TAMPERE UNIVERSITY OF TECHNOLOGY

Fawad Mazhar

Automatic Guitar Chord Detection

Master of Science Thesis

Thesis Supervisors: Adj. Prof. Tuomas Virtanen
M.Sc. Toni Heittola
Examiners and topic approved in the Department of Signal Processing meeting on 7th December 2011.

PREFACE

This work has been carried out at the Department of Signal Processing, Tampere University of Technology, Finland. I wish to express my deepest gratitude to my thesis advisor Mr. Tuomas Virtanen for his invaluable guidance, advice and patience throughout my thesis work.

A very special thanks to Mr. Toni Heittola for his day to day supervision, guidance and encouragement in the first steps of my scientific career. I would also like to thank the members of the Audio Research Team of TUT for providing me the most friendly and stimulating work atmosphere.

I wish to thank my parents Dr. Abdul Aziz Mazhar and Shabnam Hameed for their immense support and endless love. I would also like to thank my brother Ummair Mazhar, my sisters Natasha Dawood and Somia Mazhar, and my cousin Muhammed Ali. A big thanks to all my friends, including but not limited to Rakesh Mehta, Waqar Hussain, Rizwan Fazal and Vikramajeet Khatri.

Lastly, I would like to thank Satu Smolander for all understanding, support and constant love.

Tampere, February 2012

Fawad Mazhar

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

FAWAD MAZHAR : Automatic Guitar Chord Detection

Master of Science Thesis, 59 pages, 2 Appendix pages

February 2012

Major: Signal Processing

Examiners: Adjunct Professor Tuomas Virtanen, M.Sc. Toni Heittola

Financial support: Department of Signal Processing, Tampere University of Technology

Keywords: Transient, Spectral Whitening, Linear Predictive Coding, Phase Transform, Pattern Matching, Guitar Chord Detection, Recording.

Automatic guitar chord detection is a process that attempts to detect a guitar chord from a piece of audio. Generally, automatic chord detection is considered to be a part of a large problem termed as automatic transcription. Although there has been a lot of research in the field of automatic transcription, but having a reliable transcription system is still a distant prospect. Chord detection becomes interesting as chords have comparatively stable structure and they completely describe the occurring harmonies in a piece of music.

This thesis presents a novel approach for detecting the correctness of musical chords played by guitar. The approach is based on pattern matching technique applied to the database of chords and their typical mistakes. Mistakes are the versions of a chord where typical playing errors are made. Transient of a chord is skipped and its spectrum is whitened. A certain region of whitened spectra is chosen as a feature vector. Cosine distance is computed between the extracted features and the data present in a reference chord database. Finally, the system detects the correctness of a played chord based on k-Nearest Neighbor (k-NN) classifier.

The developed system uses two types of spectral whitening techniques: one is based on Linear Predictive Coding (LPC) and the other is based on Phase Transform-β (PHAT-β). The average accuracy shown by LPC based system is 72% while that of PHAT-β is 82.5%. The system was also evaluated under different noise conditions.

ABBREVIATIONS AND NOTATION

CA	Computer Audition
MIDI	Musical Instrument Digital Interface
WAV	Waveform Audio Format
MIR	Music Information Retrieval
RWC	Real World Computing
ICMC	International Computer Music Conference
PCP	Pitch Class Profile
AMT	Automatic Music Transcription
NMF	Non-Negative Matrix Factorization
HMM	Hidden Markov Model
EM	Expectation Maximization
ISMIR	International Symposium on Music Information Retrieval
SOM	Self-organized Map
SNR	Signal to Noise Ratio
CD	Compact Disk
ms	Milliseconds
DFT	Discrete Fourier Transform
k-NN	k-Nearest Neighbor
LPC	Linear Predictive Coding
AR	Auto-regressive
ADSR	Attack Decay Sustain Release
PHAT	Phase Transform
PC	Personal Computer
CVPR	Computer Vision and Pattern Recognition
NN	Nearest Neighbor
FV	Feature Vector
f_{max}	Maximum Frequency

1. INTRODUCTION

Music transcription is the analysis of music signal in order to get the parametric representation of sounding notes in the audio signal. In practise, it is accomplished by listening to a piece of audio containing music and writing down the notes manually [1]. Chord detection is a type of music transcription that only captures the harmonic components of the audio signal. Chords generally have a consistent structure that describes wholly a piece of music in terms of occurring harmonies.

Automatic Chord detection is a part of computer audition (CA) that involves all kinds of information extraction from audio data (signals). The methods used for chord detection enable us to extract the harmonies that occur in a piece of music with respect to time [2][3]. Harmony is the structure of music that is made up by the composition and progression of chords.

For a music signal that contains a variety of music timbre, harmonic constructions and transitions, it becomes almost impossible to create a mathematical algorithm for a precise detection of a musical chord. In fact, the detection becomes hard if audio data contains many instruments e.g. drums, percussions, clipping signals and background noises. Since the scope of this thesis is limited to the automatic guitar chord detection from the audio with an emphasis on different types of guitars, we simplify the problem by restricting ourselves to the isolated guitar playing material only.

In this thesis, we propose a system for detecting the correctness of the played chord by presenting a computational model based on pattern matching of the power spectrum. Based on an audio that consists of a single chord, and having the prior knowledge of the target chord, the system determines whether a correct chord is played or not. The performance of the system is calculated based on the accurate detection of the played chord.

Automatic chord detection is an important issue in music analysis. It is an emerging

field of research with many possible real-world applications such as human-computer interactive music systems, video games and many multimedia applications.

1.1 Guitar Chord Detection

Automatic chord detection refers to a process that attempts to detect a chord from a piece of audio. Guitar chord detection refers to a similar process but it only attempts to detect the chords that are played by guitar. An automatic guitar chord detection model is a system that is based on a similar process. It takes a piece of audio as an input consisting of a chord and based on some mathematical operations, the system detects the type of chord. A preliminary work in this field has been discussed in [4]. Generally in music, the sound being produced by the guitar can be annotated as chords or notes. A guitar chord is any harmonic set of two or more notes mostly played together simultaneously. The sound of the chord varies with the type of guitar being used and also the style with which it is being played.

Guitars generally are extremely capable and versatile for chording purposes, but they do exhibit some differences when it comes to different playing styles and different guitar types. Modern acoustic guitars are varying extremely in their design and construction even more than electric guitars. Some of the most common varieties are the nylon-string classical guitar, steel-string acoustic guitar, semi-acoustic guitar and lap steel guitar. Considering all these variations, we conducted a recording session in the audio laboratory of the Signal Processing Department of Tampere University of Technology. The aim of the recording was to have our own general purpose database with different guitar types, recording sources and playing styles, so that it is convenient enough to be used in every possible way.

Automatic chord detection has been a growing area of research lately. The research in this field is mainly at its initial phase so therefore there are some problems that are still needed to be addressed such as simulating different and more complexed guitar playing styles. With the growing trends and advancements in the field of computer audition, it makes it an attractive and emerging field since it can have countless practical applications.

1.2 Objectives and Scope of the Thesis

The main objective of this thesis is to propose a method for the automatic guitar chord detection. This work applies a simple and efficient framework based on pattern matching technique for automatic guitar chord detection. Particularly, the focus is set on the monotimbral music since the scope is limited to the sounds being produced by the guitar.

The second objective of this thesis is to test the accuracy of the methods proposed for guitar chord detection. For this purpose we created our own database which is a general purpose stand alone package that can be used for any other music related experiments. The performance of the algorithm is tested and compared with our own general purpose database.

1.3 Organisation of the Thesis

This thesis is organized as follows. Chapter 2 describes the literature review on automatic chord detection and related fields of interest. Chapter 3 presents the guitar chord detection model. Chapter 4 presents the general purpose guitar chord database created for this thesis. Chapter 5 presents the evaluation and discusses the results and finally Chapter 6 summarizes the observations made in this study and suggests some future work. Appendix A lists the complete pieces of chord instances that has been used to create the chord database and also to evaluate the proposed system.

2. LITERATURE REVIEW

This chapter gives a brief account on the previous work that has been carried out in this field. Based on my knowledge, there are no studies related to the guitar chord detection as it is focused in this thesis. Since the work in this thesis can be related to music transcription and chord recognition therefore we will discuss some of the most famous algorithms in this chapter. The first section gives a brief description of musical chord. Section 2.2 discusses music transcription and gives an account on some of the algorithms. Section 2.3 describes chord recognition and discusses some relevant algorithms. Finally Section 2.4 summarizes the chapter.

2.1 Musical Chord

A musical chord is the combination of sound being produced by any harmonic set of two, three or more notes [5]. Chords or sequences of chords are frequently being used in western music so therefore if one has to analyse the harmonic structure of the whole piece of audio, it is quite probable that one starts labelling individual chord manually. It is very time consuming and difficult task. Consequently, for harmonic analysis of music signal, automation of chord labelling can be very essential [6].

2.2 Music Transcription

Transcription of music refers to the analysis of a music signal in order to write down its pitch, duration, onset time and source of each sound that is present in a music piece [7]. Generally, the transcription of music have been carried out manually which is very tiresome and time-consuming task involving the knowledge of the music also. Automatic Music transcription (AMT) in particular polyphonic music transcription has been an extremely hard problem. This section discusses some of the polyphonic

music transcription techniques.

2.2.1 Non-Negative Matrix Factorization

In [8], the authors attempt to address the problem of automatic music transcription. They propose a system that is based on non-negative matrix factorization (NMF) of the music spectra. An input audio is first transformed to the DFT magnitude spectrum and then NMF is used to estimate the spectral profile for each note. The approach is mainly based on the concept of redundancy reduction and does not take any prior knowledge of the musical structure [9].

NMF is an algorithm that tends to factorize a matrix X into a product of two non-negative matrices W and H. i.e.

$$NMF(X) \rightarrow WH \tag{2.1}$$

The rows of H consists of summarized profiles of the rows of X and similarly the columns of W contains the summarized profile of columns of X. In this way, the product of WH becomes the approximation of X and the error of reconstruction is minimized. The authors have used two different cost functions that minimize the error of reconstruction.

The system takes an input audio signal $s(t)$ and transforms it to L-length time dependent magnitude spectrum $x(t)$. Then the set of all $x(t)$ are placed into the columns of matrix X. After that NMF is performed on the matrix X. It is shown that when X contains the musical spectra, the elements of W contains the spectrum of the notes while the elements of H contains the temporal information of the notes. The system was evaluated by real piano recordings by Keith Jarret of Bach's figure XVI in G minor [10]. Two different cases were considered for evaluation. The first case was related to the isolated notes while the other was about coinciding notes. The system performed comparatively better in the case of isolated notes as the transcription technique was based on the accumulated experience of the system from the input signal. The system does not extract the notes rather it extracts the unique events. The main drawback of this system was that it only performs transcription

of the notes that have static harmonic profile.

2.2.2 Note Event Modeling

In [11], the authors propose a method that automatically transcribes the notes played by pitched musical instruments. The system takes an audio as an input and produce a MIDI file. The input audio can have percussive sounds but the system does not transcribe it. The proposed transcription system uses multiple-F0 estimator as a front end [12]. Both of the channels of input audio are processed simultaneously by the front end in order to obtain few F0s and their relevant features. Since the approach is based on probabilistic modeling of note events, the output from the front end is passed to the following probabilistic models:

1. **Note Event Model:**
   The note event model takes the output of the multiple-F0 estimator and calculates the likelihoods for different notes. The model uses a three-state HMM to describe the note events.

2. **Silence Model:**
   The purpose of the silence model is to estimate the time regions where no notes are sounding. With the help of silence model, the system is able to skip the time regions where there are no notes sounding. Silence model is a one-state HMM.

3. **Musicological Model:**
   The role of the musicological model is to control the transition between note event model and the silence model. For controlling the transition it involves musical key estimator and bigram models.

At the end, the system performs transcription by searching several paths through the note model and silence model. The token passing algorithm is used for finding several paths [13].
The system was evaluated by using Real World Computing (RWC) database which consisted of stereophonic acoustic recordings from several musical genres [14]. The

evaluation results gave a reliable estimate of the performance of the music transcription system. Although the average overall performance was low but the results were quite encouraging as they became the foundation for future developments.

2.2.3 Hypothesis Search

In [15], the work has mainly been emphasized on the mutual dependency of chord-boundary detection and symbol identification. The authors present a method that recognizes musical chords from real-world audio signals in compact disc (CD) recordings.

The Mutual dependency of chord-boundary detection and chord symbol identification remains to be the major hurdle in automatic chord transcription. They are one of the biggest problems in automatic chord transcription. In order to solve this mutual dependency problem, the authors present a method that generates hypotheses about tuples of chord symbols, chord boundaries and keys, and outputs the most credible one as the recognition result. Three kinds of musical elements are used as cues for calculating the evaluation values of hypothesis:

1. Acoustic Features:

   12-dimensions chroma vectors were used as acoustic features.

2. Chord Progression Patterns:

   They are concatenations of chord symbols that are frequently used in musical pieces.

3. Bass Sounds:

   Bass sounds are used to improve the performance of automatic chord transcription since they are closely related to musical chords.

The system was tested on one minute excerpts from seven songs of RWC-MDB-P-2001 Database [16]. The system showed an average accuracy of 77%. The results showed that their method can correctly recognize chord sequences from complex musical audio signals that can contain vocal and drum sound.

2.3 Chord Recognition

Recognition of chord refers to a process of identifying the harmonic set of notes that produce the functional harmonic palette of western music. Similarly, the automatic chord recognition is a process of labelling a chord to a section of an audio. It mainly requires the harmonic analysis of the input signal. This section discusses some of the automatic chord recognition techniques.

2.3.1 Chromagram

A chromagram or a pitch class profile (PCP) based features have been mostly used in many of the previous approaches for chord recognition algorithms. A chroma vector is a vector with dimensions 12 x 1 with values representing the energy present in each of the 12 semitone pitch classes. Once a chromagram is calculated, then there are number of techniques that can be used to give a label to the chord. Finally, pattern matching techniques are used to measure the similarity of the chroma vector is to a set of chord profiles.

Fujishima developed a real time chord recognition system where he first transforms an input sound to a Discrete Fourier Transform (DFT) spectrum [17]. From the DFT spectrum, the system derived PCPs and then pattern matching was done on the PCP in order to determine the type of a chord. He introduced two main heuristics which played an important role in the overall performance of the system.

1. PCP Smoothing Over Time
2. Chord Change Sensing

The PCP smoothing over time algorithm performs averaging operation over past PCP frames. In this way the algorithm performs PCP smoothing. In chord change sensing algorithm, the system monitors the direction of the PCP vector which helps in sensing the sudden change of chord.

The analysis of the system proved that the heuristics introduced by Fujishima were very useful. Without them, the PCP's were quiet noisy. Smoothing over time reduced the noise while chord change sensing helped in preserving the chord change

points.

2.3.2 Hidden Markov Models

Chord segmentation, classification and recognition has also been done by using statistical techniques such as Hidden Markov Models (HMMs). HMM is a statistical model of a finite state machine in which state transition follows the Morkovian property [18].

One such technique has been discussed in [19]. The authors used PCP vectors as features to train the hidden Markov Model by expectation maximization (EM) algorithm. The system distinguishes each chord as one state. A single Gaussian in 24 dimensions is used to model PCP vector distribution for each state. An input audio consisting of chords is processed in frames and it is assumed that the sequence of chords are known. Although, the chords are hand-labelled but these labels are treated as hidden values within the EM framework. In order to estimate the model parameters, Baum-Welch algorithm is applied to the chord sequences [20]. Model parameters include transition and output probability. Once the model parameters are defined, the Viterbi algorithm is used to recognize the hidden states i.e. labels of the chords. This results in chord recognition.

For evaluation, the authors trained the system by 18 songs form Beatles and tested it with 2 other songs from the Beatles. The recognition accuracy was 22%.

2.3.3 Self-organized Maps

Multi-timbral chord classification using wavelet transform and self organized map neural networks have also been used for chord recognition purposes. This approach is unique since it tends to evaluate the frequency spectrum of the input audio directly rather than deriving PCPs.

The authors in [21] present a method for musical chord recognition based on a model of human perception. They used wavelet transform to evaluate the frequency spectrum directly from the sound. The results of the wavelet transform were sent to the

neural network chord classification unit. Neural network consist of a self-organized map (SOM) layer as shown in Figure 2.1.

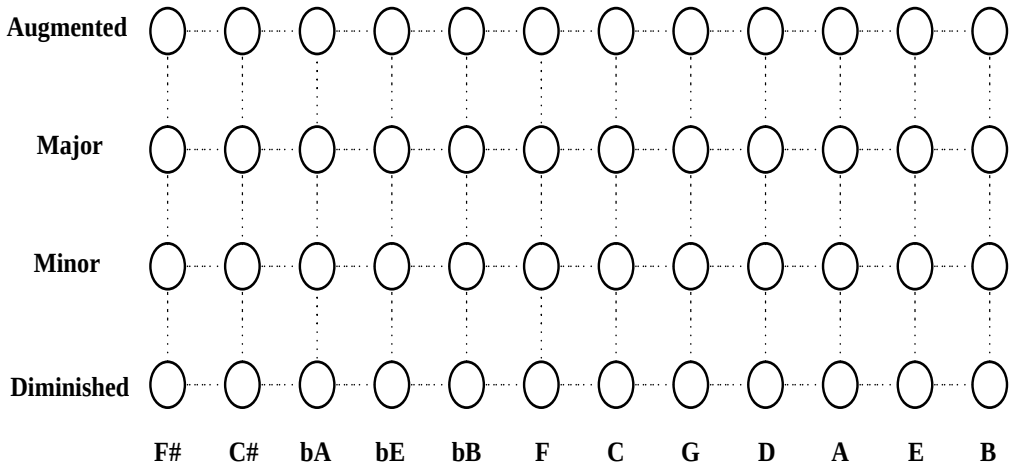

Figure 2.1: Self organized map for chord classification used in [21]

The system needs to determine two kinds of information in order to facilitate the classification. One is the tonality and the other is chord style. In Figure 2.1, the horizontal axis refer to the tonality and the vertical axis represent the chord style. In the horizontal axis, the nodes are arranged in such a way that one of the adjacent node is dominant than the other. In the vertical axis, the nodes are arranged in a manner that adjacent nodes are with high similarity with the others. Before learning, the system sets the initial weights of each neuron on the SOM. The system performs that according to the music theory. The SOM learns from the training data without any supervised information.

The system was evaluated by a test set consisting of 8 measures of Beethoven's 5^{th} symphony. The accuracy of 100% was achieved. The recognition rate stayed the same even after adding Gaussian noise to the sound signal with 0 dB signal to noise ratio (SNR).

2.4 Summary

The music transcription system proposed in [8] performs only when there are notes having static harmonic profile. The system performs poor when the notes are co-

inciding. Since a chord is a harmonic set of two, three or more notes it becomes difficult to model the chords using this algorithm. The system proposed in [11] is a state of the art music transcription system and addresses all music genres. The results are very encouraging though the transcription rate is low. The approach presented in [15] is interesting as it operates on chord sequences and also it focusses on music theoretical knowledge. The study of this algorithm is helpful as it uses a lot of music theory and is evaluated with comparatively large test data showing good results.

The chord recognition system presented by Fujishima in [17] is the ground breaking system. Although the system is quite basic but the study of algorithm is helpful as it is the pioneer in the field of chord recognition. The algorithm presented in [18] is inefficient and complex and also the recognition rate is quite poor. It can only analyze a chord sequence that have maximum of two chords. The approach presented in [21] is of least help. One of the main reasons is the inaccurate evaluation. The test data is too small and homogeneous.

Based on the above comparisons, the approach presented in [15] is the most interesting one for this thesis as it involves a lot of theoretical knowledge in its detection algorithm.

3. THE DEVELOPED SYSTEM

This chapter introduces a new chord detection algorithm that uses the idea of spectral whitening and pattern matching. The first section gives an overview of the developed chord detection algorithm. Subsequently each module of the algorithm is described in detail: Section 3.2 discusses the importance of transient in audio signals. Section 3.3 deals with the spectral whitening and different techniques used to whiten the spectrum. Section 3.4 describes the feature extraction and finally Section 3.5 discusses the pattern matching and k-nearest neighbor classifier.

3.1 Overview

The algorithm first reads an audio containing a piece of chord as input and skips its transient part. The importance of transient in audio signals has been discussed in Section 3.2. In order to skip the transient, we jump off first 250ms of the audio file and by doing so we avoid some variations that can arise due to different guitar playing styles (pick and thumb). After skipping the transient we take a windowed sequence of the audio sample (250ms) and start the process of spectral whitening. Two different techniques of spectral whitening has been discussed (see Section 3.3). After whitening, the whitened signal is transformed to a Discrete Fourier Transform (DFT) spectrum. A part of DFT magnitude spectrum is chosen to be as a feature vector. For classification, pattern matching is done between the extracted features and the data present in the chord database (containing correctly played chords and their most common mistakes) based on the prior knowledge of the input chord. Finally, the system outputs the result based on k-nearest neighbor (k-NN) classifiers. Figure 3.1 depicts the flow chart of our guitar chord detection algorithm.

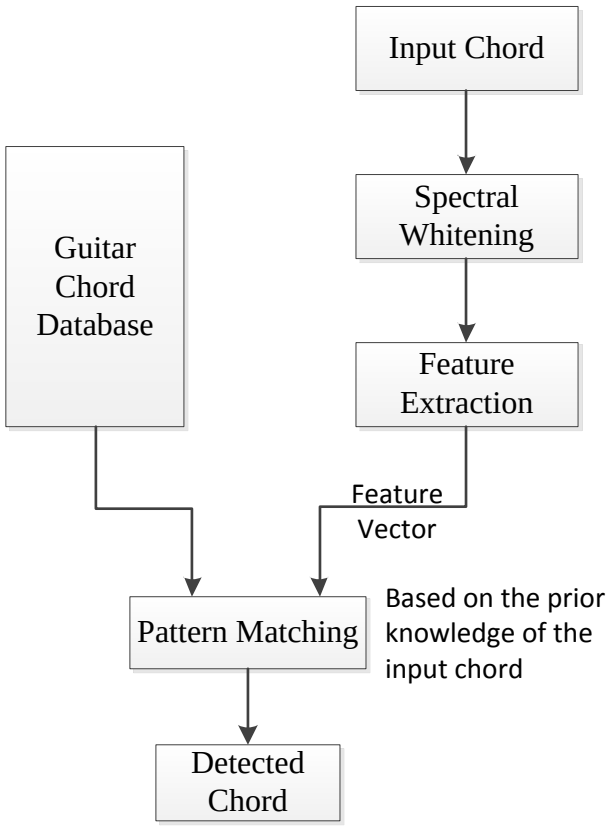

Figure 3.1: Flow chart of the chord detection model

3.2 The Transient

The detection of the transient region has been proven to be highly beneficial in various improved time and pitch scaling algorithms. It reduces the artefacts because of the transient phenomena [22]. Generally these detection tasks are difficult because of complexities in real-world musical audio signals. Fortunately, the music signals are structured enough at the signal level and also at higher levels. The music signal attributes like pitch content, presence of locations of onsets, and the boundaries of transient regions can be easily predicted [23]. Also we know that when an acoustic musical instrument produces a sound, the loudness and spectral content of the sound changes over time in a way that vary from instrument to instrument. The *attack* and *decay* of a sounds are highly effective when it comes to the instrument's sonic character. Sound synthesis techniques frequently generate a control signal. This control signal is represented as envelope which controls the parameters of a sound at any point in its duration [24]. An envelope is the evolution of amplitude of sound

over time and is mostly depicted as *ADSR* (Attack Decay Sustain Release) curve. Figure 3.2 shows a schematic of an ADSR contour.

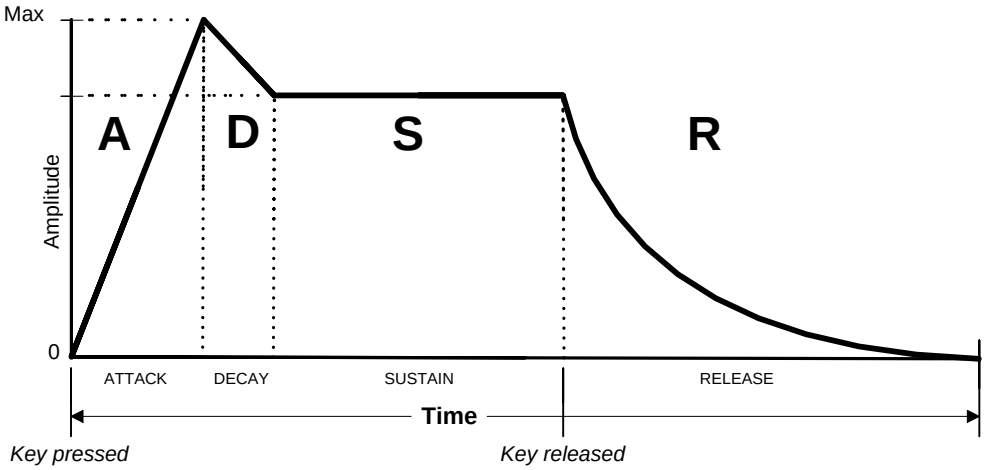

Figure 3.2: Schematic of ADSR

The contour of an ADSR envelope is specified using four parameters:

1. **Attack Time:**

   Attack time is the time taken by sound to achieve maximum amplitude. It begins when the string of a guitar is plucked.

2. **Decay Time:**

   Decay is the gradual reduction in the amplitude of a sound. It is the time taken from the end of attack to the designated sustain level.

3. **Sustain Level:**

   Sustain level is the level during which the sound settles. It also tells the ability of a guitar to hold notes.

4. **Release Time:**

   It is the rate at which the signal level decays from the sustain level to zero i.e. no sound.

Our case is limited to acoustic guitar playing materials with having chords clearly annotated, so in order to skip the transient we jump off the attack part. This is achieved by skipping the first 250ms of the audio file.

3.3 Spectral Whitening

Spectral whitening is a process to balance the frequency contributions over the range of useful signal frequencies. It is also termed as spectral normalization. The term *whitening* is originated from optics, which means the light (white) that contains all the frequencies at equal magnitudes. Spectral whitening attempts to white the frequency spectrum but it does not have any effect on phase spectrum. The amplitude of the whitened spectrum has equal contributions over frequency intervals but this does not necessarily mean that it has equal contributions at every individual frequency.

In the developed system, the main reason why the spectrum of a chord is whitened is because different guitars can have different shapes of the over all spectra. A similar chord when played by a different guitar can have different over all shape of the spectrum. Figure 3.3 shows the spectra of C major chord played by four different guitars (see Section 4.2 for the details of the guitars).

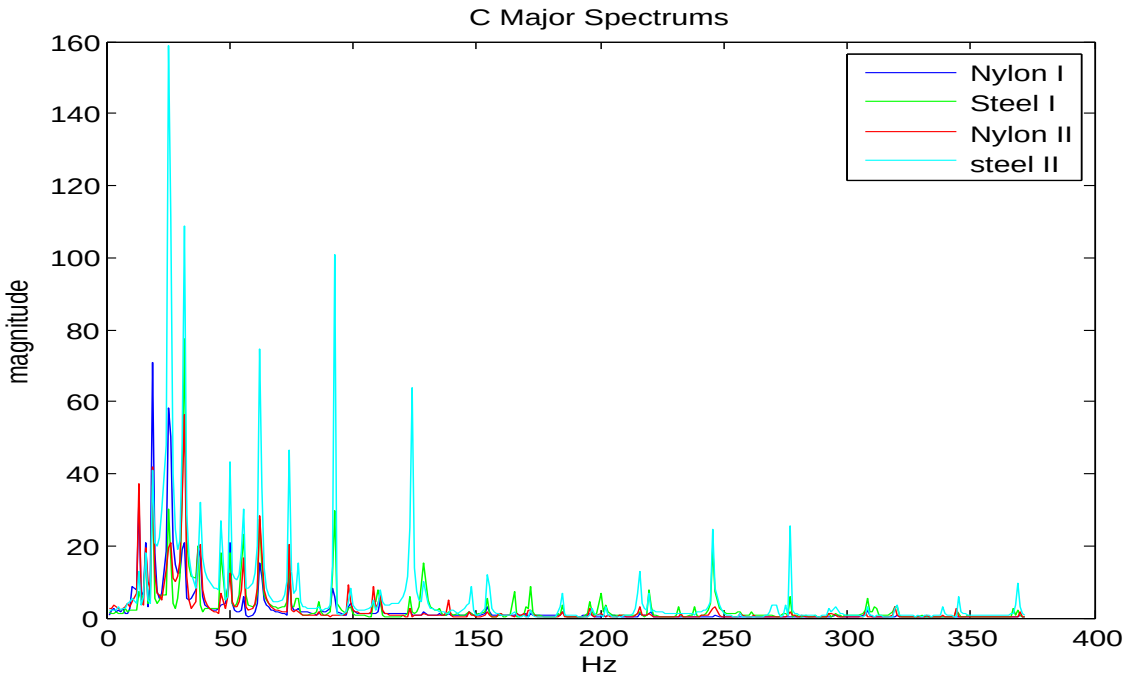

Figure 3.3: Unwhitened c major spectra

Since our approach is based on pattern matching of the magnitude spectra, the overall shape of the spectra plays a vital role in the detection process. By whitening the spectrum we attempt to compensate the difference between the overall shape of the spectra of test and training instances of chords. This compensation allows us to detect chords based on the detailed structure of the spectra. Figure 3.4 illustrates the whitened spectra of C major chord played by four different guitars.

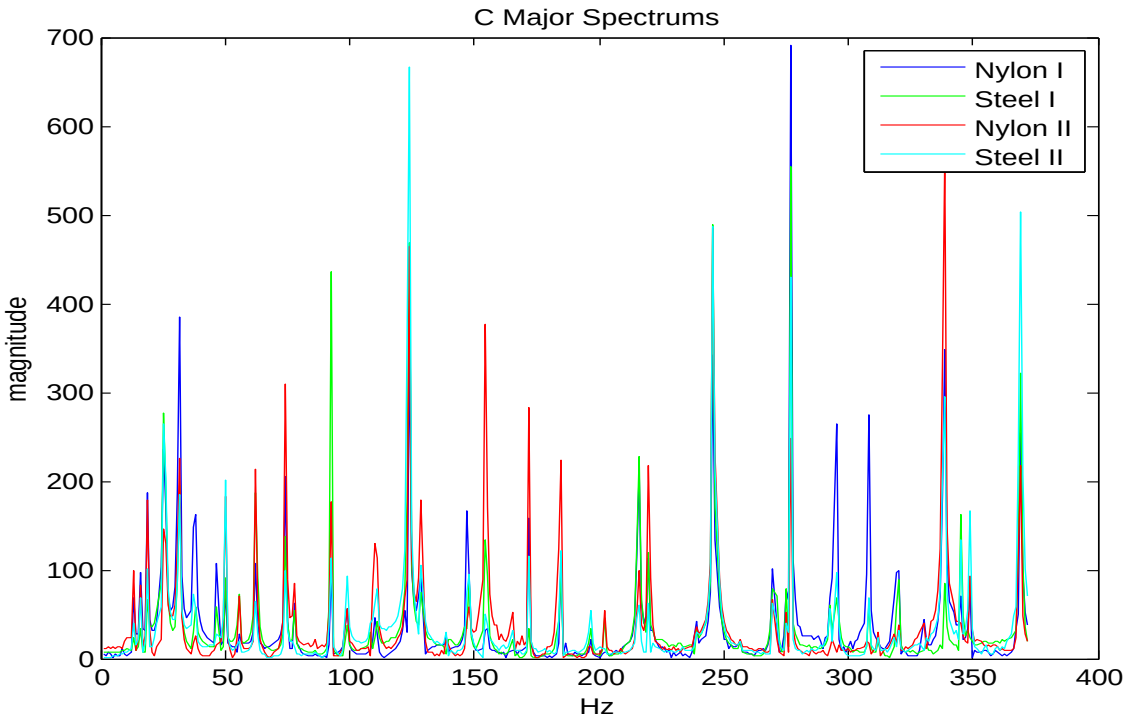

Figure 3.4: Whitened c major spectra

In the Figures 3.3 and 3.4, Linear Predictive Coding (LPC) is used as spectral whitening technique (see section 3.3.1 for details). The length of DFT was chosen to be 4096 and number of filter coefficients were 18 while upto 4KHz of features were used to plot the spectra as shown above. Two different types of techniques for spectral whitening have been discussed in this thesis:

1. LPC Inverse Filtering

2. Phase Transform-β (PHAT-β)

3.3.1 LPC Inverse Filtering

Linear Predictive Coding (LPC) is one of the most common speech analysis techniques used for estimating the basic speech parameters e.g. pitch, formants, spectra, vocal tract, and encoding good-quality speech at a low bit rate. The basic idea of LPC is that a sample of a discrete-time signal can be predicted or approximated as a linear combination of past samples. LPC is mathematically simple and easy to implement. Figure 3.5 represents a flow chart diagram of the spectral whitening model.

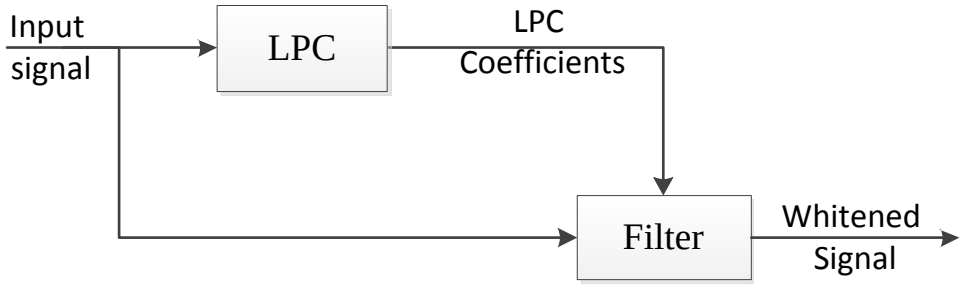

Figure 3.5: LPC based spectral whitening model

While analyzing a speech signal, LPC helps in estimating the frequencies and bandwidths of spectral poles and zeros, and shape of the vocal tract. It fundamentally provides a number of LPC coefficients (speech parameters) that are related to the configuration of the vocal tract. These speech parameters can be used as multiplier values in digital filters to produce a spectrally whitened signal [25].

Let us assume that a signal $s(n)$ is modeled by a linear combination of past samples and the source signal can be represented as [26][27]:

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \tag{3.1}$$

In the above equation G is the gain parameter, $u(n)$ is the white noise excitation signal and a_k is linear prediction coefficient. If we take the Z-transform of the above equation, we get

$$S(z) = \sum_{k=1}^{p} a_k z^{-k} S(z) + GU(z) \tag{3.2}$$

which leads us to the transfer function *H(z)*

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)} \tag{3.3}$$

where

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} \tag{3.4}$$

Now, we will observe the LPC with the perspective of estimating the sample of a signal based on the past samples. Let $\hat{s}$(n) be a random process that predicts $s(n)$ as a linear combination of past samples and can be defined as:

$$\hat{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \tag{3.5}$$

The prediction error *e(n)* is defined as:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{3.6}$$

The z-transform of the prediction error is given as:

$$E(z) = S(z)(1 - \sum_{k=1}^{p} a_k z^{-k}) = A(z)S(z) = \frac{S(z)}{H(z)} \tag{3.7}$$

Figure 3.6 depicts an inverse whitening filter model.

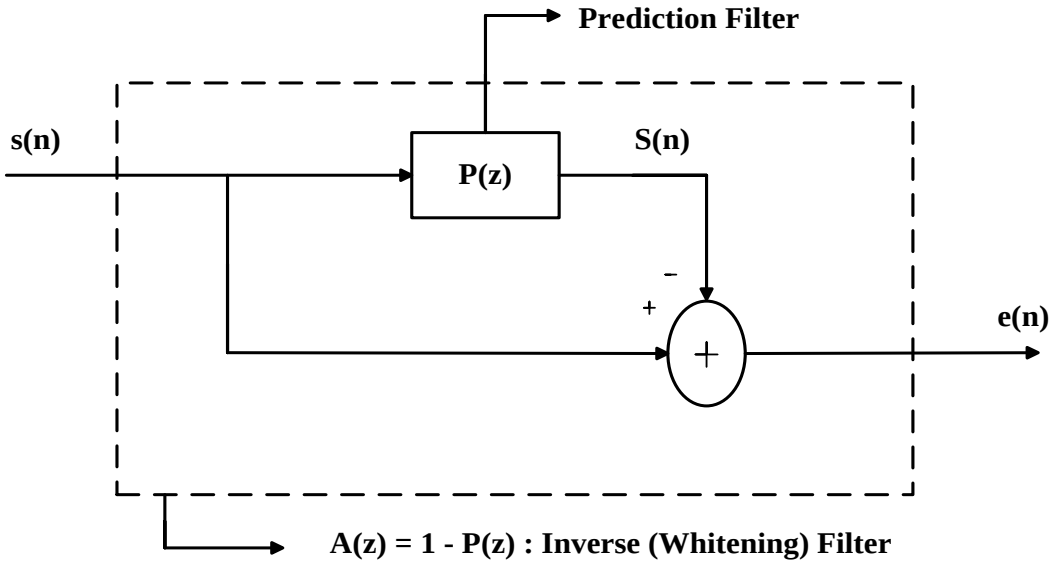

Figure 3.6: Inverse whitening filter

In Figure 3.6, P(z) is the prediction filter and can be defined as:

$$P(z) = \sum_{k=1}^{p} a_k z^{-k} \tag{3.8}$$

The fundamental objective of the linear prediction is to find the set of linear prediction coefficients $a_k{}_{k=1}^{p}$ that can minimize the prediction error variance. The prediction error variance for linear prediction of order p can be defined as:

$$D_p = E[e(n)^2] = E[(s(n) - \hat{s}(n))^2] \tag{3.9}$$

By minimizing D_p we obtain the linear prediction coefficients that are equivalent to minimizing the average ratio of the signal spectrum to its LPC spectrum. There are number of methods to obtain these prediction coefficients a_k. The method that has been used in this thesis is the autocorrelation method. To obtain a_k using autocorrelation method, we have used Levinson-Durbin algorithm [28]. After obtaining these coefficients, they are used in digital filters for defining their behaviour as multiplier values. The filter that is used is Direct Form II Transposed Structure which is a direct implementation of a standard difference equation [29]. The input signal $s[n]$ is filtered by using the LPC coefficients a_k resulting in a whitened signal. Figure 3.7 shows the response of the filter based on the LPC coefficients a_k. The number of LPC coefficients used to describe the response of this filter is 12.

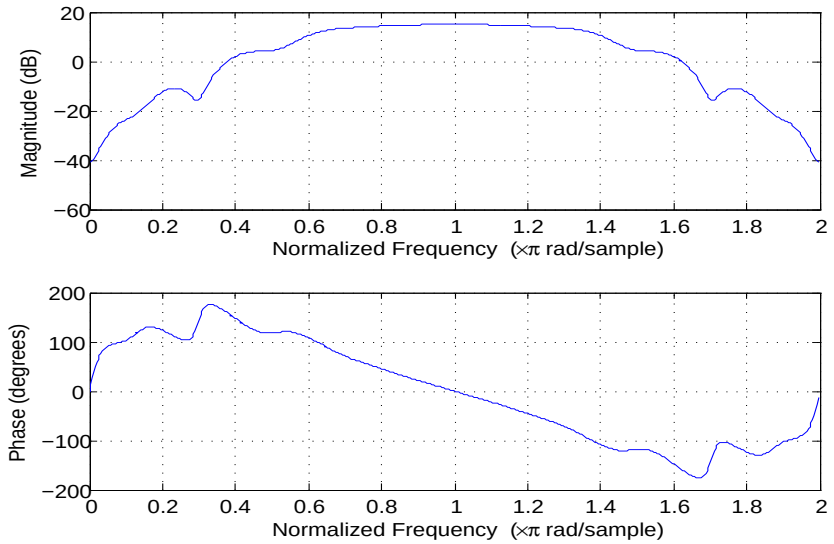

Figure 3.7: Response of the filter

Figure 3.8 shows the spectrum of c major chord before and after spectral whitening.

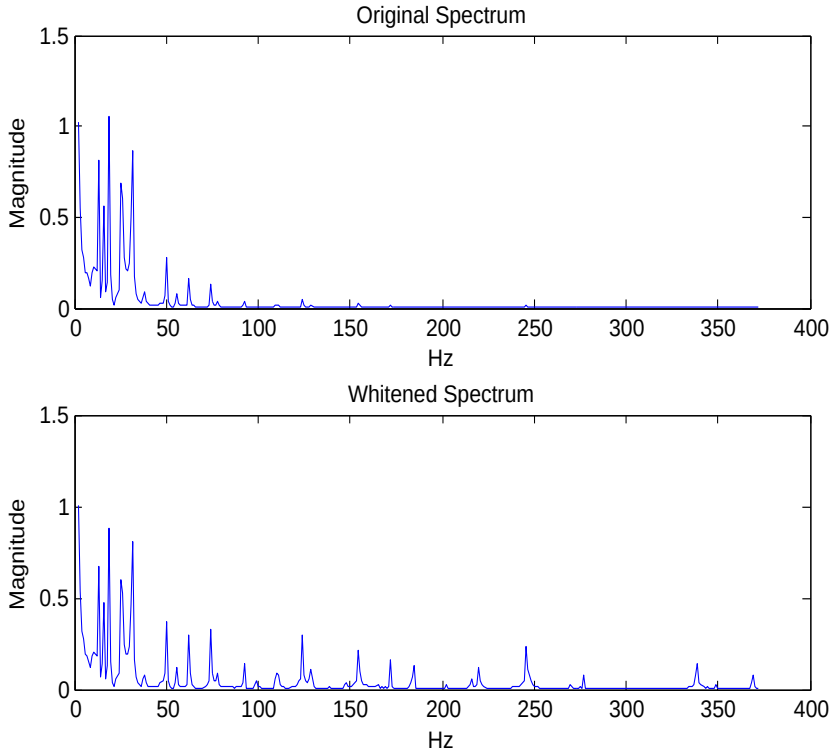

Figure 3.8: Original and whitened spectrum of c major chord.

3.3.2 PHAT-β

When a signal is represented in the frequency domain, its phase and spectral magnitude play a different role. In some cases, by just retaining the phase of the signal we are able to preserve most of the important features of a signal [30].

This method of spectral whitening is based on Phase Transform (PHAT) which correspond to Discrete Fourier Transform (DFT) normalized by the magnitude [31][32]. A parametric variant of the PHAT is also used for partial whitening of the signal [33]. The PHAT variant is referred to as β and is proven to be very beneficial in partial whitening process as it controls the whitening of the signal.

Spectral whitening can also be achieved by only using PHAT. In that case a signal is completely whitened and can have some drawbacks. One of the major shortcomings are the over-amplification of noise spectral regions especially in the case of narrowband signals where independent noise has the tendency to dominate much of the frequency band. In such cases performance of chord detection actually degrades

from the PHAT application. To overcome this problem, we use the modified version of PHAT i.e. PHAT-β where the parameter β controls the degree of whitening and limits the amount of degradation from the independent noise.
This spectral whitening technique is simple and straight-forward. Figure 3.9 represents a flow chart diagram of the spectral whitening model.

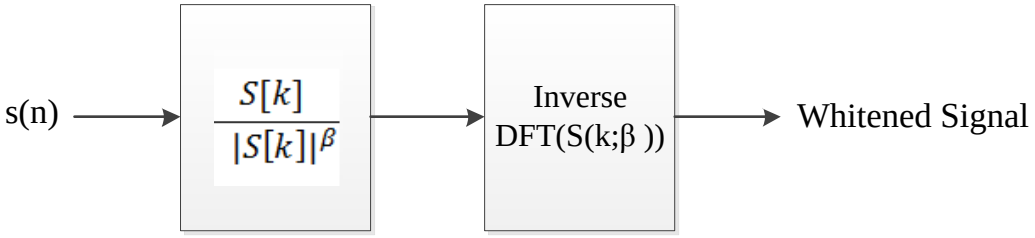

Figure 3.9: PHAT-β based spectral whitening model

Let *s(n)* be the input audio signal with N samples where n = 0,1,2,...,N-1 in the index. The resulting DFT spectrum is given for k = 0,1,2,...,N-1 as:

$$S(k) = \sum_{n=0}^{N-1} e^{-\frac{j2\pi kn}{N}} s(n) \tag{3.10}$$

where $j = \sqrt{-1}$. This results in a vector of complex numbers in the frequency domain,

$$S(k) = R(k) + jI(k) \tag{3.11}$$

R and I are the real and imaginary components of vector S. We can also represent *S(k)* in magnitude and phase angle form.

$$S(k) = M(k)e^{j\phi k} \tag{3.12}$$

where

$$M(k) = \sqrt{R^2(k) + I^2(k)} \tag{3.13}$$

and

$$\phi(k) = arctan\frac{I(k)}{R(k)} \tag{3.14}$$

While normalizing every complex number by dividing both real and imaginary parts of *S(k)* by *M(k)*, we attempt to whiten the spectrum of the signal [34]. Finally, the PHAT-β or partial whitening transform can be denoted as:

$$S(k;\beta) = \frac{S(k)}{|M(k)|^{\beta}} \tag{3.15}$$

In Equation 3.16, β is a real number that can vary between 0 and 1. If $\beta = 1$, equation 3.16 becomes the conventional PHAT and the signal is completely whitened. For $\beta = 0$, the transform has no effect on the original signal. Intermediate β values result in partial whitening. For application to time-domain implementations, the inverse DFT of the PHAT-β is taken after the transform has been applied in the frequency domain [35].

Figure 3.10 shows the spectrum of c major chord before and after spectral whitening with the β value of 0.55.

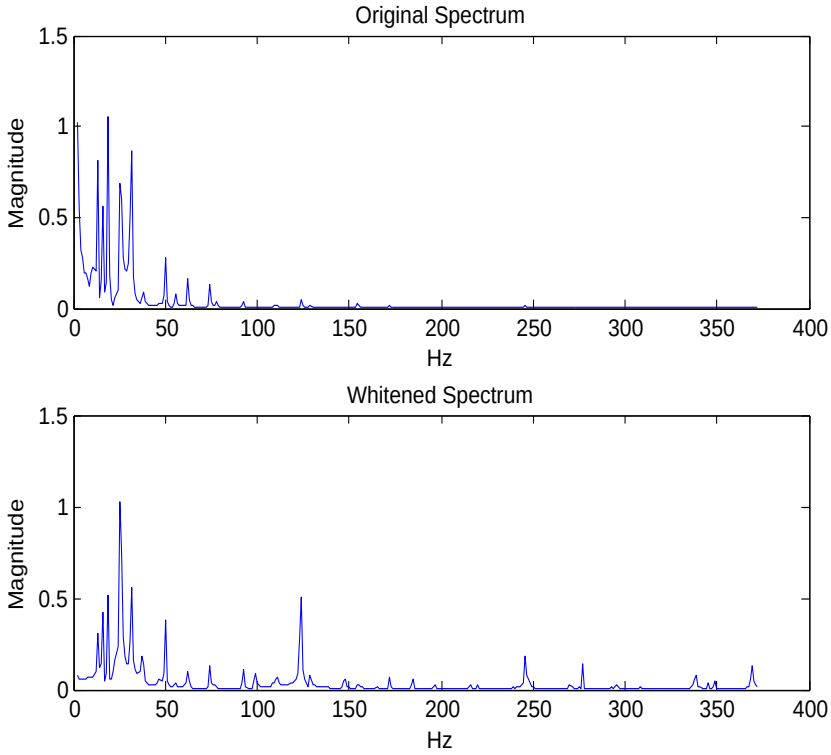

Figure 3.10: Original and whitened spectrum of c major chord.

3.4 Feature Extraction

The algorithm further transforms the input sequence to a DFT spectrum. Let *fs* be the sampling frequency and *x(n)* be the whitened signal with *N* samples where n = 0,1,2,...,N-1 in the index. The resulting DFT spectrum is given for k = 0,1,2,...,N-1 as

$$X(k) = \sum_{n=0}^{N-1} e^{-\frac{2\pi jkn}{N}} x(n) \tag{3.16}$$

The DFT magnitude spectrum yields to a trained data (features) that can be used further for pattern matching. Since most of the interesting energy lies at lower audio frequencies, therefore we only consider a certain region of feature vector for further processing. This results in a fast and efficient way of pattern matching as we tend to reduce the dimensionality of the feature vectors.

3.5 Pattern Matching

Pattern matching is done by matching the test feature vector with the pre-existing patterns present in the reference database. The reference database contains the data that has been trained using the similar feature extraction technique. For matching the patterns we use some of the following distance metrics:

Cosine Distance Metric:

Cosine distance metric which tells us how much closely related the two vectors are or in other words how similar the two vectors are. Consider we have two vectors of attributes x and y, the cosine distance metric can be calculated as follows:

$$d(x, y) = 1 - cos(x, y) \tag{3.17}$$

$$d(x, y) = 1 - \frac{< x, y >}{|x||y|} \tag{3.18}$$

$$d(x, y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} \sqrt{\sum_{i=1}^{n} (y_i)^2}} \tag{3.19}$$

As the angle between the vectors shortens, the cosine angle approaches one which means the two vectors are getting closer resulting in increase in similarity. In this thesis, cosine distance metric has been mainly used for pattern matching but the system was also evaluated by using some other most commonly used distance metrics. Based on that, some comparisons are also made (see Section 5.5).

Euclidean Distance Metric:

The distance between points x and y in a Euclidean space R^n is given by

$$d(x,y) = \sqrt{\Sigma_{i=1}^{n}|x_i - y_i|^2} \tag{3.20}$$

Correlation Distance Metric:

$$d(x,y) = 1 - \frac{(x-\bar{x})(y-\bar{y})'}{\sqrt{(x-\bar{x})(x-\bar{x})'}\sqrt{(y-\bar{y})(y-\bar{y})'}} \tag{3.21}$$

where

x and y are row vectors.

$\bar{x} = \frac{1}{n}\Sigma_j x_j$

$\bar{y} = \frac{1}{n}\Sigma_j y_j$

Spearman Distance Metric:

$$d(x,y) = 1 - \frac{(r_x-\bar{r}_x)(r_y-\bar{r}_y)'}{\sqrt{(r_x-\bar{r}_x)(r_x-\bar{r}_x)'}\sqrt{(r_y-\bar{r}_y)(r_y-\bar{r}_y)'}} \tag{3.22}$$

where

r_x and r_y are the coordinate-wise rank vectors of x and y [36].

$r_x = (r_{x1}, r_{x2}, r_{x3}, ...r_{xn})$

$$\bar{r}_x = \frac{1}{n}\Sigma_j r_{xj} = \frac{(n+1)}{2}$$

$$\bar{r}_y = \frac{1}{n}\Sigma_j r_{yj} = \frac{(n+1)}{2}$$

City Block Distance Metric:

$$d(x,y) = \Sigma_{j=1}^{n}|x_j - y_j| \tag{3.23}$$

3.5.1 k-Nearest Neighbor Classifier

K-nearest neighbor algorithm (k-NN) is one of the most straight-forward classifiers. K-NN classifications is mostly used when there is little or no prior knowledge of distribution of data. It is a type of an instance-based learning where there is no training phase explicitly. In the k-NN algorithm, an unclassified object is assigned to a class that is most heavily represented among its k nearest neighbors [37].
In k-NN classification, we assume that the data is in a feature space. The data can be scalars, vectors or even multidimensional vectors. The fact that the points are in the feature space, they have a notion of distance. The distance metric does not need to be Euclidean although it is most commonly used, it can be cosine, Minkowski, Spearman etc. In this thesis, we have mostly emphasized on the cosine distance metric although the developed system has been evaluated using other distance metrics.
The variable k in k-nearest neighbor classifier decides how many neighbor(s) are going to be used for classification or how many neighbor(s) influence the classification. k is a real number and is generally chosen to be odd. This leads us to two cases:

1. Case I : k = 1 or Nearest Neighbor Rule
2. Case II: k = K or k-Nearest Neighbor Rule

Case I: k = 1 or Nearest Neighbor Rule

This is the simplest scenario where $k = 1$. In this case, the unclassified object is assigned to the class of its nearest neighbor. Let x be an object to be classified and let y be an object nearest to x. The nearest neighbor (NN) rule will assign the class of y to x. This procedure can result a huge error if the test samples are not very large but if the number of test samples are very large, then there is a very high chance that class of x and y are same [38].

Case II : k = K or k-Nearest Neighbor Rule

In this case, the value of k is greater than 1 and is mostly an odd number. For an unclassified object x, the k-NN algorithm can be summarized as follows [39]:

- In the feature space, identify the k nearest neighbors out of N training vectors.
- Out of k chosen samples, vectors k_i are identified that belong to classes w_i, where i = 1,2,3,...,M.
- x is assigned to the class w_i based on the majority voting or based on the maximum number of k_i samples.

Figure 3.11 illustrates an example of k-NN classification.

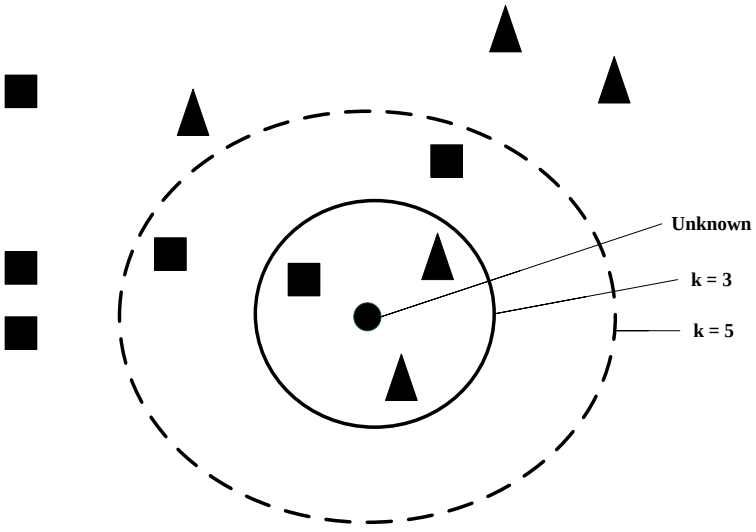

Figure 3.11: k-NN classification example

In the above figure the unclassified sample is represented by circle. Now there are two classes i.e. triangle and square. Based on the value of k, either the circle is classified to the class of squares or triangles. Let us suppose that $k = 3$. For this value of k, the k-NN algorithm will classify circle to the triangle class. For the value of $k = 5$, the k-NN algorithm would classify the circle to the square class.
It can be noticed that the accuracy of k-NN algorithm increases with the high value of k but at the same time, the computation cost is also increased.

4. DATABASE

In this chapter we discuss about the database that we created in order to test our guitar chord detection system. The first section gives an overview of the database and total number of chords played. Section 4.2 deals with the number of guitars used for playing the chords and their mistakes. Different kinds of recording sources has been discussed in Section 4.3. Section 4.4 describes the room acoustics while the format of the audio data is discussed in Section 4.5. Section 4.6 discusses the database organization and finally Section 4.7 presents the statistics of the database.

4.1 Overview

We recorded a collection of guitar chords with the sampling rate of 44.1 kHz in order to have a general purpose database for guitar related audio experiments. Chords that were recorded are listed in table 4.1:

Table 4.1: List of chords recorded

Name	e	A	D	G	B	E
A	x	0	2	2	2	0
A Minor	x	0	2	2	1	0
B	x	2	4	4	4	x
B Minor	x	2	4	4	3	2
C Major	x	3	2	0	1	0
D	x	x	0	2	3	2
D Minor	x	x	0	2	3	1
E	0	2	2	1	0	0
E Minor	0	2	2	0	0	0
F (1)	1	3	3	2	1	1
F (2)	x	x	3	2	1	1
F (3)	x	x	3	2	1	x
F Minor	1	3	3	1	1	1
G	3	2	0	0	3	3

In Table 4.1 the characters *e,A,D,G,B* and *E* are the names of the strings of the guitar. The highest string is named as *e* and the lowest string is named as *E*. The numbers *3,2,1* etc means the fret number of that particular string and number *0* means that the string is played open. The character *x* means the string is kept silent.

In addition to these standard chords, a series of typical playing mistakes were also recorded. Mistakes are the erroneous versions of a chord i.e. versions of a chord where typical playing errors are made. The total number of chords and their mistakes were 66. The mistakes included chords with extra notes (e.g. e and A strings for the D major chord), wrong notes, and missing notes. We selected the mistakes based on the opinions of an experienced guitar teacher, and they were verified by interviewing three young guitar players who still remembered what type of problems they faced while learning the chords. Table 4.2 shows a brief description of *C major* chord along with its mistakes that were recorded and stored in the database.

Table 4.2: C major chord and mistakes with their description

Name	e	A	D	G	B	E	Description
C Major	x	3	2	0	1	0	Correct version
Mistake 1	0	3	2	0	1	0	One extra note on e string
Mistake 2	x	3	x	0	1	0	Missed one note on D string
Mistake 3	x	3	x	x	1	x	Missed three notes on D, G and E strings
Mistake 4	x	3	2	0	1	x	Missed one note on E string
Mistake 5	x	3	2	0	0	1	Wrong note on E and B strings
Mistake 6	x	3	0	2	1	0	Wrong notes on D and G strings
Mistake 7	x	0	3	2	1	0	Wrong notes on A, D and G strings

4.2 Guitars

All the chords and their mistakes were played and recorded with four different guitars: two nylon-string acoustic guitars and two steel-string acoustic guitars simultaneously. The guitars are denoted as *Nylon I*, *Nylon II*, *Steel I* and *Steel II*. Quality of sound varies among the guitars e.g. sound quality of *Nylon I* is better than that of *Nylon II* and similarly the sound quality of *Steel I* is better than that of *Steel II*. We also took into consideration the two most common playing styles i.e. pick and

thumb. In *thumb* playing style, the guitar is played by plucking the strings directly from the finger tips or thumb of the hand whereas is *pick* style playing, the guitar is played by plucking the string with a pick. Pick is a plectrum often shaped as an acute isosceles triangle and it is mostly made of plastic.
By recording the chords and their mistakes with both of the playing styles, we had a number of chord instances stored in our database. An instance is a recorded piece of any chord (including mistakes). Table 4.3 shows the database statistics with respect to the guitars used.

Table 4.3: Database statistics with respect to guitars

Guitars	Number of instances
Nylon I	844
Nylon II	866
Steel I	918
Steel II	772
Total (chord instances)	3400

In the Table 4.3, we can see the number of instances vary because in the recording session we experienced crashing of some of the recording equipments.

4.3 Recording Setup

Each guitar was recorded with six different microphones; built-in microphones of three laptops and iPad 2, and two professional microphones. Details of the microphones are shown below:

Built-in Microphones

- Dell Inspiron (laptop)
- Lenovo (laptop)
- Apple macbook (laptop)
- iPad 2

Professional Microphones

- AKG C460B
- AKG C414B-XLS

Both of the professional microphones were connected to a pre-amplifier and the audio data was converted from analog to digital by RME fireface 800 AD-converter and then connected to a PC. The PC was placed in another room. All the laptops and iPad 2 were placed on a table in front of the player, the first professional microphone (AKG C460B) was placed on a stand behind the table, and the second professional microphone (AKG C414B-XLS) was placed 20-30 centimetres from the sound-hole of the acoustic guitar. Figure 4.1 provides the illustration of the recording room and the recording setup.

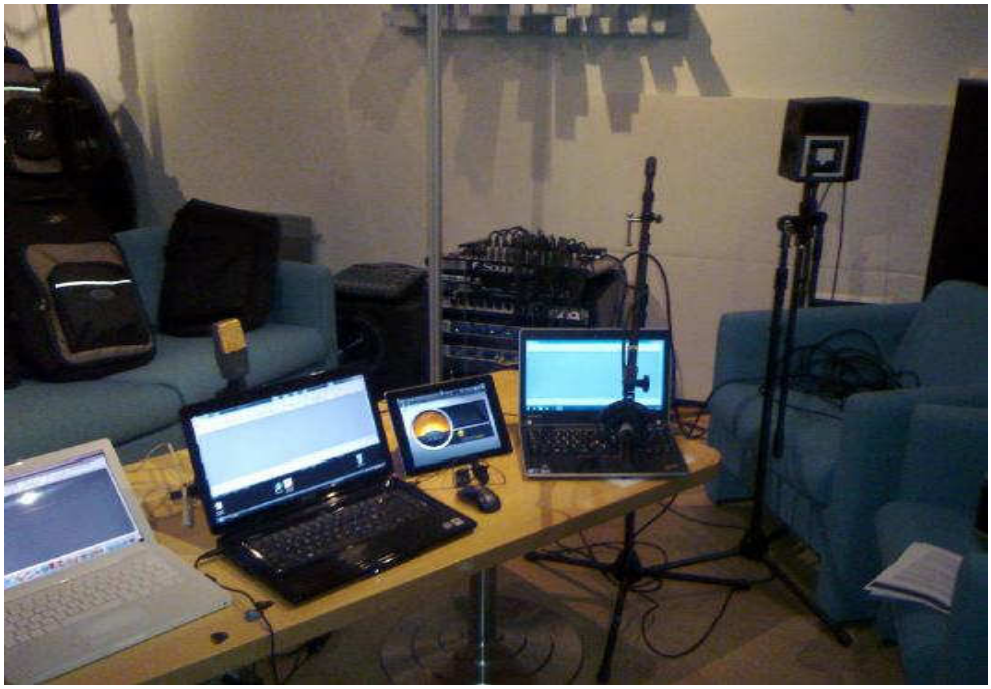

Figure 4.1: Recording room

All the six recordings were done simultaneously using a cross-platform digital audio editor called *Audacity*. Table 4.4 shows the amount of chord instances being recorded and stored in the database. The amount of chord instances from *Lenovo*

and *iPad 2* are quiet less as compare to others because they crashed number of times while recording.

Table 4.4: Database statistics with respect to recording sources

Recording Source	Number of instances
Dell Inspiron	526
Lenovo	396
Apple Macbook	526
iPad 2	374
AKG C460B	526
AKG C414B-XLS	526
Mean (professional microphones)	526
Total (chord instances)	3400

In Table 4.4 Mean (professional microphones) refers to the conversion from stereo to mono as we made audio data from *AKG C460B* to be channel 1 and that of *AKG C414B-XLS* to be channel 2 and took the average of both the channels. Figure 4.2 shows a correct instance of C major chord recorded by *AKG C460B* and played by *Steel II* guitar with the playing style *pick*.

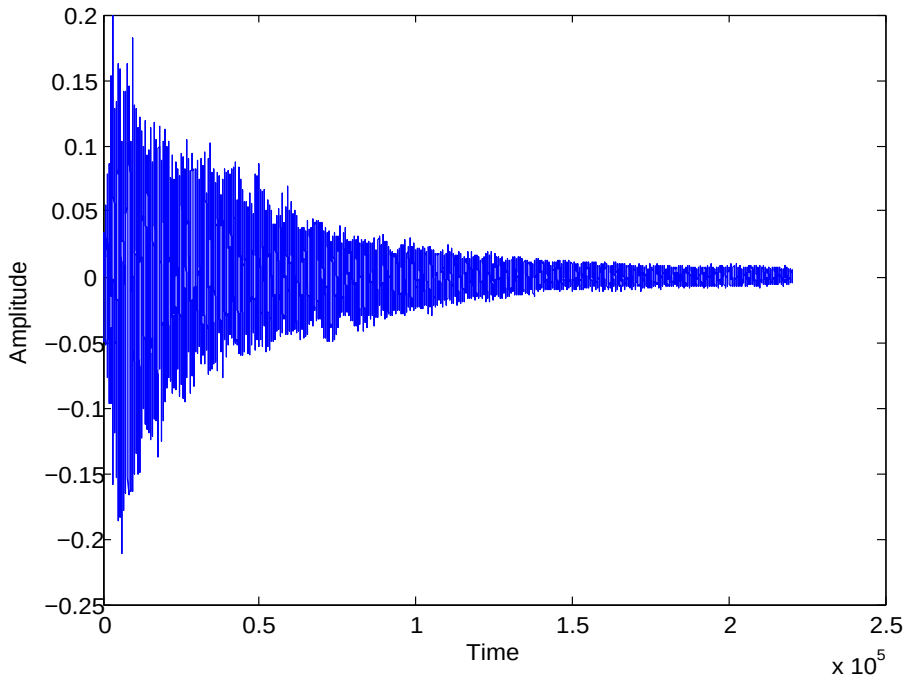

Figure 4.2: C major chord

4.4 Room Acoustics

The database was recorded during two consecutive days in the premises of Tampere University of Technology. The used room was dedicated for listening and recording purposes and its dimensions were 4.53 x 3.96 x 2.59 m. The interior of the room consisted of sofas and wooden table and some acoustic panels. Other devices in the room are audio equipment such as speakers, diffusers, racks and a projector canvas. The room contains three wall mounted sound diffractors and bass traps. The amount of background noise and other disturbances were also kept to minimum and the reverberations time T60 of the room was 260 ms.

4.5 Format of the Audio Data

Chords and their mistakes were manually annotated and stored as one file in *wav* format. The sampling rate for all the chord instances was 44.1 kHz and the resolution was kept to be 16 bits and was stored in monophonic form. The audio data from *iPad 2* was in *m4a* format and it was converted to *wav*. The recordings done by the professional microphones (AKG C460B and AKG C414B-XLS) had the sampling rate of 48 kHz so we downsample them to 44.1 kHz and stored them in the database.

4.6 Database Organization

The directory structure of the database is very simple and straight-forward. All the recordings were segmented and the chords were organized in the database according to recording source, guitar type and playing style. An example is shown below:

/Database/Chord Instance/Recording Source/Guitar Type_Playing Style.format
/Database/C_x32010_correct/iPad/Nylon1_pick.wav

In this way the database was organized so that a chord instance with any variation can be accessed easily.

4.7 Statistics of the Database

All the chords in the database do not have the same number of mistakes or in other words not every chord has the same amount of playing errors e.g. the chord *E minor* is comparatively easy played chord due to its finger placement and hence it has less amount of playing errors (see table A.1). Table 4.5 shows the number of instances from different chords with their statistics.

Table 4.5: Database statistics

Chord	No. of instances	%
A	270	7.9 %
A Minor	216	6.3 %
B	216	6.3 %
B Minor	200	5.9 %
C Major	358	10.5 %
D	250	7.3 %
D Minor	270	7.9 %
E	324	9.5 %
E Minor	150	4.4 %
F	670	19.7 %
F minor	260	7.6 %
G	216	6.3 %
Total	3400	100%

After recording such many variations, we were able to create a comprehensive database of 3400 chord instances that could then be used for experimenting with different variations of the sound sources. The database is organized so that it can be used for different experiments other than chord detection and also the size of the database can be increased.

5. EVALUATION

This chapter discusses the evaluation of the guitar chord detection system. Based on the spectral whitening techniques, both of the models i.e. LPC and PHAT-β are evaluated independently according to different sets of parameters. The first section gives an overview of the evaluation methods and procedures. Section 5.2 discusses the LPC based system's evaluation while Section 5.3 shows the evaluation of PHAT-β based system. The performance of the system under different noise conditions are discussed in Section 5.4 and finally in Section 5.5 we discuss the system's performance with different distance metrics.

5.1 Overview

The performance of the system was evaluated using both of the spectral whitening techniques i.e. LPC and PHAT-β. Using both of the techniques, the system was tested with four acoustic guitars; two nylon string and two steel string. The guitars are denoted as Nylon I, Nylon II, Steel I and Steel II.

The tests were conducted by selecting first one guitar for testing purposes then the chord instances from the other three guitars were trained and their features (patterns) were stored. Similarity was calculated between chord instances of the test data and the relevant features present in the reference database. For example if the test data is a C major chord, then based on the prior knowledge the similarity would be only calculated with the patterns (features) of C major and its mistakes. Cosine distance metric is used for similarity computation. Finally, the classification would be done based on the k-NN classifiers.This leads to a 4-fold cross validation test procedure leading to four different test cases.

The system is said to detect a chord instance correctly if a classified chord is exactly equal to the test chord based on the prior knowledge of the test chord e.g. if the

test chord is C major and the classified chord is also C major then the system has detected a chord instance correctly. With respect to this evaluation method, the performance of the chord detection system was also calculated in accordance with each guitar and can be shown as:

$$Performance = \frac{Number\ of\ correctly\ detected\ chords}{Total\ chords\ detected} \times 100\% \tag{5.1}$$

5.2 LPC Based System

The performance of the LPC based system depends on four parameters. These parameters are shown as follows:

1. Maximum frequency (f_{max}) of the feature vector (FV)
2. DFT length
3. LPC filter coefficients
4. k for k-NN classifier

We vary each parameter on an individual basis while keeping others as constant. In this way we are able to evaluate the system by selecting the best possible values for each parameter.

We begin the evaluation by first varying the f_{max} of the feature vector while the other parameters are kept constant. The f_{max} of the feature vector defines the amount of features that are selected for the evaluation. For example, if the f_{max} of the feature vector is 3KHz, then the system will only consider features up to 3KHz from the feature vector for evaluation. The other parameters i.e the length of DFT, LPC filter coefficients and k for k-NN classifier are kept constant. Table 5.1 shows the performance of the system with different maximum frequencies of the feature vector.

Table 5.1: LPC based system's performance with varying f_{max} of the feature vector

f_{max}	Nylon I	Steel I	Nylon II	Steel II	Mean
1KHz	**63.0%**	**65.4%**	**64.5%**	**73.8%**	**66.6%**
2KHz	59.7%	61.8%	61.3%	68.9%	62.9%
3KHz	58.0%	60.7%	58.9%	67.1%	61.1%
4KHz	52.3%	58.8%	60.6%	65.4%	59.2%
5KHz	50.4%	57.3%	59.9%	62.7%	57.5%
6KHz	49.7%	56.3%	58.6%	62.2%	56.7%
7KHz	49.3%	56.4%	58.8%	62.3%	56.7%
8KHz	48.1%	56.1%	58.1%	62.3%	56.1%

In Table 5.1, the DFT length was kept to be *8192* while the number of LPC filter coefficients used were *12*. The value of k was kept to be 1. We can clearly see from the above table that LPC based system gives the best detection results when features up to *1KHz* are used from feature vector. The LPC based system performs better with low frequencies since the cosine distance metric measures the weight of all the frequencies equally and the lowest are the most informative. Therefore when we do not use the higher frequencies from the feature vector, the LPC based system performs better. For further evaluation of the LPC based system we will use up to 1KHz of features.

Now we move to the next parameter on which the performance of the LPC based system is also highly dependent i.e. DFT length. For evaluating the system with different DFT lengths we keep the other parameters constant. Table 5.2 shows the performance of the system with different DFT lengths.

Table 5.2: LPC based system's performance with different DFT lengths

DFT Length	Nylon I	Steel I	Nylon II	Steel II	Mean
1024	43.1%	47.5%	44.1%	57.0%	47.9%
2048	56.4%	58.9%	53.6%	67.4%	59.1%
4096	60.2%	63.9%	61.9%	72.6%	64.6%
8192	**63.0%**	**65.4%**	**64.5%**	**73.8%**	**66.6%**
16384	63.5%	65.2%	65.4%	71.9%	66.5%
32768	62.1%	65.0%	65.1%	71.5%	65.9%

In Table 5.2, we only used frequencies up to *1KHz* from the feature vector while the number of LPC filter coefficients are *12*. The value of k is kept to be 1. We can

clearly see from the table that the system gives better detection results with DFT length of *8192*.

We further investigate the LPC based system's performance based on different amount of LPC filter coefficients. For this purpose we vary the amount of filter coefficients while the other parameters are kept constant. The other parameters include the maximum frequency of the feature vector, DFT length and the value of k for k-NN classifier. Frequencies up to *1KHz* are used from feature vector while the length of DFT is *8192*. The value of k is kept as *1*. Table 5.3 shows the system's performance with respect to different number of LPC filter coefficients.

Table 5.3: LPC based system's performance with different number of LPC filter coefficients

LPC Coefficients	Nylon I	Steel I	Nylon II	Steel II	Mean
1	61.0%	61.7%	59.9%	72.5%	63.7%
2	62.5%	67.0%	63.9%	73.0%	66.6%
3	64.8%	65.2%	66.4%	73.8%	67.5%
4	66.1%	65.3%	66.0%	73.7%	67.7%
5	67.1%	63.5%	65.2%	72.7%	67.1%
6	66.7%	62.9%	65.1%	74.3%	67.2%
7	65.1%	64.2%	63.6%	73.2%	66.5%
8	64.9%	65.2%	64.1%	74.3%	67.1%
9	64.8%	65.1%	65.2%	74.3%	67.3%
10	64.8%	65.3%	65.2%	74.3%	67.4%
11	64.2%	65.3%	64.7%	73.6%	66.9%
12	64.8%	65.4%	65.4%	73.8%	67.3%
13	65.1%	65.4%	66.2%	73.9%	67.6%
14	64.7%	65.3%	65.8%	73.6%	67.3%
15	65.5%	67.7%	65.4%	74.3%	68.2%
16	65.5%	67.4%	66.7%	75.1%	68.6%
17	66.0%	67.2%	66.7%	74.7%	68.6%
18	65.5%	67.6%	66.7%	75.0%	68.7%
19	**64.8%**	**67.7%**	**67.6%**	**75.8%**	**68.9%**
20	65.4%	66.8%	67.4%	75.6%	68.8%

From the above table, it can be seen that the system performs better when we use *19* LPC filter coefficients. For further evaluation we keep the number of LPC filter coefficients as *19*, DFT length to be *8192* and f_{max} of the feature vector as *1KHz*. The last parameter in the evaluation of LPC based system is the value of k for k-NN classifier. Table 5.4 shows the performance of the system according to different

values of k.

Table 5.4: LPC based system's performance with different values of k

k	Nylon I	Steel I	Nylon II	Steel II	Mean
1	64.8%	65.4%	65.4%	73.8%	67.3%
3	64.9%	68.6%	67.1%	73.5%	68.5%
5	67.6%	70.2%	66.7%	75.8%	70.1%
7	68.8%	71.6%	71.0%	76.0%	71.8%
9	68.8%	70.8%	71.1%	74.9%	71.4%
11	68.2%	70.0%	71.2%	76.3%	71.4%
13	69.7%	69.1%	70.2%	75.5%	71.1%
15	**72.2%**	**70.0%**	**71.1%**	**74.9%**	**72.0%**
17	70.3%	70.1%	71.2%	73.6%	71.3%

It can be seen from Table 5.4 that the LPC based system performs best when the value of k is *13*. The LPC based system gives the best chord detection results with the following parameters:

- f_{max} of the feature vector: **1KHz**
- DFT length: **8192**
- Number of LPC filter coefficients: **19**
- k for k-NN classifier: **15**

The LPC based system's performance and random guess rate are shown in Table 5.5.

Table 5.5: LPC based system's performance and random guess rate

Case	Nylon I	Steel I	Nylon II	Steel II	Mean
Efficiency	72.2%	70.0%	71.1%	74.9%	72.00%
Random guess rate	22.1%	22.1%	22.2%	22.2%	22.15%

5.3 PHAT-β Based System

The performance of the PHAT-β based system also depends on four parameters. These parameters are shown as follows:

1. f_{max} of the feature vector
2. DFT length
3. Whitening parameter β
4. k for k-NN classifier

The PHAT-β based system is also evaluated in the similar manner as the LPC based system. We select and vary each parameter one at a time while keeping others as constant. In this manner we are able to get the best possible values for parameters with which the system performs best. We begin the evaluation by first varying the f_{max} of the feature vector while keeping other parameters constant.The other parameters include the DFT length, whitening parameter β and k for k-NN classifier. Table 5.6 shows the performance of the PHAT-β based system with varying f_{max} of the feature vector.

Table 5.6: PHAT-β based system's performance with varying f_{max} of the feature vector

f_{max}	Nylon I	Steel I	Nylon II	Steel II	Mean
1KHz	67.2%	78.4%	72.7%	83.4%	75.4%
2KHz	73.6%	80.6%	74.8%	82.2%	77.8%
3KHz	**73.7%**	**80.9%**	**75.0%**	**83.7%**	**78.3%**
4KHz	72.8%	80.7%	74.9%	83.2%	77.9%
5KHz	73.5%	80.5%	75.2%	82.7%	77.9%
6KHz	73.5%	80.6%	75.5%	82.2%	77.9%
7KHz	73.2%	80.4%	74.7%	82.0%	77.5%
8KHz	73.3%	80.7%	74.3%	82.0%	77.5%

In Table 5.6, the DFT length was kept to be *8192* while the whitening parameter β was kept as *0.55*. The value of k was kept to be 1. It can be clearly seen from the above table that, when we select features up to 3KHz from the feature vector, the PHAT-β based system gives the best detection results. For further evaluation of the PHAT-β based system we will use up to 4KHz of features from feature vector.

Now we move to the next parameter i.e. DFT length. The performance of the PHAT-β based system is also highly dependent on the length of DFT. For evaluating the system with different DFT lengths we keep the other parameters constant. Table 5.7 shows the performance of the system with different DFT lengths.

Table 5.7: PHAT-β based system's performance with different DFT lengths

DFT Length	Nylon I	Steel I	Nylon II	Steel II	Mean
1024	43.7%	49.6%	48.4%	57.7%	49.8%
2048	57.1%	65.8%	60.4%	69.7%	63.2%
4096	65.8%	75.7%	68.9%	79.6%	72.5%
8192	**73.7%**	**80.9%**	**75.0%**	**83.7%**	**78.3%**
16384	73.8%	77.3%	76.3%	82.3%	77.4%
32768	75.1%	77.1%	76.7%	82.1%	77.7%

In table 5.7, we used up to *3KHz* of features from feature vector. The whitening parameter β was kept to be *0.55* while the value of k was kept as *1*. It can be noticed that DFT with length *8192* gives the best detection results. For further evaluation we will use *8192* as the length of DFT.

We further investigate the performance of the PHAT-β based system by varying the whitening parameter β. For this purpose we vary the value of β and keep the other parameters as constants. The other parameters include the f_{max} of the feature vector, DFT length and the value of k for k-NN classifier. The maximum frequency of the feature vector remains to be *3KHz* while the DFT length is kept as *8192*. The value of k is kept to be *1*.

Table 5.8 shows the PHAT-β based system's performance with respect varying whitening parameter β.

Table 5.8: PHAT-β based system's performance with different values of β

β	Nylon I	Steel I	Nylon II	Steel II	Mean
0.00	58.3%	66.2%	63.1%	79.4%	66.7%
0.05	63.5%	67.8%	64.2%	80.6%	69.0%
0.10	65.5%	69.5%	65.9%	82.1%	70.7%
0.15	67.2%	70.9%	67.1%	82.4%	71.9%
0.20	68.5%	72.6%	68.8%	83.4%	73.3%
0.25	70.6%	74.2%	69.6%	84.7%	74.7%
0.30	71.9%	75.9%	71.3%	84.5%	75.9%
0.35	73.2%	77.5%	72.6%	84.9%	77.0%
0.40	72.6%	79.5%	74.9%	84.9%	77.9%
0.45	74.4%	80.7%	75.9%	84.3%	78.8%
0.50	**74.6%**	**80.9%**	**76.7%**	**83.7%**	**78.9%**
0.55	73.7%	80.9%	75.0%	83.7%	78.3%
0.60	72.7%	80.6%	74.4%	81.6%	77.3%
0.65	71.0%	77.8%	72.4%	80.3%	75.4%
0.70	69.0%	75.1%	70.5%	79.6%	73.5%
0.75	67.1%	71.4%	67.1%	77.3%	70.7%
0.80	61.3%	65.3%	62.8%	72.5%	65.4%
0.85	56.2%	57.7%	56.0%	68.1%	59.5%
0.90	50.5%	51.1%	51.4%	62.9%	53.9%
0.95	45.2%	45.4%	45.5%	56.8%	48.2%
1.00	41.0%	37.1%	40.7%	51.3%	42.5%

From the above table, it can be observed that the system performs best when the value of β is *0.5*. For further evaluation we keep the value of β to be *0.5*, DFT length to be *8192* and f_{max} of the feature vector to be *3KHz*. The last parameter in the evaluation of PHAT-β based system is the value of k for k-NN classifier. Table 5.10 shows the performance of the system according to different values of k.

Table 5.9: PHAT-β based system's performance with different values of k

k	Nylon I	Steel I	Nylon II	Steel II	Mean
1	74.6%	80.9%	76.7%	83.7%	78.9%
3	72.7%	82.7%	76.8%	84.4%	79.1%
5	75.0%	82.4%	80.3%	86.2%	80.9%
7	74.5%	83.0%	80.6%	87.5%	81.4%
9	75.4%	83.1%	81.4%	86.1%	81.5%
11	73.6%	83.1%	82.6%	87.4%	81.7%
13	**75.7%**	**84.6%**	**83.9%**	**86.0%**	**82.5%**
15	75.1%	82.1%	83.4%	87.0%	81.9%
17	72.9%	83.9%	82.4%	87.7%	81.7%

It can be seen from table 5.9 that the system performs best when the value of k is *11*. The PHAT-β based system gives the best chord detection results with the following parameters:

- f_{max} of the feature vector: **3KHz**
- DFT length: **8192**
- Whiteneing parameter β: **0.50**
- Value of k: **13**

The overall PHAT-β based system's performance and random guess rate are shown in table 5.10.

Table 5.10: PHAT-β based system's performance and random guess rate

Case	Nylon I	Steel I	Nylon II	Steel II	Mean
Efficiency	75.7%	84.6%	83.9%	86.0%	82.50%
Random guess rate	22.1%	22.1%	22.2%	22.2%	22.15%

5.4 System's Performance Under Different Noise Conditions

The LPC based system and PHAT-β based system were evaluated under different noise conditions. In order to check the robustness of the system we added a number of realistic noises and white noise to our test data and conducted all the four test cases. The signal to noise ratio (SNR) was kept to be 0dB for all the realistic noises while it varied from -20dB to 20dB for the white noise. All the test parameters were

kept constant. For evaluating the system with realistic noise, a database was used that was created by the Audio Research Team of TUT in 2001 [40]. The details related to the recording equipment that was used to record the realistic noises are as follows:

- Two AKG C460B microphones
- Sony TCD-D10 Pro DAT Recorder
- PSP-2 stereo pre-amplifier
- Bruel and Kjaer Type 4321 microphone calibrator

Some daily life noises were added to the test data and then the performance of the system was evaluated with noisy test data. Noise of automobiles, outdoor noise of streets and roads, noise of indoor public social places like restaurants and cafeterias, natural noises, and indoor noise from offices and homes were used for evaluation.

Outdoor Places

The outdoor sounds heard in our daily lives e.g. traffic sounds, sounds of people talking, wind sound, cell phone ringing etc, were added to the test data and then the performance of the system was evaluated. Table 5.11 shows the performance of the system under different outdoor noises.

Table 5.11: System's performance after adding different outdoor noises to the test data

System	Case	Nylon I	Nylon II	Steel I	Steel II	Mean
LPC	Natural 1	48.0%	60.3%	53.5%	67.2%	57.2%
	Natural 2	63.9%	64.9%	62.4%	71.7%	65.7%
	Street 1	42.8%	53.7%	41.4%	60.6%	49.6%
	Street 2	55.0%	61.8%	55.1%	68.3%	60.0%
	Road 1	53.8%	63.9%	57.1%	69.4%	61.0%
	Road 2	59.3%	64.0%	56.2%	70.3%	62.4%
PHAT-β	Natural 1	60.5%	73.2%	69.0%	78.4%	70.2%
	Natural 2	70.0%	81.0%	79.5%	85.7%	79.0%
	Street 1	56.4%	63.7%	63.8%	69.6%	63.4%
	Street 2	63.1%	76.0%	72.5%	80.3%	72.9%
	Road 1	63.9%	77.2%	73.7%	81.8%	74.1%
	Road 2	65.2%	78.4%	76.5%	84.9%	76.2%

It can be observed from Table 5.11 that the LPC based system shows on average 59.3% while the PHAT-β based system shows 72.6% accuracy when evaluated under outdoor noises.

Indoor Public Places

The performance of the system was also evaluated by adding the noise of indoor public places to the test data. Noises like inside shopping malls, cafeterias, restaurants, offices and indoor crowded places were used for the evaluation. Table 5.12 shows the performance of the system under different indoor public place noises.

Table 5.12: System's performance after adding different indoor public place noises to the test data

System	Case	Nylon I	Nylon II	Steel I	Steel II	Mean
LPC	Supermarket 1	47.2%	59.1%	50.2%	66.4%	55.7%
	Supermarket 2	56.1%	62.1%	55.3%	69.2%	60.6%
	Cafeteria 1	56.3%	63.8%	55.9%	69.8%	61.4%
	Cafeteria 2	57.8%	65.2%	57.8%	70.0%	62.7%
	Restaurant 1	38.3%	55.3%	43.4%	58.6%	48.9%
	Restaurant 2	57.9%	64.2%	57.8%	70.5%	62.6%
	Office 1	61.1%	65.9%	57.7%	69.2%	63.4%
	Office 2	32.1%	48.1%	38.7%	49.7%	42.1%
	Croud 1	35.6%	52.5%	39.3%	55.2%	45.65%
	Croud 2	48.0%	61.6%	49.7%	66.0%	56.3%
PHAT-β	Supermarket 1	57.6%	69.8%	63.8%	72.5%	65.9%
	Supermarket 2	67.5%	74.8%	77.2%	84.9%	76.1%
	Cafeteria 1	62.4%	75.6%	76.4%	78.7%	73.2%
	Cafeteria 2	62.4%	75.6%	73.6%	79.4%	72.7%
	Restaurant 1	59.2%	67.0%	59.7%	68.1%	63.5%
	Restaurant 2	64.5%	74.7%	70.9%	81.7%	72.9%
	Office 1	67.6%	81.2%	79.3%	85.2%	78.3%
	Office 2	40.1%	62.8%	43.3%	57.8%	51.0%
	Croud 1	43.7%	58.7%	44.7%	57.9%	51.2%
	Croud 2	55.4%	69.0%	64.9%	75.9%	66.3%

From the Table 5.12, it can be observed that the LPC based system shows on average 56.8% while PHAT-β based system shows 67.1% accuracy when evaluated with a set of noises from indoor social places.

Indoor Private Places

System's performance was also evaluated by indoor residential noises. Sounds inside a house e.g. Television noise in the living room, sounds inside of the bathroom, sounds of utensils in kitchen etc. were added to the test for the evaluation of the system. Table 5.13 shows the performance of the system under different indoor place noises.

Table 5.13: System's performance after adding different indoor private place noises to the test data

System	Case	Nylon I	Nylon II	Steel I	Steel II	Mean
LPC	Kitchen 1	62.5%	65.2%	62.3%	69.8%	64.9%
	Kitchen 2	54.0%	65.1%	56.3%	68.8%	61.0%
	Living Room	23.6%	23.9%	18.3%	24.2%	22.5%
	Bathroom 1	63.8%	63.5%	62.7%	69.5%	64.8%
	Bathroom 2	60.5%	63.9%	64.6%	71.5%	65.1%
PHAT-β	Kitchen 1	70.5%	80.7%	79.2%	85.5%	78.9%
	Kitchen 2	67.2%	75.1%	72.2%	80.5%	73.7%
	Living Room	25.8%	25.6%	26.5%	33.1%	27.7%
	Bathroom 1	74.5%	85.7%	81.4%	85.3%	81.7%
	Bathroom 2	74.3%	83.2%	82.4%	85.9%	81.4%

We can observe from the Table 5.13 that the LPC based system shows 55.6% accuracy on average while the PHAT-β based system shows 68.7%. The performance of both of the system is poor when it is tested with the living room noise. One of the major reasons for the poor performance is the very loud television sound that was added in the test data.

Automobiles

Both of the systems were evaluated from the sounds produced by an automobile. The sound produced by a moving bus and a car when recorded from inside were used to evaluate the performance. The noise include moving of vehicle, opening and closing of doors, car stereo, conversation of passengers, brake sounds, thumping etc. Table 5.14 shows the performance of the system under automobile noise.

Table 5.14: System's performance after adding automobile noise to the test data

System	Case	Nylon I	Nylon II	Steel I	Steel II	Mean
LPC	Bus 1	42.2%	60.0%	48.7%	67.0%	54.5%
	Bus 2	57.4%	64.3%	57.7%	71.3%	62.6%
	Car 1	67.0%	67.4%	66.7%	71.9%	68.2%
	Car 2	71.5%	68.9%	69.2%	76.1%	71.4%
PHAT-β	Bus 1	39.7%	58.1%	46.8%	60.9%	51.4%
	Bus 2	58.1%	72.1%	63.8%	77.8%	67.9%
	Car 1	69.5%	77.8%	75.0%	84.3%	76.6%
	Car 2	73.4%	83.8%	80.4%	85.0%	80.6%

From the Table 5.14, it can be observed that the LPC based system shows on average 64.2% accuracy while the PHAT-β based system shows 69.1%.

White Noise

Both of the systems were also evaluated under white noise. To have an over all performance of the system, we added white noise to our test data with varying SNR and conducted all the four tests. All the test parameters were kept constant while only the SNR of the test data was varied. Table 5.15 shows the performance of the LPC based system after adding white noise.

Table 5.15: LPC based system's performance after adding white noise to the test data

SNR (dB)	Nylon I	Steel I	Nylon II	Steel II	Mean
-20	43.9%	52.2%	50.5%	52.9%	49.8%
-15	51.7%	59.9%	51.6%	59.2%	55.6%
-10	56.7%	60.1%	58.4%	68.3%	60.8%
-5	58.2%	62.5%	60.4%	72.2%	63.3%
0	60.8%	63.9%	65.1%	73.3%	65.7%
5	63.0%	63.7%	70.4%	71.3%	67.1%
10	66.8%	63.2%	65.6%	71.2%	66.7%
15	67.0%	64.4%	68.5%	72.6%	68.1%
20	69.6%	66.1%	68.8%	72.3%	69.2%

To have an average performance of the LPC based system against a unique SNR, we plot the mean of the performance of all the four test cases. Figure 5.1 shows a graph between different ratios of SNR and performance of the system.

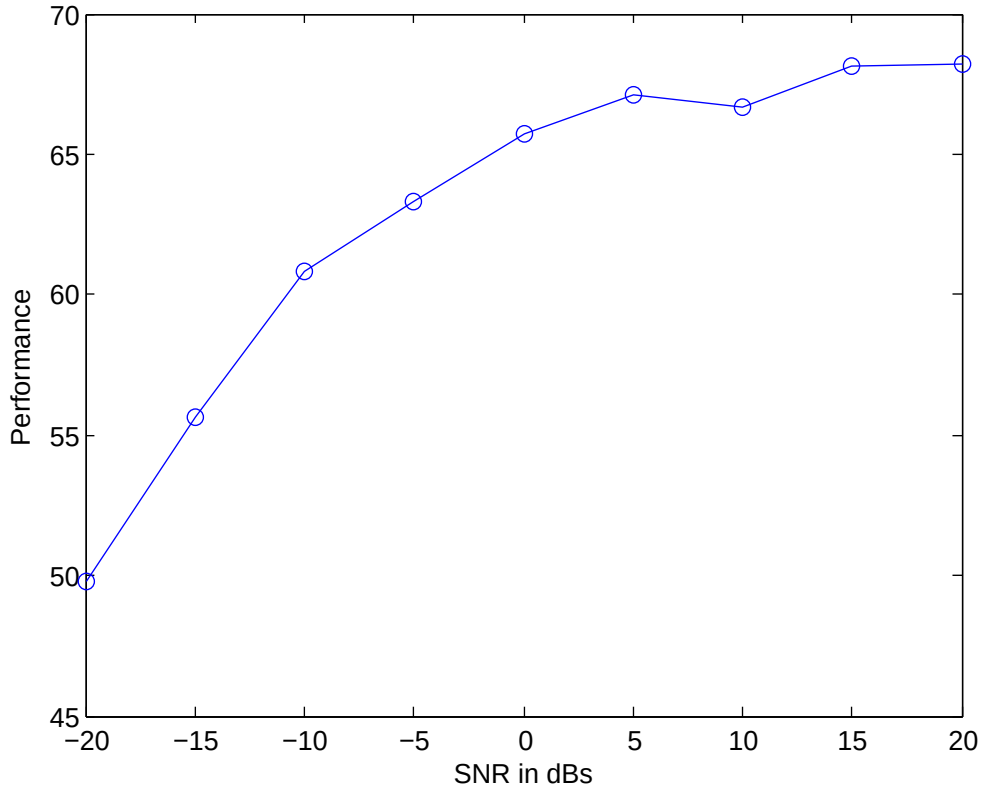

Figure 5.1: LPC based system's performance with varying signal to noise ratio

It can be seen from Figure 5.1 that the LPC based system shows stable response from SNR = 20dB to SNR = 0dB with an average efficiency being 67%. However, the performance of the system starts fall down when SNR is less than 0dB.

Similarly, the performance of the PHAT-β based system after adding white noise is shown in Table 5.16 shows.

Table 5.16: PHAT-β based system's performance after adding white noise to the test data

SNR (dB)	Nylon I	Steel I	Nylon II	Steel II	Mean
-20	34.2%	37.2%	34.9%	36.3%	35.6%
-15	47.1%	49.6%	52.3%	49.8%	49.7%
-10	55.3%	61.6%	63.6%	68.6%	62.2%
-5	63.1%	69.2%	66.7%	71.3%	67.6%
0	65.8%	73.4%	71.3%	81.5%	73.0%
5	68.8%	78.1%	77.7%	83.3%	76.9%
10	70.5%	78.7%	79.0%	83.9%	78.0%
15	71.3%	82.1%	80.3%	85.4%	79.8%
20	71.1%	82.5%	80.0%	85.2%	79.7%

To have an average performance of the system against a unique SNR, we plot the mean of the performance of all the four test cases. Figure 5.2 shows a graph between different ratios of SNR and efficiency of the system.

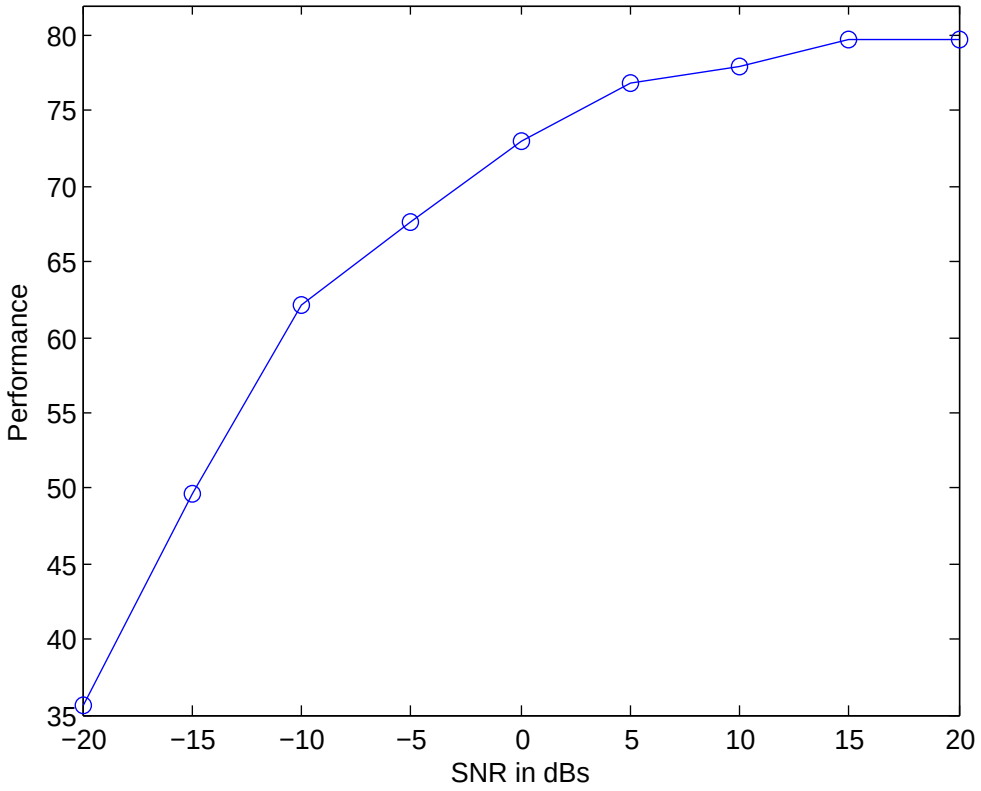

Figure 5.2: PHAT-β based system's performance with varying signal to noise ratio

It can be seen from Figure 5.2 that the PHAT-β based system shows stable response from SNR = 20dB to SNR = 0dB with an average efficiency being 77%. However, the performance of the system starts to decay when the SNR is less than 0dB.

5.5 System's Performance With Different Distance Metrics

The system was also tested with different distance metrics. The details related to different kinds of distance metrics have been discussed in section 3.5. The results from each distance metric are compared with the ones with cosine distance metric.

Euclidean

Table 5.17 shows the list of values of the parameters that give the best result using euclidean distance metric.

Table 5.17: Parameters of the system that gives the best result

System	Parameters	Value
LPC	f_{max} of the FV	1KHz
	DFT Length	16384
	LPC Coefficients	14
	k	9
PHAT-β	f_{max} of the FV	4KHz
	DFT Length	16384
	β	0.55
	k	13

Table 5.18 shows the performance of the LPC and PHAT-β based system using euclidean distance metric.

Table 5.18: System's performance with euclidean distance metrics

System	Nylon I	Steel I	Nylon II	Steel II	Mean
LPC	70.6%	69.3%	69.5%	74.9%	71.0%
PHAT-β	68.8%	78.5%	74.4%	83.2%	76.2%

From the Tables 5.18 and 5.19, we can make few observation. Some of them are as follows:

- Both of the systems show good performance, in fact the performance of LPC based system with euclidean distance metric is very close to the one with cosine distance metric.
- The number of LPC coefficients used are quite less (14 compare to 19) and also the number of nearest neighbors (9 compare to 15).
- The LPC based system uses less nearest neighbors (13 compare to 15) when correlation distance metrics is used.

- The PHAT-β based system uses more features (4KHz compare to 3KHz) when we use correlation distance metrics.
- Both of the systems are computationally heavy because of very long DFT lengths.

Correlation

Table 5.19 shows the list of values of the parameters that give the best result using correlation distance metric.

Table 5.19: Parameters of the system that gives the best result

System	Parameters	Value
LPC	f_{max} of the FV	1KHz
	DFT Length	16384
	LPC Coefficients	19
	k	13
PHAT-β	f_{max} of the FV	4KHz
	DFT Length	16384
	β	0.5
	k	15

Table 5.20 shows the performance of the LPC and PHAT-β based system using correlation distance metric.

Table 5.20: System's performance with correlation distance metrics

System	Nylon I	Steel I	Nylon II	Steel II	Mean
LPC	71.6%	67.1%	72.4%	76.0%	71.7%
PHAT-β	79.5%	79.3%	79.1%	87.3%	81.3%

From the Tables 5.19 and 5.20, there can be number of observations that can be made. Some of them are as follows:

- The performance of both of the systems with correlation distance metrics are very close to the ones with cosine distance metric.
- Both of the systems use long DFT lengths as compare to the DFT lengths of the systems using cosine distance metrics i.e 16384 compare to 8192.

- The LPC based system uses less nearest neighbors (13 compare to 15) when correlation distance metrics is used.
- The PHAT-β based system uses more features (4KHz compare to 3KHz) when we use correlation distance metrics.
- As a whole, both of the systems are computationally heavy since they use very long DFT lengths.

Spearman

Table 5.21 shows the list of values of the parameters that give the best result using Spearman distance metric.

Table 5.21: Parameters of the system that gives the best result

System	Parameters	Value
LPC	f_{max} of the FV	2KHz
	DFT Length	16384
	LPC Coefficients	16
	k	11
PHAT-β	f_{max} of the FV	2KHz
	DFT Length	8192
	β	0.5
	k	13

Table 5.22 shows the performance of the LPC and PHAT-β based system using Spearman distance metric.

Table 5.22: System's performance with Spearman distance metrics

System	Nylon I	Steel I	Nylon II	Steel II	Mean
LPC	71.5%	74.6%	72.1%	81.8%	75%
PHAT-β	61.6%	64.1%	66.2%	68.7%	65.1%

From the Table 5.22, we can see that PHAT-β based system with Spearman distance metric shows poor performance as compared to the one with cosine distance metric

(65.1% compare to 82.5%). The LPC based system however shows a better performance. Some interesting observations can be made about the LPC based system with Spearman distance metric.

- The performance of the LPC based system with Spearman distance metric is better than the one with cosine distance metric (75% compare to 72%).
- The LPC based system with Spearman distance metric uses more features from the feature vector (2KHz compared to 1KHz). This increases the amount of computation.
- The number of LPC coefficients used are less (16 compare to 19) and also the number of nearest neighbors (11 compare to 15). This reduces the amount of computation.
- The DFT length is long (16384 compare 8192). This increases the amount of computation by a factor 2.
- The performance of the LPC based system with Spearman distance metric is 3% better than that of cosine distance metric. Although this long DFT length makes the system slow and computationally heavy but due to better performance the Spearman distance metric should be preferred over cosine.

City Block

Table 5.23 shows the list of values of the parameters that give the best possible results using city block distance metric.

Table 5.23: Parameters of the system that gives the best result

System	Parameters	Value
LPC	f_{max} of the FV	3KHz
	DFT Length	16384
	LPC Coefficients	17
	k	11
PHAT-β	f_{max} of the FV	2KHz
	DFT Length	16384
	β	0.45
	k	11

Table 5.24 shows the performance of the LPC and PHAT-β based system using city block distance metric.

Table 5.24: System's performance with city block distance metrics

System	Nylon I	Steel I	Nylon II	Steel II	Mean
LPC	62.1%	66.7%	64.3%	69.8%	65.7%
PHAT-β	66.7%	74.2%	71.7%	78.0%	72.6%

From the Table 5.24, it can be observed that the performance of both of the systems with city block distance metric is less than when compared with the system using cosine distance metric. Both of the systems use very long DFT lengths, this criteria alone makes this system computationally less efficient and heavy when compared with the system with cosine distance metric.

It can be concluded that the LPC based system using Spearman distance metric gives the best detection results while the PHAT-β based system with cosine distance metric gives the best detection results.

6. CONCLUSION

In this thesis, we have studied automatic guitar chord detection using isolated chords as test and trained data. Furthermore we have presented a system based on spectral whitening and pattern matching that detects the correctness of the played chord by guitar. An audio consisting of a single chord and having the prior knowledge about the target chord determines whether the system has performed the detection correctly or not.

Two different types of spectral whitening techniques have been discussed in this thesis. In both of the techniques, spectrum of the chord is whitened or partially whitened and a certain region is selected as a feature vector based on the energy distribution in the frequency domain. The cosine distance is calculated between the test chord and a reference chord database. At the end, chord detection is done using k-NN classifiers.

The result show that the performance of the LPC based system is 72% while that of PHAT-β is 82.5%. Both of the systems have also been tested under realistic noises. Some very interesting comparisons have been made while testing the system with different distance metrics. The results show that the research conducted in this thesis can have significant impact on some practical applications. Applications like computer games, human-computer interactive systems and many other musical applications can gain benefit from this research.

There is still a lot of work to be done before we can have a complete and robust guitar chord detection system. Future works can include detecting the chords that are played with different strumming patterns and complex finger picking technique e.g. rasgueado technique.

REFERENCES

[1] M. Ryynänen. *Automatic transcription of pitch content in music and selected applications.* PhD thesis, Tampere University of Technology, 2008.

[2] V. Zenz and A. Rauber. *Automatic chord detection incorporating beat and key detection.* IEEE Internation Conference on Signal Processing and Communication (ICSPC 2007), pp. 1175-1178, November 2007.

[3] V. Zenz. *Automatic chord detection in polyphonic audio data.* M.Sc. thesis, Vienna University of Technology, 2007.

[4] F. Mazhar, T. Heittola, T. Virtanen, J. Holm. *Automatic scoring of guitar chords.* AES 45^{th} Conference on Applications of Time Frequency Processsing in Audio, Helsinki, Finland, March 2012.

[5] A. D. Stark and M. D. Plumbley. *Real-time chord recognition for live performances.* In proceedings of International Computer Music Conference (ICMC), 2009.

[6] K. Lee. *Automatic chord recognition from audio using enhanced pitch class profile.* In proceedings of the ICMC, 2006.

[7] A. P. Klapuri and M. Davy. *Signal processing methods for music transcription.* Springer, New York, NY, 2006.

[8] P. Smaragdis and J. C. Brown. *Non-negative matrix factorization for polyphonic music transcription.* IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, October 2003.

[9] H. B. Barlow. *Sensory mechanisms, the reduction of redundancy and intelligence.* In Symposium on the Mechanization of Thought Processes. National Physical Laboratory Symposium No. 10. (1959).

[10] K. Jarret. *J. S. Bach, Das Wohltemperierte Klavier, Buch I.* ECM Records, CD 2, Track 8 (1988).

[11] M.P. Ryynänen and A. Klapuri. *Polyphonic music transcription using note event modeling.* IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, October 2005.

[12] A. Klapuri. *A perceptually motivated multiple-F0 estimation method.* In proceedings of IEEE workshop on Applications of Signal Processing to Audio and Acoustics, October 2005.

[13] S. J. Young, N. H. Russel and J. H. S. Thornton. *Token passing: a simple conceptual model for connected speech recognition systems.* Cambridge University Engineering Department, Tech. Rep., July 1989.

[14] M. Goto, H. HashiGuchi, T. Nishimura and R. Oka. *RWC music database: Music genre database and musical instrument sound database.* In proceedings of 4^{th} ISMIR, October, 2003.

[15] T. Yoshioka, T. Kitahara, K. Komatani, T. Ogata, and H. G. Okuno. *Automatic Chord Transcription with Concurrent Recognition of Chord Symbols and Boundaries.* In proceedings of 5th ISMIR, 2004, pp. 100-105.

[16] M. Goto, H. Hashiguchi, T. Nishimura and R. Oka. *RWC Mucis Database: Popular, Classical and Jazz Music Databases.* In proceedings of ISMIR, pp. 287-288, 2002.

[17] T. Fujishima. *Realtime chord recognition of musical sound: a system using common liso music.* In proceedings of ICMC, Beiging: International Computer Music Association, 1999.

[18] L. R. Rabiner. *A tutorial on Hidden Markov Models and selected applications in speech recognition.* In proceedings of IEEE, vol. 77, no. 2, pp. 257-286, 1989.

[19] A. Shah and D. P. Ellis. *Chord segmentation and recognition using EM-trained Hidden Markov Models.* In proceedings of the International Symposium on Music Information Retrieval (ISMIR), Baltimore, MD, 2003.

[20] P. M. Baggenstoss. *A modified baum-welch algorithm for hidden markov models with multiple observation spaces.* IEEE Transactions on Speech and Audio Processing, vol. 9, no. 4, May 2001.

[21] B. Su and S. K. Jeng. *Multi-timbre chord classification using wavelet transform and self organized map neural networks.* In proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2001, pp.3370-3380.

[22] F. X. Nsabimana and U. Zolzer. *Transient encoding of audio signals using dyadic approximations.* In Proceedings of 10^{th} International Conference on Digital Audio Effects (DAFx-07), Bordeaux, France, September 2007.

[23] H. Thornburg. *Detection and modeling of transient audio signals with prior information.* PhD thesis, Stanford University, 2005.

[24] K. Jensen. *Envelope model and isolated musical sounds.* In proceedings of the 2nd COST G-6 Workshop on Digital Audio Effects (DAFx99), NTNU, Trondheim, December 1999.

[25] Douglas O'Shaughnessy. *Linear Predictive Coding-one popular technique of analyzing certain physical signals.* Potentials IEEE, issue: 1, volume: 7, pp. 29-32, February 1988.

[26] J. Makhoul. *Linear prediction: a tutorial review.* IEEE Proceedings, volume 63, pp. 561-580, 1975.

[27] S. Kay and S. Marple. *Spectrum analysis-a modern perspective.* IEEE Proceedings, volume 69, pp. 1380-1418, 1981.

[28] J. Durbin. *The fitting of time series model.* Review of the International Statistical Institute 28, pp. 233-243, (1960).

[29] A. V. Oppenheim and R. W. Schafer. *Discrete Time Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, pp. 311-312, 1989.

[30] A. V. Oppenheim and J. S. Lim. *The importance of phase in signals.* IEEE Proceedings, Volume 69, Issue 5, pp. 529-541, 1981.

[31] J. H. DiBiase, H.F. Silverman and M. S. Brandstein. *Robust localization in reverberant rooms.* Microphone Arrays, Signal Processing Techniques and Applications, Springer, Berlin, 2001, pp. 157-180.

[32] B. Mungamuru and P. Aarabi. *Enhanced sound localization.* IEEE Transactions on Systems, Man and Cybernetics-Part B, Volume 34, No. 3, 2004

[33] K. D. Donohue, J. Hannemann and H. G. Dietz. *Performance of phase transform for detecting sound sources with microphone arrays in reverberant and noisy environments.* Signal Processing, Volume 87, Issue 7, July 2007, pp. 1677-1691.

[34] D. Aiger and H. Talbot. *The Phase Only Transform for unsupervised surface defect detection.* IEEE conference on Computer Vision and Pattern Recognition (CVPR), June 2010, pp. 295-302.

[35] K.D. Donohue, A. Agrinsoni and J. Hannemann. *Audio signal delay estimation using partial whitening.* Proceeding of the IEEE, Southeastcon, pp. 466-471, March 2007.

[36] C. Spearman. *The proof of measurement of association between two things.* The American Journal of Psychology, Vol. 15, No. 1, January 1904, pp. 72-101.

[37] S. A. Dudani. *The Distance-Weighted k-Nearest-Neighbor Rule.* IEEE Transaction on Systems, Man and Cybernetics, Issue 4, April 1976, pp. 325-327.

[38] T. Cover and P. Hart. *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, Volume 13, Issue 1, January 1967, pp. 21-27.

[39] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, 1999, pp. 44-46.

[40] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho and J. Huopaniemi. *Audio-based context recognition.* IEEE Transactions on Audio, Speech and Language Processing, Vol. 14, No. 1, January 2006.

A. APPENDIX

Table A.1: Chord instances with their description (1)

Name	e	A	D	G	B	E	Description
C Major	x	3	2	0	1	0	Correct version
Mistake 1	0	3	2	0	1	0	One extra note on e string
Mistake 2	x	3	x	0	1	0	Missed one note on D string
Mistake 3	x	3	x	x	1	x	Missed three notes on D, G and E strings
Mistake 4	x	3	2	0	1	x	Missed one note on E string
Mistake 5	x	3	2	0	0	1	Wrong note on E and B strings
Mistake 6	x	3	0	2	1	0	Wrong notes on D and G strings
Mistake 7	x	0	3	2	1	0	Wrong notes on A, D and G strings
D	x	x	0	2	3	2	Correct version
Mistake 1	x	0	0	2	3	2	One extra note on A string
Mistake 2	0	0	0	2	3	2	Two extra notes on e and A strings
Mistake 3	x	x	0	2	3	x	Missed one note on E string
Mistake 4	x	x	0	2	3	0	Wrong note on E string
D Minor	x	x	0	2	3	1	Correct version
Mistake 1	x	0	0	2	3	1	One extra note on A string
Mistake 2	0	0	0	2	3	1	Two extra notes on e and A strings
Mistake 3	x	x	0	2	3	x	Missed one note on E string
Mistake 4	x	x	0	2	3	0	Wrong note on E string
E	0	2	2	1	0	0	Correct version
Mistake 1	0	2	2	1	x	x	Missed two notes on B and E strings
Mistake 2	x	2	2	1	0	0	Missed one note on e string
Mistake 3	0	2	2	x	0	0	Missed one note on G string
Mistake 4	x	0	2	2	1	0	Wrong chord
Mistake 5	0	2	2	0	0	0	Wrong note on G string
E Minor	0	2	2	0	0	0	Correct version
Mistake 1	0	2	2	0	x	x	Missed two notes on B and E strings
Mistake 2	0	2	2	x	0	0	Missed one note on G string
F (1)	1	3	3	2	1	1	Correct version
Mistake 1	1	3	3	2	x	x	Missed two notes on B and E strings
Mistake 2	1	3	3	x	x	1	Missed two notes on G and B strings
Mistake 3	1	3	3	x	x	x	Missed three notes on G,B and E strings
Mistake 4	1	3	3	x	1	1	Missed one note on G string
Mistake 5	1	3	3	2	x	1	Missed one note on B string

Table A.2: Chord instances with their description (2)

Name	e	A	D	G	B	E	Description
F (2)	x	x	3	2	1	1	Correct version
Mistake 6	x	x	3	2	1	x	Missed one note on E string
Mistake 7	x	0	3	2	1	1	One extra note on A string
Mistake 8	x	x	3	x	1	1	Missed one note on G string
F (3)	x	x	3	2	1	x	Correct version
Mistake 9	x	0	3	2	1	x	One extra note on A string
Mistake 10	x	x	3	x	1	1	Missed one note on G string
F Minor	1	3	3	1	1	1	Correct version
Mistake 1	1	3	3	1	x	x	Missed two notes on B and E strings
Mistake 2	1	3	3	x	x	1	Missed two notes on G and B strings
Mistake 3	1	3	3	x	x	x	Missed three notes on G,B and E strings
Mistake 4	1	3	3	x	1	1	Missed one note on G string
G	3	2	0	0	3	3	Correct version
Mistake 1	3	x	0	0	3	3	Missed one note on A string
Mistake 2	3	2	0	3	0	3	Wrong notes on G and B strings
Mistake 3	3	2	x	0	3	3	Missed one note on D string
A	x	0	2	2	2	0	Correct version
Mistake 1	0	0	2	2	2	0	One extra note on e string
Mistake 2	x	0	2	2	2	x	Missed one note on E string
Mistake 3	x	0	2	2	3	0	Wrong note on B string
Mistake 4	x	0	2	2	1	0	Wrong note on B string
A Minor	x	0	2	2	1	0	Correct version
Mistake 1	0	0	2	2	1	0	One extra note on e string
Mistake 2	x	0	2	2	1	x	Missed one note on E string
Mistake 3	0	2	2	1	0	0	Wrong chord
B	x	2	4	4	4	x	Correct version
Mistake 1	0	2	4	4	4	x	One extra note on e string
Mistake 2	x	2	4	x	4	x	Missed one note on G string
Mistake 3	x	2	4	4	x	x	Missed one note on B string
B Minor	x	2	4	4	3	2	Correct version
Mistake 1	0	2	4	4	3	2	One extra note on e string
Mistake 2	x	2	4	4	3	x	Missed one note on E string
Mistake 3	x	2	4	4	x	2	Missed one note on B string