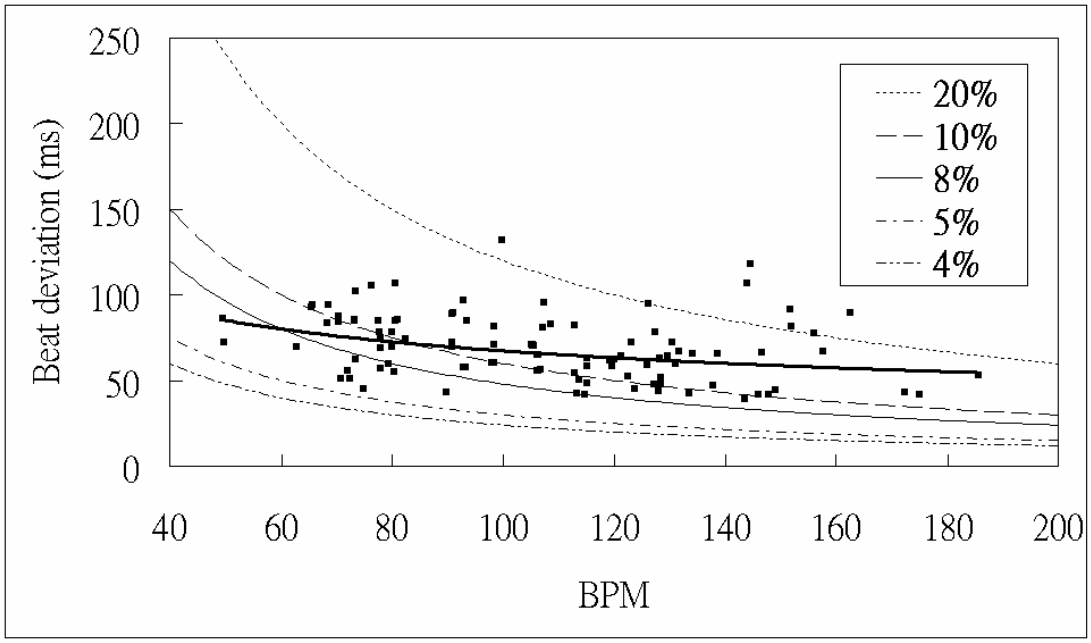

Figure 3.6-3: Accepted tolerance of the final BPM result for papers included in Table 3.3, with our suggested curve "$y = 320.67x^{-0.3388}$" shown superimposed. Each of the 100 plotted data points represents the 2SD point for one of the sets of tapping data in our study

As discussed before, our curve is selected to approximate the 'natural' beat deviation of the 50 songs in our set. We believe it is appropriate to apply the same measure of deviation to BPM tolerance. Accordingly, we suggest that many of the values used in previous research are inappropriate. Values of 4%, 5% and also 8% appear to be too restrictive for measures of BPM tolerance. Higher values of 10% come close to our curve; yet we encountered higher values still, (20% in one paper) giving a range which includes almost every beat in our set of 50 songs, however this may be too liberal. We propose that the same power curve equation of Equation 3 would be a more appropriate choice. In the event that a percentage measure is preferable, our analysis demonstrates that 10% or 12% would have similar results.

3.7 Conclusion

In this chapter we have looked at the two related issues of permitted beat deviation and BPM deviation. By analyzing the performance of musicians entering tapping sequences for 50 songs we determined appropriate expressions for permitted beat deviation and BPM deviation. Both parameters are considered in the light of previous studies. We are hopeful that this work will improve the performance of all algorithms which require some form of beat identification. Our system would use Equation 3 for the permitted IOI deviation.

CHAPTER 4

DISCRETE WAVELET TRANSFORM

4.1 Introduction of discrete wavelet transform

In this chapter, the details of the discrete wavelet transform are described. After the analysis of the input data, the next stage is to use the discrete wavelet transform to change the input data from the time domain into the frequency domain. It is very common to analyze the audio data in the frequency domain, as some important features can only be seen in the frequency domain.

In order to explain what the discrete wavelet transform is, we must first explain what a continuous wavelet transform is. This is because a discrete wavelet transform is devised by a continuous wavelet transform. So, we would like to introduce what a continuous wavelet transform is. It is defined as the sum over all the time of the signal multiplied by scaled, shifted versions of the wavelet function Ψ:

Equation 4: $C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position, t)dt$

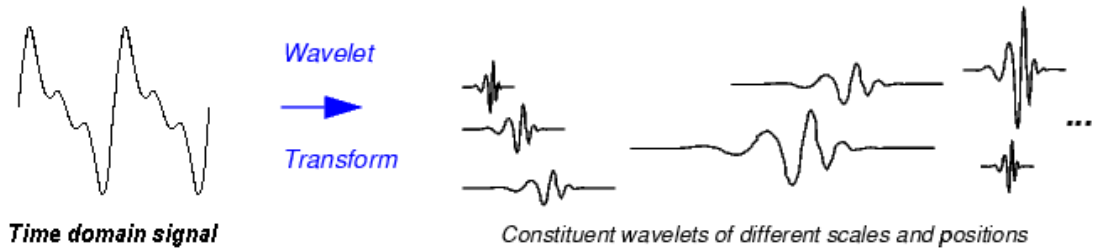

Figure 4.1-1: A continuous wavelet transform breaks down a signal into constituent wavelets of different scales and positions

A continuous wavelet transform breaks down a signal into constituent wavelets of different scales and positions (see Figure 4.1-1). The results of the CWT are many wavelet coefficients C, which are a function of scale and position. Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelets of the original signal.

It is a fact that the output of a wavelet transform is a time-scale view of a signal. What is the meaning of scale? Scaling a wavelet means stretching or compressing it (see Figure 4.1-2). The highest scale corresponds to the most stretched wavelet. It compares the longest portion of the signal, so is responsible for the low frequency part.

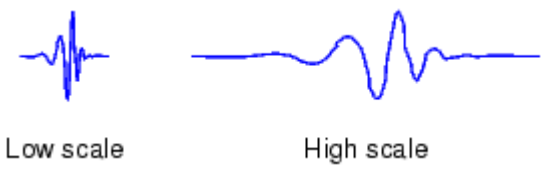

Figure 4.1-2: Low scale (left) and high scale (right) of the wavelet

Another feature of the wavelet is the position. It is determined by the shifting of a wavelet. Shifting means delaying or hastening its onset (see Figure 4.1-3).

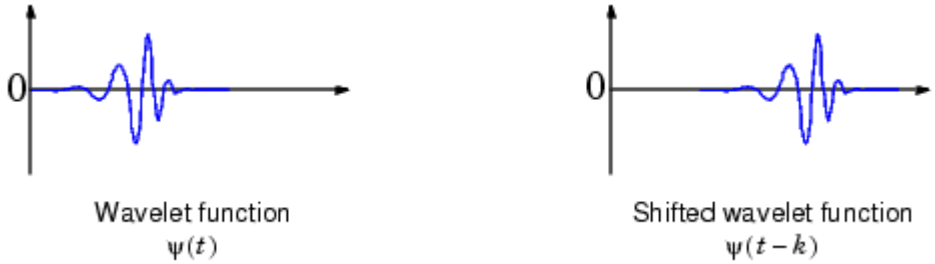

Figure 4.1-3: Original wavelet function (left) and shifted wavelet function (right)

After knowing what scaling and shifting are, how does the wavelet transform actually work? This process produces wavelet coefficients that are a function of scale and position. There are a total of five steps to perform this process:

I. Take a wavelet and compare it to a section at the start of the original signal.

II. Calculate the coefficient, C, that represents how closely correlated the wavelet is with this section of the signal. The higher C is, the more the similarity. Figure 4.1-4 shows an example that results in a low value of C because it is not similar. More precisely, if the signal energy and the wavelet energy are equal to one, C may be interpreted as a correlation coefficient.

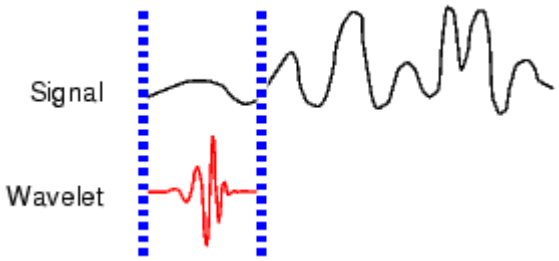

Figure 4.1-4: Calculate a coefficient, C, to show how similar the wavelet and the signal are.

III. Perform a time shift so as to shift the wavelet to the right. Repeat steps 1 and 2 until you have covered the whole signal (see Figure 4.1-5).

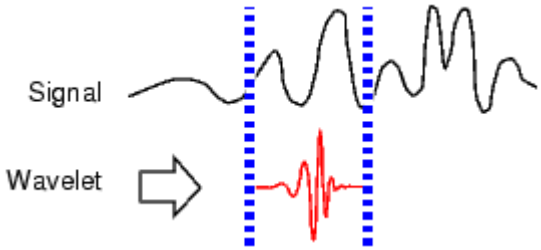

Figure 4.1-5: Shift the wavelet to another position

IV. Scale the wavelet and repeat steps 1 through 3 (see Figure 4.1-6).

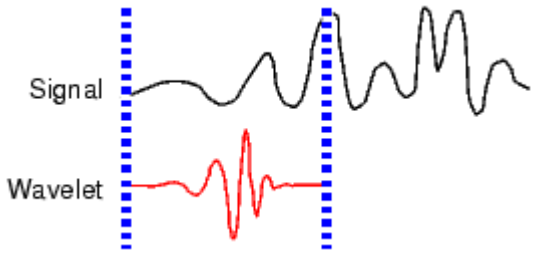

Figure 4.1-6: Scale the wavelet by stretching it

V. Repeat steps 1 through 4 for all scales.

When the above five steps are done, the coefficients produced at different scales by different sections of the signal are obtained. However, calculating the coefficients at every possible scale is a lot of work. It is possible to choose only a subset of scales and positions to be much more efficient. If we choose the scales and positions based on powers of two, it is called the discrete wavelet transform.

4.2 Why the discrete wavelet transform is used

There are a few methods that perform a transformation from time domain to frequency domain. The reason this system uses the discrete wavelet transform but not the other transformations is discussed in the following paragraphs.

Some of the tempo extraction algorithms analyze the signal in the frequency domain, so, we need to apply a transformation to change the input signal from the time

domain into the frequency domain. There are many types of transformation which can perform the above transform, such as the Fourier transform (see Figure 4.2-1).

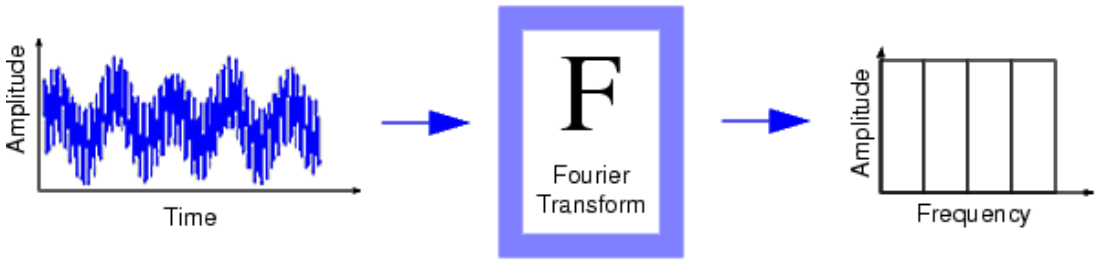

Figure 4.2-1: The original signal in the time domain (left). After the Fourier transform, the signal is transformed into the frequency domain (right)

The Fourier transform breaks down a signal into constituent sinusoids of different frequencies (see Figure 4.2-2). This is extremely useful because the signal's frequency content is very important.

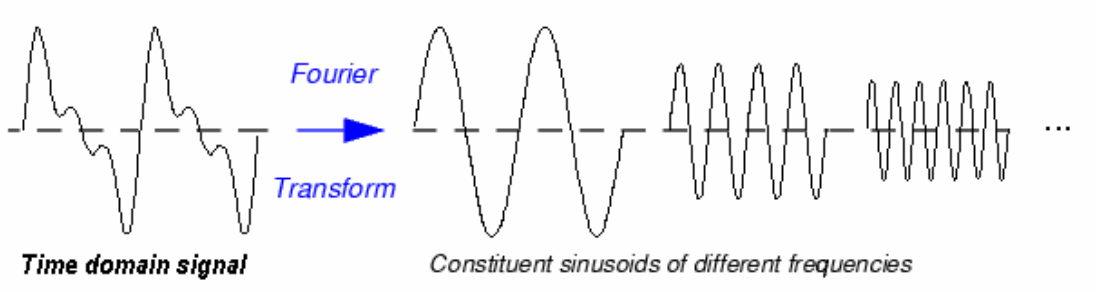

Figure 4.2-2: A Fourier transform breaks down a signal into constituent sinusoids of different frequencies with different amplitudes

However, it has a serious drawback. After the signal is transformed to the frequency domain, time information is lost. When looking at a Fourier transform of a signal, it is impossible to tell when a particular event took place. If the signal properties do not change much over time, that is, it is a stationary signal, this drawback is not very

important. However, in our case we want to extract the tempo from a signal which is music from an audio CD, which would not be a stationary signal. So, the Fourier transform is not suitable as a tempo extraction algorithm.

In order to correct this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyze only a small section of the signal at a time. This technique is called windowing the signal. The whole method is called short-time Fourier transform (STFT) (see Figure 4.2-3). By using STFT, it maps a signal into a two-dimensional function of time and frequency.

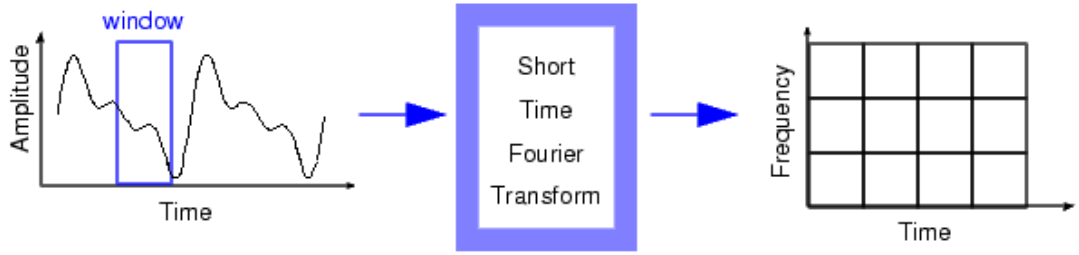

Figure 4.2-3: The original signal in time domain (left). By using the short time Fourier Transform with windowing technique, the output would become frequency against time (right)

The STFT provides some information about both when and at what frequencies a signal event occurs. This information can be obtained with limited precision, and that precision is determined by the size of the window. However, when you choose a particular size for the time window, the window size remains the same for all

frequencies. Many signals require a more flexible approach, such as varied window sizes, to determine more accurately either time or frequency.

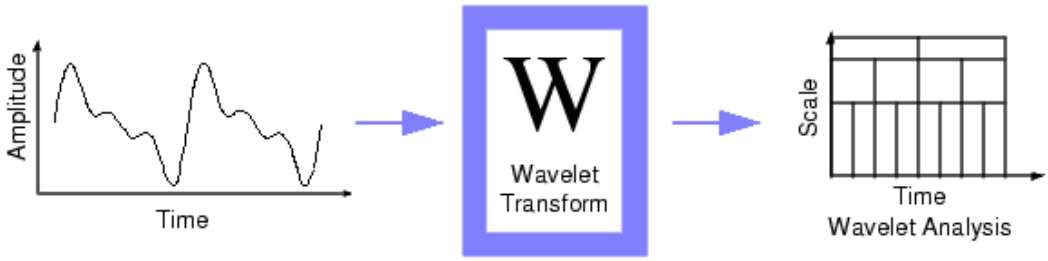

Figure 4.2-4: The original signal in time domain (left). By using the Wavelet Transform, the output would become scale against time (right). Low scale means high frequency while high scale means low frequency

A wavelet transform uses the windowing technique with variable sized regions (see Figure 4.2-4). This allows the use of long time intervals when we want more precise low frequency information, and shorter time intervals when we want high frequency information.

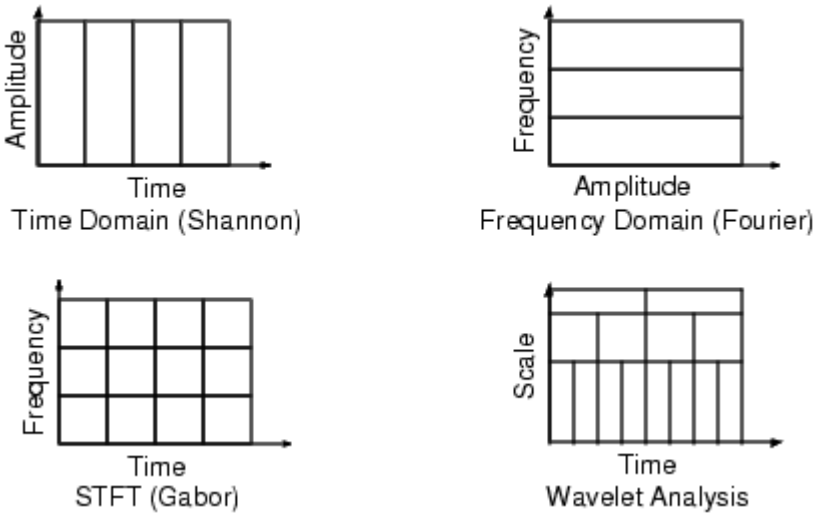

Figure 4.2-5: Time domain, frequency domain, short time Fourier transform and wavelet analysis views of a signal

A comparison between time-based, frequency-based, STFT and wavelet analysis can be easily seen in Figure 4.2-5. Wavelet transform is the best among these for the purpose of tempo extraction because it contains the time information with a flexible window size. So, it is used for our tempo extraction system. The details of how to apply this transformation are discussed in section 5.2.

CHAPTER 5

TEMPO EXTRACTION ALGORITHM

5.1 Overview of the system

For our tempo extraction system, there are total five stages from top to bottom (see Figure 5.1-1). First, a discrete wavelet transform is applied to the audio data for four iterations to obtain the DWT coefficients, cD_1 to cD_4. Second, a peak detection is performed for cD_1 to cD_4 concurrently. After getting the peaks of these coefficients, the beat intervals are calculated from these peaks. Then, we combine four sets of beat interval information to create a histogram. Finally, we improve the histogram by smoothing with a Gaussian function. After these five stages, the tempo of the audio data can be obtained.

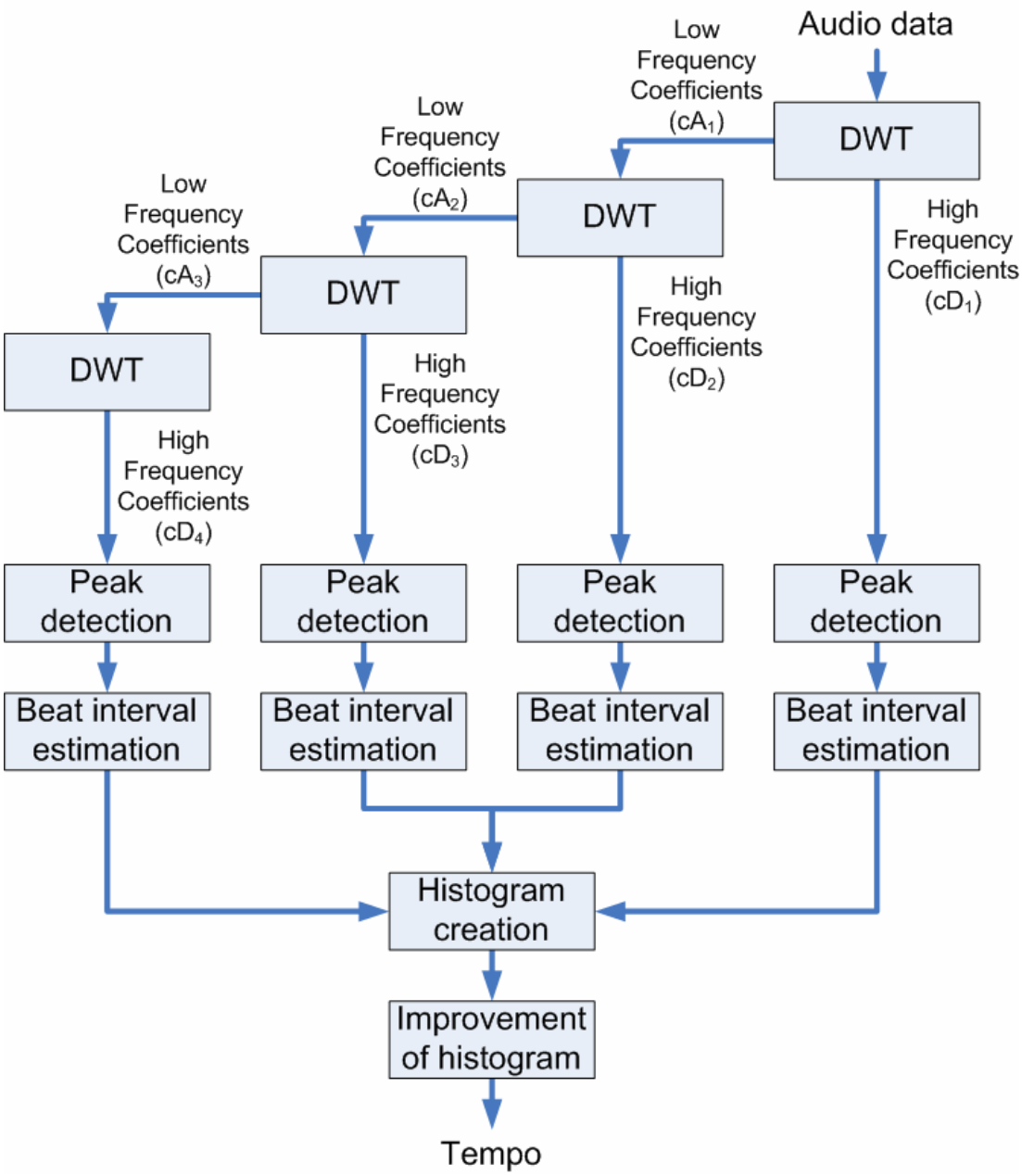

Figure 5.1-1: Overview of the tempo extraction system

5.2 Data Input

Any audio file can be the input data. It can be a song or some music without vocals. The duration of the audio can be any length but with a minimum of 5 seconds. The tempo cannot be accurate if the duration is less than 5 seconds for our system.

5.2.1 Input file format

For the file format, our system supports the Microsoft WAV and mp3 format. For a better result, an uncompressed WAV file is preferred. This is because mp3 format has been compressed so that some raw data has been lost in order to decrease the file size. It is the same for the compressed WAV file format. So, an uncompressed WAV is the best format as none of the raw data have been altered.

The sampling rate is also important. It is better to use a 44100Hz WAV file than a 22050Hz or lower sampling rate. The Nyquist-Shannon sampling theorem states that the sampling frequency has to be greater than twice the Nyquist frequency. In other words, the sampling frequency must be at least twice the maximum frequency component of the signal. If the sampling rate is smaller than or equal to twice the Nyquist frequency, the high frequency components will no longer be reconstructed. Figure 5.2-1 shows an example that using a low sampling rate to sample a high frequency sinusoidal signal. As a result, the high frequency components will become low frequency components as shown in Figure 5.2-1. Then those components mix with the original lower frequency components. This is called aliasing. In conclusion, a high sampling rate is important.

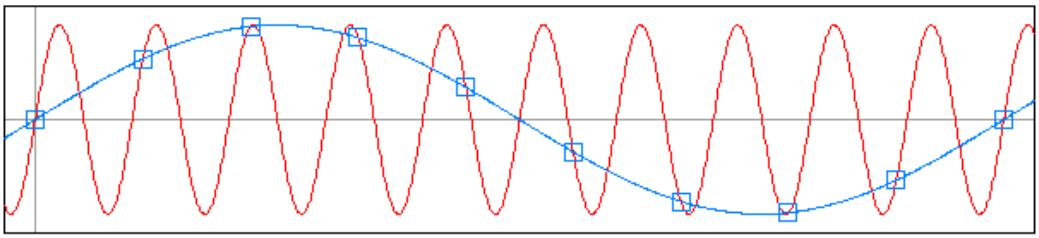

Figure 5.2-1: A sinusoidal signal of frequency f sampled using a sampling rate s where $s \leq 2f$

5.2.2 Our input data sets

In our system, two sets of input data are utilized:

I. One of them consists of 50 uncompressed WAV files that are extracted from audio CDs in a 44100Hz, stereo format. The duration of each song is at least 4 minutes. These 50 WAV files contain several genres as discussed in section 3.2.

II. The other set was used by a tempo induction contest which was organized during the International Conference on Music Information Retrieval (ISMIR 2004). It contains 698 excerpts of dance music and 465 song excerpts of 30 seconds and 20 seconds length respectively. They are all uncompressed WAV files in 44100Hz and mono format.

For the input data, we need to have a tempo value in order to compare our system result. We define these tempo values to be the ground truth tempo. We obtain the

ground truth data of our 50 songs by the experiment described in section 3.3. If two subjects agree on the time signature, the ground truth is obtained by taking the average of two tempos. If not, the author of this thesis would be the third party to determine the time signature. The tempo derived by the subject, which is not the same time signature as the author, are multiplied or divided by two or three to make the time signature the same. Finally, taking the average of these two tempos will give the ground truth tempo. For example, assume that subject A says 90 BPM and subject B says 160 BPM. As the two subjects do not agree on the time signature, the author as a third party will decide on the time signature. If it is a relatively slow song, the tempo of subject A might remain 90 BPM while that of subject B becomes 160 BPM divided by 2, which is equal to 80 BPM. Finally, the ground truth tempo of this example is 85 BPM by taking the average of these two values.

5.3 Applying the discrete wavelet transform

A discrete wavelet transform is applied to all these input data to change from time domain into frequency domain for further analysis. In discrete wavelet transform, we use the term *approximations* and *details*. The approximations are the high-scale, low-frequency components of the signal, while the details are the low-scale, high-frequency components.

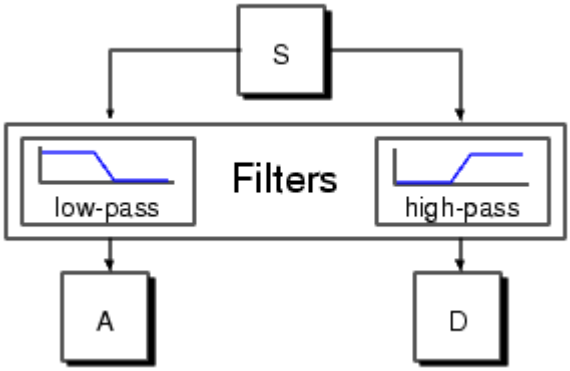

Figure 5.3-1: Signal S passes through two filters to get signal A and signal D

Consider an original signal, S, passes through two complementary filters and emerges as two signals A and D (see Figure 5.3-1). Note that the number of samples of output signal A and that of output signal D combined together is twice as much data as we started with. Suppose the original signal S consists of 1000 samples of data. Then the resulting signals A and D would total 2000 samples of data. However, we want 1000 samples instead of 2000 samples. We use a technique called downsampling. For this we can use different downsampling factors. As we want to get 1000 samples instead of 2000 samples, we use the downsampling factor equal to 2. So, we may keep only one sample out of two in both A and D. After performing downsampling, we get two sequences called cA and cD (see Figure 5.3-2). These sequences are called DWT coefficients.

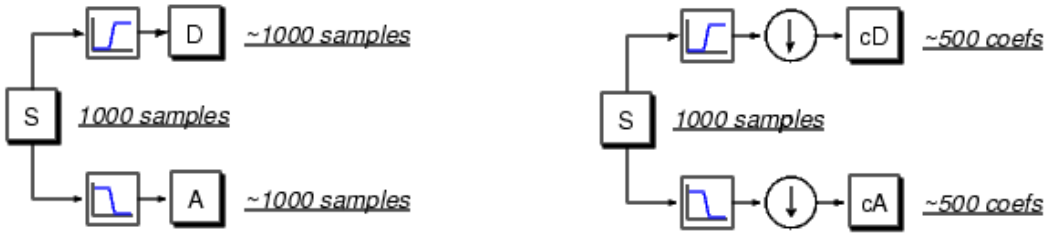

Figure 5.3-2: Signal A and D get 1000 samples (left). Signal cD and cA get 500 coefficients by downsampling (right). (↓) in the right figure means downsampling.

To gain a better appreciation of this process, let us perform a DWT of a signal (see Figure 5.3-3). Our signal, S, is a pure sinusoid with high-frequency noise added to it. The detail coefficients cD are small and consist mainly of a high-frequency noise, while the approximation coefficients cA contain much less noise than the original signal S. This is because the detail coefficients cD filter out the low frequency components of the original signal which remains the high frequency components that in this example are the noise.

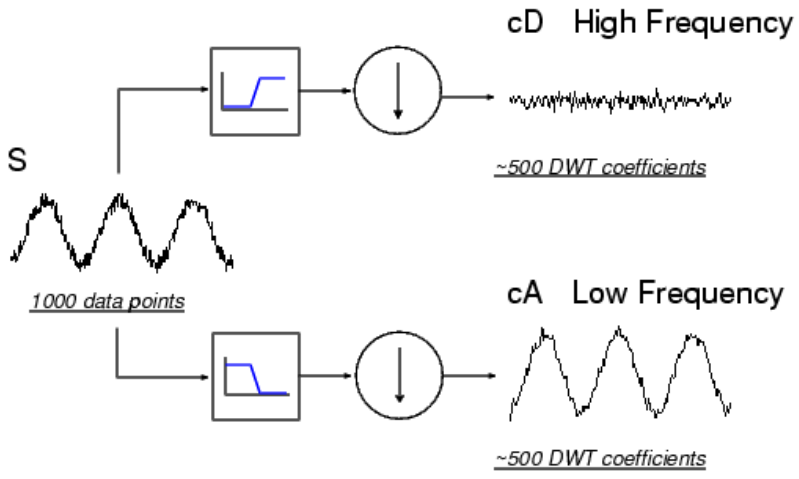

Figure 5.3-3: A DWT is applied to signal S to obtain cD and cA

The decomposition process can be iterated, with successive approximations being decomposed in turn. The original signal is broken down into many lower resolution components. This is called the wavelet decomposition tree (see Figure 5.3-4).

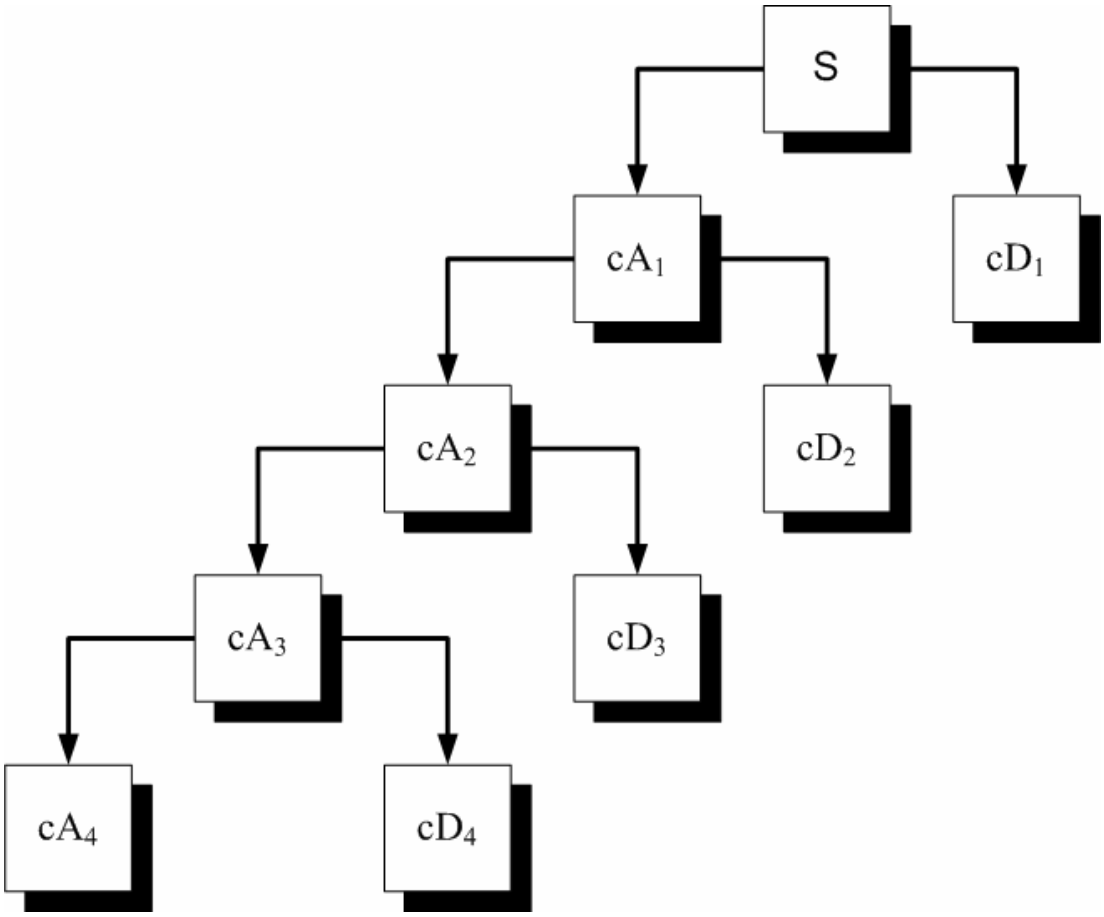

Figure 5.3-4: A wavelet decomposition tree with four iterations

For our tempo extraction system, we use four iterations to obtain cD_1 to cD_4. For an input data which is in format 44100Hz, the frequency band of cD_1 to cD_4 is shown in Figure 5.3-5.

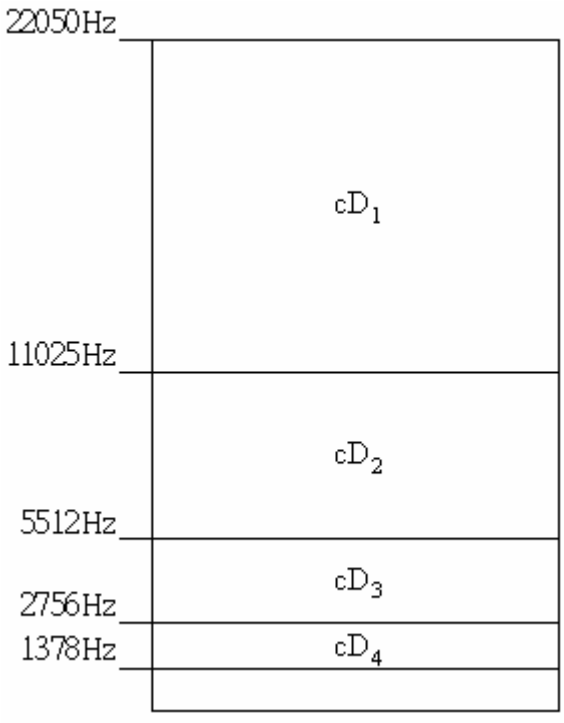

Figure 5.3-5: The frequency band of cD_1 to cD_4 if the input data is in format 44100Hz

We choose several iterations because different songs are better for tempo analysis at different frequency bands. We informally examined several songs. For frequencies lower than approximately 1000Hz we found too much acoustic influence by audio sources which were often not directly relevant to the beat of the music. Therefore more than four iterations is not appropriate for the tempo extraction system.

5.4 Peak detection

After we obtain cD_1 to cD_4 by four iterations of discrete wavelet transform, a peak detection process is applied for each of cD_1 to cD_4. We perform the peak detection because the beat may occur at the peak of the DWT coefficients. If all the peaks are

detected, we can focus on the peaks only to get the beat intervals. There are two stages in the peak detection process. They are full wave rectification and the moving window technique.

5.4.1 Full wave rectification

Full wave rectification is a process to convert all the negative values into positive values, while the positive values remain unchanged. Here is an example of a full wave rectification. The original signal is a sine wave, and the rectified signal is shown in Figure 5.4-1.

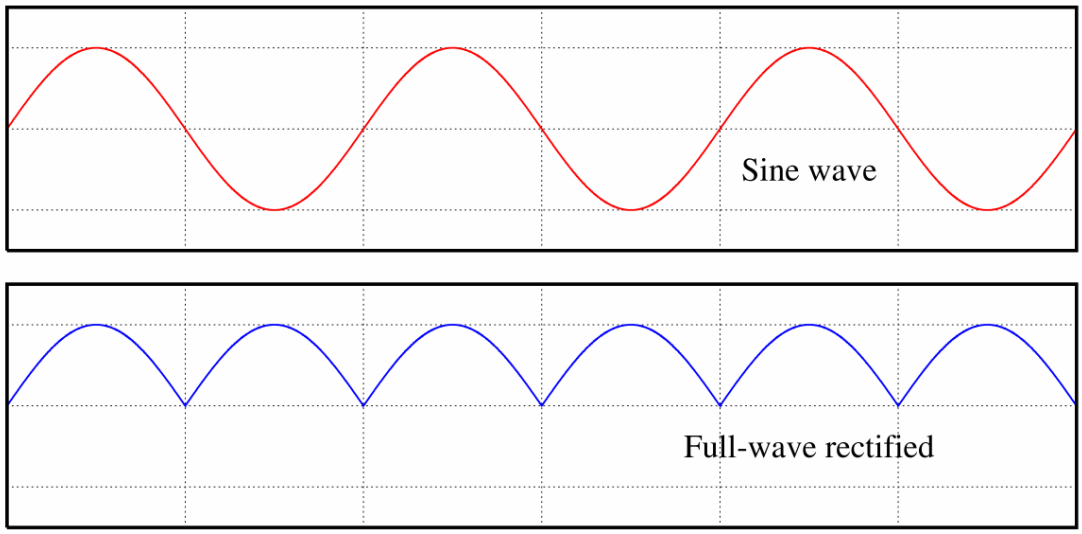

Figure 5.4-1: The original signal is a sine wave (upper graph). After applying full wave rectification, the rectified signal is obtained (lower graph)

The DWT coefficients contains positive values and negative values. We need to apply the full wave rectification to the DWT coefficients so that all the negative values become positive values. This is because peaks may appear in either the

positive or the negative values, so we need to compare both parts to find the peak. However, if full wave rectification is applied, all peaks can be obtained by finding the local maximum.

5.4.2 Moving window

After the full wave rectification is applied, all the DWT coefficients become positive values. We can obtain the peak by locating the local maximum of a moving window. For our system, we assume the tempo of the input is less than 240 BPM. For a song that is 240 BPM, it will have one beat every 0.25 second. By the above assumption, we can ensure there are no two peaks within 0.25 of a second. So, a window of 0.25 of a second is chosen. This window is slided along the entire DWT coefficients to find the local maximum.

Next, we need to define the step size of the moving window. If the step size is too small, the running time of the system is increased. If the step size is too big, the accuracy of the peak detection algorithm is decreased. There is a trade off for setting the step size. The step size of our system is 1/20 of the window's width, that is 12.5 milliseconds. So, for example, it follows that the window needs to move twenty times to process 0.25 of a second.

A peak is obtained if the local maximum position is the same for over 90% of the window width, that is more than or equal to 18 times. For example, you can see a moving window in Figure 5.4-2. There are overlapping windows moving along at 12.5 milliseconds for 20 steps. Point A, from the time that the window touches it, is a local maximum position. It is still the local maximum position when the window leaves point A. As the window moves 20 steps, point A is a local maximum position for 20 times while the window is moving. So, point A is considered to be a peak. Point B only reaches the maximum 9 times while the window is moving, so point B will not be considered a peak because its maximum is less than 18 times.

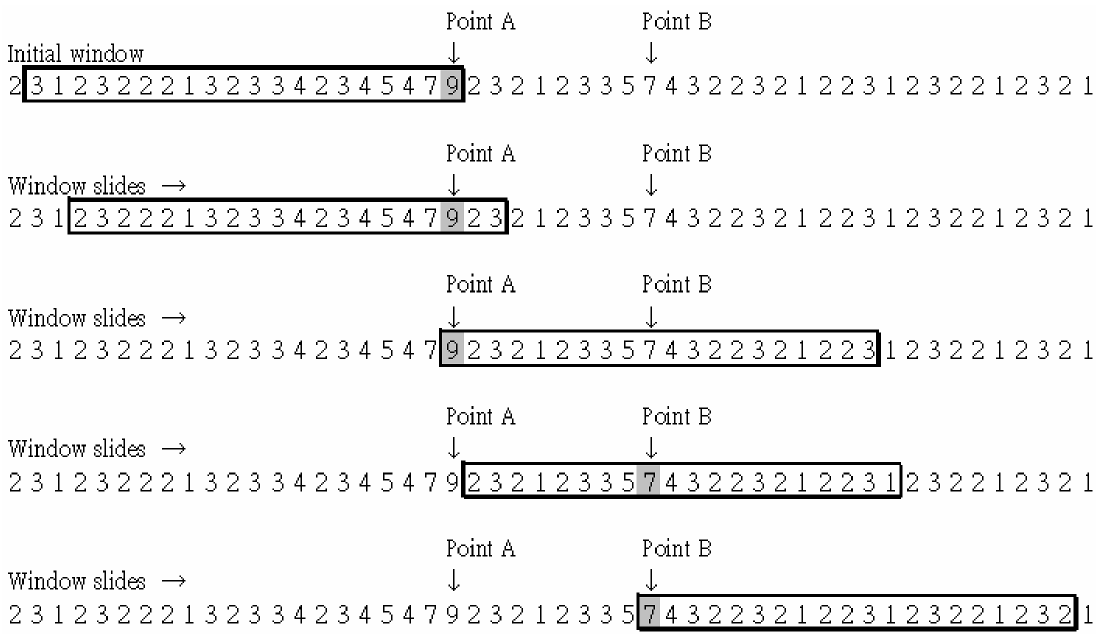

Figure 5.4-2: An example showing the moving window with each number representing the full wave rectified DWT coefficient. The width of the moving window is 0.25 of a second. The local maximum within the window is shown in gray. Point A is a peak as it is a local maximum over 90% of the assessed length of time while point B is not.

5.5 Beat interval estimation

After getting all the peaks from four DWT coefficients, we assume all these peaks are the beat positions. By calculating the time difference of each beat position, we can obtain N-1 beat intervals from N peaks. For example, four peaks are detected. The positions of these peaks are 0 second, 0.5 second, 1.1 second and 1.4 second as shown in Figure 5.5-1. We can calculate the time difference by subtracting from the neighbors. The beat intervals of these peaks are 0.5 of a second, 0.6 of a second and 0.3 of a second.

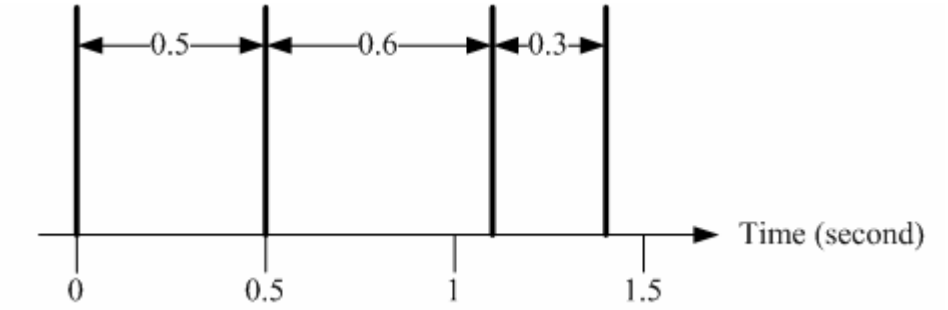

Figure 5.5-1: Three beat intervals are calculated from four peaks

5.6 Histogram of beat interval

By collecting all the beat intervals, we can create a histogram to analyze the data. Each beat interval is added to a histogram counting the number of occurrences. An example of a beat interval histogram is shown in Figure 5.6-1. The beat interval which has the maximum occurrence is most likely to be the tempo of the input data. From this example, we can see the maximum occurrence beat interval is 0.478 of a

second. We use the following equation to convert the beat interval into beats per minute (BPM).

$$\text{BPM} = 60 / \text{beat interval}$$

So, the tempo of this example is 60 / 0.478 = 125.5 BPM.

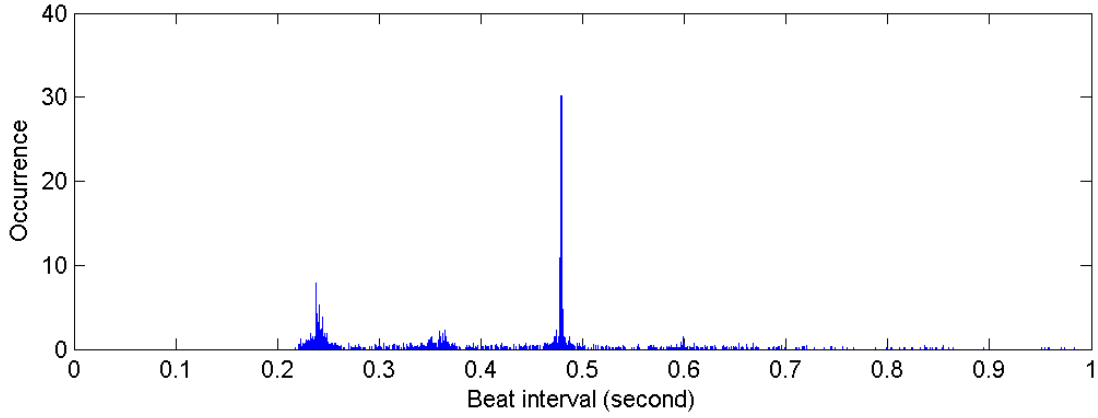

Figure 5.6-1: Histogram of beat interval

5.7 Improvement of the histogram

By using the above histogram alone, the performance of this tempo extraction may not be so good. In order to improve the performance of the system, we need to improve the histogram so as to make the system more accurate.

5.7.1 Introducing weight to the occurrence of beat intervals

We can process occurrences of beat intervals with some weights to increase the accuracy of the system. There are many methods to introduce weighting of

occurrence of beat intervals. We need to define the weight in a reasonable way in order to increase the performance.

For example, we get a beat interval called ‘B’. If all the neighbors have similar values of ‘B’, then the weight of ‘B’ should be very high. Similarly, if all the neighbors get different values of ‘B’, the weight of ‘B’ should be very low. So, we can define the weight according to how many neighbors of ‘B’ get similar values. There are two parameters that we can set. The first one is how many neighbors we should compare. The second one is the definition of similar values.

For the first parameter, our system uses four beat intervals on each side of ‘B’. Therefore, there are eight beat intervals for comparison. If we use too many beat intervals for comparison, the weight would be not accurate as the far neighbors may get a different value. If the number of beat intervals is too small, the weights are not so useful. So, our system uses a total of eight beat intervals for comparison (see Figure 5.7-1.

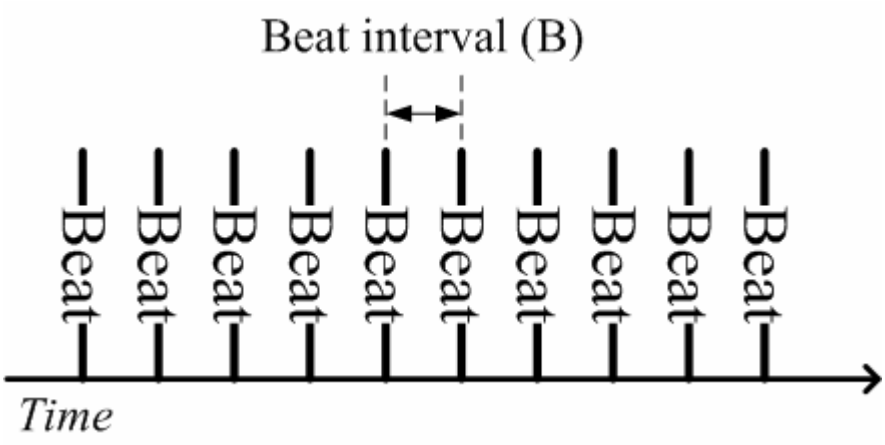

Figure 5.7-1: Eight beat intervals are used for comparison.

For the second parameter, as we discussed in section 3.5, the permitted IOI deviation should be a value which is relative to the tempo of the song. Our proposed value is calculated by the formula $y=320.67x^{-0.3388}$ where x is the tempo of the song and y is the proposed permitted IOI deviation value. However, we do not know what the tempo of the song is at this stage. So, we use the beat interval value 'B' for a reference in order to calculate the formula x = 60 / B. Then, we can get the value of y by using the above x. By using the value of y, we can define two IOIs are similar if the difference between y and the beat interval value 'B' is smaller than y (see Figure 5.7-2).

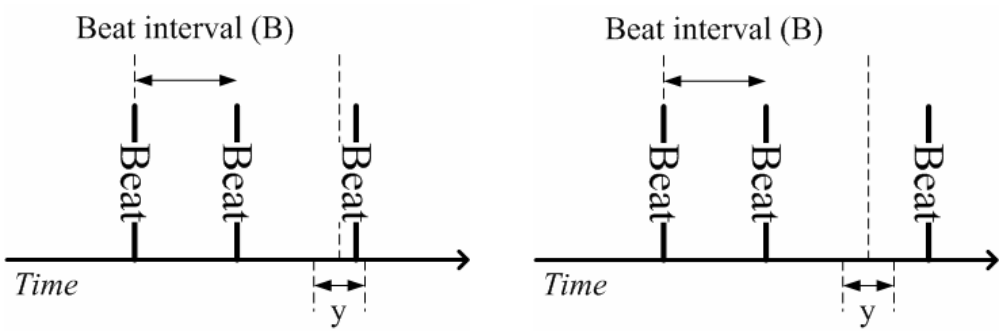

Figure 5.7-2: Similar beat (left) and not similar beat (right).

After fixing these two parameters, we calculate the weight by counting the number of similar beat intervals out of nine beat intervals. We use weighting of occurrence of beat intervals instead of the occurrence of beat intervals alone in order to increase the performance.

5.7.2 Smoothing out the histogram

Sometimes, the histogram would encounter some special cases.

I. Consider an example where the histogram of beat intervals gets multiple maximum occurrences (see Figure 5.7-3). In this example, which beat interval should we choose to calculate the tempo as there is more than one maximum occurrence point?

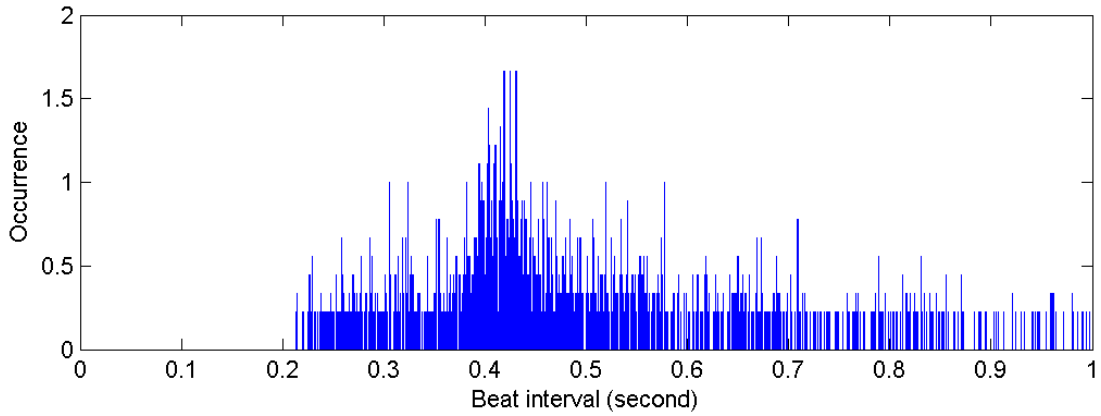

Figure 5.7-3: A histogram of beat interval which gets multiple maximum occurrences

II. Consider another case where the maximum occurrence occurs at the low density region of beat interval (see Figure 5.7-4). We can simply choose the maximum occurrence position to calculate the tempo. However, this maximum is not the majority in the distribution of the beat interval. The major region is around 0.4 second but not 0.8 second. So, it may not be accurate enough if we simply choose the maximum occurrence without consider the distribution.

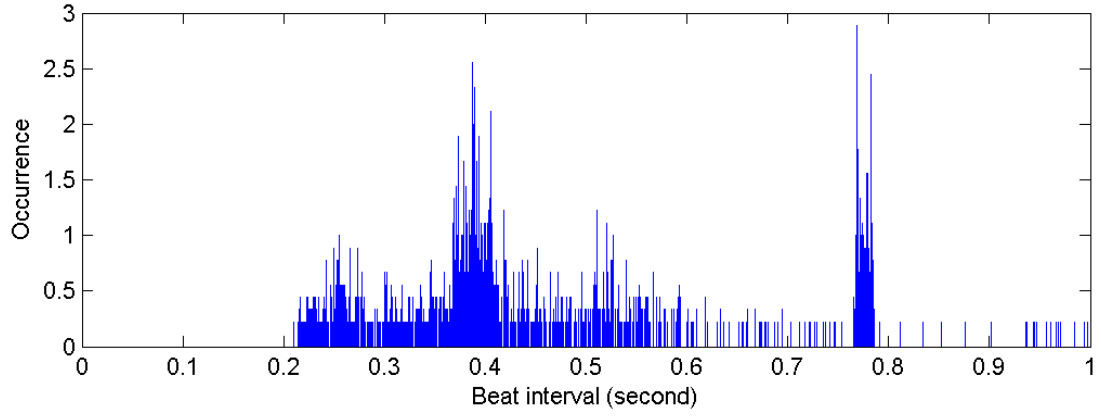

Figure 5.7-4: A histogram of beat intervals in which the maximum occurs at the lowest density region of beat intervals

For these two problems discussed previously, we can solve both problems with a single solution: smoothing the histogram by multiplying a Gaussian function to each data item. After the histogram is smoothed out, we can get the beat interval from the maximum amplitude.

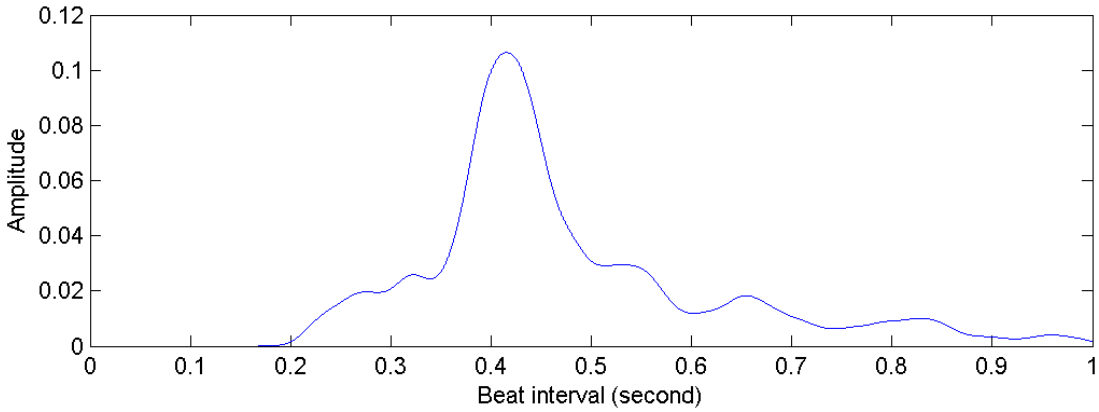

Figure 5.7-5: The smoothed histogram of the first example which is shown in Figure 5.7-3

Figure 5.7-5 shows the smoothed histogram of the first example. The multiple maximum occurrences have disappeared. So, we can now get the maximum amplitude of the new figure to calculate the tempo. For this example, the beat interval is 0.416 second at the maximum amplitude. So, the tempo is 144.2 BPM.

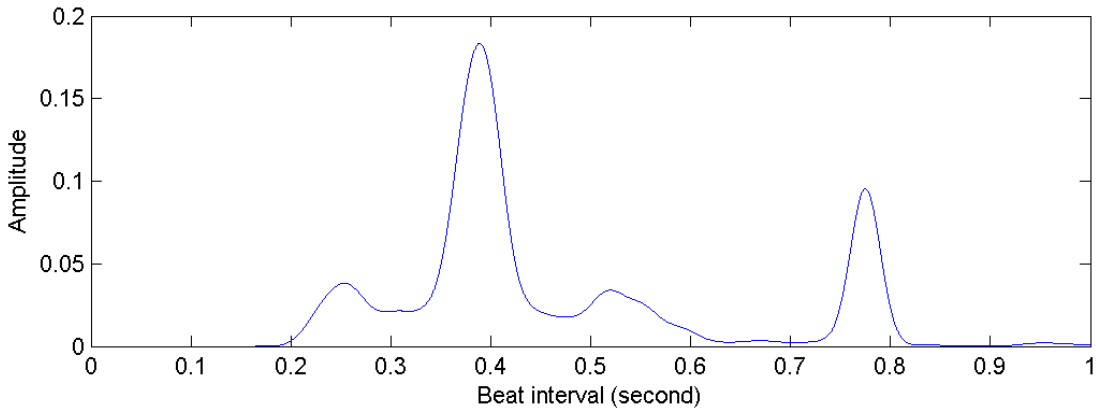

Figure 5.7-6: The smoothed histogram of the second example which is shown in Figure 5.7-4

The smoothed histogram of the second example is shown in Figure 5.7-6. The original maximum occurrence is no longer the maximum amplitude. This is because the Gaussian function can make use of the distribution of the beat interval. It will get higher amplitude in the major region while the minor region will get smaller amplitude. For this example, the beat interval is at 0.389 of a second at the maximum amplitude. So, the tempo is 154.2 BPM.

5.7.3 Analyzing the left and right channels

If the input is in stereo format, we can perform the tempo extraction algorithm in a different way. We can make use of the left and right channels to do a further analysis. If we listen to a song carefully, we may find that sometimes the sounds coming from the left channel are different to that of the right channel. For example, in some songs the background music is in the left channel and the vocals are in the

right channel. Also, different instruments often occur in different channels. This means that the beat may be easier to be extracted in a channel compared to another.

The overview of the system for stereo input is shown in Figure 5.7-7. For an input which is in stereo format, we treat it as three different inputs; the left channel, the right channel and the mono channel. The mono channel is the average of the left and right channels. We pass these three inputs into our system which are described in section 5.1. Then, we can select the best one to be our final result. The question is how to select the best one among these three histograms.

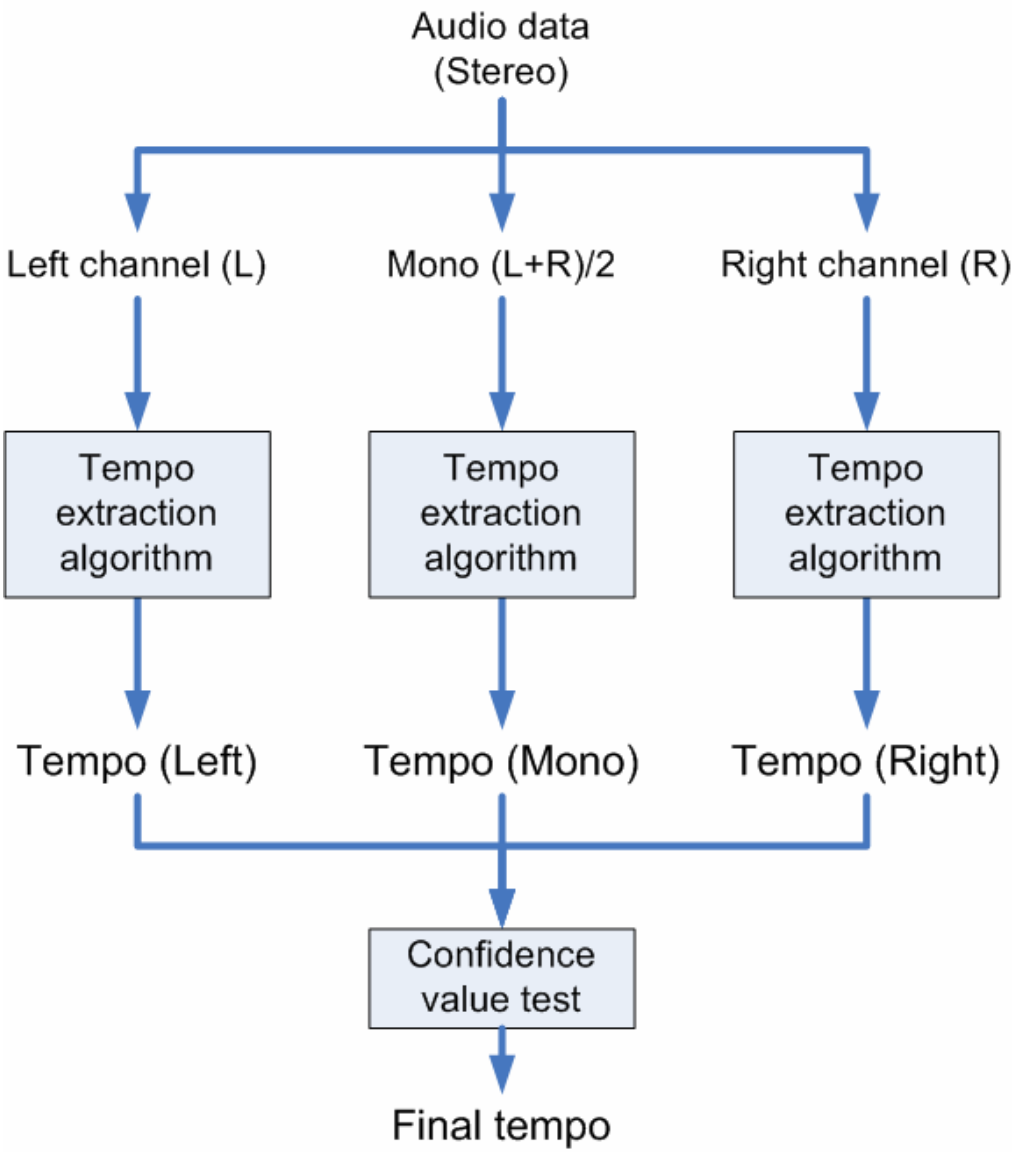

Figure 5.7-7: Overview of our algorithm if the input is in stereo format

We can first define a confidence value for a given histogram. Our system defines it as the area under the peak with a width of 0.1 of a second over the entire area under the curve (see Figure 5.7-8). We use the value 0.1 second because it is the same width as that of the 3SD of the gaussian function. Therefore, every histogram will get a confidence value. Finally, our system will choose the tempo by a confidence

value test, which obtains the final tempo by getting the maximum confidence value, to be the final tempo.

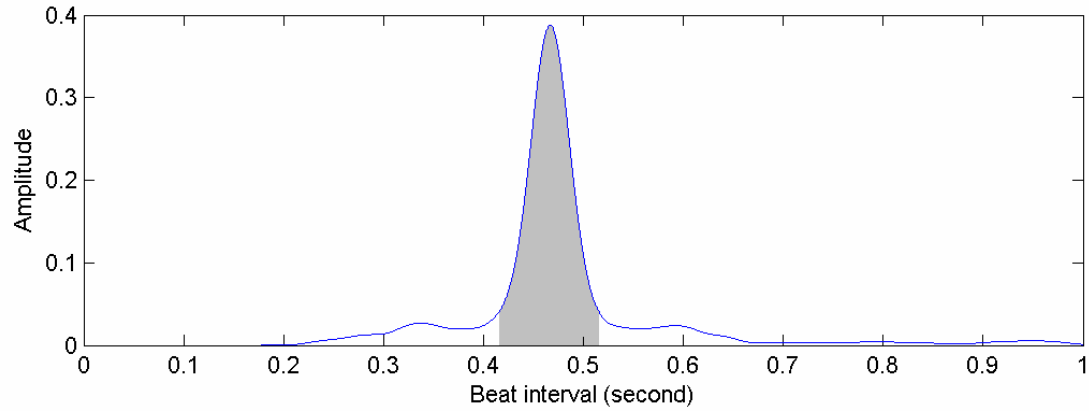

Figure 5.7-8: The confidence value is defined as the area under the peak with 0.1 second width (gray in color) over the entire area under the curve

CHAPTER 6

EXPERIMENTATION AND RESULTS

6.1 Implementation

In this chapter, we present the implementation of our system and some experimental results. The system was implemented using Matlab version 7.0.4. It was developed and run on a Pentium 4 2.4GHz machine with 512MB RAM operated in a Microsoft Windows XP environment.

For any input audio data, our system checks if the audio is stereo or not. If it is stereo, we can treat it as three different inputs as discussed in section 5.7.3. Next, the input is read into the memory. Our system can let the user select how many seconds to be analyzed and the starting time. By default, it would be the first 180 seconds of the input. We select 180 seconds because it is always the majority part of a typical song which is less than six minutes. If the input is less than 180 seconds, the entire input is read into memory.

After reading the input data into memory, a discrete wavelet transform is applied using the command ‘dwt’ in Matlab. After the transformation, we keep the DWT

coefficients only. The input data which is stored in memory are freed in order to have more free memory in the system.

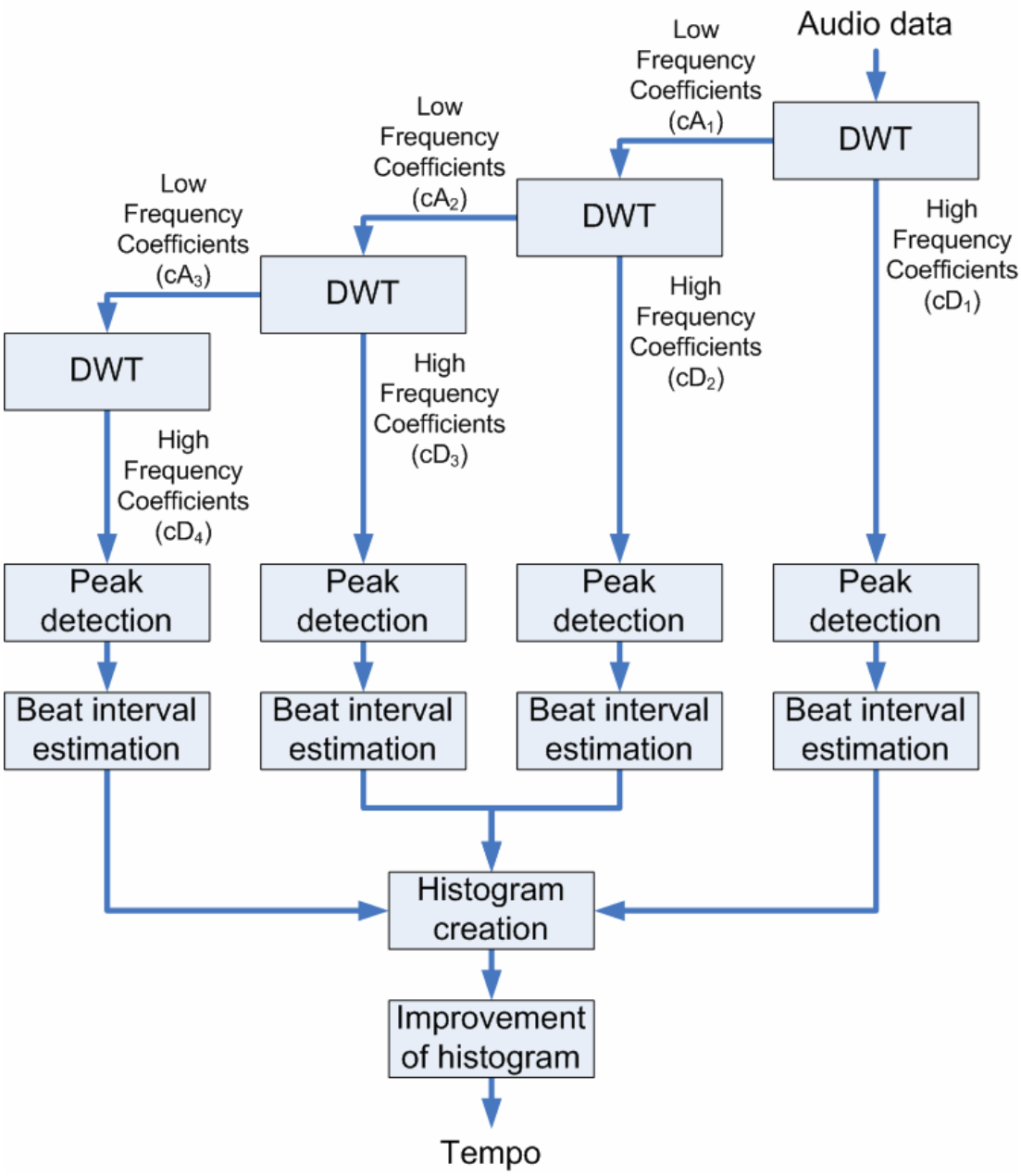

Figure 6.1-1: An overview of the tempo extraction system

For a reminder, the overview of the tempo extraction system is shown again in Figure 6.1-1. For these DWT coefficients, the detail coefficients cD consist mainly

of the high-frequency part of the input, while the approximation coefficients cA consist mainly of the low-frequency part. This is the first level of the transformation. We name the detail coefficients and the approximation coefficients as cD_1 and cA_1 respectively. For the cA_1, DWT is applied again for further decomposition. For our system, four levels of DWT is used. In the following part, we focus on the detail coefficients.

For the detail coefficients, we apply a peak detection to it. The details of the peak detection have been discussed in section 5.4. While we are writing the code, we need to remind ourselves that our system uses a 0.25 second width window to find the local maximum position. Then, the window moves with the step size to 12.5 milliseconds to find the next local maximum position until the window reaches the end of the detail coefficients using a while loop in Matlab. Every time a local maximum position is recorded, it will assign a counter for it. If the same position is acting as the local maximum, the counter of that position is increased by 1. In the end, we will typically end up with a lot of local maximum positions with counter values.

Remember that a peak is obtained if the local maximum position is the same for over 90% of the window width. The step size is 12.5 milliseconds and the window width is 0.25. That means a window needs to move 20 times for one window width.

So, a position can have a maximum counter value equal to 20. By calculating 90% of the maximum counter value, our system removes the positions which get a counter value less than 18. Finally, the remaining positions are the peaks.

Our system calculates the IOIs between all the peaks. The details have been described in section 5.5. For every IOI, a weight is introduced to it like we discussed in section 5.7.1. The weight is calculated by checking if the neighbor IOIs is within the beat deviation value. By setting a different beat deviation value, a different weight is obtained. We used the value which was suggested in section 3.5 for our system. The weighted IOI are added to a histogram for further analysis.

There are four levels of detail coefficients. All four levels repeat the above steps to get four histograms. These histograms are added together to obtain the final result by finding the maximum position using the command ‘max’. Finally, Our system calculates the tempo by converting the IOI into BPM for final output.

6.2 Using our song set

In Table 3.2, a complete song list is shown. All 50 songs are in stereo format, so we can get three outputs for each one as explained in section 5.7.3. We then select the final result among them by getting the maximum confidence value.

When we get the final result from the system, we compare it with the ground truth tempo. Our system evaluates the result by calculating the BPM deviation percentage. This is defined by the minimum percentage of tempo estimates of the ground truth tempo, double, half, three times, and one third of the ground truth tempo. The mathematical expression is shown in Equation 5.

Equation 5: $$E = Min\left(\frac{|S-G|}{G}, \frac{|S-2\times G|}{G}, \frac{\left|S-\frac{G}{2}\right|}{G}, \frac{|S-3\times G|}{G}, \frac{\left|S-\frac{G}{3}\right|}{G}\right)\times 100\%$$ where

E is the BPM deviation percentage, S is the system estimated tempo and G is the ground truth tempo.

For our system, we use 4% to determine whether the result is correct or not. Our system chooses 4% as ISMIR 2004 [10] used this value as well. Table 6.1 shows the complete results of 50 songs. For more song details, see Table 3.2. There are 3 songs out of 50 which are regarded as not correct as the BPM deviation percentage is over 4%. The average BPM deviation percentage of these 50 songs is 2.15%.

Song	Ground truth tempo (BPM)	System tempo output (BPM)	Error (%)	Mono	Left	Right
Addicted to love	113.5	111.5	1.8	111.5	111.5	111.5
Baby please don't go	148.6	144.9	2.5	145.5	145.1	144.9
Bachelorette	98.3	96.2	2.1	96.4	96.4	96.2
Besito pa ti	158.2	154.5	2.3	154.3	154.5	157.7
Breakfast in bed	72.9	144.4	1.9	144.0	142.7	144.4
Buenas noches from a lonely room	98.5	96.8	1.7	96.6	96.7	96.8
Chariots of fire	68.5	136.4	0.9	137.0	137.3	136.4
Country feedback	73.4	144.6	3.0	144.9	144.3	144.6
Do you want to	126.9	123.4	2.8	123.4	123.4	123.8
Domingo	128.4	125.3	2.4	125.3	125.3	125.3

Firestarter	138.2	141.8	2.6	140.3	96.4	141.8
Five circles	65.6	132.7	2.3	130.4	130.7	132.7
Freeddie freeloader	130.8	128.8	1.5	128.5	128.1	128.8
Friendly fire	182.6	89.0	1.3	89.0	89.0	89.0
Guns in the ghetto	138.2	136.0	1.6	136.0	135.9	136.0
Hunter	80.7	80.2	0.6	80.1	80.2	80.0
I am a rock	114.9	113.0	1.7	113.5	113.0	113.2
I cried for you	75.4	150.8	0.0	150.8	150.8	150.8
I got you	173.7	84.6	1.3	84.6	84.6	84.7
I wish I know	122.4	122.6	0.2	122.9	122.6	122.8
Knock on wood	119.7	118.5	1.0	118.2	118.5	118.7
Layla	115.1	113.2	1.7	112.8	113.3	113.2
Lie to me	133.8	132.0	1.3	132.0	132.0	132.0
Loco de amor	130.7	127.4	2.5	127.0	127.4	127.4
Maggic McGill	93.2	92.9	0.3	92.7	92.9	92.6
Maybe someday	90.8	90.3	0.6	90.1	90.1	90.3
Mi tonada montuna	159.4	155.5	2.4	156.0	154.3	155.5
Mofo	91	89.4	1.8	89.7	89.7	89.4
My heart will go on	99.4	99.1	0.3	99.2	99.1	99.5
Nerve centre	80.1	158.9	1.6	158.9	158.7	158.8
No expectations	80.8	80.3	0.6	80.3	80.3	80.6
No surprises	77.2	155.2	1.0	154.8	155.2	154.1
Paranoid android	83.6	81.9	2.0	81.9	81.9	81.9
Play dead	77.7	154.3	1.4	154.5	154.3	154.2
Pride	107.2	105.8	1.3	104.7	105.8	103.6
Right here, right now	125.9	123.9	1.6	123.8	123.9	124.0
Roadhouse blues	122.8	121.2	1.3	121.4	121.2	121.6
Song for Bob Dylan	70.3	138.2	3.4	138.2	138.1	138.1
Summertime	72.2	156.9	17.3	156.9	156.9	156.9
The pan piper	73.4	145.1	2.3	145.5	145.1	145.8
The ride of the valkyries	93.3	82.0	12.1	82.0	82.8	90.8
The soul cages	106.3	105.0	1.2	105.0	105.0	105.0
The thing that should not be	112.9	111.7	1.1	111.7	111.6	111.3
Una fuerza inmensa	78	154.9	1.4	154.9	155.3	155.7
Wednesday morning 3am	107.6	101.6	5.6	105.4	105.2	101.6
Who's crying now	105.4	104.0	1.3	104.0	104.1	103.7
Will O the wisp	71.4	141.8	1.4	142.2	141.8	142.6
With god on our side	150.3	149.2	0.7	149.2	148.5	147.5
Word up	119.4	117.4	1.7	117.4	117.1	117.1
Work	127.8	126.8	0.8	126.8	127.1	126.9

Table 6.1: Result of our song set. The boxes with bold borders are chosen by the system as the preferred tempo ‘answer’ because the confidence value (not shown) is the maximum among the mono / left / right signals.

By sorting the BPM deviation percentage, we can have a better view for the results.

This is shown in Figure 6.2-1.

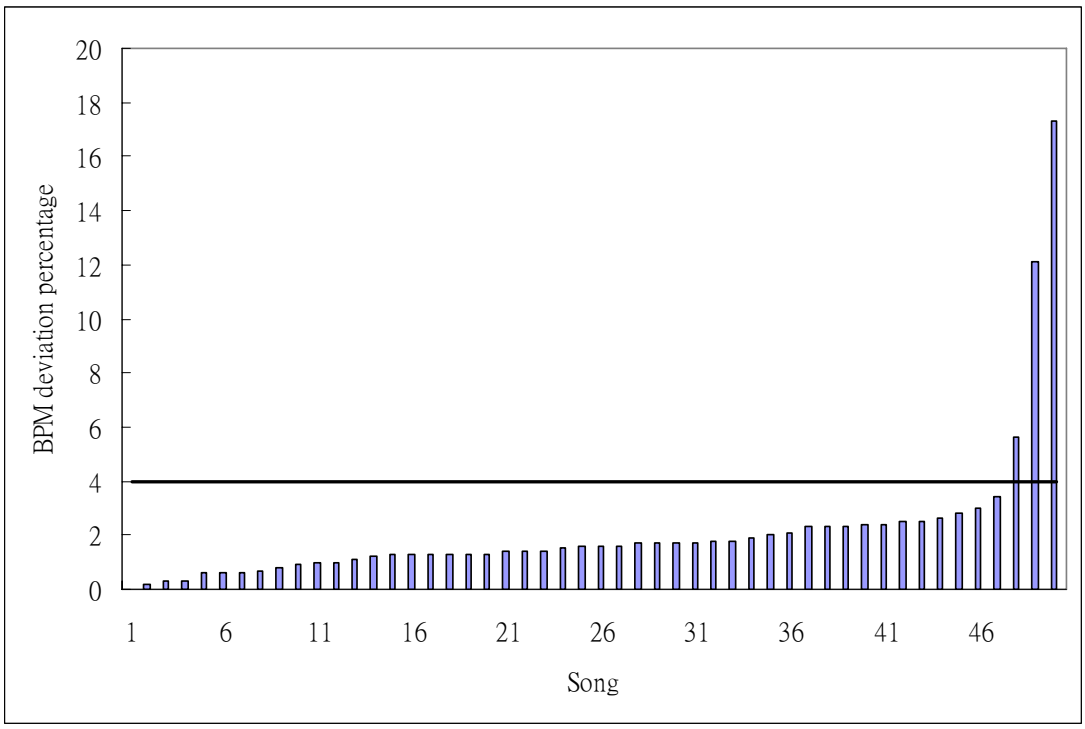

Figure 6.2-1: BPM deviation percentage sorted in ascending order

For the evaluation percentage of BPM deviation, we can consider what happens if we use other values instead of 4%. Table 6.2 shows a range of values used in evaluating our song set, with the corresponding result. Based on these results, it is difficult to choose what percentage should be used for evaluation. We choose 4% mainly because there is a contest that also used this value. This issue is further discussed later.

BPM deviation (%)	1	2	3	4	5	6	7
Number of songs consider as correct (out of 50)	12	35	46	47	47	48	48

Table 6.2: Using a range of evaluation percentage of BPM deviation on our song set

Many IOI tolerances have been discussed previously in section 3.6.1. These values are used for our system when we calculate the weight of an IOI. We have tried two different types of IOI tolerance. One type is a constant value and the other type is the percentage value of IOI.

For the first type of IOI tolerance which is a constant time, the result is listed in Table 6.3 and plotted in Figure 6.2-2. The best constant value is 40ms among these four values proposed by different papers. Table 6.3 shows that if the IOI tolerance is too small like 20ms, it would be too strict for the neighbor beats. As long as the IOI tolerance is more than 40ms, the result would be good. However, it gets worse for higher values.

IOI tolerance	20ms	40ms	50ms	70ms	100ms
Average BPM deviation percentage	2.85	2.15	2.16	2.18	2.29

Table 6.3: Using a range of constant IOI tolerance in our algorithm

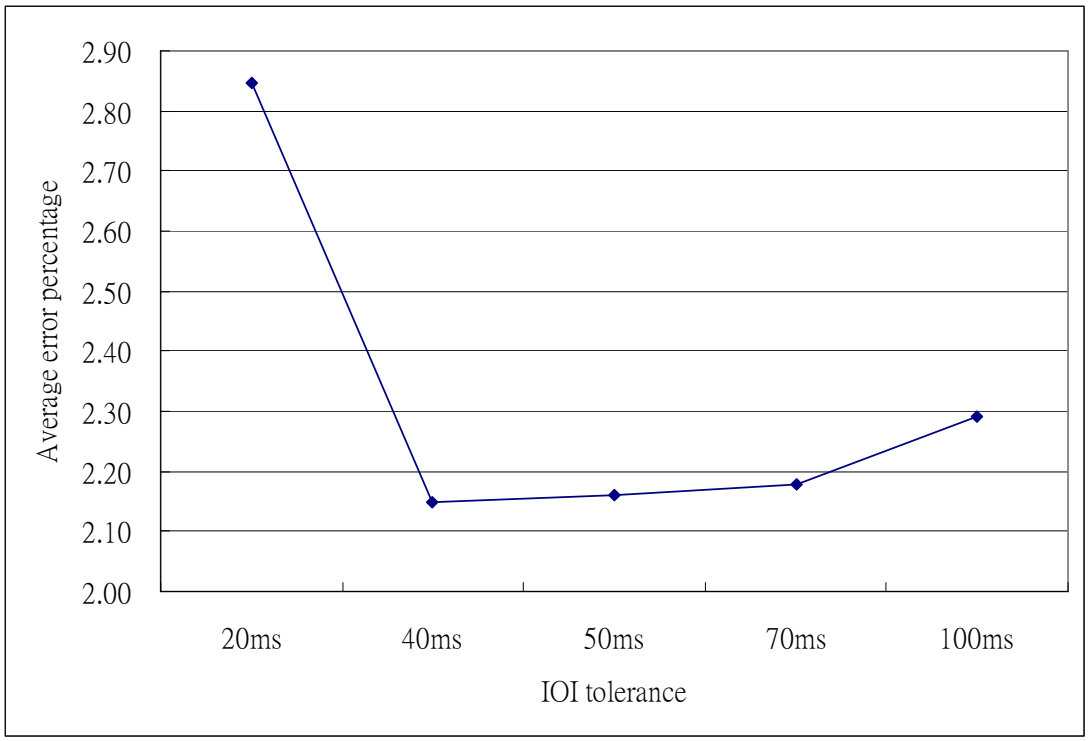

Figure 6.2-2: A graph of average percentage error across a range of constant IOI tolerance

The result for the second type of IOI tolerance which is relative to the current IOI, is listed in Table 6.4 and plotted in Figure 6.2-3. Previous research used 15% and 17.5% only. We show the trend by testing four values which are two values from each side. They are 10%, 12.5%, 20% and 22.5%. This table shows that if we keep increasing the percentage, we can get a better result up to 20%. However, the result becomes worse if we keep increasing it.

IOI tolerance	10%	12.5%	15%	17.5%	20%	22.5%
Average BPM deviation percentage	2.89	2.56	2.55	2.42	2.10	2.18

Table 6.4: Using a range of percentage IOI tolerance in our system

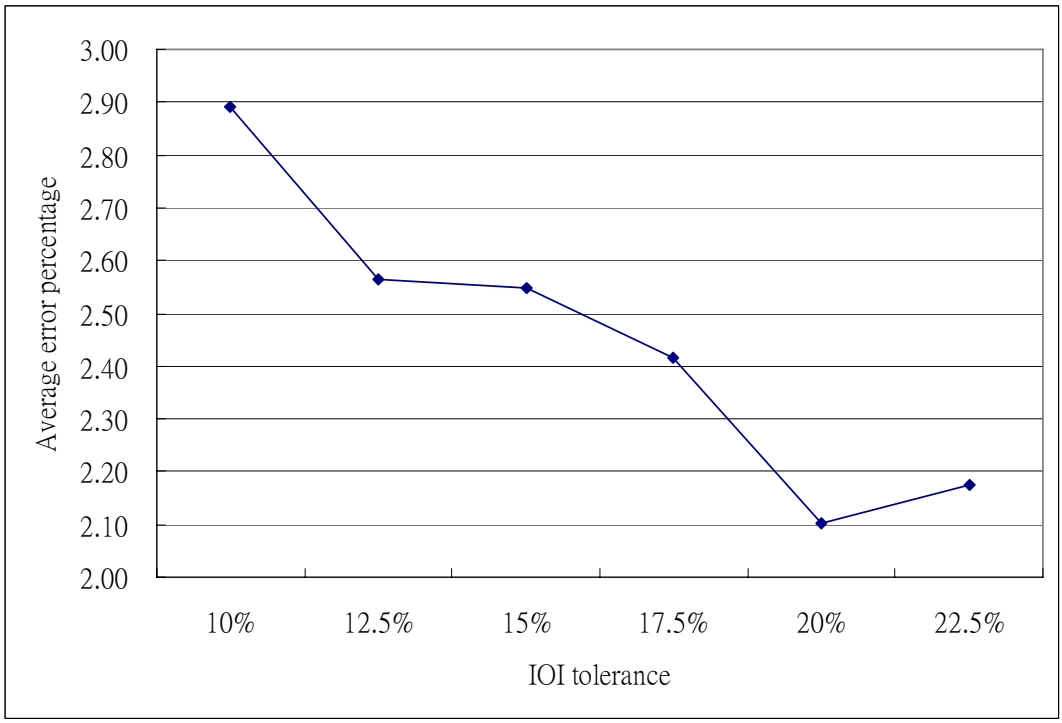

Figure 6.2-3: A graph of average percentage error across a range of percentage IOI tolerance

Without any improvements of the histogram, our system can only get an average value of BPM deviation for our 50 song set of 2.79%. After the smoothing of the histogram, the average becomes 2.66%. By introducing the weight to the occurrence of beat intervals, the average becomes 2.45%. Finally, with the analysis of the mono/ left/ right signals, the final result becomes 2.15%. Table 6.5 summarises the results.

Gaussian smoothing	off	on	on	on
Introduce weighting	off	off	on	on
Mono / Left / Right	off	off	off	on
Average BPM deviation	2.79%	2.66%	2.45%	2.15%

Table 6.5: Average BPM deviation for the 50 songs in out set achieved by adding successive improvements

6.3 Using the set from a previous contest – ISMIR 2004

For the previous contest set (ISMIR 2004 [10]) there are dance music and song excerpts. For the dance music, there are 698 inputs of 30 second duration. For the song excerpts, there are 465 inputs, 20 seconds in length. For the result, we use the same evaluation method as used for the contest.

The evaluation method calculates two accuracy values. Accuracy 1 is the percentage of tempo estimates within 4% of the ground truth tempo. Accuracy 2 is the percentage of tempo estimates within 4% of either the ground truth tempo, or double, half, three times, or one third of the ground truth tempo. Accuracy 2 is the same as Equation 5 defined in section 6.2.

Twelve algorithms were submitted to the contest organizer. All twelve algorithms were evaluated, eleven of which are reported in the released document. Figure 6.3-1 and Figure 6.3-2 present the results for each algorithm including our algorithm, ordered alphabetically with our algorithm at the end: A1 is AlonsoACF, A2 is AlonsoSP, D1 is DixonACF, D2 is DixonI, D3 is DixonT, KL is Klapuri, SC is Scheirer, T1 is TzanetakisH, T2 is TzanetakisMM, T3 is TzanetakisMs, UH is Uhle and RT is our algorithm. For each algorithm, accuracy 1 and 2 are given in light and dark shadings respectively.

The eleven dance music results, including ours, are shown in Figure 6.3-1. The winning algorithm gets the result 63.18% and 90.97% for accuracy 1 and accuracy 2 respectively. For our system, we get 58.88% and 83.24% for accuracy 1 and accuracy 2 respectively. If our algorithm is compared with these eleven algorithms, our position is 2nd out of 12 for accuracy 1 and 3rd out of 12 for accuracy 2.

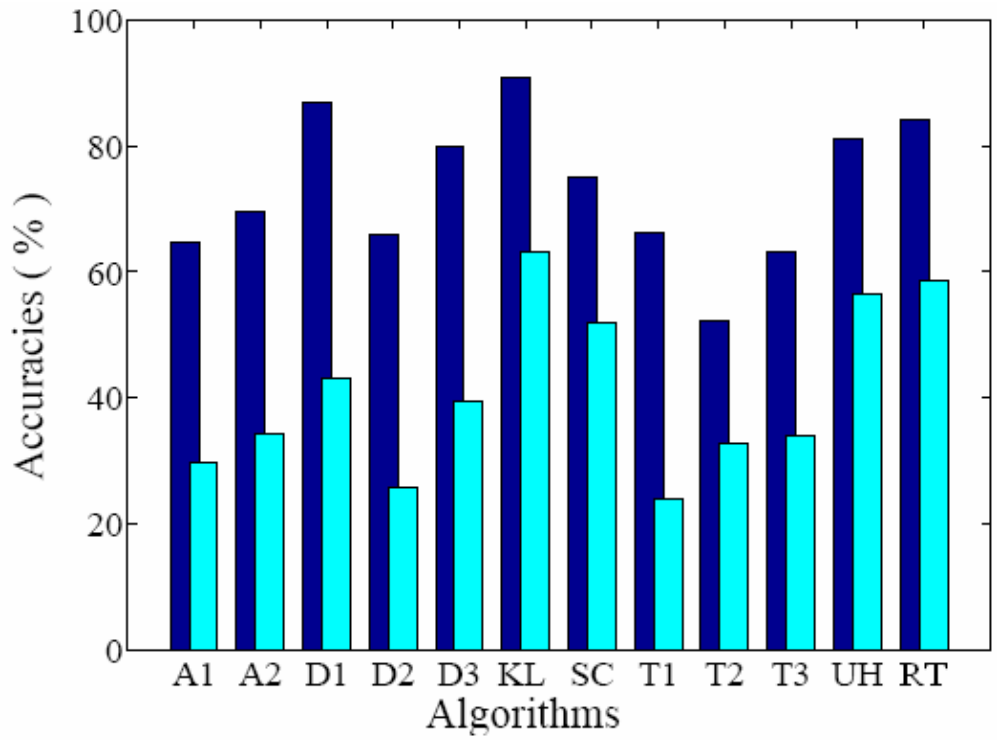

Figure 6.3-1: Accuracies 1 (light) and 2 (dark) on the dance music data set

The eleven song excerpt results, including ours, are shown in Figure 6.3-2. The winning algorithm gets 58.49% and 91.18% for accuracy 1 and accuracy 2 respectively. Our system gets 41.51% and 66.45% for accuracy 1 and accuracy 2 respectively. If our algorithm is compared to the other eleven algorithms, our position is 3rd out of 12 for accuracy 1 and 7th out of 12 for accuracy 2.

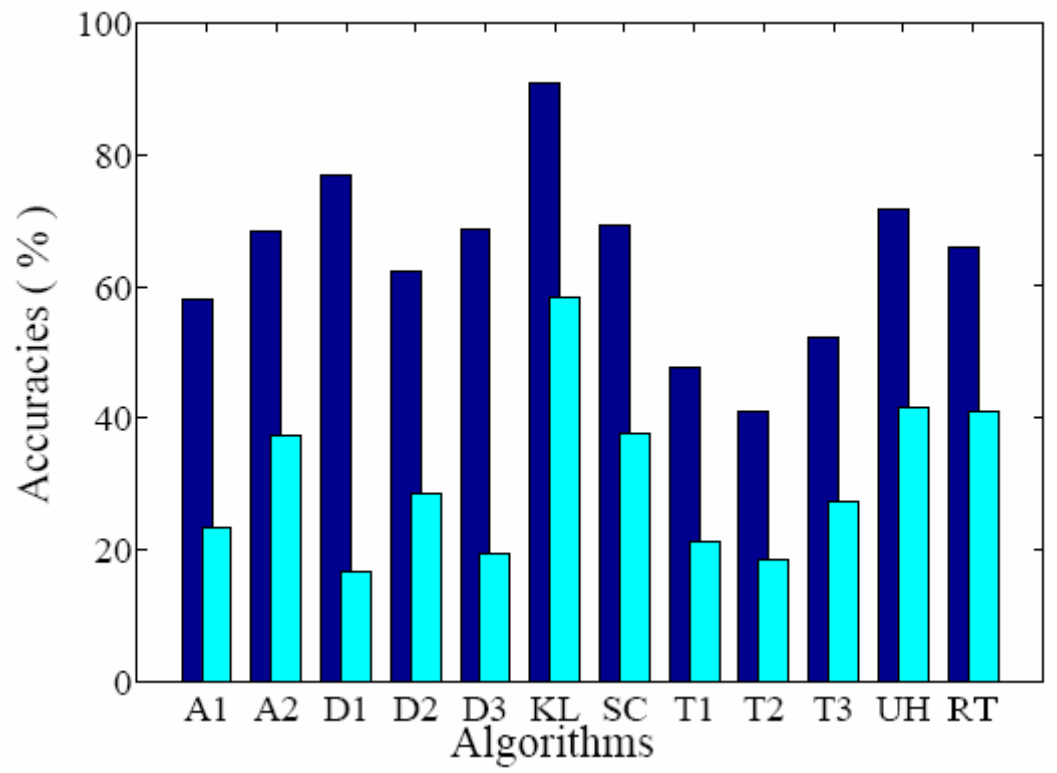

Figure 6.3-2: Accuracies 1 (light) and 2 (dark) on the songs data set

6.3.1 Using other BPM deviation values

In this section, we use a range of BPM deviation values. The results of our system using the dance music data set are listed in Table 6.6 and plotted in Figure 6.3-3. We cannot apply the BPM deviation percentage values to the other algorithms. So, only our algorithm was assessed. From Figure 6.3-3 we can see the BPM deviation percentage needs to be large enough to permit an acceptable level of BPM deviation,

and it should not be too large that every algorithm has a good result. We agree with the ISMIR 2004 contest choice of 4%.

BPM deviation (%)	1	2	3	4	5	6	7
Accuracy 1	39.11	47.99	55.59	58.88	60.32	60.89	61.46
Accuracy 2	51.58	67.19	78.65	83.24	85.53	86.96	88.40

Table 6.6: Using a range of evaluation percentage of BPM deviation on the dance music data set

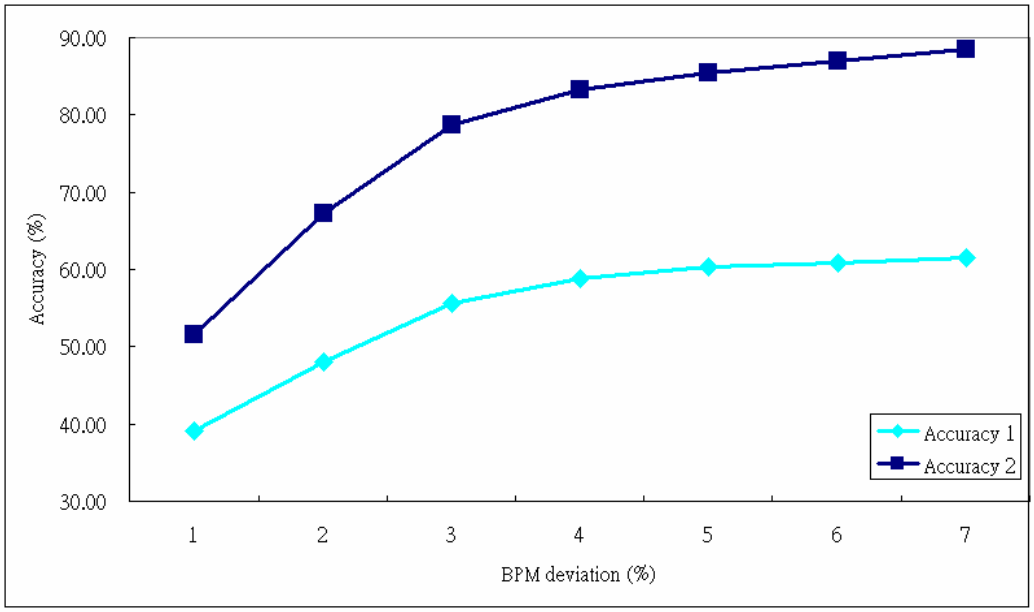

Figure 6.3-3: A graph of accuracy 1 & 2 using a range of percentage of BPM deviation on the dance music data set

Similarly, we can also use different evaluation percentage of BPM deviation values to test with. The results of our system on the songs data set are listed in Table 6.7 and plotted in Figure 6.3-4. The result is similar to the dance music data set.

BPM deviation (%)	1	2	3	4	5	6	7
Accuracy 1	24.09	33.55	39.14	41.51	43.01	43.87	44.73
Accuracy 2	37.42	53.33	61.72	66.45	70.32	72.69	73.98

Table 6.7: Using a range of evaluation percentage of BPM deviation on the songs data set

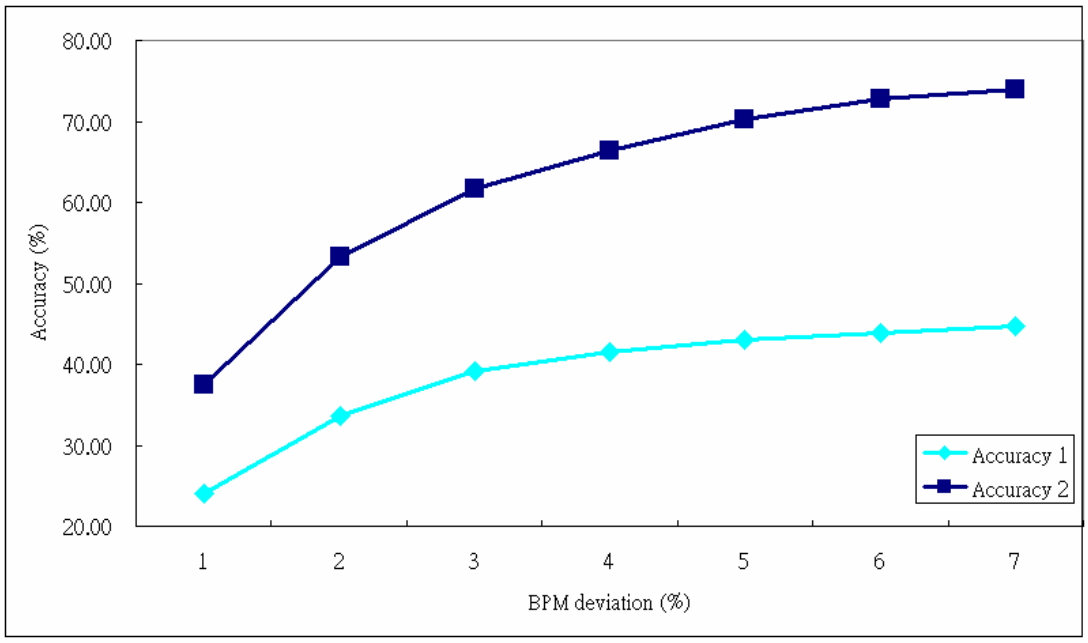

Figure 6.3-4: A graph of accuracy 1 & 2 using a range of percentage of BPM deviation on the songs data set

6.4 Using songs with karaoke version

It is interesting to find out whether tempo extraction is easier for a normal song or a karaoke (i.e non-vocal) version of the same song. Our system is applied to two songs that have a karaoke version. The first one called ‘Heaven’ and the second one called ‘Make my day’. The result is shown in Table 6.8. It indicates that a normal song may be easier for the tempo extraction process. One of the reasons for this may be that the voice contains some extra information compared to the karaoke version. However, this test was performed on only two songs, and so cannot be considered as a general rule.

Song	Ground truth tempo (BPM)	System tempo output (BPM)	Error (%)
Heaven	135.8	135.7	0.1
Heaven (Karaoke)	135.8	135.9	0.1
Make my day	114.8	113.8	0.9
Make my day (karaoke)	114.8	110.0	4.2

Table 6.8: Comparison of tempo extraction for two normal songs with their karaoke (i.e. non-vocal) version

CHAPTER 7

CONCLUSION

7.1 Summary

In this thesis, a tempo extraction algorithm has been proposed. Our algorithm is an off-line algorithm, which means it is not running in real time. Audio data with WAV or MP3 format can be the input for tempo extraction. The best format for the input is the uncompressed WAV format with 44100Hz and stereo.

As our system is an off-line one, the whole input data is read into the memory in the first step. Then, a discrete wavelet transform is applied to the audio data, which is stored in the memory, for four iterations to obtain the DWT coefficients. A peak detection algorithm is applied to these four sets of DWT coefficients. Four sets of peaks are obtained by the peak detection algorithm. The IOIs are calculated from these peaks and then a weight is introduced to each IOI value. A histogram is created by using all these weighted IOIs. The histogram is then improved by smoothing with a Gaussian function. We can get the beat interval from the maximum amplitude of the improved histogram. Finally, the tempo can be calculated from the value of beat interval.

We have proposed an equation to calculate the beat deviation from the tempo. This equation is the power curve fit based on actual data obtained from two musicians. Our algorithm uses this equation when determining the weight of an IOI, so as to improve the performance of the tempo extraction system.

Our algorithm is tested with 50 songs from audio CDs. The complete list is shown in Table 3.2. The tempo extracted from the system is compared with the tempo obtained from two musicians. To evaluate the result, we use an error measure which was used in the tempo extraction contest in ISMIR 2004 [10]. We find 47 out of 50 are correct if we use the same error measure used in the contest.

Our algorithm is also tested with the data used in the tempo extraction contest in ISMIR 2004 [10]. The evaluation method calculates two accuracy values. Accuracy 1 is the percentage of tempo estimates within 4% of the ground truth tempo. Accuracy 2 is the percentage of tempo estimates within 4% of either the ground truth tempo, or double, half, three times, or one third of the ground truth tempo. There are in total two sets of data we can test with. For accuracy 1, our position for one set is 2^{nd} out of 12 and 3^{rd} out of 12 for the other set. For accuracy 2, our position for one set is 3^{rd} out of 12 and 7^{th} out of 12 for the other set.

7.2 Future Work

Our tempo extraction system can be extended in numerous ways. We can improve the peak detection algorithm of our system. The current peak detection algorithm uses a moving window. We can adjust the step size of the moving window and the number of times it should peak for the local maximum. Or, we can change the peak detection algorithm using another method such as calculating the derivative.

Another possible improvement would be to use agents to determine the weight of each IOI. One method would be the use of a set of different agents working independently from each other. Each agent would calculate its own weight. The final weight would then be the sumation of all the values given by the agents. Finally, the weight would be added to the histogram for further analysis.

The current system is an off-line system. We can extend it into a real time system by getting the input continuously and then apply the DWT to it. While the coefficients are updating, the peak detection could be performed at the same time, as it is using the moving window technique. As long as the moving window is fast enough to output the peaks the histogram can be updated in real time. The tempo value can be retrieved from the histogram while it is updating. So, it can be extended to be a real time system.

We could also extend the system to identity the beat locations. This has a number of useful applications. For example, a lighting effect can be displayed on the beat while the music is playing. This could be achieved by extending the current system. The peak detection algorithm described previously has identified all the peaks. One simple method to determine the beat is to identify the peak with the minimum beat deviation as the best peak. When the best peak is obtained, we can identity all the beat locations by adding or subtracing multiples of IOI from the best peak.

BIBLIOGRAPHY

[1] M. Goto and Y. Muraoka, “A beat tracking system for acoustic signals of music”, in Proceedings of the Second ACM International Conference on Multimedia, pages 365-372, 1994.

[2] M. Goto and Y. Muraoka, “Music understanding at the beat level – real-time beat tracking for audio signals - In Working Notes of the IJCAI-95 Workshop on Computational Auditory Scene Analysis, pages 68-75, 1995.

[3] M. Goto and Y. Muraoka, “A real-time beat tracking system for audio signals”, in Proceedings of the 1995 International Computer Music Conference, pages 171-174, 1995.

[4] R. Curtis, S. John, A. Curtis, G. John and G. Philip. The Computer Music Tutorial, MIT Press, pages 525-527 & 581-589, 1996.

[5] M. Goto and Y. Muraoka, “Beat tracking based on multiple-agent architecture – a real-time beat tracking system for audio signals”, in Proceedings of the Second International Conference on Multiagent Systems, pages 103-110, 1996.

[6] M. Alonso, B. David, G. Richard, "Tempo and beat estimation of musical signals", in Proceedings of the International Conference on Music Information Retrieval (ISMIR), pages 158-163, Barcelona, Spain, October 2004.

[7] R. B. Dannenberg and B. Mont-Reynaud, “Following an Improvisation in Real Time”, in Proceedings of the 1987 International Computer Music Conference, International Computer Music Association, San Francisco, pages 241-248, 1987.

[8] George Tzanetakis, Georg Essl, and Perry Cook, “Audio analysis using the Discrete Wavelet Transform”, in Proceedings WSES International Conference in Acoustics and Music: Theory and Applications (AMTA 2001), pages 205-210, Skiathos, Greece, September 2001.

[9] K. Jensen and T. H. Andersen, “Beat estimation on the beat”, in IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, pages 87-90, October 19-22, 2003.

[10] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the Meter of Acoustic Musical Signals", IEEE Transactions on Speech and Audio Processing, Volume 14, Issue 1, pages 342-355, January 2006.

[11] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle , P. Cano, "An Experimental Comparison of Audio Tempo Induction Algorithms", IEEE Transactions on Speech and Audio Processing, to appear in September 2006 issue.

[12] G. Peeters, "Time Variable Tempo Detection and Beat Marking", in

International Computer Music Conference, Barcelone, Spain, 2005.

[13] C. Uhle and J. Herre, "Estimation of tempo, micro time and time signature from percussive music", in Proceedings of the 6th International Conference on Digital Audio Effects. (DAFx-03), London, UK, September 2003.

[14] F. Gouyon and S. Dixon, "Dance music classification: A tempo-based approach", in Proceedings of the International Conference on Music Information Retrieval (ISMIR), pages 501-504, Barcelona, Spain, October 2004.

[15] M. Alonso, B. David, G. Richard, "Tempo and beat estimation of musical signals", in Proceedings of the International Conference on Music Information Retrieval (ISMIR), pages 158-163, Barcelona, Spain, October 2004.

[16] M. Alonso, B. David, G. Richard, "A Study of Tempo Tracking Algorithms from Polyphonic Music Signals", Proceedings of the 4th COST 276 Workshop, Bordeaux, France, March 2003.

[17] M. Alonso, R. Badeau, B. David et G. Richard, "Musical tempo estimation using noise subspace projections", IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA '03), pages 95-98, New Paltz, New York, 19-22 October 2003.

[18] M. McKinney, Music Information Retrieval Evaluation eXchange (MIREX

2005) – Audio Tempo Extraction,

http://www.music-ir.org/evaluation/mirex-results/audio-tempo/index.html

[19] M. E. P. Davies and P. M. Brossier, "Fast implementations for perceptual tempo extraction", In Proceedings of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX 2005), Audio Tempo Extraction contest, September 2005.

[20] M. E. P. Davies, M. D. Plumbley, "Beat tracking with a two state model", in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005), Vol. 3, pages 241-244, Philadelphia, USA, March 19-23, 2005.

[21] A. Klapuri, "Musical meter estimation and music transcription", Cambridge Music Processing Colloquium, pages 40-45, Cambridge University, UK, March 2003.

[22] S. W. Hainsworth, M. D. Macleod, "Beat tracking with particle filtering algorithms", In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, pages 91-94, Mohonk, New York, 2003.

[23] Simon Dixon, "Automatic extraction of tempo and beat from expressive performances", Journal of New Music Research, 30(1), pages 39-58, 2001.

[24] Simon Dixon, "Beat Tracking with Musical Knowledge", Proceedings of the

14th European Conference on Artificial Intelligence (ECAI), pages 626-630, ed. W.Horn, IOS Press, Amsterdam, 2000.

[25] K. Jensen and T. H. Andersen, "Beat estimation on the beat", In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pages 19-22, New York, October 2003.

[26] K. Jensen and T. H. Andersen, "Real-time beat estimation using feature extraction", In Proceedings of the Computer Music Modeling and Retrieval Symposium, Lecture Notes in Computer Science, pages 13-22. Springer Verlag, 2003.

[27] W. A. Sethares, R. D. Morris, J. C. Sethares, "Beat Tracking of Musical Performances Using Low-Level Audio Features", IEEE Transactions on Speech and Audio Processing, Issue Vol.13 Issue.2, 2005.

APPENDIX A

In this appendix we show the actual codes used in our tempo extraction system. The following codes were written in Matlab Version 7.0.4.365 (R14) Service Pack 2.

There are two files, ‘run.m’ and ‘peakdetection.m’. The file ‘run.m’ extracts all the tempo from each song in a directory called ‘batch’. All the input files should be placed in ‘batch’ under the working directory in order to run these codes. The result will be shown in a text file called ‘result.txt’. The file ‘peakdetection.m’ is a function which is called by ‘run.m’. It performs the tempo extraction from the input signal ‘d’, and stores the tempo value in a variable called ‘indexbpm’.

<run.m>

```
%This program will extract all the tempo from each song in ‘batch’ and output the result to ‘result.txt’.
%Any sampling rate of the input file is fine.
clear;
timesforloop = 4;

%Read all the files in the directory called ‘batch’
D = dir('batch');
for numsong=1:length(D)
    if (D(numsong).isdir == 0)
        file=strcat('batch\',D(numsong).name)
        clear d;
        clear dstereo;
        l=length(file);

        %Check the format of input (mp3 or wav)
        if strcmp(file(l-2:l),'mp3')
            SIZ=mp3read(file,'size');
            [dstereo,sr] = mp3read(file,min(SIZ(1),44100*180));
```

```
end
if strcmp(file(l-2:l),'wav')
    SIZ=wavread(file,'size');
    [dstereo,sr] = wavread(file,min(SIZ(1),44100*180));
end

%Check if the input is stereo or not, SIZ(2) equals to 2 means it is stereo
if SIZ(2)==2
    %For the left channel
    d=dstereo(1:length(dstereo),1);
    peakdetection;
    for i=1:timesforloop+1
        indexbpm3(2,i)=indexbpm(i);
        accuracy3(2,i)=accuracy(i);
    end
    hgcf = gcf;
    saveas(hgcf, [file(1:l-4) 'left'], 'png');

    %For the right channel
    d=dstereo(1:length(dstereo),2);
    peakdetection;
    for i=1:timesforloop+1
        indexbpm3(3,i)=indexbpm(i);
        accuracy3(3,i)=accuracy(i);
    end
    hgcf = gcf;
    saveas(hgcf, [file(1:l-4) 'right'], 'png');

    %For the mono
    d=mean(dstereo');
    peakdetection;
    for i=1:timesforloop+1
        indexbpm3(1,i)=indexbpm(i);
        accuracy3(1,i)=accuracy(i);
    end
    hgcf = gcf;
    saveas(hgcf, [file(1:l-4) 'mono'], 'png');
else
    %If the input is not stereo
```

```
                d=dstereo;
                peakdetection;
                for i=1:timesforloop+1
                    indexbpm3(1:3,i)=indexbpm(i);
                    accuracy3(1:3,i)=accuracy(i);
                end
                hgcf = gcf;
                saveas(hgcf, [file(1:l-4) 'mono'], 'png');
            end

            result(numsong).bpm = indexbpm3;
            result(numsong).accuracy = accuracy3;
        end
    end

    %Prepare the output format of 'result.txt' for appropriate import into Excel later
    clear res;
    for numsong=1:length(D)
        if (D(numsong).isdir == 0)
            for loop=1:timesforloop+1
                for channel=1:3
                    res(numsong,1,channel,loop) = result(numsong).bpm(channel,loop);
                    res(numsong,2,channel,loop) = result(numsong).accuracy(channel,loop);
                end
            end
        end
    end
    for numsong=1:length(D)
        if (D(numsong).isdir == 0)
            for loop=1:timesforloop+1
                for channel=1:3
                    fil(numsong,channel+3*(loop-1))=res(numsong,1,channel,loop);
                    reswrite(numsong,channel+3*(loop-1))=res(numsong,2,channel,loop);
                end
            end
        end
    end
    fil=round(fil*10)/10;
    for loop=1:timesforloop+1
```

```
        quickwrite(:,loop*6-5:loop*6-3) = fil(:,loop*3-2:loop*3);
        quickwrite(:,loop*6-2:loop*6) = reswrite(:,loop*3-2:loop*3);
end
save 'batch\result.txt' quickwrite -ASCII;
```

<peakdetection.m>

```
%This program extracts the tempo from the input signal ‘d’, and the result is stored in ‘indexbpm’

%Initialization
indexbpm = 0;
valuepeak = 0;
numpeak = 0;
accuracy = 0;
firstTime = true;
start = 1;
secondToRun = 180;
clear peak_diff2
clear peak_diff2int
clear peak_diff2loop
peak_diff2(ceil(sr/2))=0;
peak_diff2int(ceil(sr/2))=0;
for i=1:timesforloop
    peak_diff2loop(i,ceil(sr/2))=0;
end
section=1;
loopcounter = 1;
clear ca
clear cd
ca(1).data = d(1:min(length(d),44100*secondToRun));

%Repeat the following for 4 iterations for cD1 to cD4 in order to create the histogram
while (loopcounter <= timesforloop)
    loopcounter = loopcounter + 1
    %Apply the discrete wavelet transform
    [ca(loopcounter).data,cd(loopcounter).data] = dwt(ca(loopcounter-1).data,'db5');

    %Apply full wave rectification
    abs_cd(loopcounter).data = abs(cd(loopcounter).data);

    %Apply moving window for getting the peak position
    clear peak_window;
    i=1;
    thold=20;
    window_size=round( (sr/8) / (power(2,loopcounter-2)) );
```

```
window_move=round(window_size/thold);
width=window_size;
while(i<length(abs_cd(loopcounter).data)-width)
    [mm,ii]=max(abs_cd(loopcounter).data(i:i+width));
    peak_window.data(ceil(i/window_move)) = i+ii-1;
    i=i+window_move;
end

%Calculate the IOI from the peak position
clear peak_count;
peak_count.sample(1) = peak_window.data(1);
peak_count.count(1) = 1;
j=1;
for i=2:length(peak_window.data)
    if peak_window.data(i)==peak_count.sample(j)
        peak_count.count(j) = peak_count.count(j) + 1;
    else
        j=j+1;
        peak_count.sample(j)=peak_window.data(i);
        peak_count.count(j)=1;
    end
end

%Filter out any peaks which are not the maximum for 90% of the time
clear peak_thold;
j=1;
for i=1:length(peak_count.sample)
    if peak_count.count(i)>=(0.9*thold)
        peak_thold(j)=peak_count.sample(i);
        j=j+1;
    end
end
peak_diff = diff(peak_thold);

%Compute the weight of each IOI
num_delta = 4;
clear delta_peak_diff;
for i=num_delta+1:length(peak_diff)-num_delta
    delta_peak_diff(i)=0;
```

```
            %By using Equation 3 "y = 320.67 x^-0.3388", we set the beat deviation
            TEMPBPM = (sr/2*60) / peak_diff(i);
            ms = 320.67*power(TEMPBPM,-0.3388);
            W = ms * 22050 / 1000;          %change ms to W

            W = W / (power(2,loopcounter-2));
            for j=-num_delta:num_delta
                %To check whether two IOIs are similar
                if abs(peak_thold(i+j) - (peak_thold(i)+j*peak_diff(i)))<=W
                    delta_peak_diff(i) = delta_peak_diff(i)+1;
                end
            end
        end

        %Construct the histogram
        for i=num_delta+1:length(peak_diff)-num_delta
            new_peak_diff = peak_diff(i) * (power(2,loopcounter-2));
            if (new_peak_diff <= sr/2)
                peak_diff2(new_peak_diff)=peak_diff2(new_peak_diff) +
delta_peak_diff(i)/(2*num_delta+1);
                peak_diff2int(new_peak_diff)=peak_diff2int(new_peak_diff) + 1;
                peak_diff2loop(loopcounter-1,new_peak_diff) =
peak_diff2loop(loopcounter-1,new_peak_diff) + delta_peak_diff(i)/(2*num_delta+1);
            end
        end
end

%Smooth the histogram with a Gaussian function
h = fspecial('gaussian',[1 2205],360);
peak_diff3 = conv(h,peak_diff2);
peak_diff4 = peak_diff3(1103:(round( length(peak_diff2) )+1102));

%Extract the tempo from the maximum position in the histogram
[valuepeak,indexdiff] = max(peak_diff4);

%Plot the graph
y = (0:1/22050:1-1/22050);
subplot(2,1,1), stairs(y,peak_diff2,'b');
```

```
ylabel('Occurrence');
xlabel('Beat interval (second)');
subplot(2,1,2), plot(y,peak_diff4);
ylabel('Amplitude');
xlabel('Beat interval (second)');

%Change the unit of the final output from number of samples into beats per minute
indexbpm = (sr/2*60)/indexdiff;

%Compute the confidence value
integ = sum(peak_diff4(indexdiff-1102:indexdiff+1102));
accuracy = 100*integ/sum(peak_diff4);

%Calculate the individual band results
for i=1:timesforloop
    peak_diff3loop(i).data = conv(h,peak_diff2loop(i,:));
    peak_diff4loop(i).data = peak_diff3loop(i).data(1102:(round( length(peak_diff2loop(i,:))
+1103)));

    [valuepeakloop(i),indexdiffloop(i)] = max(peak_diff4loop(i).data);

    indexbpmloop(i) = (sr/2*60)/indexdiffloop(i);
    integ(i) = sum(peak_diff4loop(i).data(indexdiffloop(i)-1102:indexdiffloop(i)+1102));
    accuracyloop(i) = 100*integ(i)/sum(peak_diff4loop(i).data);
    indexbpm = [indexbpm indexbpmloop(i)];
    accuracy = [accuracy accuracyloop(i)];
end
```