PITCH DETECTION AND PITCH CORRECTION

Visveswaraiah Technological University,Belgaum

A PROJECT REPORT

Submitted by

KHOUSHIKH S	**(1BM16TE018)**
PRAJWAL S RAO	**(1BM16TE029)**
R ADVAITH ANANTH KRISHNAN	**(1BM16TE031)**
SRIRAM R	**(1BM16TE046)**

In partial fulfillment for award of the degree

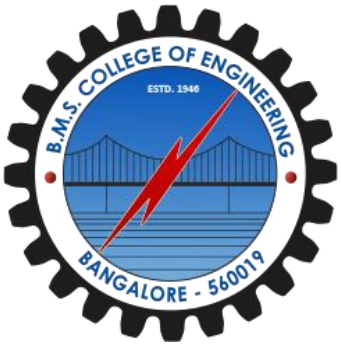
Of Bachelor of Engineering

In

TELECOMMUNICATION ENGINEERING

Under the supervision of

Dr. Balachandra K
Assistant Professor, TCE, BMSCE



Department Of Telecommunication Engineering
B.M.S COLLEGE OF ENGINEERING
(Autonomous Institution under Visveswaraiah Technological University, Belgaum)

Bull Temple Road, Basavangudi, Bengaluru- 560019

PITCH DETECTION AND PITCH CORRECTION

A PROJECT REPORT

Submitted by

KHOUSHIKH S (1BM16TE018)

PRAJWAL S RAO (1BM16TE029)

R ADVAITH ANANTH KRISHNAN (1BM16TE031)

SRIRAM R (1BM16TE046)

In partial fulfillment for award of the degree

Of Bachelor Of Engineering

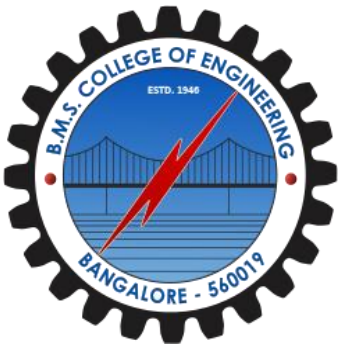
in

TELECOMMUNICATION ENGINEERING

Under the supervision of

Dr.Balachandra K

Assistant Professor, TCE, BMSCE



Department Of Telecommunication Engineering

B.M.S COLLEGE OF ENGINEERING
(Autonomous Institution under Visveswaraiah Technological University, Belgaum)

Bull Temple Road, Basavangudi, Bengaluru- 560019

B.M.S COLLEGE OF ENGINEERING
(Autonomous Institution under Visveswaraiah Technological University, Belgaum)

Bull Temple Road, Basavangudi, Bengaluru- 560019

Department Of Telecommunication Engineering

CERTIFICATE

This is to certify that project entitled "**PITCH DETECTION AND PITCH CORRECTION**" is a bonafide work carried out by **Khoushikh S(1BM16TE018), Prajwal S Rao(1BM16TE029), R Advaith Ananth Krishnan(1BM16TE031) and SriramR(1BM16TE046)** in fulfillment for completion of MAJOR PROJECT[16TE8DCMPJ] during the academic year 2019-2020.This project has been approved as it satisfies the academic requirement in respect of project work prescribed for Bachelor Of Engineering degree.

Dr.Balachandra K	Dr.B Kanmani	Dr. B. V. Ravishankar
Assistant Professor,TCE	HOD,TCE,BMSCE	Principal,BMSCE

Examiners **Signature with date**

1.

2.

B.M.S COLLEGE OF ENGINEERING
(Autonomous Institution under Visveswaraiah Technological University,Belgaum)

Bull Temple Road,Basavangudi,Bengaluru- 560019

BONAFIDE CERTIFICATE FROM THE SUPERVISOR

Department: Telecommunication Engineering

Candidate's Degree: Registered for Bachelor of Engineering

Candidate Details:

Sl.No	Student Name	USN	Student Signature
1.	KHOUSHIKH S	1BM16TE018	
2.	PRAJWAL S RAO	1BM16TE029	
3.	R ADVAITH ANANTH KRISHNAN	1BM16TE031	
4.	SRIRAM R	1BM16TE046	

Certified that the project is titled as "**PITCH DETECTION AND PITCH CORRECTION"** The candidates have carried out the project work to my satisfaction and the work is original. This project work dissertation report was thoroughly scrutinized by me. All the corrections are incorporated by the students and the project report is final and is of high standard. I duly certify the same.

Sl.No	Supervisor Name	Department/Organization/Designation	Signature
1.	Dr. BALACHANDRA K	TCE/BMSCE/ Assistant Professor	

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pitch correction is a controversial and commonly used method of music editing, probably better know as **'auto-tune'**. Auto tune is a specific audio processor licensed by **Antares Audio Technologies**. The question of pitch correction being a form of musical cheating has been a great debate by many professionals and layman alike. Pitch correction can be used to make very subtle changes to a singer's pitch, but it has also started a new style of music, popularized by Cher with her song "Beliver" in 1998. It seemed like a straightforward concept, that is changing the undesired pitch in a musical signal to a desired one.

Auto tune is an important application of **Audio Signal Processing**. Audio Signal Processing can be defined as a field of engineering that focuses on computational methods for altering sound intentionally. Figure 1.1 shows the block diagram of a generalised Audio Signal Processing System which takes audio signal as input and generates another audio signal using human or automatic controls.

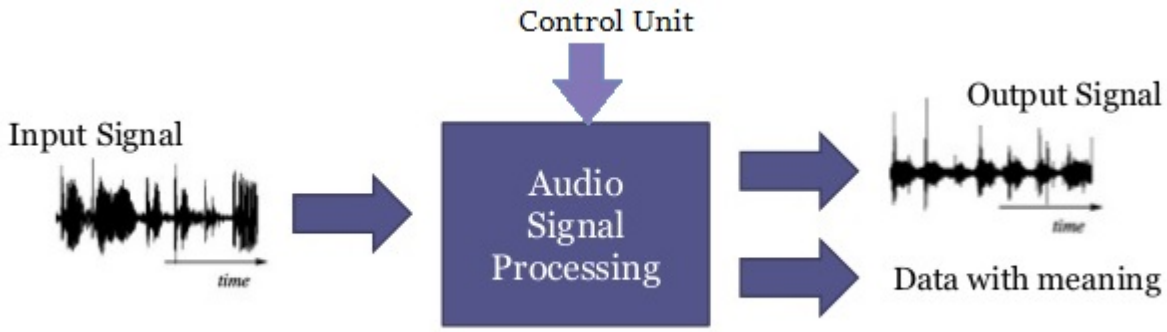


Figure 1.1: Block Diagram Of Audio Signal Processing System

Since audio signals can be represented electronically in analog or digital formats, signal processing occurs in either domain. An analog sound is usually electrical. It is a voltage level that represents the air pressure variation of the sound. It is a continuously varying function. A digital method of representation uses the pressure waveform as binary numbers, thus, as a discrete function.

Figure 1.2 shows continuous and discrete sine waves. The digital representation is used in microprocessors and computers. Although there is a loss in conversion from analog to digital, many modern audio systems use this method because digital signal processing is a much more powerful and efficient technique and can be easily implemented in a computer.

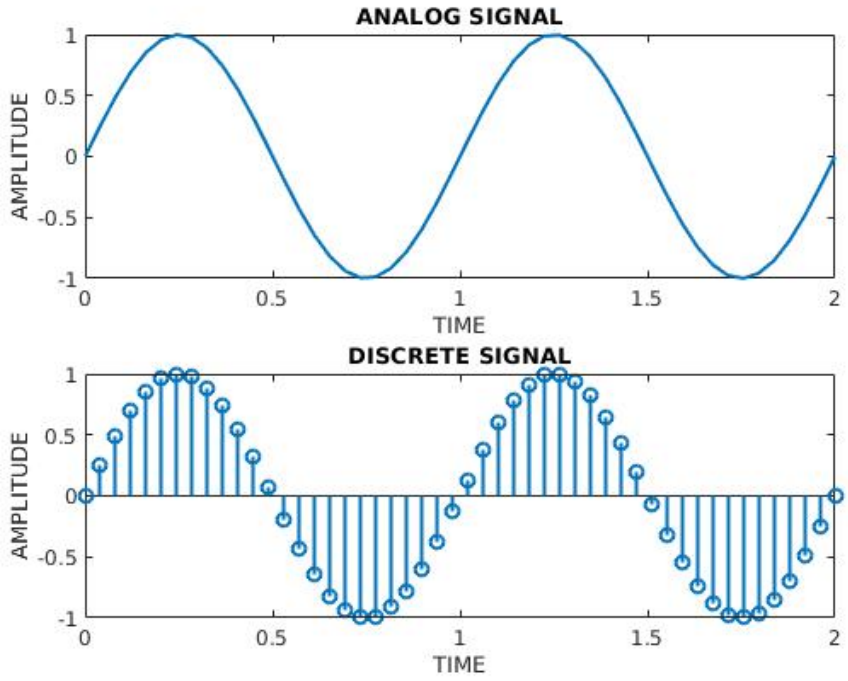


Figure 1.2: Analog v/s Discrete

Applications of Audio Signal Processing

1. **Storage**: In storage of audio signals, the two main steps are recording and recreation. Audio signals such as spoken speech, sounds or music can be recorded as electrical or mechanical inscriptions in a media. At the end it is a simple task of recreating audio signals from these inscriptions.

2. **Data compression**: Data compression or **Audio coding** is a method to reduce the bandwidth and storage sizes of audio signals. When high compression is required a method known as lossy compression is used. In this method, some information is lost, but to a listener there is no perception of loss. The other method of compression is lossless compression, in which there is no loss of information.

3. **Sound Transformation**: Sound Transformation is the application in which the audio signal is subjected to effects such as 3D audio, echo, equalizer, flagne, phaser, chorus, time stretching, morphing, pitch shift, and many more. In our project, we concentrate on this application as our intention is to correct and shift pitch.

4. **Sound description**: Modelling meaningful characteristics of an audio signal is a field that has grown tremendously over the past few years. There are several descriptors in audio signals. Some of the low-level descriptors or labels such as loudness, timbre, pitch are very useful in music information retrieval. Some of the mid-level descriptors or labels that are musically more meaningful include concepts like rhythm, harmony, or melody. Some of the high-level descriptors, such as genre, emotions, similarity are the things that are common to human perception of music.

5. **Sound synthesis**: Signal processing plays a vital role in sound synthesis. In this application, the audio signal can be generated by using simple techniques such as additive syntheses of sine waves as shown in Figure 1.3. This figure shows generation of sound by taking the summation of the outputs of oscillators generating sinusoids. Other techniques include physical modelling, wave shaping, modelling using differential equations, and many more. In this project, we are targeting some of these methods.

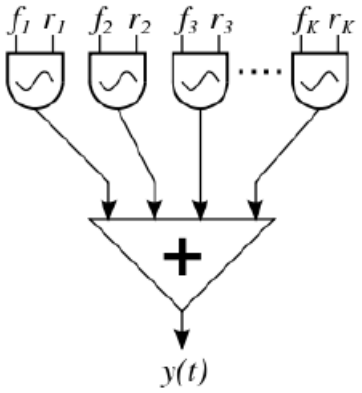


Figure 1.3: Additive Synthesis of Complex Sound

Chapter 2

Literature Survey

As discussed in Chapter 1, Speech and audio signal processing are two major areas of audio signal processing. The main areas of research in music signal processing are music synthesis, instrument and voice identification, detection and correction of fundamental frequency. This project discusses about the pitch detection and correction algorithms. The majority of the contents have been largely influenced by an online course on coursera [13]. This course starts from fourier transform, STFT, basics of music, harmonic model, finding the fundamental frequency, pitch correction and synthesising audio.

2.1 Basics of Signal Processing

The need of Fourier transform, mathematical modelling to convert a time varying signal to its frequency domain were discussed by Oppenheim, A. V., Schafer, R. W. and Buck, J. R. [1]. In STFT the need for windowing/segmenting the audio signal and the significance of window function, drawbacks of various window functions were studied to implement the project. The inverse fourier transform a tool which was used to move back to time domain from frequency domain was explored from this book. Definition of pitch and pitch detection algorithms in time domain like auto correlation function, square difference function was explored from Rabiner.L.R and R.W.Schafer [2].

2.2 Pitch Detection Algorithms

2.2.1 Autocorrelation

The algorithm to find pitch using autocorrelation function is discussed in Rabiner.L.R.[3]. In this method the music samples are broken down into short intervals using windowing technique and the signal in each window is convolved with the time reversal of the signal samples in that window to obtain the peak. The distance between the first and second peak gives the fundamental time period and the inverse of that will give the fundamental frequency or pitch.

2.2.2 Average Magnitude Difference Function

The algorithm for computing pitch using Average Magnitude Difference Function was explored from Rabiner.L.R and R.W.Schafer [2]. In this method the music samples are broken down into short intervals using windowing technique and the signal in each window is subtracted with the circular shifted version of windowed signal, to avoid cancellation absolute value is taken. In this the location the main aim is to find the minima index and when this minima index is divided by the sampling rate to get fundamental time period. Inverse of fundamental time result in Pitch or fundamental frequency.

2.2.3 Yin Algorithm

The other method for pitch detection is Yin algorithm. This algorithm is discussed in De Cheveign, A., Kawahara, H.[4]. The signal is broken into short intervals using the window technique, the difference between the signal of that window and the signal obtained by circular shift is computed and the absolute value is squared and the minimum of that is computed. In this the location of local minima is found and divided by the sampling rate to get fundamental time period. Inverse of fundamental time result in Pitch or fundamental frequency.

2.2.4 Cepstrum

Pitch detection using cepstrum is discussed in Noll, A. M. [5]. In this method the audio samples are broken down using windows, the short time fourier transform of the windowed signal is computed and logarithm of the magnitude values of the STFT is taken and finally the inverse fourier transform is computed, this results in obtaining the cepstrum of the signal, To obtain the pitch from the cepstrum, a peak is located in the quefrency region corresponding to speech fundamental frequencies.

2.3 Chord Detection

Oudre, Laurent & Grenier, Yves & Févotte, Cédric. (2009) [6] and Oudre, Laurent, Yves Grenier and Cédric Févotte. [7] speaks about the template matching method used for chord detection. Machine learning and neural network methods for computation of chord was explored from J. Osmalskyj, J-J. Embrechts, S. Piérard, M. Van Droogenbroeck [8]. Chord is polyphonic audio and normal pitch detection algorithm cannot be used hence a different approch was explored using these 3 papers and implemented the same.

2.4 Scale Detection

Scale is a sequence of notes and detection of such scales was done using the Krumhansl-Schmuckler algorithm which is described in David Temperly [9]. This algorithm involves correlation between the input vector and the key profile and the one with maximum correlated value is the scale present in the audio.

2.5 Pitch Shifting Algorithms

2.5.1 Phase Vocoder

Phase vocoder is a widely used method for pitch shifting. The concept of phase vocoder was proposed by Flanagan J.L. and Golden, R. M. [10]. In this method, the audio signal is broken down into windows and using the phase information, the signal is reconstructed for a different frequency, without changing the duration of the signal. This brings about a change in pitch.

2.5.2 Harmonic Transformation

An important model for analysing musical signals is the harmonic and stochastic model of audio signals. This model is proposed by Serra, X., [12] in which a signal is decomposed into harmonics and stochastic processes, which vary with time. Shifting these harmonics by the incremental value so as to move to a higher or lower pitch was done and preservation of timbre and formant structure was achieved. The method to transform and resynthesize music is described in the paper by Serra, X., [11].

2.6 Sound Models

An important part of analysing music is to model these sound signals. The method to model, analyze, transform and resynthesize music is described in the paper by Serra, X., [11]. An important model for analysing musical signals is the harmonic and stochastic model of audio signals. This model is proposed by Serra, X., [12] in which a signal is decomposed into harmonics and the errors by stochastic processes, which vary with time.

Chapter 3

Music

3.1 Introduction

Music is an art form and cultural activity whose medium is sound organized in time. Music and Noise both are time varying quantity, but music is a periodic time varying quantity where as noise is a random signal and shows no periodicity. There are two types of signal being periodic, one being ideal periodic and another being quasi periodic. Artifically synthesised audio is Ideal periodic because they repeat after a given interval of time called the **Fundamental Time period** and its inverse is called **Fundamental Frequency** or **Pitch**. So Frequency can be defined as the number of times a signal repeats itself in unit time. On the other hand quasi periodic signal is more or less periodic but not perfectly periodic over a given time interval. Natural Music are always Quasi Periodic. Figure 3.1 is a plot which differentiates Music from Noise, Periodicity is the key for Music.

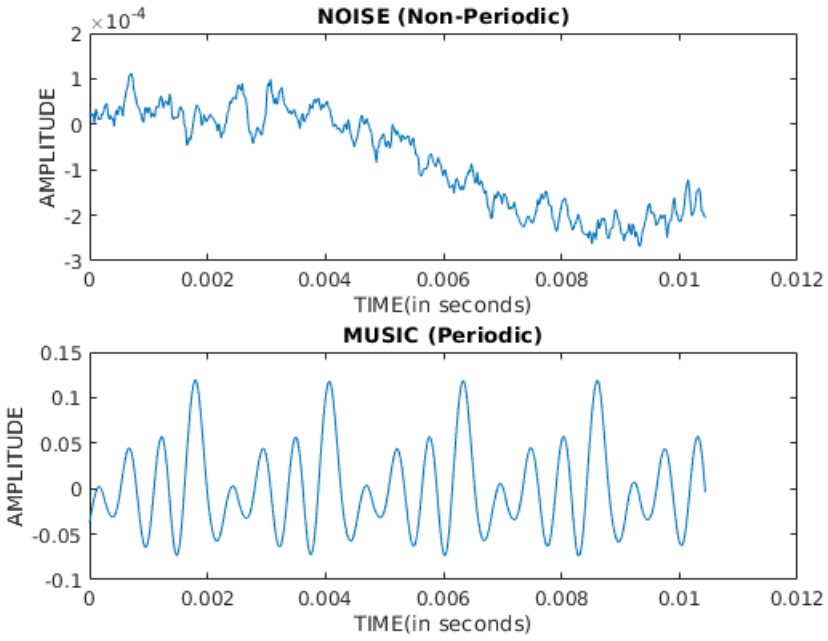


Figure 3.1: Comparision of Noise and Music

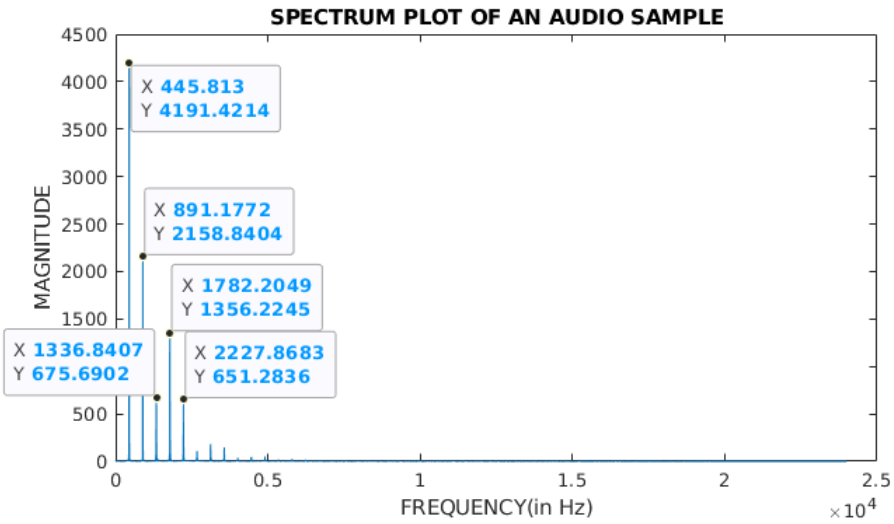


Figure 3.2: Spectrum Plot of an audio whose fundamental frequency is 445Hz

The common terminologies associated with music pitch (which governs melody and harmony), rhythm, tempo, meter, dynamics (loudness and softness), and the sonic qualities of timbre and texture.

3.2 Harmonic Model

As stated earlier in the chapter, if a audio is periodic or quasi periodic there exists a single fundamental frequency called the pitch if it is monophonic audio. Then there exists multiple frequencies which is integral multiple of fundamental frequency. Some instruments may have odd harmonics few may have even harmonics and some instruments can have all the harmonics. Harmoic Model of a music is given according to Equation 3.1, where A[n] is instantaneous amplitude, F_0 is fundamental frequency, N is number of harmonic components.

$$y[n] = \sum_{m=1}^{N} A[n] \times cos(2\pi m F_0[n] n) \tag{3.1}$$

Figure 3.2 shows the harmonic model of an audio whose fundamental frequency is 445Hz. From the spectrum plot we can see peaks at 445Hz(F_0), 891Hz($2F_0$), 1336Hz($3F_0$), 1782Hz($4F_0$) and so on. It is clear that for music it has harmonics and structure of Harmonics is an identify of every instrument.

Octave: If the harmonic frequency is a multiple of 2 powers, then those frequencies are said to be in octaves. From one octave to another frequency doubles. A_4 has a frequency of 440 Hz whereas A_5 note has a frequency of 880 Hz and similarly A_3 has a frequency of 220 Hz. From Equation 3.2, we can see that for frequencies to be in octave, n must be a multiple of 12. So a frequency must step up 12 steps to reach its next octave.

3.3 Description Elements of Music

Pitch or "Musical Notes"

Music can be composed of notes at any arbitrary frequency, which is explained in previous section. In Western music, only twelve notes of fixed frequencies are used. These fixed frequencies are related to each other, and these are defined around the central note, A_4. The current **"standard pitch"** or **"modern concert pitch"** for A_4. note is 440 Hz, although this might vary in actual practice. Let the distance between next note to be calculated be denoted n. If the note is above A_4, then n is positive; if it is below A_4, then n is negative. The frequency of the note (f) (assuming equal temperament) is found according to Equation 3.2.

$$f = 2^{\frac{n}{12}} \times 440Hz \tag{3.2}$$

For example, the frequency of C_5, since C_5 is above A_4 'n' is positive. There are 3 steps between A_4 and C_5, i.e. $A_4 \longrightarrow B_4^b \longrightarrow B_4 \longrightarrow C_5$, so n = +3. The note's frequency is around 523.2 Hz. Now if one want to find the frequency of F_4, since F_4 is below A_4 'n' is negative. There are 4 steps between F_4 and A_4, i.e. $F_4 \longrightarrow G_4^b \longrightarrow G_4 \longrightarrow A_4^b \longrightarrow A_4$, so n = -4. The note's frequency is around 349.23 Hz. Table 3.1 shows the frequency of various musical notes calculated in the similar fashion mentioned above.

Nomenclature of musical notes

A, B, C, D, E, F and **G** are used to name main musical notes within an octave. There are 5 more frequencies/notes which lie between the main musical notes usually named along with main musical notes as **sharps** or **flat**. $A^{\#}$ (pronounced as **A sharp**) or B^b (pronounced as **B flat**) lies between A and B main musical note. Similarly $C^{\#}$ or D^b lies between C and D, $D^{\#}$ or E^b lies between D and E, $F^{\#}$ or G^b lies between F and G, $G^{\#}$ or A^b lies between G and A notes. Subscript number denotes the **octave** number.

Melody

A melody is also known as **tune, voice,** or **line**. It is a linear succession of musical tones that the listener perceives as a single entity. In its most literal sense, a melody is a combination of pitch and rhythm, while more figuratively, the term can include successions of other musical elements such as tonal color. It may be considered the foreground to the background accompaniment. A line or part need not be a foreground melody.

Melodies often consist of one or more musical phrases, and are usually repeated throughout a composition in various forms.

Harmony

Harmony is the process by which the composition of individual sounds, or superpositions of sounds, is analysed by hearing. Usually, this means simultaneously occurring frequencies, pitches (tones, notes), or chords. The study of harmony involves chords, construction of chords and chord progressions and the principles of connection that govern them.

NOTE	Hz	NOTE	Hz	NOTE	Hz	NOTE	Hz
C_0	16.35	C_2	65.41	C_4	261.63	C_6	1046.50
$C_0^\#/D_0^b$	17.32	$C_2^\#/D_2^b$	69.30	$C_4^\#/D_4^b$	277.18	$C_6^\#/D_6^b$	1108.73
D_0	18.35	D_2	73.42	D_4	293.66	D_6	1174.67
$D_0^\#/E_0^b$	19.45	$D_2^\#/E_2^b$	77.78	$D_4^\#/E_4^b$	311.13	$D_6^\#/E_6^b$	1244.51
E_0	20.60	E_2	82.41	E_4	329.63	E_6	1318.51
F_0	21.83	F_2	87.31	F_4	349.23	F_6	1396.91
$F_0^\#/G_0^b$	23.12	$F_2^\#/G_2^b$	92.50	$F_4^\#/G_4^b$	370.00	$F_6^\#/G_6^b$	1479.98
G_0	24.50	G_2	98.00	G_4	392.00	G_6	1567.98
$G_0^\#/A_0^b$	25.96	$G_2^\#/A_2^b$	103.83	$G_4^\#/A_4^b$	415.30	$G_6^\#/A_6^b$	1661.62
A_0	27.50	A_2	110.00	A_4	440.00	A_6	1760.00
$A_0^\#/B_0^b$	29.14	$A_2^\#/B_2^b$	116.54	$A_4^\#/B_4^b$	466.16	$A_6^\#/B_6^b$	1864.44
B_0	30.87	B_2	123.47	B_4	493.88	B_6	1975.53
C_1	32.70	C_3	130.81	C_5	523.25	C_7	2093.00
$C_1^\#/D_1^b$	34.65	$C_3^\#/D_3^b$	138.59	$C_5^\#/D_5^b$	554.37	$C_7^\#/D_7^b$	2217.63
D_1	36.71	D_3	146.83	D_5	587.33	D_7	2349.32
$D_1^\#/E_1^b$	38.89	$D_3^\#/E_3^b$	155.56	$D_5^\#/E_5^b$	622.25	$D_7^\#/E_7^b$	2489.02
E_1	41.20	E_3	164.81	E_5	659.25	E_7	2637.02
F_1	43.65	F_3	174.61	F_5	698.46	F_7	2793.83
$F_1^\#/G_1^b$	46.25	$F_3^\#/G_3^b$	185.00	$F_5^\#/G_5^b$	740.00	$F_7^\#/G_7^b$	2959.96
G_1	49.00	G_3	196.00	G_5	784.00	G_7	3135.96
$G_1^\#/A_1^b$	51.91	$G_3^\#/A_3^b$	207.65	$G_5^\#/A_5^b$	830.61	$G_7^\#/A_7^b$	3322.44
A_1	55.00	A_3	220.00	A_5	880.00	A_7	3520.00
$A_1^\#/B_1^b$	58.27	$A_3^\#/B_3^b$	233.08	$A_5^\#/B_5^b$	932.32	$A_7^\#/B_7^b$	3729.31
B_1	61.74	B_3	246.94	B_5	987.77	B_7	3951.07

Table 3.1: Frequency of various Musical notes.

Descriptions and definitions of harmony and harmonic practice often show bias towards European (or Western) musical traditions, although many cultures practice vertical harmony.

Timbre or "Tone Color"

In music, timbres is the perceived sound quality of a **musical note, sound** or **tone**. Timbre distinguishes different types of production of sound, such as musical instruments and choir voices, such as wind, percussion,string instruments. It enables listeners to distinguish different instruments in the same category. For example, an oboe and a flute have different timbre even though both are wind instruments.

Figure 3.2 shows the timbre nature of 2 wind instruments oboe and flute. The harmonic structures are different and hence timbre is an identity of an instrument. Even if different players play the same instrument play and same note, their notes might sound

different due to differences in instrumental technique, different types of accessories or strings made out of different materials for string players. Timbre is also known as **tone color** or **tone quality**.

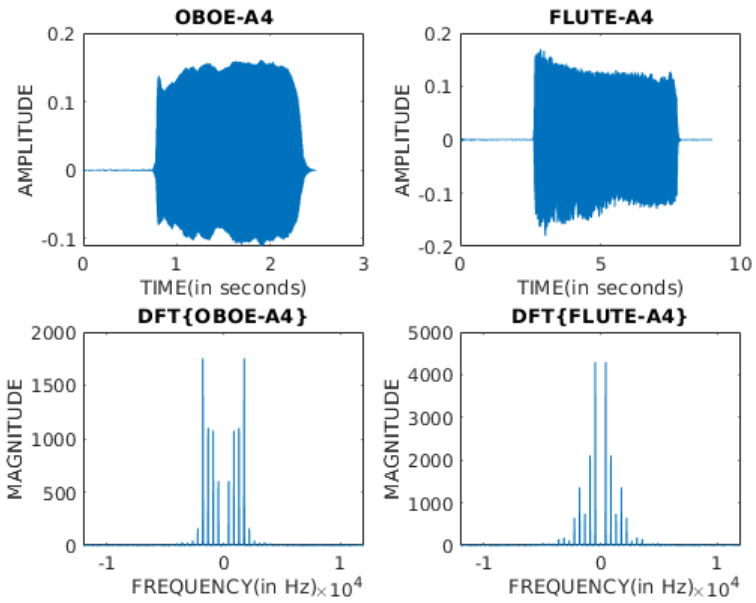


Figure 3.3: Spectrum Plot of Oboe and Flute playing A4 note

The physical characteristics of sound that determine the perception of timbre include spectrum and envelope. Singers and instrumental musicians can change the timbre of the music they are singing/playing by using different singing or playing techniques. For example, a violinist can use different bowing styles or play on different parts of the string to obtain different timbres. In simple terms we can tell that Timbre is the identity of the instrument as it maps uniquely for each instruments.

Tempo

Tempo in music mean the speed or pace at which a given piece of audio is played or sung. In classical music, tempo is indicated with an instruction at the start of a piece and is measured in beats per minute (bpm). For example, a tempo of 60 beats per minute signifies one beat per second, while a tempo of 120 beats per minute is twice as rapid, signifying two beat every second and tempo of 30 beats per minute is half the speed compared to 60 bpm having one beat every two seconds. BPM is the most precise way of indicating fast tempo or slow tempo. It's used in applications where musical durations must be completely precise, such as film scoring. It's also used to set metronomes that are used on the highest level professional recordings. In fact, some people use the term "metronome marking" to describe beats per minute.

Basic tempo markings

⋆ **Larghissimo**: Larghissimo has a very low tempo with tempo less than 20 bpm.

⋆ **Grave**: Grave has slow and solemn tempo of 20 to 40 bpm.

⋆ **Lento**: Lento has a tempo range of 40 to 60 bmp.

⋆ **Largo**: Largo has a range of tempo from 40 to 60 bpm.

⋆ **Larghetto**: Larghetto hastempo ranging from 60 to 66 bpm.

⋆ **Adagio**: Adagio has tempo in the range 66 to 76 bpm.

⋆ **Adagietto**: Adagietto has a tempo range of 70 to 80 bpm.

⋆ **Andante moderato**: Andante moderato has a bit slower than andante.

⋆ **Andante**: Andante has tempo ranging from 76 to 108 bpm.

⋆ **Andantino**: Andantino has slightly faster than andante.

⋆ **Moderato**: Moderato has a moderate tempo of 108 to 120 bpm.

⋆ **Allegro moderato**: Allegro moderato has moderate tempo measuring around 112 to 124 bpm.

⋆ **Allegro**: Allegro has a fast tempo of 120 to 168 bpm.

⋆ **Vivace**: Vivace has approximately 140 bpm tempo.

⋆ **Presto**: Presto has a very fast tempo of 168 to 200 bpm.

⋆ **Prestissimo**: Prestissimo is extremely fast with tempo starting from 200 bpm and above.

Texture

In music, texture is how the harmonic, tempo and melodic materials are combined in a composition, thus examining the overall quality of the sound sample. Texture is often described in regard to the density, or thickness, and range, or width, between lowest and highest pitches, in relative terms as well as more specifically distinguished according to the number of voices, or parts, and the relationship between these voices. **Types of Texture**

1. **Monophonic**:Monophony is the basic musical textures, consisting of a melody , typically sung by a single singer or played by a single instrument player without accompanying harmony or chords. Many folk songs and traditional songs are monophonic.

2. **Biphonic**:Two distinct pitches, the lower sustaining a drone (constant pitch) while the other pitch creates a more elaborate melody above it. Pedal tones or ostinati would be an example for biphonic.

3. **Polyphonic**:Polyphony is a type of musical texture consisting of two or more simultaneous pitches/notes of independent melody, as opposed to a musical texture with just one voice, monophony, or a texture with one dominant melodic voice accompanied by chords, homophony. Baroque forms such as fugue, which might be called polyphonic.

4. **Homophonic**:Homophony is a texture in which a primary part is supported by one or more additional strands that flesh out the harmony and often provide rhythmic contrast.

Rhythm

Rhythm, in music, the placement of sounds in time. In its most general sense, rhythm is an ordered alternation of contrasting elements. Repetitions of elements and intervals between these repetitions can create a sence of rhythm. Musicians create rhythm in the spacing between notes, effectively making these "silent" gaps play off the notes. Designers insert spacing between elements to make rhythm. There are 3 types of rhythm.

- ⋆ **Random Rhythm**:Repeating elements without any specific regular interval creates random rhythms. The spacing could be a millimeter at one instance, a centimeter at another instance, while the elements could be all over the place. It's also worth noting that a rhythm may appear random if you examine a small section of the rhythm. However, if you step back and examine a larger section, it may be that there is a regular but complex rhythm applied to the design.

- ⋆ **Regular Rhythm**:Regular Rhythm is like beating of heart, it repeats over and over again over the same intervals of time. Regular Rhythm can be made by creating a grid or series of notes. There is a risk that when you're using a regular rhythm in a design that it can become monotonous.

- ⋆ **Alternating Rhythm**:Repetitions of more than one element in a design in an alternating pattern like 1-2-1-2-1-2 (where 1 and 2 are two distinct repetition elements). An alternating rhythm is, in fact, a regular rhythm with more complexity. It can be simple or complex as we want to make an alternating rhythm, it can be an easy way to break up the monotony of a regular rhythm.

3.4 Musical Scales and Chords

3.4.1 Scales

Scale is any set of musical notes ordered by fundamental frequency or pitch. A scale placed in increasing pitch is an ascending scale, and a scale placed in decreasing pitch is a descending scale. Some scales contain different pitches when ascending than when descending. Often, in common practice period, most or all of the melody and harmony of a musical work is built using the notes of a single scale, which can be conveniently represented on a staff with a standard key signature. Because of the principle of octaves, scales are generally considered to span within the same octave, with lower or higher octaves simply repeating pattern. Furthur details about types, duration are provided in Chapter 5.

3.4.2 Chords

Chord is any harmonic set of pitches or musical notes that are sound as if they were played simultaneously. A sequence of chords is known as a **chord progression** or **harmonic progression**. These are frequently used in Western music. One example of a widely used chord progression in Western traditional music and blues is the 12 bar blues progression.

Usually one source can produce one pitch, when there are more than one source at a given time for production of sound chords are generated. When 2 or more strings are plucked in string instruments like guitar, cello or pressing 2 or more keys simultaneously in piano or keyboard produces chord. Wind instruments can have only one source so producing chord from wind instruments is not possible. In singing pitch are produced by vibration of vocal chords and since we have only one source for production of sound chords producced by vocal is not possible. Furthur details about types, detection are provided in Chapter 5.

Both scales and chords contain multiple pitches/musical notes but the difference is evident. In scales multiple pitches are played one after the other forming a chain of sequence where as Chord has multiple pitches played together.

Chapter 4

Pitch Detection

4.1 Introduction

Speech is produced by vibration of vocal chords when air moves from lungs (which being the power house of speech) to oral cavity through trachea or windpipe. The rate at which the vocal chords vibrate is known as **Fundamental frequency** or **Pitch** of an individual. The rate per second of a vibration of instruments constituting a wave, in a material such as in sound waves is called as **Frequency** or **Pitch**.

Males generally have **low pitched voices** and frequency ranges from 55Hz to 131Hz (A1 to C3) while females have **high pitched voices** and frequency ranges from 170Hz to 262Hz (F3 to C4). Males tend to use full pitch range during speech while females vary pitch range during speech. Whereas musical instruments have a large range of frequency starting from as less as 16Hz and can go upto as high as 4000Hz (C0 to B7). But frequency above 1000Hz becomes less significant so we limit our area of interest below 1000Hz.

Pitch can be calculated in time domain or frequency domain. Calculation in time domain is direct by using the actual data of the audio whereas pitch detection in frequency domain involves in moving from time space to frequency spaces using operations like Fourier transform.

4.2 Pitch Detection in Time Domain

Time domain algorithms process the data in its raw form as it is usually read from a sound card - a series of uniformly spaced samples representing the movement of a waveform over time. For example 44100 samples per second is a common recording speed. Each input sample x[n], is assumed to be a real number in the range -1 to 1 inclusive, with its value representing the height of the waveform at discrete time instance 'n'. This section summarizes time domain pitch algorithms, like **Autocorrelation** (AUTO), **Square Difference Function** or the **Yin method** (YIN), the **Average Magnitude Difference Function** (AMDF), Simple Feature Based Method.

4.2.1 Modified Autocorrelation Method

One the most popular methods for pitch estimation is **autocorrelation**. It takes an Input function, x[n], and cross-correlates it with itself; that is, each element is multiplied by a shifted version of x[n], and the results summed to get a single autocorrelation value.

$$r[n] = \sum_{m=0}^{N} x[m] \times x[m-n] \tag{4.1}$$

Where n = 1, 2... N/2; N being the frame length. Autocorrelation methods need atleast two pitch periods to detect pitch.This means that in order to detect a fundamental frequency of 40Hz, atleast 50 milliseconds(ms) of the speech signal must be analysed, hence we must move half of the window size. Equation 4.1 represents auto correlation of an input signal x[n]. If a signal is periodic with period p, then the autocorrelation will have maxima at Multiples of p where the function matches itself. The auto correlated function will have global maxima for n=0 (origin) which is the primary peak and auto correlated function will have local maxima at integral multiple of 'p'. So the distance of first local maxima from origin represent the fundamental time period. Inverse of fundamental time period gives the pitch or the fundamental frequency. We eliminate all the peaks below a threshold value so that peak picking becomes easier and amount of error is checking peaks are reduced. The thresholding makes it a difference between auto correlation and modified autocorrelation method. Figure 4.1 is the block diagram to calculate pitch using modified autocorrelation method.

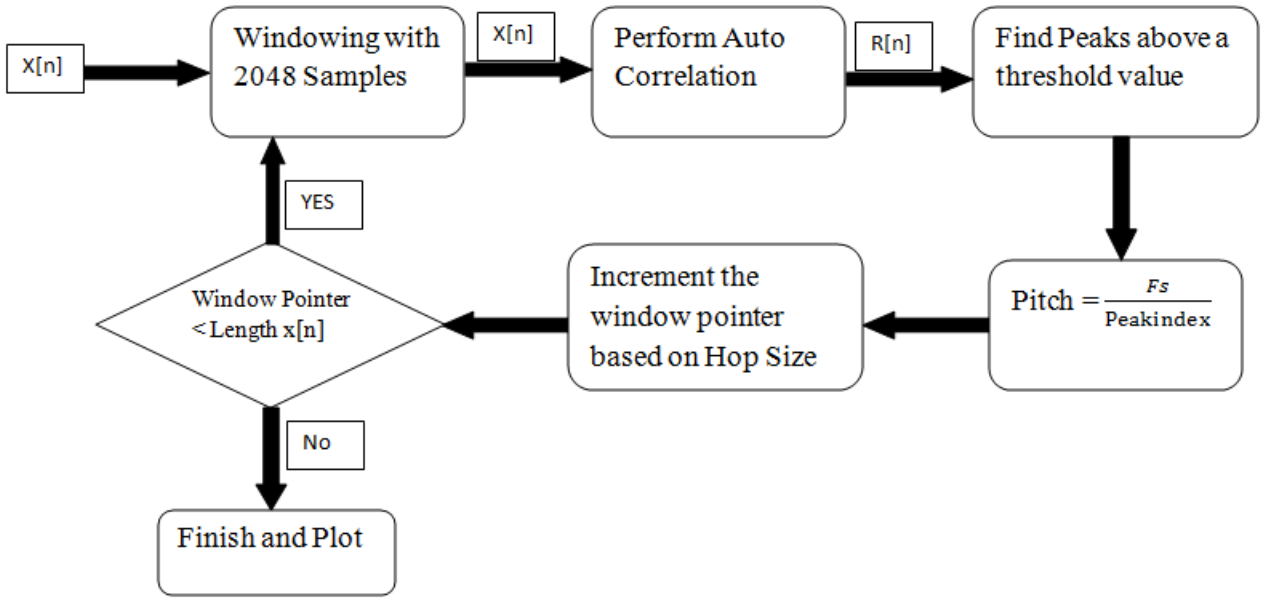


Figure 4.1: Block diagram of Pitch detection using Modified Autocorrelation method

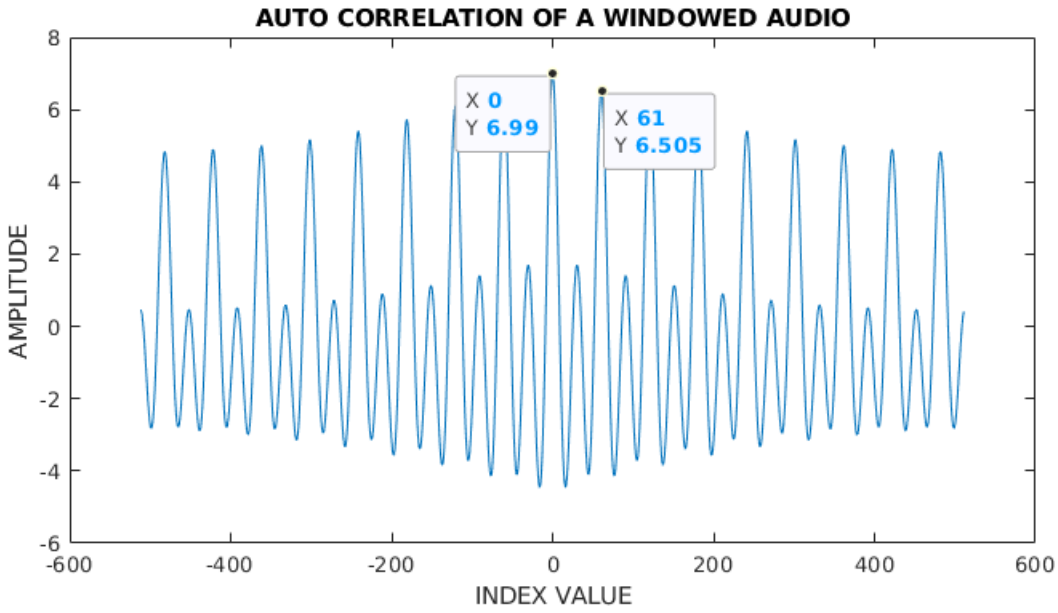


Figure 4.2: Autocorrelation applied on a windowed audio playing G5 note

Steps involved in finding the pitch of an audio sample by modified auto-correlation:

1. The entire audio is broken in smaller segments which is known as windowing. We chose window size as 2048 samples. Minimum frequency or pitch obtained using 2048 samples is approximately 50 Hz. We multiply the segmented audio by a suitable window function.

2. Auto correlation is performed on the windowed audio.

3. We eliminate all the values below a threshold value and now we check for peaks or maximas. The index of maxima is noted.

4. Pitch or fundamental frequency of the windowed audio is obtained by dividing the sampling frequency(Fs) by the maxima index.

5. We increment the window pointer by hop length. If we use 50 percentage overlap we increment the pointer by 1024 samples.

6. We check if the window pointer has reached the end of audio. If the window pointer has not reached the end of the audio we go back to step 1 else coputation of pitch for entire audio is complete and perform desired operation on pitch like plotting etc.

Example illustrating autocorrelation method used for computing pitch

Figure 4.2 shows autocorrelation applied on a windowed audio. Since autocorrelation is an even function it is symmetric about origin and we consider only one half of the correlated signal. Figure shows a global maximum value of 6.99 at origin which is the energy of the signal. First local maxima is present at 61st sample. The sampling frequency(Fs) for the windowed sample is 48000 samples/sec. So the **pitch** or **fundamental frequency**

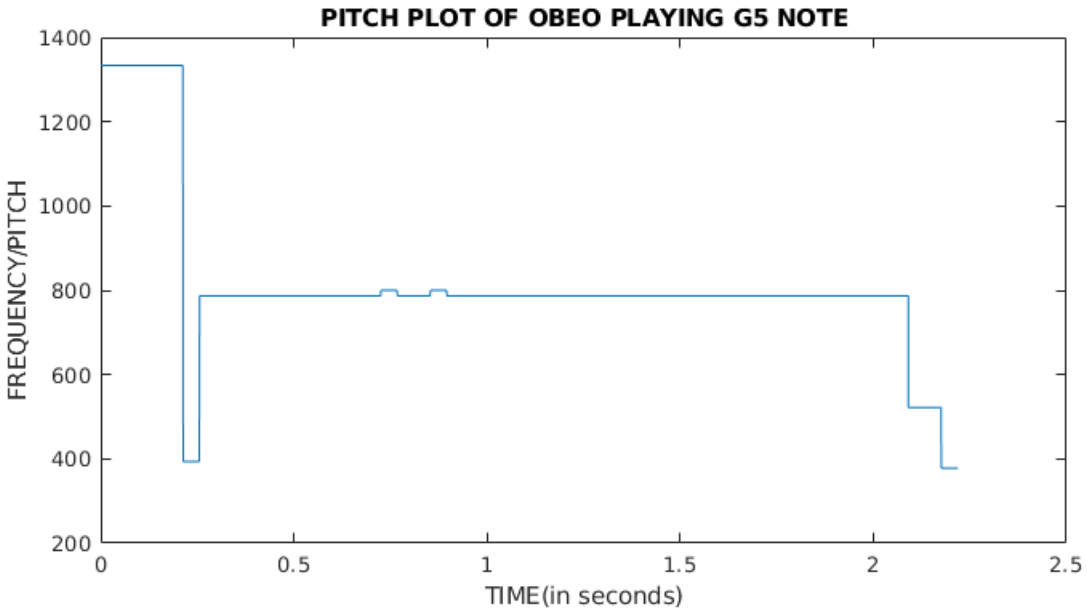


Figure 4.3: Complete pitch plot of Obeo playing G5 note

is given by Fs/maxima index which is $48000/61 \simeq 787$Hz. The entire pitch plot of obeo playing G_5 note is shown in Figure 4.3. G_5 corresponds to 784Hz.

4.2.2 Average Magnitude Difference Function

The idea for using the **Average Magnitude Difference Function** (AMDF) is that if a signal is pseudo-periodic then any two adjacent periods of the waveform are similar in shape. So if the waveform is shifted by one period and compared to its original self, then most of the peaks and troughs will line up well. If one simply takes the differences from one waveform to the other and then sums them up, the result is not useful, as some values are positive and some negative, tending to cancel each other out. This could be dealt with by using the absolute value of the difference and averaging them, as discussed in Equation 4.2.

$$s[n] = \frac{1}{N} \times \sum_{m=0}^{N} |x[n] - x[n-m]| \tag{4.2}$$

Where n = 1, 2...N; N being the frame length. When the waveform is shifted by an amount, τ, that is not the period the differences will become greater, and cause an increased sum. Whereas, when τ equals the period it will tend to a minimum. In this algorithm we find location of local minima. Figure 4.4 is the block diagram to calculate pitch using Average Magnitude Difference Function.

Steps involved in finding the pitch of an audio sample by AMDF:

1. The entire audio is broken in smaller segments which is known as windowing. We chose window size as 2048 samples. Minimum frequency or pitch obtained using 2048 samples is approximately 50 Hz. We multiply the segmented audio by a suitable window function.

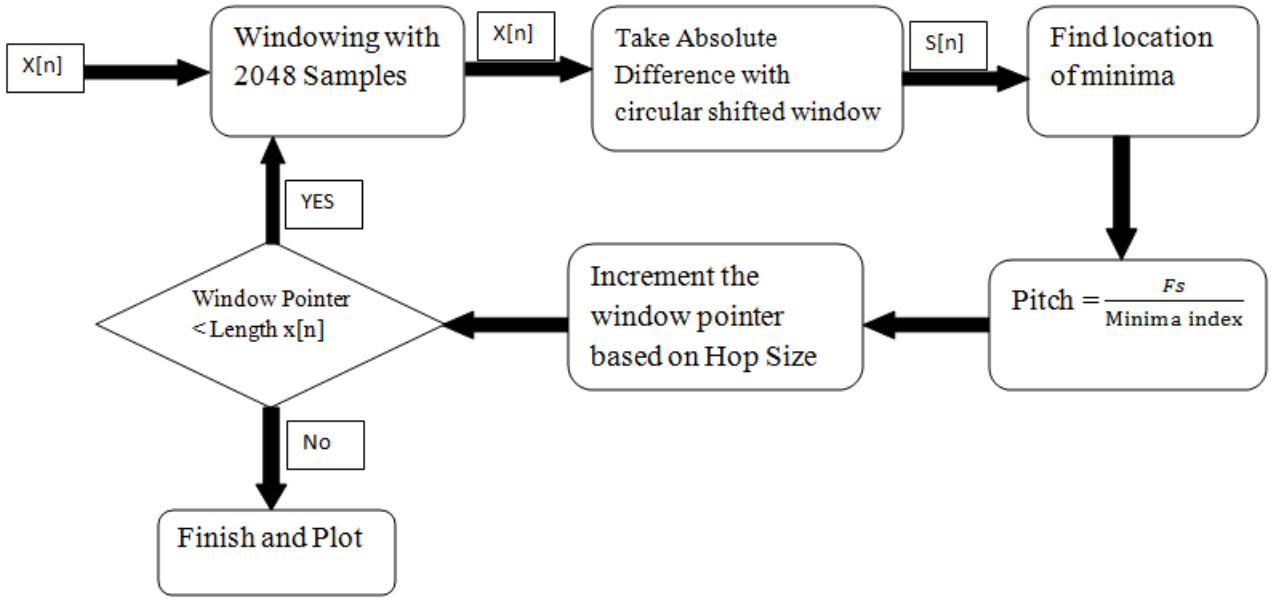


Figure 4.4: Block diagram of Pitch detection using AMDF

2. We take absolute difference of the windowed audio with circular shifted windowed sample.

3. We sum the result obtained in step 2 and sum value is stored.

4. Circular shift windowed audio and go back to step 2. The process of circular shift and taking absolute difference is repeated till we get back the initial windowed signal.

5. We check for the minimum value obtained in step 3 i.e. the minimum value if the sums. The index of minima is noted.

6. Pitch or fundamental frequency of the windowed audio is obtained by dividing the sampling frequency (Fs) by the minima index.

7. We increment the window pointer by hop length. If we use 50 percentage overlap we increment the pointer by 1024 samples.

8. We check if the window pointer has reached the end of audio. If the window pointer has not reached the end of the audio we go back to step 1 else computation of pitch for entire audio is complete and perform desired operation on pitch like plotting etc.

Example illustrating AMDF used for computing pitch

Figure 4.5 shows average magnitude difference function applied on a windowed audio. First local minima is present at 133rd sample. The sampling frequency(Fs) for the windowed sample is 44100 samples/sec. So the **pitch** or **fundamental frequency** is given by Fs/minima index which is $44100/133 \simeq 331.57$Hz. The entire pitch plot of piano playing E_4 note is shown in Figure 4.6. E_4 corresponds to 330Hz.

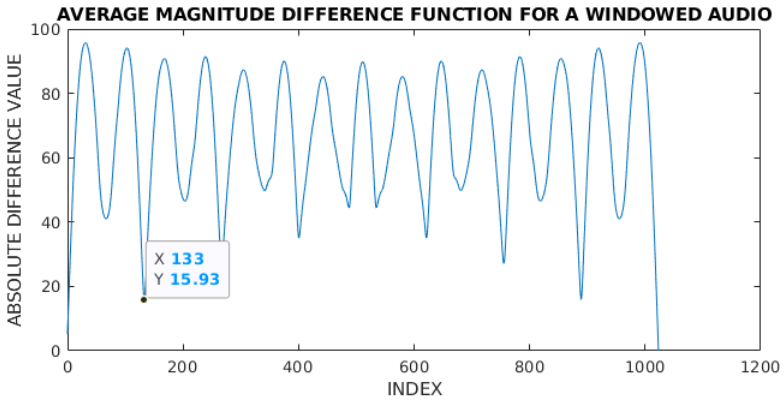


Figure 4.5: AMDF applied on a windowed audio playing E4 note

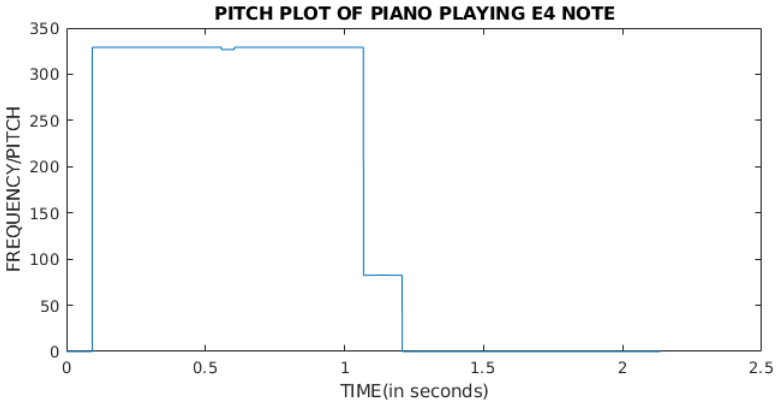


Figure 4.6: Complete pitch plot of Piano playing E4 note

4.2.3 Yin Method

The idea for using the **square difference function** (SDF) or **Yin method** is that if a signal is pseudo-periodic then any two adjacent periods of the waveform are similar in shape. So if the waveform is shifted by one period and compared to its original self, then most of the peaks and troughs will line up well. If one simply takes the differences from one waveform to the other and then sums them up, the result is not useful, as some values are positive and some negative, tending to cancel each other out. This could be dealt with by using the absolute value of the difference, as discussed in Equation 4.3 however it is more common to sum the square of the differences, where each term contributes a non negative amount to the total.

$$s[n] = minimum[\sum_{m=0}^{N}(x[m] - x[m-n])^2] \tag{4.3}$$

When the waveform is shifted by an amount, τ, that is not the period the differences will become greater, and cause an increased sum. Whereas, when τ equals the period it will tend to a minimum. In this algorithm we find location of local minima. Figure 4.7 is the

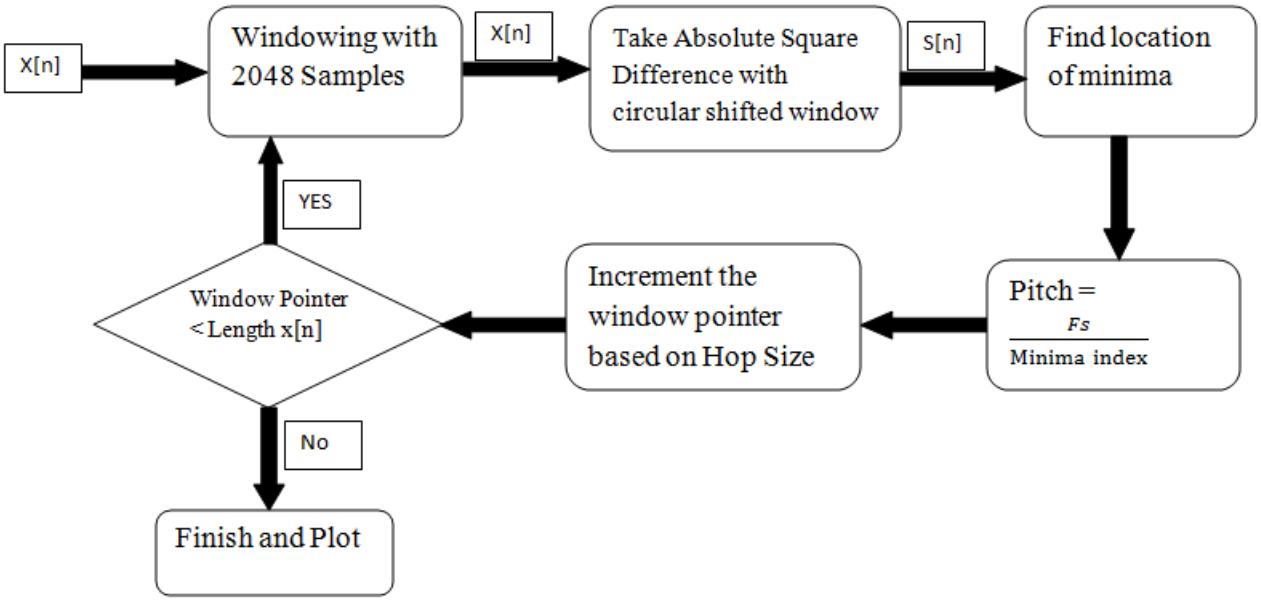


Figure 4.7: Block diagram of Pitch detection using Yin method

block diagram to calculate pitch using Yin method.

Steps involved in finding the pitch of an audio sample by Yin method:

1. The entire audio is broken in smaller segments which is known as windowing. We chose window size as 2048 samples. Minimum frequency or pitch obtained using 2048 samples is approximately 50 Hz. We multiply the segmented audio by a suitable window function.

2. We take absolute difference square of the windowed audio and the circular shifted windowed sample.

3. We sum the result obtained in step 2 and sum value is stored.

4. Circular shift windowed audio and go back to step 2. The process of circular shift and taking absolute difference is repeated till we get back the initial windowed signal.

5. We check for the minimum value obtained in step 3 i.e. the minimum value if the sums. The index of minima is noted.

6. Pitch or fundamental frequency of the windowed audio is obtained by dividing the sampling frequency (Fs) by the minima index.

7. We increment the window pointer by hop length. If we use 50 percentage overlap we increment the pointer by 1024 samples.

8. We check if the window pointer has reached the end of audio. If the window pointer has not reached the end of the audio we go back to step 1 else computation of pitch for entire audio is complete and perform desired operation on pitch like plotting etc.

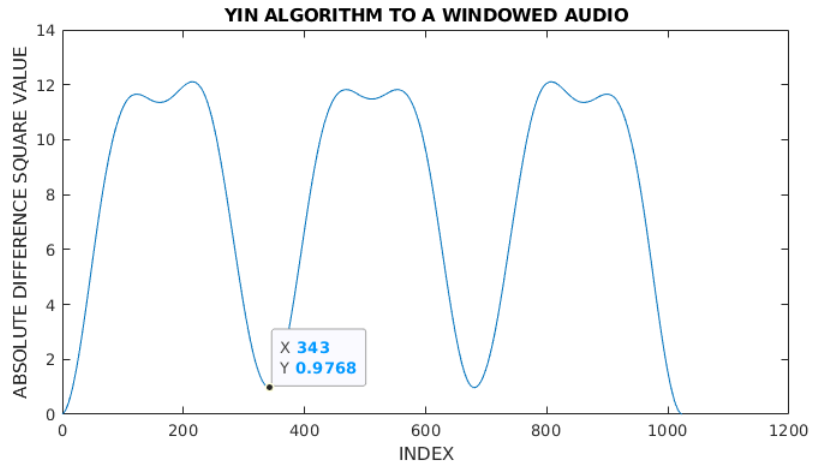


Figure 4.8: YIN applied on a windowed audio playing B2 note

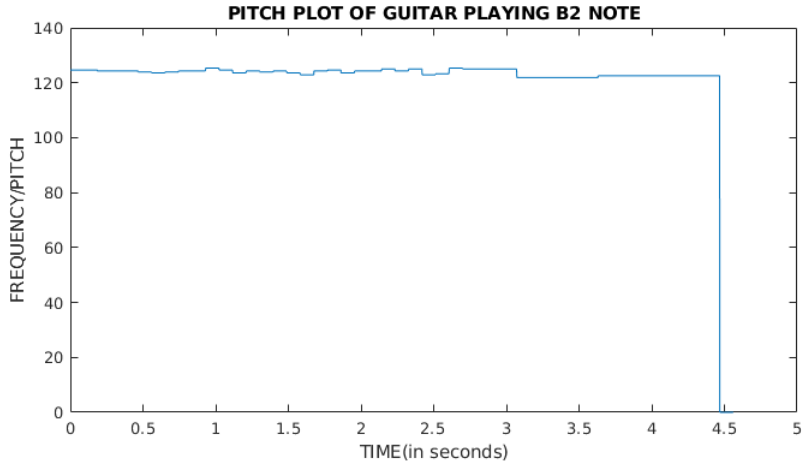


Figure 4.9: Complete pitch plot of Piano playing B2 note

Example illustrating Yin method used for computing pitch

Figure 4.8 shows Yin method applied on a windowed audio. First local minima is present at 343rd sample. The sampling frequency(Fs) for the windowed sample is 44100 samples/sec. So the **pitch** or **fundamental frequency** is given by Fs/minima index which is $44100/343 \simeq 128.57$Hz. The entire pitch plot of piano playing B_2 note is shown in Figure 4.9. B_2 corresponds to 123.47Hz.

4.3 Pitch Detection in Frequency Domain

Frequency domain algorithms do not investigate properties of the raw signal directly, but instead first we pre-process the raw, time domain data, transforming it into the frequency space. This is done using the Fourier transform. Frequency domain pitch algorithms, includes **Harmonic Product Spectrum**, **Sub harmonic-to-Harmonic Ratio**, **Complex Cepstrum** and **Spectrum Peak Method**.

4.3.1 Cepstrum Method

Cepstrum method is used for pitch detection when we have signals which are convolutively related to each other. First we make these convolutively related signals to linearly related. This can be done by the following method. First we move to frequency domain where convolution is converted into multiplication. Later we take logarithm so that they become linearly dependent and finally move back to time domain.

$$X(\omega) = \sum_{n=0}^{N-1} x[n] \times e^{-j\omega n} \tag{4.4}$$

$$x'[n] = \frac{1}{2\pi} \times \int_{0}^{2\pi} log[X(\omega)] \times e^{j\omega n} \tag{4.5}$$

$$log[X(\omega)] = log \mid X(\omega) \mid + jarg(X(\omega)) \tag{4.6}$$

Equations 4.4, 4.5, 4.6 represent Discrete Time Fourier Transform (DTFT), the Inverse Fourier Transform (IDFT) and the complex logarithm respectively. This method is valid if the fundamental frequency is dominant over its harmonics. Figure 4.10 shows the block diagram used for computation of pitch using Cepstrum method

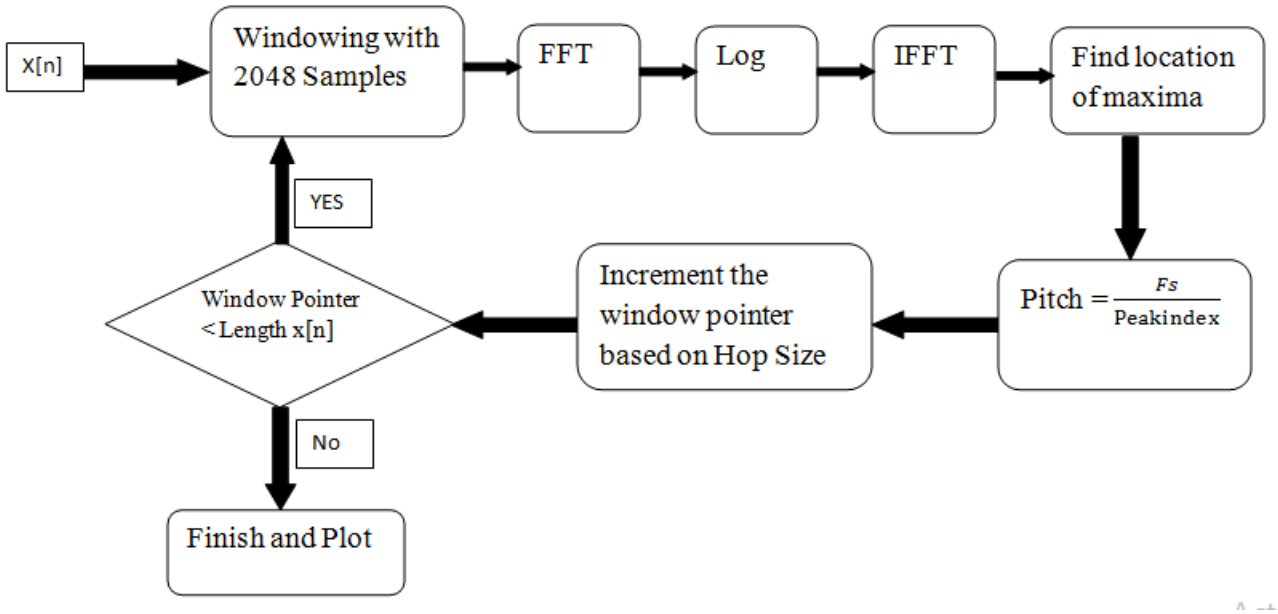


Figure 4.10: Block diagram of Pitch detection using Cepstrum method

Steps involved in finding the pitch of an audio sample by Cepstrum method:

1. The entire audio is broken in smaller segments which is known as windowing. We chose window size as 2048 samples. Minimum frequency or pitch obtained using 2048 samples is approximately 50 Hz. We multiply the segmented audio by a suitable window function.

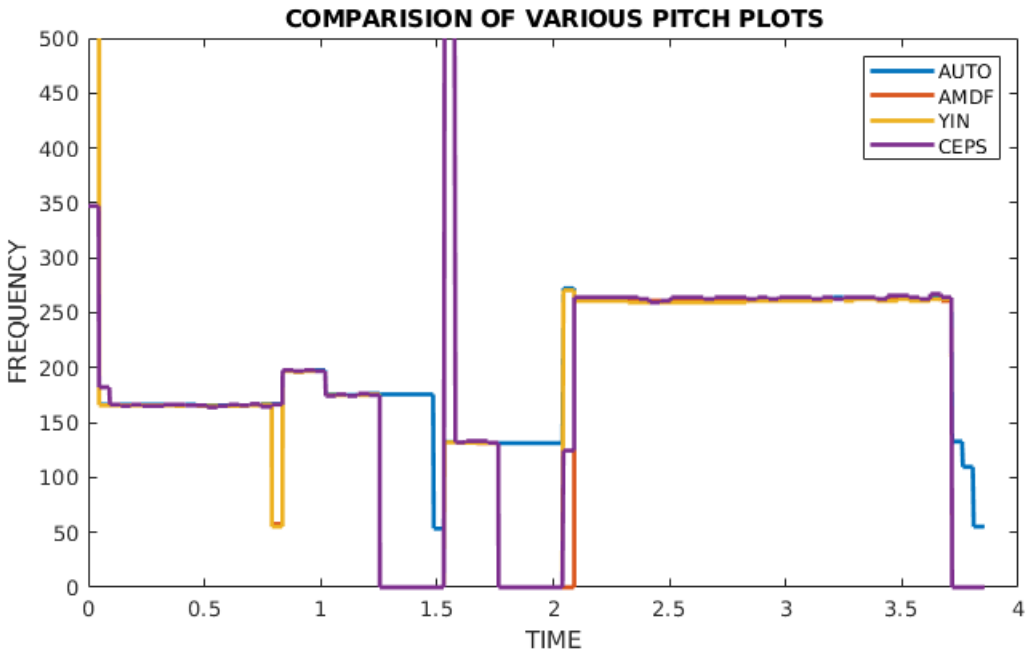


Figure 4.11: Pitch plot of all the pitch detection algorithm for a given audio

2. We then move to frequency domain by applying FFT (Fourier Transform) on the windowed audio.
3. We apply logarithm for the transformed audio.
4. We move back to time domain by taking IFFT (Inverse Fourier Transform).
5. Now we check for peaks or maxima. The index of maxima is noted.
6. Pitch or fundamental frequency of the windowed audio is obtained by dividing the sampling frequency (Fs) by the minima index.
7. We increment the window pointer by hop length. If we use 50 percentage overlap we increment the pointer by 1024 samples.
8. We check if the window pointer has reached the end of audio. If the window pointer has not reached the end of the audio we go back to step 1 else computation of pitch for entire audio is complete and perform desired operation on pitch like plotting etc.

Figure 4.11 shows the various pitch detection algorithm applied on a piano sample. We can see the difference in various algorithm and its closeness to each other.

Chapter 5

Chord and Scale detection

5.1 Chords

A chord is the layering of several tones played simultaneously, usually built on superposed thirds. Chords are defined by their root note and their quality (major, minor, 7 etc) and eventually by their inversion. They have a harmonic set of pitches consisting of **multiple notes** (also called **"pitches"**) that are heard as if sounding simultaneously. Chords and sequences of chords are frequently used in modern West African, Oceanic, Western classical and pop music as well. The most frequently encountered chords are triads, so called because they consist of three distinct notes: the root note, and intervals of a third and a fifth above the root note. Chords with more than three notes include added tone chords, extended chords and tone clusters, which are used in contemporary classical music, jazz and other genres. Table 5.2 lists few important chords in music.

We all know that chord consist of multiple pitches at a given time slot i.e. **Polyphonic audio**. So basic pitch detection algorithms like modified autocorrelation , average magnitude difference function , yin and cepstrum may not work as these are used for audio with single pitch at a given time slot i.e. **monophonic audio**. Hence we must follow a different method for chord detection. In our project we used 2 methods for chord detection. One by making use of **Chromagram** and another using **Machine learning** approach.

5.1.1 Chord recognition using Chromagram

In this approach, we have used a Chromagram estimation method to detect chords. Chromagram is a two dimensional image, showing time or number of frames on the x axis and 12 pitch classes on the y axis. We can easily identify which pitches being the strongest according to the colour on the plot. Figure 5.1 is the block diagram used for chord detection using chromagram.

Chromagram representation

The Chromagram could be constructed as spectrograms, which represents the relationship between time and frequency spectrum. The frequency spectrum is formed by 12-dimensional chroma vectors, which is a set of pitch classes $\{C, D^b/C^\#, D, E^b/D^\#, E, F, G^b/F^\#, G, A^b/G^\#, A, B^b/A^\#, B\}$ and each element of the vector shows the strength of the input. The computation of the Chromagram is to calculate the frequency and amplitudes

CHORD	NOTES / PITCHES	CHORD	NOTES / PITCHES
C Major	C, E, G	C Minor	C, $D^{\#}$, G
$C^{\#}$ Major	$C^{\#}, F, G^{\#}$	$C^{\#}$ Minor	$C^{\#}, E, G^{\#}$
D Major	D, $F^{\#}$, A	D Minor	D, F, A
$D^{\#}$ Major	$D^{\#}, G, A^{\#}$	$D^{\#}$ Minor	$D^{\#}, F^{\#}, A^{\#}$
E Major	E, $G^{\#}$, B	E Minor	E, G, B
F Major	F, A, C	F Minor	F, $G^{\#}$, C
$F^{\#}$ Major	$F^{\#}, A^{\#}, C^{\#}$	$F^{\#}$ Minor	$F^{\#}, A, C^{\#}$
G Major	G, B, D	G Minor	G, $A^{\#}$, D
$G^{\#}$ Major	$G^{\#}, C, D^{\#}$	$G^{\#}$ Minor	$D^{\#}, B, D^{\#}$
A Major	A, $C^{\#}$, E	A Minor	A, C, E
$A^{\#}$ Major	$A^{\#}$, D, F	$A^{\#}$ Minor	$A^{\#}, C^{\#}, F$
B Major	B, $D^{\#}, F^{\#}$	B Minor	B, D, $F^{\#}$
$CMajor^7$	C, E, G, B	$CMinor^7$	C, $D^{\#}, G, A^{\#}$
$C^{\#}Major^7$	$C^{\#}, F, G^{\#}, C$	$C^{\#}Minor^7$	$C^{\#}, E, G^{\#}, B$
$DMajor^7$	D, $F^{\#}$, A, C	$DMinor^7$	D, F, A, C
$D^{\#}Major^7$	$D^{\#}, G, A, C^{\#}$	$D^{\#}Minor^7$	$D^{\#}, F^{\#}, A^{\#}, C^{\#}$
$EMajor^7$	E, $G^{\#}$, B, $D^{\#}$	$EMinor^7$	E, G, B, D
$FMajor^7$	F, A, C, E	$FMinor^7$	F, $G^{\#}, C, D^{\#}$
$F^{\#}Major^7$	$F^{\#}, A^{\#}, C^{\#}, F$	$F^{\#}Minor^7$	$F^{\#}, A, C^{\#}, E$
$GMajor^7$	G, B, D, $F^{\#}$	$GMinor^7$	G, $A^{\#}$, D, F
$G^{\#}Major^7$	$G^{\#}, C, D^{\#}, G$	$G^{\#}Minor^7$	$G^{\#}, B, D^{\#}, F^{\#}$
$AMajor^7$	A, $C^{\#}$, E, $G^{\#}$	$AMinor^7$	A, C, E, G
$A^{\#}Major^7$	$A^{\#}, D, F, A$	$A^{\#}Minor^7$	$A^{\#}, C^{\#}, F, G^{\#}$
$BMajor^7$	B, $D^{\#}, F^{\#}, A^{\#}$	$BMinor^7$	B, D, $F^{\#}$, A

Table 5.1: Various Musical chords and it‘s notes.

of the corresponding note from the spectrogram. There are many ways to calculate the chromagram like Short-time Fourier Transform, Constant Q Transform and Fast Fourier Transform. We have used the Constant Q-Transform in this method, which is explained below.

Constant-Q Transform (CQT)

In mathematics and signal processing, the constant-Q transform transforms a data series to the frequency domain. It is related to the Fourier transform and very closely related to the complex Morlet wavelet transform.

The transform can be thought of as a series of filters f_k, logarithmically spaced in frequency, with the k^{th} filter having a spectral width δf_k equal to a multiple of the

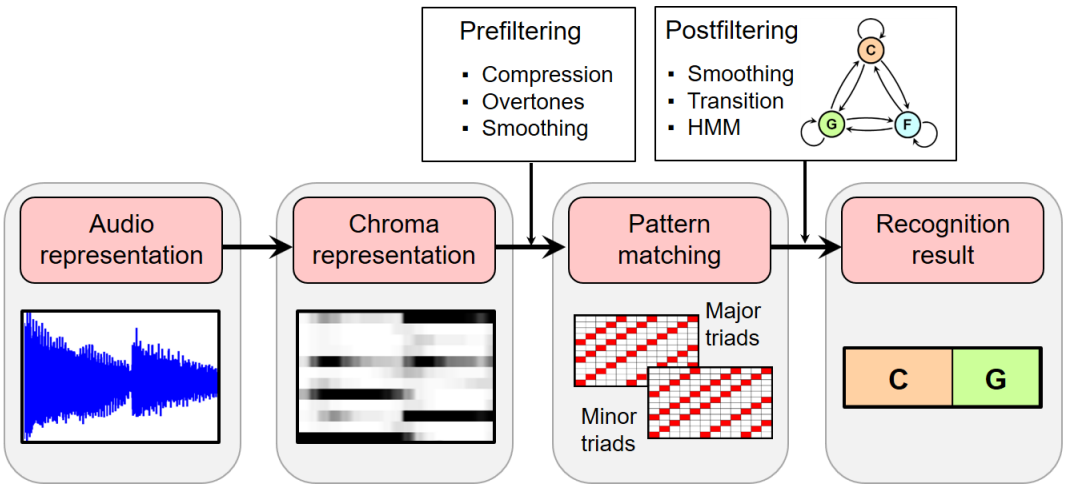


Figure 5.1: Block diagram for chord detection using templates and chromagram

previous filter's width, given below.

$$\begin{aligned} \delta f_k &= 2^{\frac{1}{n}} \times \delta f_{k-1} \\ &= (2^{\frac{1}{n}})^k \times \delta f_{min} \end{aligned}$$

Where δf_k is the bandwidth of the k^{th} filter, f_{min} is the central frequency of the lowest filter, and n is the number of filters per octave.

Pitch Class Profile (PCP)

Pitch class profile (PCP) is a group of features that a computer program extracts from an audio signal, based on a descriptor proposed in the context of a chord recognition system. PCP are an enhanced pitch distribution feature that are sequences of feature vectors describing tonality and measuring the relative intensity of each of the 12 pitch classes of the equal-tempered scale within an analysis frame. Often, the twelve pitch spelling attributes are also referred to as chroma and the PCP features are closely related to what is called chroma features or chromagram.

By processing musical signals, software can identify PCP features and use them to estimate the key of a piece, to measure similarity between two musical pieces (cover version identification), to perform content-based audio retrieval (audio matching), to extract the musical structure (audio structure analysis) and to classify music in terms of composer, genre or mood. The process is related to time-frequency analysis. In general, chroma features are robust to noise (e.g., ambient noise or percussive sounds), independent of timbre and instrumentation and independent of loudness and dynamics.

Steps involved in PCP feature extraction :

1. Input musical signal.

2. Do spectral analysis to obtain the frequency components of the music signal.

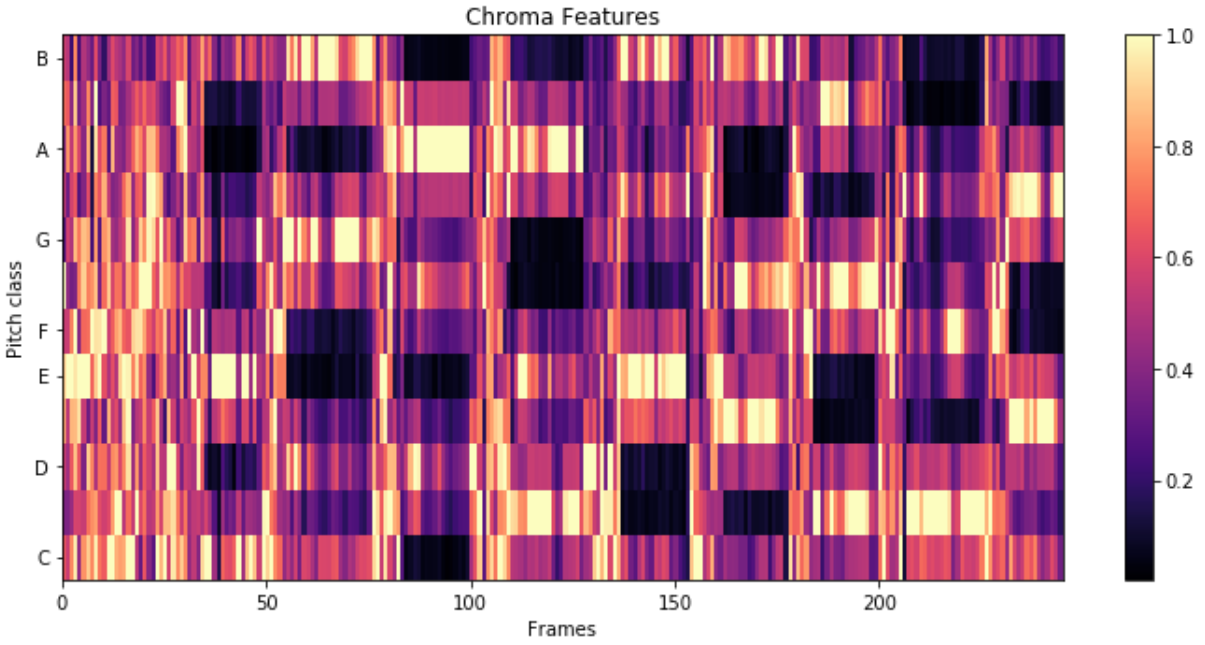


Figure 5.2: Chromagram

3. Use Fourier transform to convert the signal into a spectrogram.
4. Do frequency filtering. A frequency range of between 100 and 5000 Hz is used.
5. Do peak detection. Only the local maximum values of the spectrum are considered.
6. Do reference frequency computation procedure. Estimate the deviation with respect to 440 Hz.
7. Do Pitch class mapping with respect to the estimated reference frequency. This is a procedure for determining the pitch class value from frequency values. A weighting scheme with cosine function is used. It considers the presence of harmonic frequencies (harmonic summation procedure), taking account a total of 8 harmonics for each frequency. To map the value on a one-third of a semitone, the size of the pitch class distribution vectors must be equal to 36.
8. Normalize the feature frame by frame dividing through the maximum value to eliminate dependency on global loudness.
9. The chromagram is then plotted with the frequency or frame on the x-axis and pitch class on the y-axis as shown in Figure 5.2

Chordgram Calculation

The next thing we need to do is to determine the chord probabilities from the chroma vectors we have calculated from the previous step. It is basically a comparison between the typical distributions of chords and the energy computed in the chroma vectors to estimate the correct chord. We will do this analysis frame by frame and compare with a set a chord template.

Chord templates

The set of chords we want to detect is the **12 major chords**, **12 minor chords** and a "N.C" chord, which stands for **"no chord"**. Given the template for C Major and C Minor as follows, we can generate the templates for all 24 chords simply by shifting the numbers in the array. For the special chord, the "no chord", all pitch classes are weighted equally. The template for "N.C" is given as No Chord = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

CHORD	TEMPLATE	CHORD	TEMPLATE
C Major	[1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1]	C Minor	[1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0]
$C^{\#}$ Major	[1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0]	$C^{\#}$ Minor	[0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]
D Major	[0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0]	D Minor	[0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0]
$D^{\#}$ Major	[0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0]	$D^{\#}$ Minor	[0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0]
E Major	[0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1]	E Minor	[0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1]
F Major	[1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0]	F Minor	[1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0]
$F^{\#}$ Major	[0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0]	$F^{\#}$ Minor	[0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0]
G Major	[0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1]	G Minor	[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0]
$G^{\#}$ Major	[1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0]	$G^{\#}$ Minor	[0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1]
A Major	[0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0]	A Major	[1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]
$A^{\#}$ Major	[0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0]	$A^{\#}$ Minor	[0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
B Major	[0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1]	B Minor	[0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1]

Table 5.2: Chord and it's Template

Template comparison

We can now match the chroma vector calculated for the current frame against the template. It is trivial to use the Euclidean Distance as many researchers do and so we decided to adopt the cosine similarity computation. Since we want to improve the ability of our algorithm to detect complex chords, we find this method gives us a slightly more accurate result than the Euclidean Distance can offer. By computing the cosine similarity

between chroma vectors and the template for each frame, we obtain the Chordgram as a result.

Cosine similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is defined to equal the cosine of the angle between them, which is also the same as the inner product of the same vectors normalized to both have length 1. It is thus a judgment of orientation and not magnitude. The cosine similarity is particularly used in positive space, where the outcome is neatly bounded.

The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula as shown in Equation 5.1 and 5.2.

$$A.B = \|A\|\|B\|cos\theta \tag{5.1}$$

$$cos\theta = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.2}$$

Chordgram plot

Based on the chord probabilities calculated using cosine similarity, we can plot the chordgram with frames on the x-axis and chord probability on the y-axis as in Figure 5.3.

5.1.2 Chord detection using Machine Learning

In the method, we establish that a naive application of the definition of chords to classify PCPs fails to provide good results. In addition, we also prove that once this diversity is correctly handled by machine learning methods, chords form an adequate description for recognizing music. We have used machine learning techniques for chords recognition. However, such algorithms usually need a labelled database in order to learn a classification model.

PCP features are good mid-level features which provide a more reliable and straightforward representation of songs than melody. The spectral energy is usually collapsed into a 12-bin octave independent histogram representing the relative intensity of each of the 12 semitones of an equal-tempered chromatic scale.

PCP*(p) be a vector defined for p = 0, 1, . . ., 11 according to Equation 5.3

$$PCP*(p) = \sum_{l} \|X(l)\|^2 \delta(M(l), p) \tag{5.3}$$

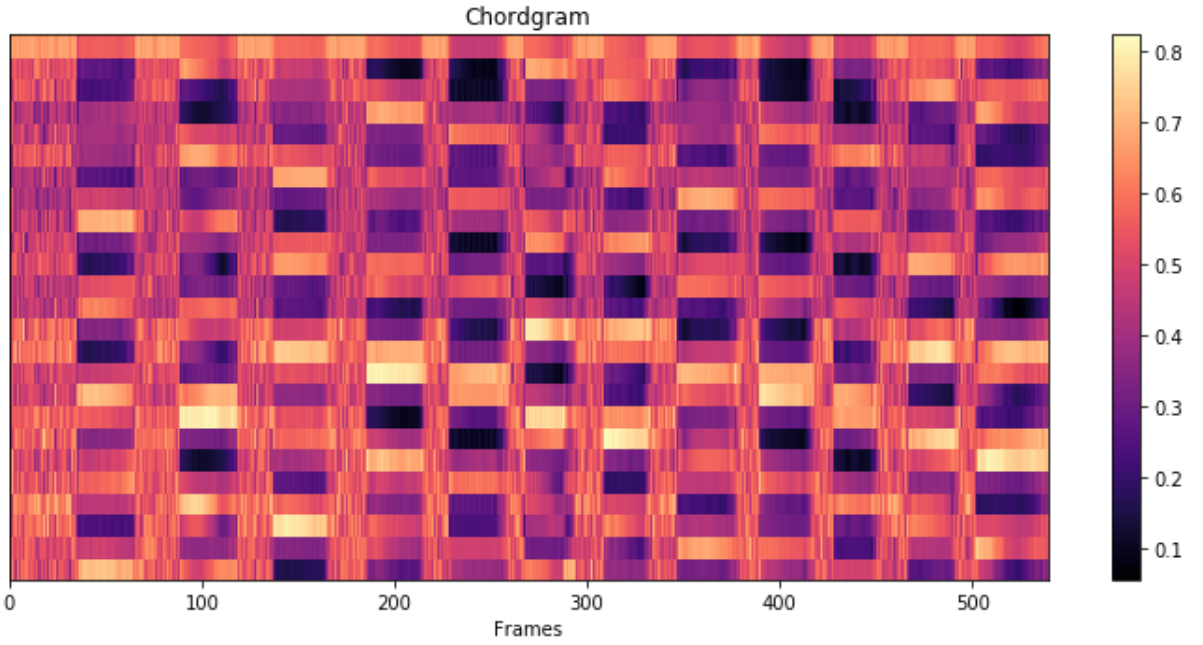


Figure 5.3: Chordgram

Where $\delta(M(l), p)$ denotes Kronecker's delta. M(l) is defined according to Equation 5.4

$$M(l) = \begin{cases} -1, & l = 0. \\ round(12 \times \log_2(Fs \times \frac{l}{N}))mod12, & l = 1, \ldots \frac{N}{2} - 1. \end{cases} \tag{5.4}$$

Where N the number of bins in the DFT of the input signal, and Fs is the sampling frequency. In order to compare PCP vectors, it is necessary to normalize them. Indeed, a chord can be played louder or softer and therefore, energy distribution can vary from one trial to another. To normalize a PCP vector, we divide each energy bin by the total energy of original PCP according to Equation 5.5

$$PCP(p) = \frac{PCP*(p)}{\sum_{j=0}^{11} PCP*(j)} \tag{5.5}$$

Database

In our database, we gathered audio files (recorded in the WAV format, sampled at fs = 44100 Hz, and quantized with 16 bits), and the corresponding precomputed PCP vectors. The PCP vectors were computed on windows comprising each 16384 samples, which correspond to 0.37 seconds. The window size was chosen experimentally. We noticed that windows containing only 2048 samples produce correct results, however, best results for our application were achieved using a bigger window size. Since our final goal is not to recognize all the existing chords, but to develop a music recognition system, we can limit chords to the most frequent ones. Therefore, we chose a subset of 10 chords: A Major, A Minor, B Minor, C Major, D Major, D Minor, E Major, E Minor, F Major, G Major.

In our database, these chords are represented by an identifier ranging respectively from 0 to 9. Note that if other chords are also played in a song; the main chords can suffice.

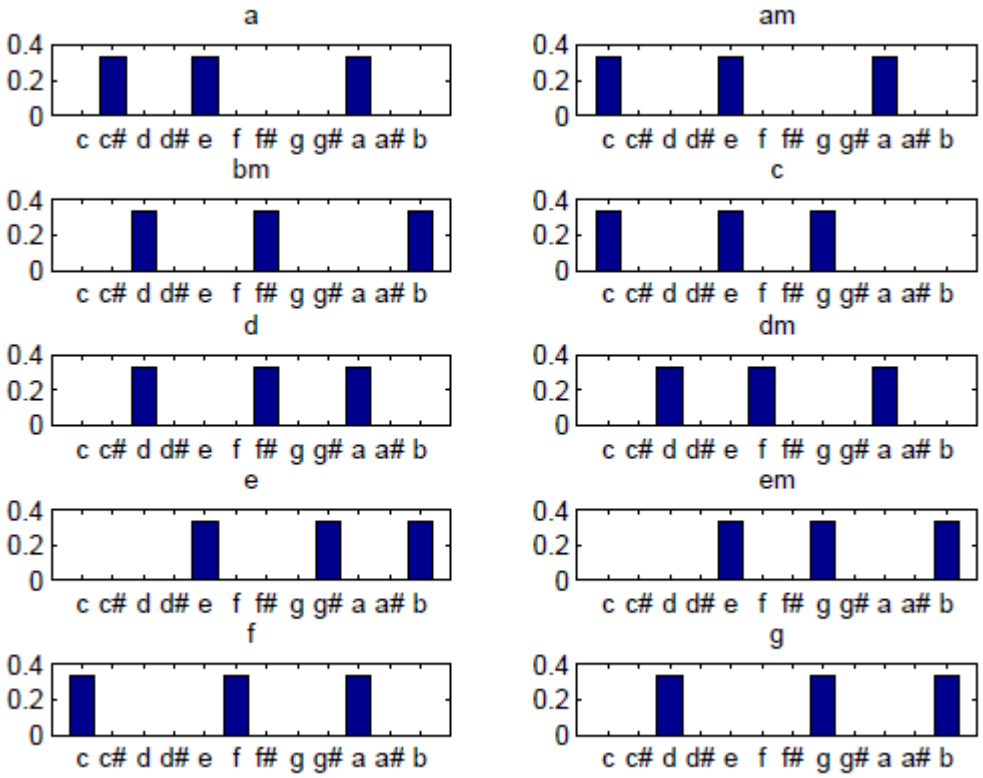


Figure 5.4: PCP representation of ten prominent chords for chord detection

Moreover, many modern songs played in Western Europe are based on these 10 chords. Therefore, it seems to be a suitable starting point to validate our recognition method. In practical terms, all PCP vectors are stored in a unique file which is organized as follows. Each line consists in a normalized PCP vector of twelve elements and one more element for the corresponding chord identifier. These were stored as feature vectors for training and testing the model.

Modelling

Most techniques proposed in the literature for chord recognition do not use machine learning methods. Since our final goal is not to develop a new chord descriptor, we chose to use a very simple, though powerful, technique to recognize chords.

We modelled the extracted features on several machine learning algorithms like SVM, KNN, logistic regression, random forest, neural networks etc. On training the model and comparing their test accuracies, we chose a feed-forward neural network using a classical gradient descent algorithm with a negative log-likelihood as cost function. The neural network architecture is described in the Table 5.3. There are twelve input attributes, which correspond to the twelve semi-tones of the PCP vector representing the chord. The neural network outputs a vector of 10 values, corresponding to the output neurons, each one being the probability of the detected chord to be issued from the corresponding chord. The final settings of the neural network were optimized by a 10-fold cross-validation on

SL.NO	NEURAL NETWORK HYPERPARAMETERS	VALUES USED
1	Number of hidden layers	1
2	Number of neurons in input layer	12
3	Number of neurons in hidden layer	35
4	Number of neurons in output layer	10
5	Learning rate	0.001
6	Momentum	0.25
7	Weight decay	0.0

Table 5.3: Neural network hyperparameters

the learning database.

5.2 Scales

A set of musical notes ordered by fundamental frequency is defined as Scale. A scale can be categorised as increasing or decreasing scale depending upon increasing or decreasing pitch. Some scales contain different pitches when ascending than when descending. Harmony is built using the notes in a single scale, this can be represented with a standard key signature.

Because of the principle of octaves, scales are generally considered to span within the same octave, with lower or higher octaves simply repeating pattern. In a musical scale the octave space can be divided into a certain number of scale steps, the distance or interval between two successive notes of the scale is called scale step. the scale steps need not be equal within any scale. As said in microtonal music, the number of notes that can be injected in any musical interval is indefinite.

The uniqueness of a scale defined by a special note and its characteristic interval pattern, the special note is known as its first Tonic. the note selected at the beginning of the octave is called the tonic of the scale, and therefore as the beginning of the adopted interval pattern. The name of the scale specifies both its interval pattern and its tonic. For example, C major indicates a major scale with a C tonic.

Scales define the emotions present in the music, it can be happieness, sadness , fear, disgust and many more. It has been scientifically proven that **"B"** is the "**happiest**" musical note. Historically, classical composers felt that **D minor** was the most melancholy of the keys, i.e. **Saddest scale**. Table 5.4 shows some of the known scales and the notes involved with it.

Types of Scales

We can classify scales into 2 main divisions based on number of notes present in a scale and the origin of scales.

SCALE	NOTES	SCALE	NOTES
C Major	C, D, E, F, G, A, B	C Major Pentatonic	C, D, E, G, A
$C^{\#}$ Major	$C^{\#}, D^{\#}, F, F^{\#}, G^{\#}, A^{\#}, C$	$C^{\#}$ Major Pentatonic	$C^{\#}, D^{\#}, F, G^{\#}, A^{\#}$
D Major	$D, E, F^{\#}, G, A, B, C^{\#}$	D Major Pentatonic	D, E, $F^{\#}$, A, B
$D^{\#}$ Major	$D^{\#}, F, G, G^{\#}, A^{\#}, C, D$	$D^{\#}$ Major Pentatonic	$D^{\#}, F, G, A^{\#}, C$
E Major	$E, F^{\#}, G^{\#}, A, B, C^{\#}, D^{\#}$	E Major Pentatonic	$E, F^{\#}, G^{\#}, B, C^{\#},$
F Major	$F, G, A, A^{\#}, C, D, E$	F Major Pentatonic	F, G, A, C D
$F^{\#}$ Major	$F^{\#}, G^{\#}, A^{\#}, B, C^{\#}, D^{\#}, F$	$F^{\#}$ Major Pentatonic	$F^{\#}, G^{\#}, A^{\#}, C^{\#}, D^{\#}$
G Major	$G, A, B, C, D, E, F^{\#}$	G Major Pentatonic	G, A, B, D, E
$G^{\#}$ Major	$G^{\#}, A^{\#}, C, C^{\#}, D^{\#}, F, G$	$G^{\#}$ Major Pentatonic	$G^{\#}, A^{\#}, C, D^{\#}, F$
A Major	$A, B, C^{\#}, D, E, F^{\#}, G^{\#}$	A Major Pentatonic	$A, B, C^{\#}, E, F^{\#}$
$A^{\#}$ Major	$A^{\#}, C, D, D^{\#}, F, G, A$	$A^{\#}$ Major Pentatonic	$A^{\#}, C, D, F, G$
B Major	$B, C^{\#}, D^{\#}, E, F^{\#}, G^{\#}, A^{\#}$	B Major Pentatonic	$B, C^{\#}, D^{\#}, F^{\#}, G^{\#}$
C Minor	$C, D, D^{\#}, F, G, G^{\#}, A^{\#}$	C Minor Pentatonic	$C, D^{\#}, F, G, A^{\#}$
$C^{\#}$ Minor	$C^{\#}, D^{\#}, E, F^{\#}, G^{\#}, A, B$	$C^{\#}$ Minor Pentatonic	$C^{\#}, E, F^{\#}, G^{\#}, B$
D Minor	$D, E, F, G, A, A^{\#}, C$	D Minor Pentatonic	D, F, G, A, C
$D^{\#}$ Minor	$D^{\#}, F, F^{\#}, G^{\#}, A^{\#}, B, C^{\#}$	$D^{\#}$ Minor Pentatonic	$D^{\#}, F^{\#}, G^{\#}, A^{\#}, C^{\#}$
E Minor	$E, F^{\#}, G, A, B, C, D$	E Minor Pentatonic	E, G, A, B, D
F Minor	$F, G, G^{\#}, A^{\#}, C, C^{\#}, D^{\#}$	F Minor Pentatonic	$F, G^{\#}, A^{\#}, C, D^{\#}$
$F^{\#}$ Minor	$F^{\#}, G^{\#}, A, B, C^{\#}, D, E$	$F^{\#}$ Minor Pentatonic	$F^{\#}, A, B, C^{\#}, E$
G Minor	$G, A, A^{\#}, C, D, D^{\#}, F$	G Minor Pentatonic	$G, A^{\#}, C, D, F$
$G^{\#}$ Minor	$G^{\#}, A^{\#}, B, C^{\#}, D^{\#}, E, F^{\#}$	$G^{\#}$ Minor Pentatonic	$G^{\#}, B, C^{\#}, D^{\#}, F^{\#}$
A Minor	A, B, C, D, E, F, G	A Minor Pentatonic	A, C, D, E, G
$A^{\#}$ Minor	$A^{\#}, C, C^{\#}, D^{\#}, F, F^{\#}, G^{\#}$	$A^{\#}$ Minor Pentatonic	$A^{\#}, C^{\#}, D^{\#}, FG^{\#}$
B Minor	$B, C^{\#}, D, E, F^{\#}, G, A$	B Minor Pentatonic	$B, D, E, F^{\#}, A$

Table 5.4: Scale and it‘s notes

Based on number of notes present:

- ⋆ **Diatonic scale**: Has 7 notes and includes the major scale and the natural minor.
- ⋆ **Chromatic scale**: This scale has 12 notes.
- ⋆ **Whole tone scale**: This scale has 6 notes.
- ⋆ **Pentatonic scale**: This scale has 5 notes.
- ⋆ **Octatonic scale**: This scale has 8 notes.

Based on origin and usage:

⋆ **Phrygian dominant scale**

⋆ **Arabic scales**

⋆ **Hungarian minor scale**

⋆ **Byzantine music scales**

⋆ **Persian scale**

5.2.1 The Krumhansl-Schmuckler Key-Finding Algorithm

In western music Key detection plays a vital role. This algorithm uses Key profiles to determine the key.There are 12 vectors in a Key profile which represents the stability of the pitch classes with respect to the key. These key profiles are obtained from the data experiments done by Kessler and Krumhansl. In this experiment people were asked to rate how well the pitch classes fit with a prior contest establishing a key. There are 24 keys and each one has its own key profile. The key profiles for C major and C minor are shown in Figure 5.5 and Figure 5.6.

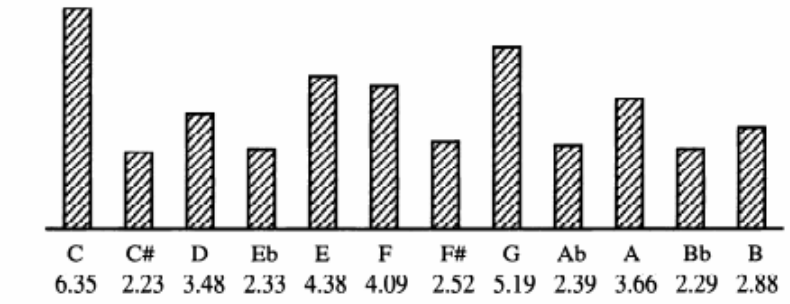


Figure 5.5: Key profiles for C major

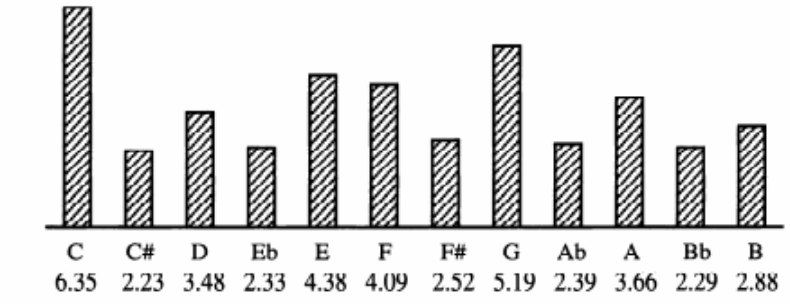


Figure 5.6: Key profiles for C minor

The profiles of other can be generated by shifting the values with the appropriate number of steps.

This algorithm detects the key of a part of audio or music by using correlation. Correlation of each key profile with the input vector. Input vector is a 12 valued vector, which contains the total duration of the pitch class in each piece.

Consider a sample audio file, let the tempo of a quarter note be 120, The pitch class 'G' in the audio file has a duration of 0.75 seconds, similarly pitch class 'A' has a duration of 0.5 seconds,duration of pitch class 'B'is 0.5 seconds and 'D' has a duration of 0.25 seconds, the duration of the remaining eight pitch classes is 0 since they do not occur in the part of the audio. The input vector for this is shown in Figure 5.7.

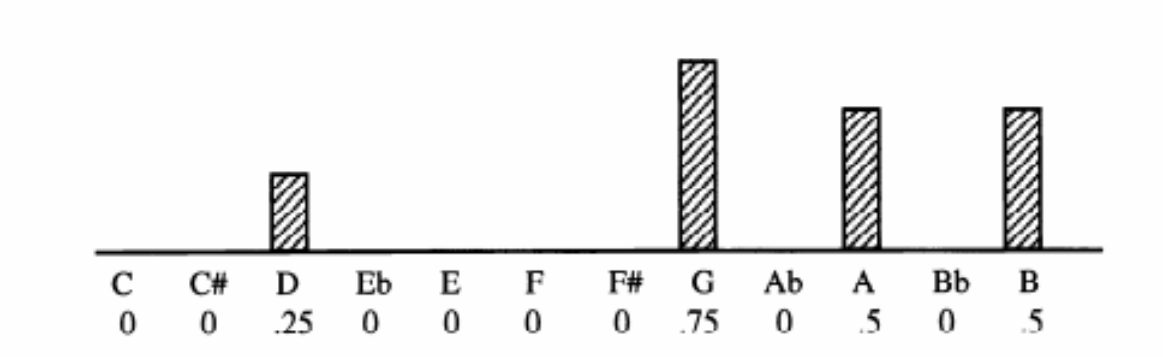


Figure 5.7: Input vector from an audio file

The formula for finding the correlation between the input vector and the key profile is given in Equation 5.6.

$$r = \frac{\sum(x - x')(y - y')}{\sqrt{(\sum(x - x')^2 \times \sum(y - y')^2)}} \tag{5.6}$$

Where x = input vector values, x' = the average of the input vector values, y = the key-profile values for a given key and y' = the average key-profile value for that key.

To determine the key, correlation should be done between each key profile and the input vector. The output with the highest correlation value is said to be the key. Template matching is used in case of Krumhansl algorithm, if the peaks from the input vector and the key profile coincide the correlation value is high, similarly when the input and the key profile of a particular pitch class do not coincide the correlation value is low. The above given correlation formula normalizes the input vector as well as the key profile for their mean and variance. Based on the output of correlation the key of the particular audio file is determined.The other method of obtaining the same result is by defining the key profile score as the sum of products of input vector and the key profile vector.

5.2.2 Scale Detection using energy thresholding

When compared to chord detection, scale detection is relatively easier as it has only one fundamental frequency at a given point of time. We used the method of musical key detection and comparing the otained set of musical keys with standard scale keys and hence decide the scale. Figure 5.8 shows the block diagram of scale detection by usinf key detection. The energy of the audio is low at the transition of notes and using this concept we isolate the notes.

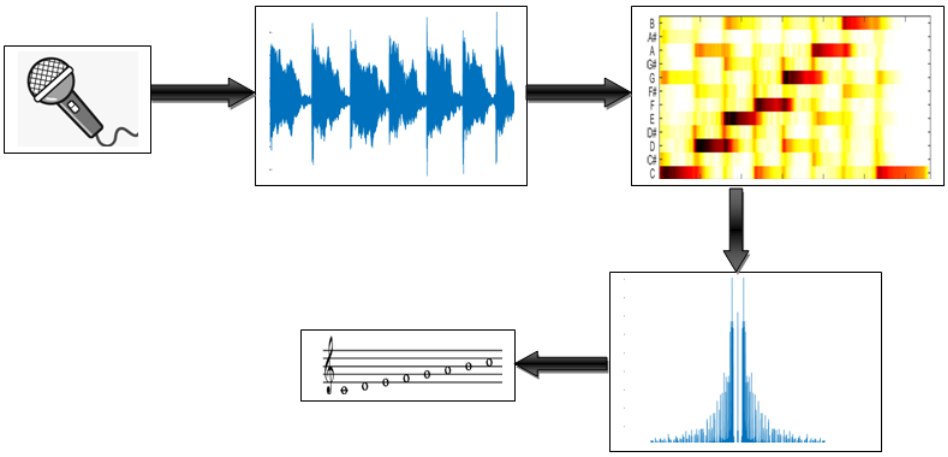


Figure 5.8: Block diagram for scale detection using energy thresholding

Steps involved in scale detection :

1. We input the audio using a microphone, record and stored digitally.
2. We plot the raw data and check for start and end index of the note by means of thresholding.
3. The start and end index of the note is determined from the spectrogram. We find end index of a note where energy is less i.e transition part of the note. We even get the start index of next note in the similar way.
4. We would have got the start and end index of a note from Step 3 so we determine the pitch. From pitch we get the note and octave details. We repeat for all the detected start and end points.
5. Once all the octave and frequency of notes are determined we compare the sequence with standard scales and hence appropriate scales is determined.

Chapter 6

Pitch Shifting

6.1 Introduction

Pitch shifting is a technique which is used to change the original pitch of a signal to higher or a lower frequency. Units that increase or decrease pitch by a musical interval such as a semi-note are called **"pitch shifters"** or **"pitch benders"**. Pitch Shifting is a way to shift the pitch of a signal without altering the total time duration. This can be achieved by altering the length of a sound. Pitch correction is a form of pitch shifting and is found in music editing and tuning softwares such as Auto-Tune to correct intonation inaccuracies in a recording or performance.

Pitch shifting requires accurate pitch detection followed by shifting the frequency by desired amount. The whole signal is not shifted by a constant amount because the harmonics would just get shifted but not scaled. The first harmonic must be shifted twice the amount fundamental frequency is shifted and the second harmonic by three times the fundamental frequency is shifted and so on.

6.2 Phase Vocoder

It is one of the most widely used pitch shifting techniques. The three stages involved in pitch shifting are analysis, processing and synthesis.

Figure 6.1 shows the block diagram of a phase vocoder system.

6.2.1 Analysis

The frequency domain provides many useful information except the information of when it happens. If an audio wave consists of 10 Hz component for 1 second and 50 Hz component for 5 seconds, the Fourier spectrum indicates just the presence of 10 Hz and 50 Hz components. To overcome this, we use Short Time Fourier Transform (STFT). In STFT the audio signal is broken down into different time durations called windows, and the discrete fourier transform of each window is computed.

The windowing process alters the spectrum of the signal and this effect should be minimized. In order to reduce the effect of windowing on the signal frequency representation, a Hamming window of size N is used. The distance between frames can be a division of N such as N/2, N/4 and so on.

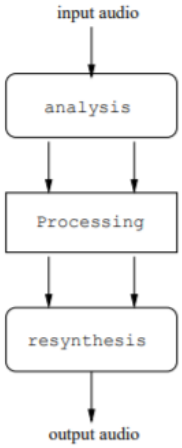


Figure 6.1: Block Diagram of Phase Vocoder

6.2.2 Processing

Once the Fourier transform is applied, we get N frequency bins from 0 to (N-1)/N*fs with an interval of fs/N where fs is the sampling frequency. The phase helps in improving the accuracy of the frequency estimation of each bin. The phase difference between two frames is called **phase shift** and it lies between (-π, π). The frequency deviation from the bin is calculated and then wrapped. Equation 6.1 and 6.2 is used to calculate frequency deviation and wrapped frequency deviation respectively.

$$(\omega_{true}[k])_i = \frac{(\Delta\phi_a[k])_i}{\Delta t_a}$$

$$(\Delta\omega[k])_i = \frac{(\phi_a[k])_i - (\phi_a[k])_{i-1}}{\Delta t_a} - \omega_{bin}[k] \tag{6.1}$$

$$\Delta\omega_{wrapped}[k])_i = mod[(\omega_{true}[k])_i + \pi), 2\pi] - \pi \tag{6.2}$$

True frequency is given by Equation 6.3.

$$(\omega_{true}[k])_i = \omega_{bin}[k] + (\Delta\omega_{wrapped}[k])_i \tag{6.3}$$

The new phase is calculated using Equation 6.4 to avoid discontinuity.

$$(\phi_a[k])_i = \phi_a[k]_{i-1} + \Delta t_a \times (\omega_{true}[k])_i \tag{6.4}$$

6.2.3 Synthesis

Once the phase is adjusted in the frequency domain of our current frame inverse fourier transform is performed on each frame spectrum and then passed through a hamming window.

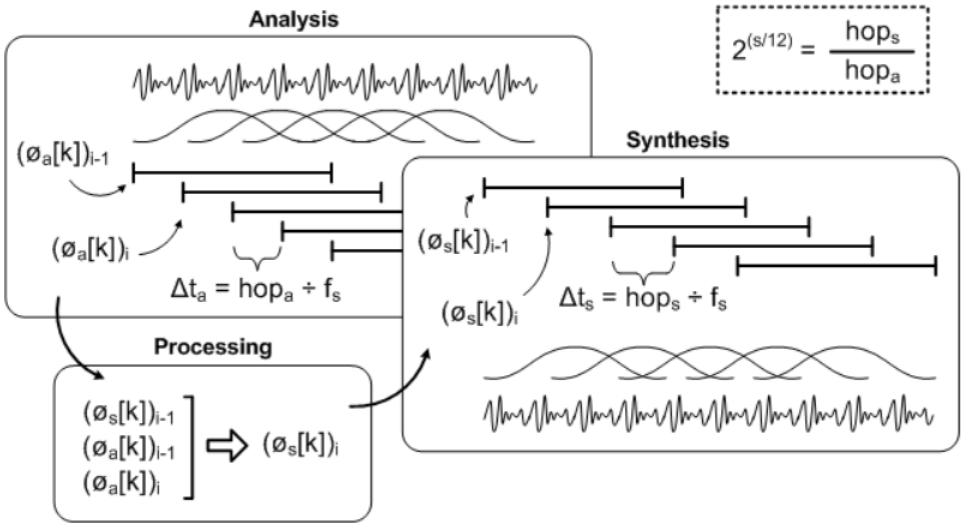


Figure 6.2: Representation of Phase Vocoder

6.2.4 Resampling

The signal after the synthesis process is either time stretched or compressed but the pitch still remains the same. Now resampling is done to get back the original signal with change in pitch. By using this method, we can easily resample at twice the previous rate. But in cases where scaling factors are not integers, linear interpolation is used to approximate the sample and that should lie in this location. Linear interpolation behaves similar to a low pass filter and it can remove aliasing when down sampling is done..

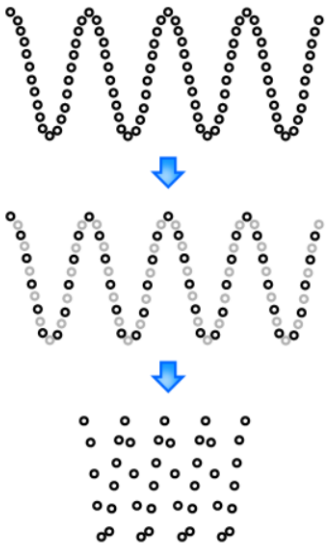


Figure 6.3: Resampling and sampling at Fs/2 for example

Linear Interpolation: (x1,y1) and (x2,y2) are two samples resulting from time stretching and xt is the expected time index for the next sample in pitch-shifting and yt is the expected amplitude value is given according to Equation 6.5

$$y_t = \frac{y_2 - y_1}{x_2 - x_1} \times (x_t - x_1) + y_1 \tag{6.5}$$

6.3 Harmonic Transformation

6.3.1 Introduction

Harmonic model was described in section 3.2 in which it was described that a given music sample can be expanded as a sum of harmonics. This looks very similar to the representation of Fourier series, i.e., sine model, but there is a small difference which helps in analyzing and synthesizing music samples very well. The equation is repeated here for better understanding.

$$y[n] = \sum_{m=1}^{N} A[n] \times cos(2\pi m F_0[n] n) \tag{6.6}$$

Where, A[n] is instantaneous amplitude, F_0 is fundamental frequency, N is number of harmonic components.

Figure 6.4 shows the harmonic analysis applied to 2 music samples. The tracking of the different harmonic components is clearly visible in the Figure 6.4. These harmonic components, can then be scaled in time or frequency to give different effects.

6.3.2 Pitch shifting using Harmonic Transformation

The Figure 6.5 shows the block diagram which illustrates the steps in pitch shifting using Harmonic Transformation.

Steps involved in pitch shifting using Harmonic Transformation. :

1. The music signal is windowed by using Blackman window, by taking 1201 samples of the music signal at a time (The odd value is for zero phase windowing) and padding zeros to make it 2048 samples (Padding is done to smoothen the spectrum), with a 25% overlap.

2. The FFT of all these windowed segments are taken and the peaks in the spectrum is found by using parabolic interpolation, where a peak and its two neighbouring values are taken and by applying parabolic interpolation, the exact frequency of the peak is obtained. The magnitude and phase values of these peaks for each of the segment is collected.

3. With these frequency, magnitude and phase of the peaks of the segments, the fundamental frequency is calculated.

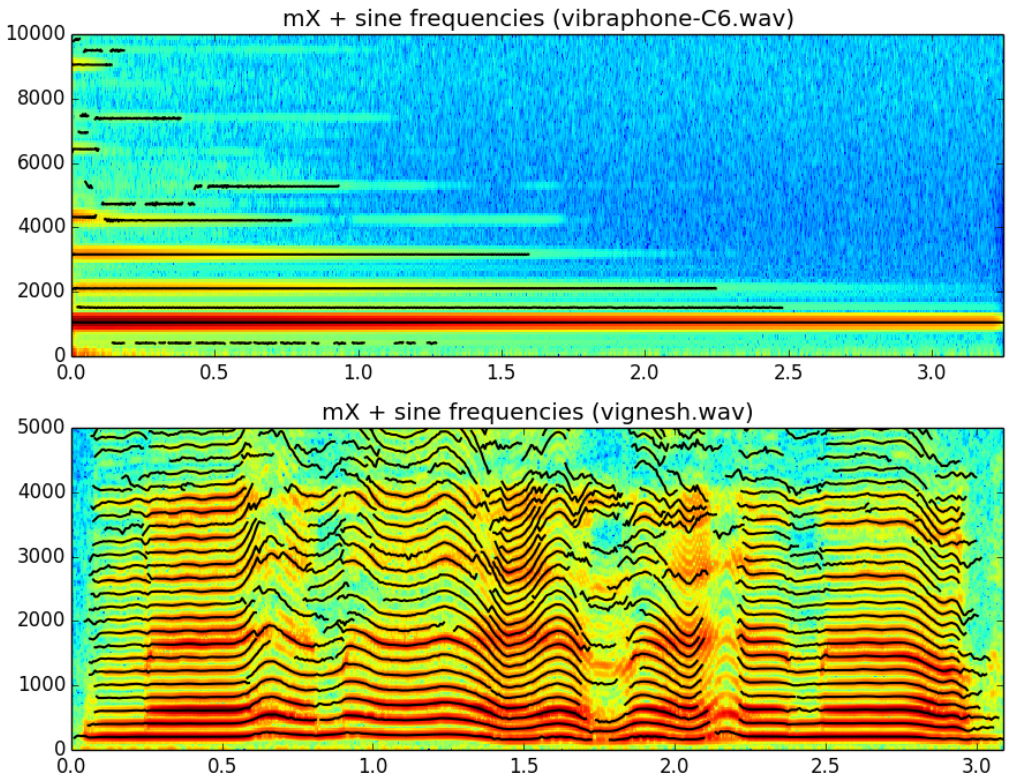


Figure 6.4: Harmonic analysis applied to 2 music samples

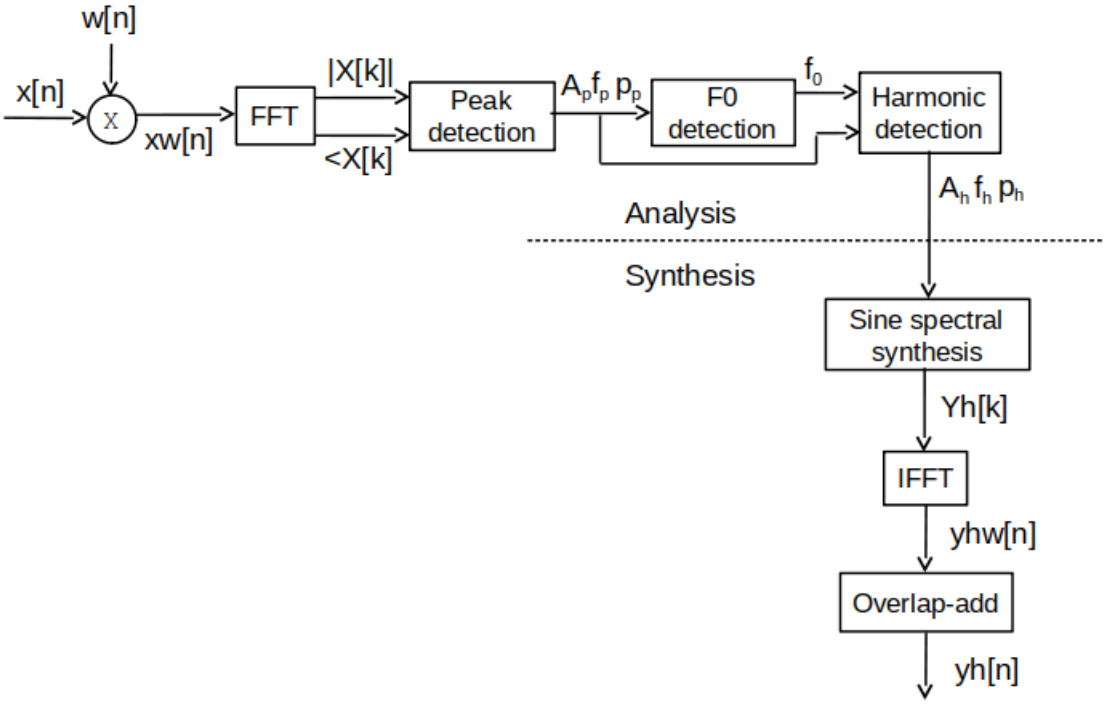


Figure 6.5: Block diagram illustrating the steps in Harmonic Transformation

4. These fundamental frequencies f0 are then analyzed by checking the amplitudes of the frequency spectrum at $2F_0, 3F_0$, etc. If the magnitudes are above a threshold, then they are labeled as harmonics and rest of the F_0 values are discarded as they are stray fundamental frequencies which are not a part of any harmonic series.

5. These harmonic values are then used to transform the pitch of the music sample. If the sample has to be upshifted, the fh value calculated in the above step is added with the change Δf to get fh+Δf. Similarly, if the audio sample has to be downshifted, the changed frequency becomes fh-Δf.

6. The modified frequency values are then reconstructed or synthesized using the Equation 6.7

$$Yh_l[k] = \sum_{r=1}^{R_l} A_{(r,l)} \times W[k - rf'_{(0,l)}] \tag{6.7}$$

   Where W is spectrum of analysis window, R is number of harmonics, A is amplitude of harmonic, f' is normalized fundamental frequency, l is frame number and r is harmonic number.

   This gives the modified spectrum of the music signal.

7. This needs to be converted back to time domain, which is done by taking the inverse Fourier transform. This is done for each windowed segment.

8. Finally, all the segments are overlap added to give the modified audio signal.

6.3.3 Pitch shifting for some audio samples

The audio sample is analyzed and synthesized using harmonic model

Figure 6.6 shows the harmonic contours, showing how the harmonics varies with time.

Transformation and resynthesis

Figure 6.7 shows the modified harmonic tracks according to the transformation parameters and resynthesized to give the modified audio sample.

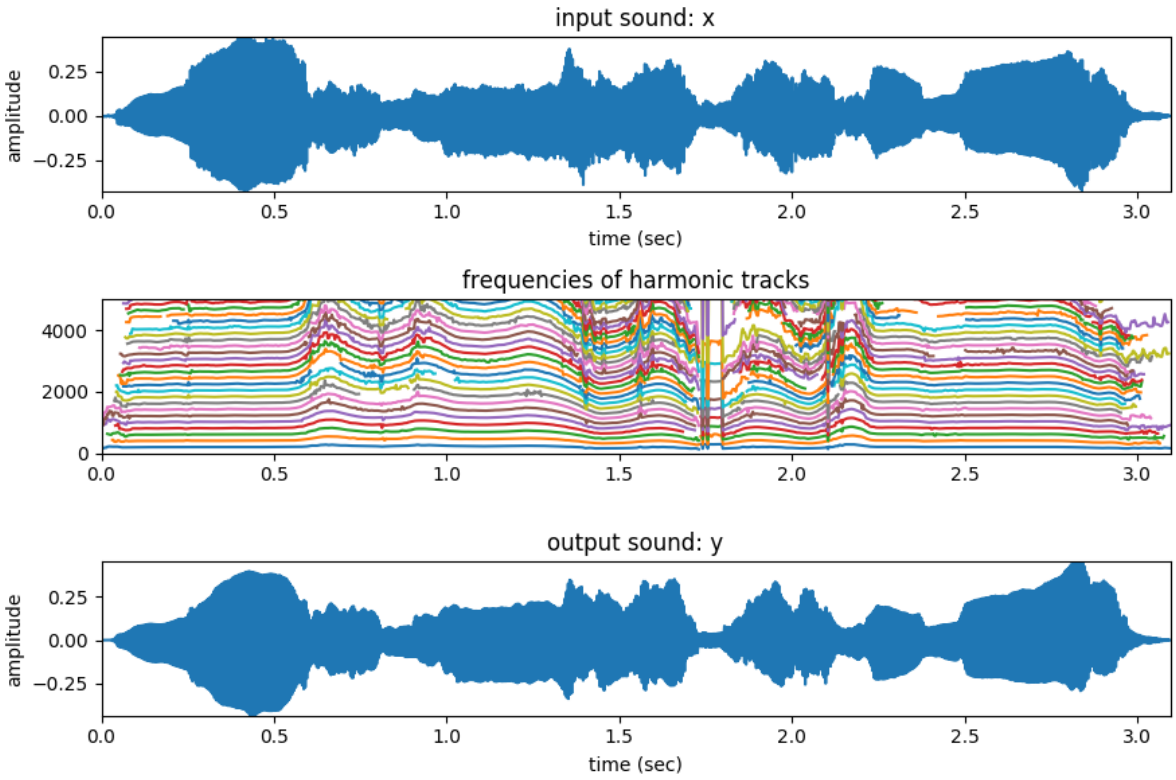


Figure 6.6: Harmonic contours and variation of harmonics wrt time

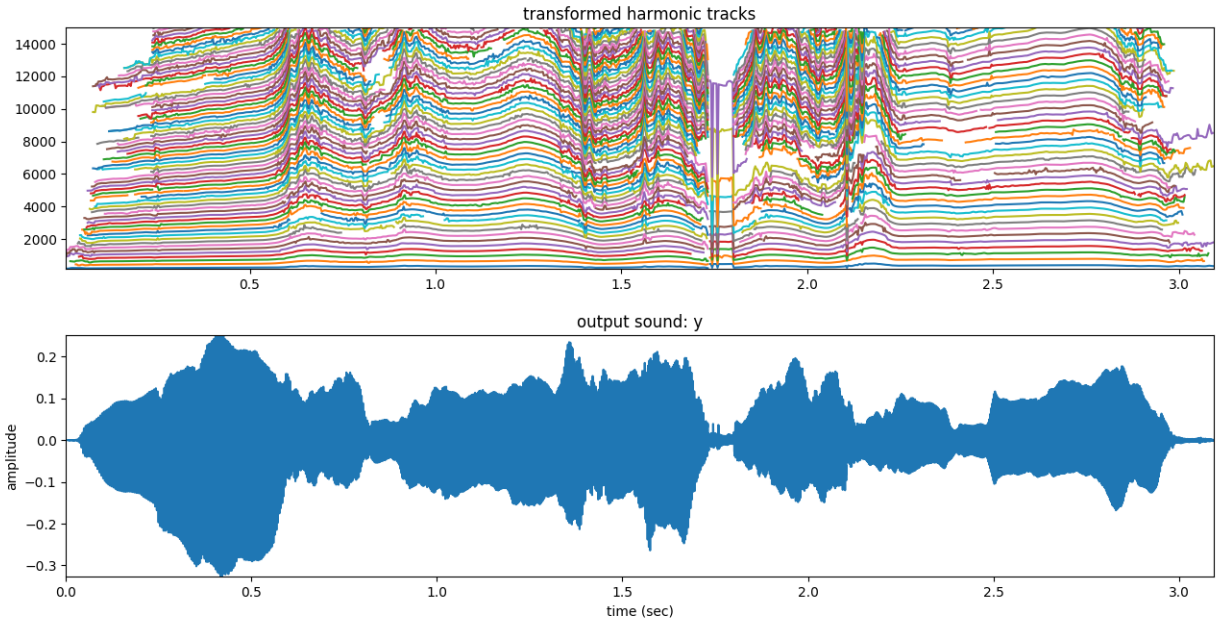


Figure 6.7: Harmonic model tracking the transformation parameter and resynthesized audio

Chapter 7

Audio Effects

7.1 Introduction

Audio effects are hardware or software devices that manipulate how an audio signal sounds. Effects can be controlled via various parameters including rate, feedback or drive. They are useful when playing live or as studio tools while recording or mixing music. It can also be defined as an enhanced sound or one that is artificially created, these effects are used to emphasize artistic content in movies and other recordings.

7.1.1 Techniques used in Audio effects

Filtering

Equalization is a form of filtering. The range of frequencies can be selected using low/high/band-pass/band-stop filters accordingly. The effect of telephone can be simulated by using band pass filters.

Time varying filters

In this type of effect we have bandpass filters whose center cut off frequency is time dependent. Wah-wah, Phaser are some of the effects which involves time varying filters. In Wah-wah effect we linearly varying the center cut off frequency from 500 - 3000Hz.

Delays

In this type of effect we usthe delayed version of the audio and actual audio for processing. Vibrato, Flanger, Chorus, Echo are some of the audio effects involving Delays.

Modulations

To change the carrier frequency or amplitude of a signal in relation to a predefined signal. Ring Modulation is nothing but amplitude modulation. Effects like Tremolo and Vibrato are frequency modulation effects.

Non Linear Processing

Compression, Limiters, Distortion, Exciters/Enhancers are some of the audio effects which involves Non Linear Processing.

Spatial Effects

This involves spatial effects like 4D, 5D effects which make us feel the sound is surrounding us. Panning, Reverb, Surround Sound aresome of the methods

Pitch shift

Moving the fundamental frequency to desired region without destroying the timbre of the instruments. Wehave discussed methods for shifting pitch in Chapter 6.

In our project we have implemented 2 audio effects, they are Vibrato and Tempo. So we try to explain what ,its types and method used to implement the same.

7.2 Vibrato

7.2.1 Introduction

Pulsating change of pitch observed in music is called vibrato. The periodic and regular wavering that you hear in many musical instrument sounds including singing voices. Basically, it is a slight frequency shift that is applied on a music. The volume is kept constant and the variation in frequency is not more than 1.5% above or below the steady state of pitch at that particular instant. As such, it would be rather uninteresting to listen to but as soon as the signal with vibrato leaves a speaker and becomes audible tone, the effect becomes a complex vibrato. The reason for this is that sound travels at around 1120 feet/second, thus when sound waves propagate through the air, there is an increasing time delay as the distance from the speaker increases. Soundwaves travel in a spherical pattern.

For example, let‘s say a person is listening to a tone with a simple vibrato that is a small change in frequency in a typical room in a house, and the speaker is on one side of a room that is 11.2 feet long. It will take the sound 0.01 second to traverse the room, after which it will reflect from the opposite wall. There will be a slight delay, because of the slight delay, the pitch of the sound wave at the wall will be slightly different at any instant in time from the frequency at the speaker. A listener standing in the room, however will hear both the speaker‘s sound and that reflected from the wall. Walls of the room are straight and flat whereas sound travels in spherical pattern. So, given this scenario the effect becomes very complex as sound reflects from the interior surfaces in a room and also from objects within the room. Thus, the simple vibrato produces an infinitely complex effect in a listening space and it is why vibrato becomes a very nice effect to hear, adding richness and complexity to even a simple tone.

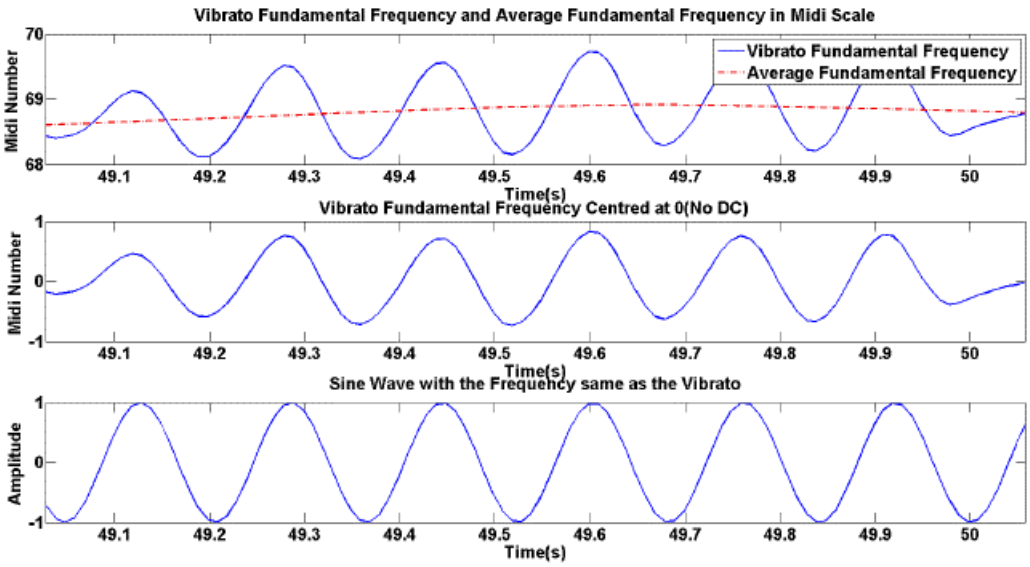


Figure 7.1: Fundamental Frequency and Fundamental frequency with vibrato

7.2.2 Types of Vibrato

Simple Vibrato

As the name says simple vibrato is easy to understand. It is just variation of frequency about a mean fundamental frequency.

Heterodyne Vibrato

Consider two sound waves such that the difference between them is less. During the propagation of the waves they get in and out of phase. The waves add up when they are in phase and produce a louder sound. When they are out of phase they cancel each other resulting in reduction of sound level. Therefore, one might think the final effect would be a simple tremolo. However, this is not so this type of vibrato is referred to as a heterodyne vibrato. The Figure 7.2 shows two sound waves with slight variation in frequencies and the addition of waves when they are in phase.

Polyphase Vibrato

A polyphase vibrato is derived out of simple vibrato, when a simple vibrato leaves the speaker it gets converted into polyphase vibrato. A polyphase vibrato is generated electronically. It can be simple or complex. For example there are two identical signals with phase difference of 180 degrees in the vibrato of the X66 Hammond, this means that one is increasing and the other is decreasing. These two vibrato signals must are not combined electrically, but mixed as sound waves in the air after they leave their respective speakers.

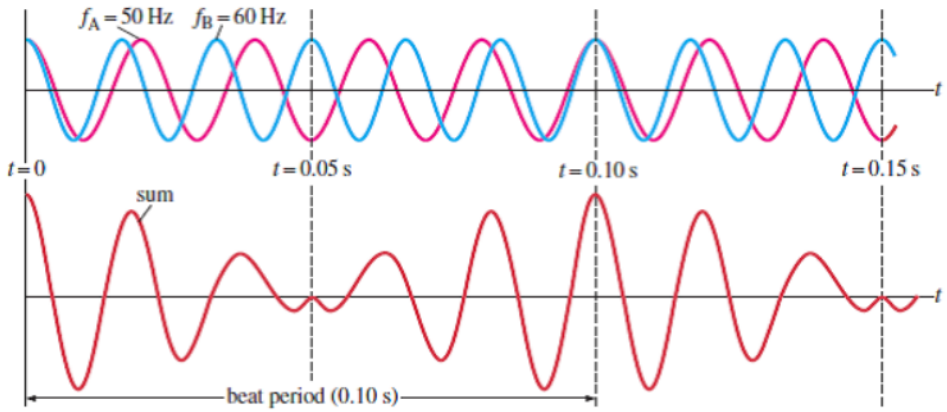


Figure 7.2: Addition of waves with sight variation in frequency

Constant Pitch Shift Vibrato

This effect requires gating of two separate constant pitch shift vibrato signals, electrically we shift the phase of one signal by a constant amount. When the two signals are out of phase, the signals are alternately trimmed on and off, the output signal has continuous phase shift and thus has a pitch vibrato.

Hammond Scanner Vibrato

Theoretically a Hammond scanner vibrato is regular, simple vibrato, but practically it is a complex vibrato even before it leaves the speaker or speakers. This is due to the presence of some low amplitude reflected waves that form in the vibrato line box. This is scanned as the line box and the final output is a complex polyphase vibrato.

Often, we get confused between Vibrato and Tremolo, so we have explained a little bit about tremolo here, Periodic and regularly-recurring change in the instantaneous volume of a musical tone is called tremolo, and vibrato is a periodic and regularly-recurring change in the instantaneous pitch of a musical tone. In Tremolo the amplitude of the sound wave is continuously varying. Figure 7.3 shows Tremolo.

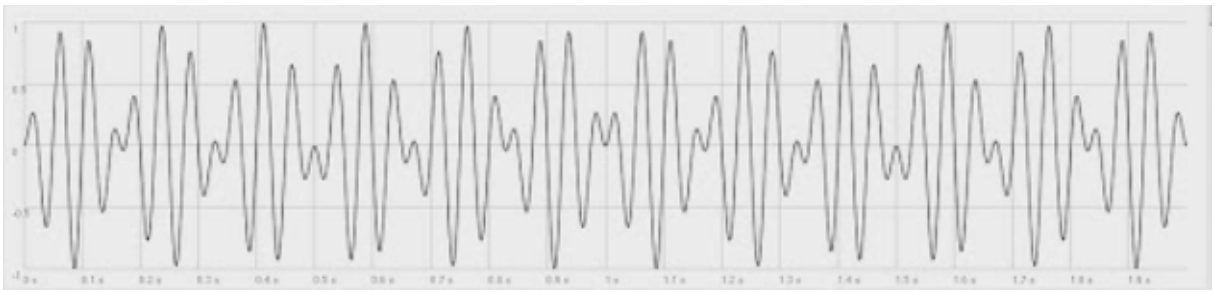


Figure 7.3: Addition of waves with sight variation in frequency

7.3 Tempo

Tempo is defined as the speed at which a part of the music is played. The speed of the music helps depict the mood or character of that piece of the song, for example a fast speed might convey a bright lively mood while a slower piece could convey a melancholy message. The key element of a musical performance is tempo. Just like melody, rhythm, dynamics and harmony tempo is equally important in music. The three ways in which tempo can be communicated is BPM, modern language and Italian terminology.

Beats Per Minute (BPM)

BPM is the method of assigning a numeerical value to tempo. A tempo communicated as 60BPM means that there is one beat every one second and a 120 BPM means to beats per second and it would be twice as fast as 60 BPM. A beat corressponds to a piece's time signature.

A time signature containing 4 at the bottom (2/4, 4/4, 3/4, 5/4), beat corresponds with quarter notes. To complete a full measure in a 4/4 time we need four beats, similarly in a 5/4 time, every five beats make a full measure. A time signature containing 8 at the bottom (3/8, 6/8, or 9/8), an eight note corresponds with a tempo beat.

At times beats other durations. For instance, if a person wants through a measure of 12/8, he could choose a tempo that represents dotted eighth notes (one measure = 4 tempo beats) or one that represents eighth notes(on measure = 12 tempo beats). The best way to indicate a slow or fast temp is using BPM. Tempo plays a vital role in musical applications where the precision is necessary. It is used in high level professional recordings to set the metronomes.

Italian Music Terminology

Italian words are used to convey change in speed of a part of music through specific information. Italian tempos are widely used when compared to others (popular among those are largo, presto, and andante). Italian music uses the following terminologies for tempo markings:

- ⋆ **Larghissimo**: Larghissimo has a very low tempo with tempo less than 20 bpm.
- ⋆ **Grave**: Grave has slow and solemn tempo of 20 to 40 bpm.
- ⋆ **Lento**: Lento has a tempo range of 40 to 60 bmp.
- ⋆ **Largo**: Largo has a range of tempo from 40 to 60 bpm.
- ⋆ **Larghetto**: Larghetto hastempo ranging from 60 to 66 bpm.
- ⋆ **Adagio**: Adagio has tempo in the range 66 to 76 bpm.
- ⋆ **Adagietto**: Adagietto has a tempo range of 70 to 80 bpm.
- ⋆ **Andante moderato**: Andante moderato has a bit slower than andante.

- ⋆ **Andante**: Andante has tempo ranging from 76 to 108 bpm.
- ⋆ **Andantino**: Andantino has slightly faster than andante.
- ⋆ **Moderato**: Moderato has a moderate tempo of 108 to 120 bpm.
- ⋆ **Allegro moderato**: Allegro moderato has moderate tempo measuring around 112 to 124 bpm.
- ⋆ **Allegro**: Allegro has a fast tempo of 120 to 168 bpm.
- ⋆ **Vivace**: Vivace has approximately 140 bpm tempo.
- ⋆ **Presto**: Presto has a very fast tempo of 168 to 200 bpm.
- ⋆ **Prestissimo**: Prestissimo is extremely fast with tempo starting from 200 bpm and above.
- ⋆ **Accelerando**: Gradually becoming faster.
- ⋆ **Ritardando**: Gradually becoming slower.

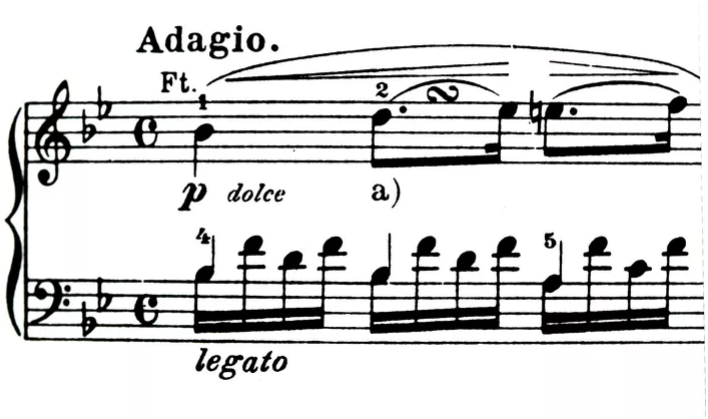


Figure 7.4: Musical sheet showing two different tempo

7.4 MIDI

Musical Instrument and Digital Interface is a connectivity standard for transferring digital instrument data. Using MIDI one can edit the the notes in it, change the velocity of the note, tempo of a part of music can be changed, alter the duration of the note played, define the track, increase or decrease the volume of a particular note that is played, change the articulation. MIDI is a collection of messages like **"NOTE ON", "DURATION", "NOTE OFF", "TEMPO", "PITCH BEND", "VELOCITY"**.

Midi has many advantages:

- ⋆ **It is compact**: An audio file which has a large size can be converted to MIDI which will be much lesser in size since it consists of only few hundred MIDI messages.
- ⋆ **Notes can be modified accordingly**: The pitch, duration, tempo and other parameters of audio file converted to MIDI format can be easily altered as required.
- ⋆ **Change Instrument**: MIDI is just a series of messages which describe the notes being played, these notes can be sent to any instrument and the overall sound of composition can be changed.

In a keyboard when a key is pressed, the instrument creates a "NOTE ON" message. This message records the pitch and the velocity of the key i.e. how fast a particular key is pressed.

MIDI Number

This indicates the pitch of the key that is pressed with a value ranging between 0-127. The formula for conversion of pitch from frequency to MIDI number is given in Equation 7.1.

$$MIDINumber = 12 \times (log_2(\frac{Pitchfrequency}{440}) + 69) \tag{7.1}$$

"Velocity" is a message in MIDI which between 0-127, this describes the gain(volume) of a particular note, if the velocity is more it will sound louder. Different velocities create different timbre in an instrument. Once the key is released the instrument creates another message "NOTE OFF". When two keys are pressed at once and one of them released, the "NOTE OFF" message of one won't cause the end of both notes, it creates a "NOTE OFF" message only for the key that was released.

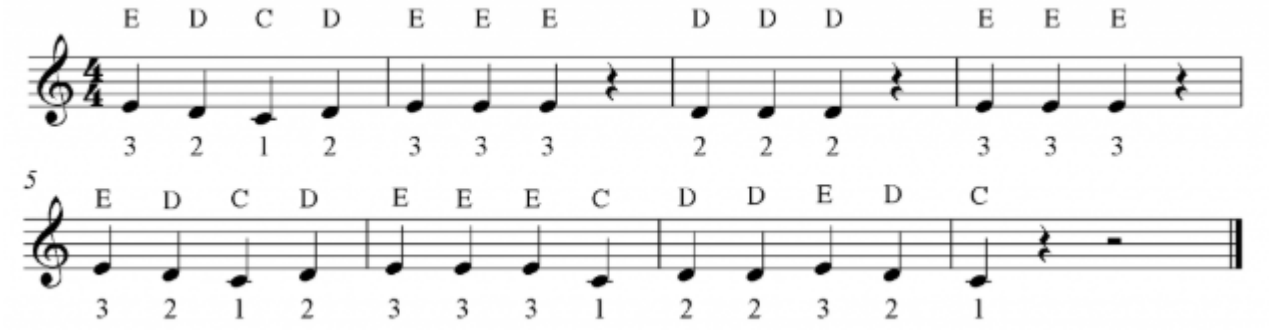


Figure 7.5: Music Sheet for a piano audio

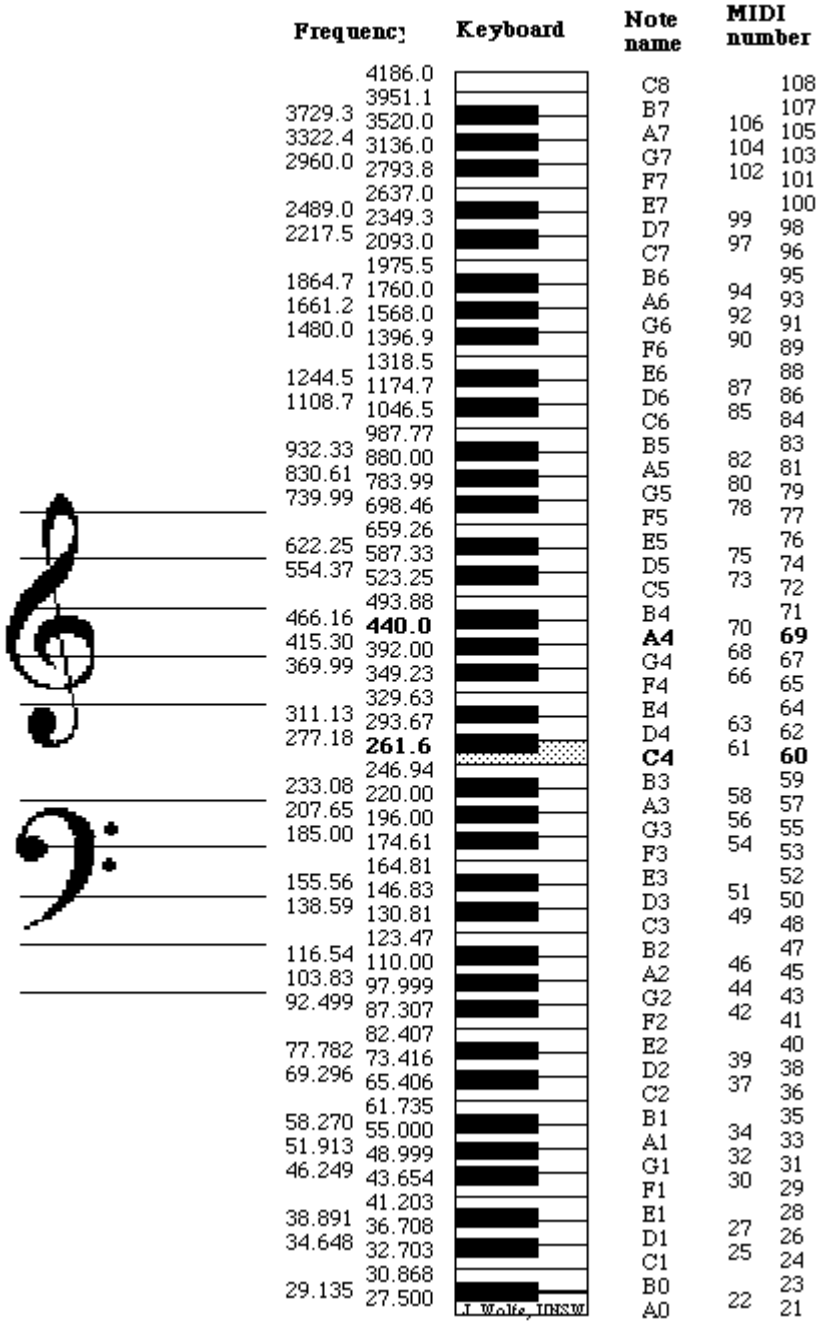


Figure 7.6: Note Frequency and the corresponding MIDI Number

Chapter 8

Methodology

In chapter 4 we spoke about various pitch detection algorithms, in chapter 5 we spoke about detection of chord and scales and in chapter 6 we spoke about pitch shifting algorithms. All these algorithms were implemented in MATLAB software. We integrated all these under a single platform and built a **Graphical User Interface(GUI)** using the same MATLAB software. We have 2 tabs in our app namely "PITCH TUNER/CORRECTOR" and "PRACTICE SINGING". Figure 8.1 and Figure 8.2 shows the frontend design of **PITCH TUNER/CORRECTOR** and **PRACTICE SINGING** respectively.

8.1 App Layout and Explanation of panel controls

In this section we discuss the description, working and requirements of various panels and controls.

8.1.1 PITCH TUNER/CORRECTOR

Figure 8.1 shows the frontend design of our app built using MATLAB for **PITCH TUNER/CORRECTOR**. This is the main core of our project where we integrated each and every features under one single platform. This was built if a singer or a musician finds mistakes in the recording or a recorded audio and wants to correct them. User can move pitches up/ down according to his wish. He can check for standard scales or chords if present. User can also change audio effects like by adding vibrato or by changing tempo.

TYPE OF AUDIO TO BE USED PANEL

- ⋆ **RECORDED AUDIO FILE** Radio Button: When this Radio Button is selected the user need to upload an audio file. **BROWSE** is clicked to select any audio file in the system. Text box next to BROWSE button displays the name of audio chosen.
- ⋆ **RECORD LIVE** Radio Button: When this Radio Button is selected the user need to record an audio file using RECORD and STOP buttons.

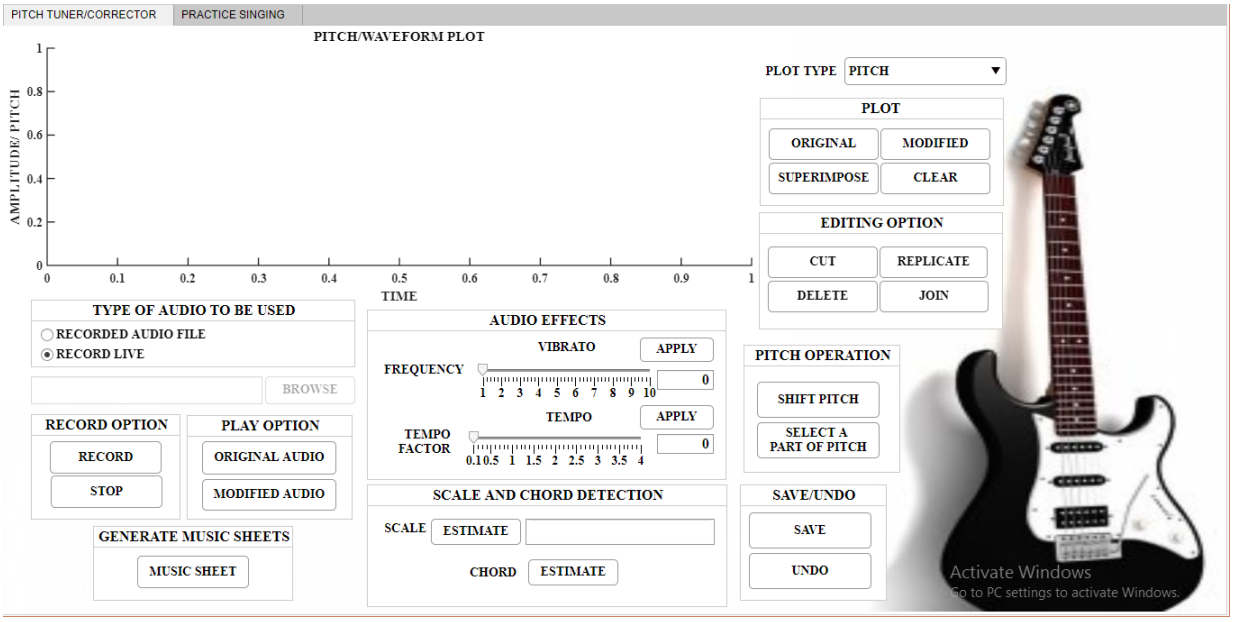


Figure 8.1: An App build on MATLAB software to correct pitch,find chord and scales

RECORD OPTION PANEL

- ⋆ **RECORD**: This button starts the recording. As the user records corresponding data is also plotted (plotted in purple) in Axes so that user can see what he is singing. A working microphone and sound card must be installed in your computer.
- ⋆ **STOP**: This button stops the recording. Immediately after pressing STOP, the tool will run the pitch detection algorithm.

PLAY OPTION PANEL

- ⋆ **ORIGINAL**: This button allows the user to play original audio.
- ⋆ **MODIFIED**: This button allows the user to play modified audio.

PLOT PANEL

User must select if he wants a **PITCH** plot or **WAVEFORM** plot from the **PLOT TYPE** Dropdown list box and later one can select any of the following buttons.

- ⋆ **ORIGINAL**: This button helps the user in plotting the original audio vs time. Original audio can be a recorded one or an audio which is already recorded and uploaded by the user. We have chosen purple colour for plotting original audio.
- ⋆ **MODIFIED**: This button helps the user in plotting the modified audio vs time. Modification can be anything like pitch shifting, editing options. We have chosen blue colour for plotting modified audio.

- ⋆ **SUPERIMPOSE**: This button allows the user to plot original audio vs time(purple) and modified audio vs time(blue) simultaneously so that user can find the difference between the original and modified audio.

- ⋆ **CLEAR**: This button clears the Axes/Plot.

PITCH OPERATION

- ⋆ **SELECT A PART OF PITCH**: Using this button user can select a part of pitch and any operations can be applied on the selected region. When this button is clicked user will be given an option to draw a rectangle so that he can fit the ROI within the rectangle. Selected region gets highlighted with red colour in order to check if desired region is selected.

- ⋆ **PITCH SHIFT**: Using this button user can shift the pitch up or down in the plot itself. When this button is clicked user can move pitch up or down flat region of the plot i.e. a single note at a time.

EDITING OPTION PANEL

- ⋆ **CUT**: On selecting the ROI user can cut a part of audio and replace it to a place he wish to replace it. On clicking this button one can remove certain part of the audio temporarily and replace it to another position. This is used when a correct pitch is present but at a wrong place, so one can cut that part of audio and replace it to the desired place using JOIN button.

- ⋆ **REPLICATE**: On selecting the ROI an user can duplicate a part of audio and place it any number of times and anywhere in the audio. Point of joining is found using the same selection of pitch where the starting point of rectangle is the point of interest for joining the pitch. This is used when a correct pitch is present but needed at multiple places, so once can replicate that part of audio and place it to the desired place using JOIN button.

- ⋆ **DELETE**: Unwanted noise or pitch can be removed by the user when he selects and delete corresponding part of the audio. On clicking this button one can remove certain part of the audio permanently. Noise or/and unwanted pitch can be removed from the audio.

- ⋆ **JOIN**: This is used mostly after using CUT and REPLICATE option. Segmented/ Replicated audio can be joined at any point of the audio. Point of joining is found using the same selection of pitch where the starting point of rectangle is the point of interest for joining the pitch.

CHORD AND SCALE DETECTION PANEL

- ⋆ **ESTIMATE CHORD**: There is a **ESTIMATE** button next to text box when pressed it checks if there is any important chord present in the audio. Detected chord is displayed in the corresponding text box and variation of chords wrt time is plotted in the axes.

- ⋆ **ESTIMATE SCALE**: There is a **ESTIMATE** button next to text box when pressed it checks if there is any important scale present in the selected part of audio. Detected scale is displayed in the corresponding text box.

AUDIO EFFECTS

- ⋆ **VIBRATO**: The silder is used to select desired oscillating frequency between 1 and 10Hz and text box next to it displays the oscillating frequency selected. On clicking APPLY button we can apply vibrato effect to the selected part of audio
- ⋆ **TEMPO**: The silder is used to select desired temp rate between 0.1 and 4 and text box next to it displays the tempo rate selected. On clicking APPLY button we can apply tempo effect to the selected part of audio

GENERATE MUSIC SHEET

MUSIC SHEET button helps in generating midi files and displaying the music sheet of the audio selected

SAVE/UNDO MODIFICATION PANEL

- ⋆ **SAVE**: Once user is satisfied with the pitch operations or modifications he has performed he can save the modified audio. On clicking the button user saves his changes in ".wav" format which can be later accessed.
- ⋆ **UNDO**: If te user is not satisfied with the pitch operations he has performed he undo all the changes and retain the same audio. User can restart his pitch operations without saving the changes

8.1.2 PRACTICE SINGING

Figure 8.2 shows the frontend design of our app built for **PRACTICE SINGING**. This was built mainly for those users who are new to music and need a guidance about there errors. A user can upload his teacher's audio and try to mimic it and practice the same, later find how close he was wrt his teacher. If he doesnt have any reference audio he can upload the table and follow the pitch accordingly.

UPLOAD TABLE/AUDIO PANEL

- ⋆ **UPLOAD A REFERENCE AUDIO FILE** Radio Button: When this Radio Button is selected the user need to upload an audio file. **BROWSE** is clicked to select any audio file from the system. Text box next to BROWSE button displays the name of audio chosen.
- ⋆ **UPLOAD TABLE WITH PITCH TIME DETAILS** Radio Button: When this Radio Button is selected the user need to upload the table with pitch he wish to sing and duration of the pitch he will be singing. User needs to select pitch from **SELECT PITCH** dropdown list and type **DURATION** in the textbox, he need to press **APPEND** button and corresponding details enter the table. After entering

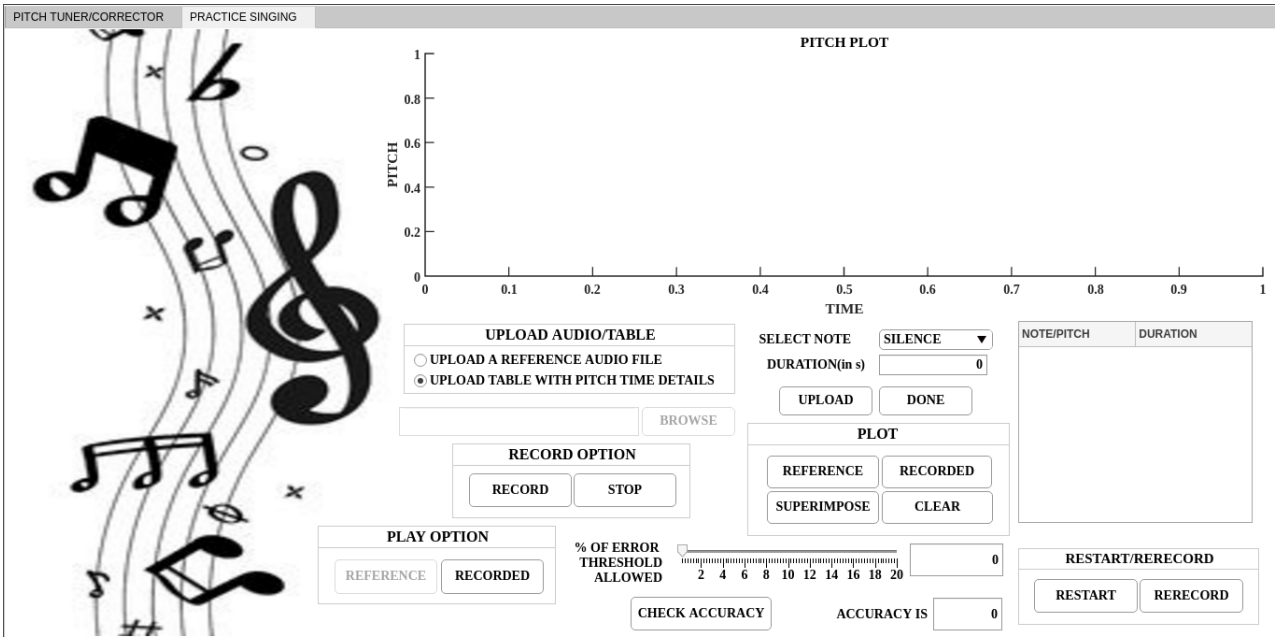


Figure 8.2: An App build on MATLAB software to practice singing

all the pitch and duration details user must press **DONE** button for completion of table.

RECORD OPTION PANEL

- ⋆ **RECORD**: This button starts the recording and reference audio (purple) will be plotted in the Axes so that user can follow the reference. As the user records corresponding data is also plotted (blue) in Axes so that he can see what he is singing. A working microphone and sound card must be installed in the computer
- ⋆ **STOP**: This button stops the recording. Immediately after pressing STOP, the tool will run the pitch detection algorithm.

PLAY OPTION PANEL

- ⋆ **REFERENCE**: This button allows the user to play reference audio.
- ⋆ **RECORDED**: This button allows the user to play recorded audio.

PLOT PANEL

- ⋆ **REFERENCE**: This button allows the user to plot pitch of reference audio vs time. We have chosen purple colour for plotting reference audio.
- ⋆ **RECORDED**: This button allows the user to plot pitch of recorded audio vs time. We have chosen blue colour for plotting reference audio.
- ⋆ **SUPERIMPOSE**: This button allows the user to plot pitch of reference audio vs time(purple) and recorded audio vs time(blue) simultaneously.

⋆ **CLEAR**: This button clears the Axes/Plot.

RESTART/RERECORD PANEL

⋆ **RESTART**: By clicking this button user can start his practice from beginning i.e. the reference audio and recorded audio will be erased.

⋆ **RERECORD**: By clicking this button user can record his audio again for the same refernce audio file i.e. only recorded audio will be erased where as reference audio will be unaltered.

CHECK ACCURACY

⋆ % **OF ERROR THRESHOLD ALLOWED** Slider: This slider has values from 0.25% to 20%. This slider value is the maximum percentage of pitch deviating from reference pitch which is permicible for it to be considered as a right pitch. Textbox next to the slider gives the value of slider value. If the recorded pitch is in the range **Reference pitch** $\pm$ **Reference pitch** $\times$ **slider value**% then it is considered as a correct frame else wrong frame.

⋆ **CHECK ACCURACY**: When this button is clicked recorded audio is compared with the reference audio and % of accuracy is displaed in the textbox.

8.2 Operation of the App built

In this section we try to explain the procedure of using our Graphical User Interface(GUI).

8.2.1 PITCH TUNER/CORRECTOR

This is the core of our project and involves many musical operations. We can chose a recorded audio or record live and use that for audio processing. Figure 8.3 shows uploading an audio and plotting the same. User can upload required audio using the **BROWSE** button. User is allowed to select only **.wav or .mid or .mp3** files else error will be generated. If user wishes to record then he can make use of **RECORD** and **STOP** button to start and stop his recording. The sampling rate chosen was 48000, mono channel. Pitch shifting can be achieved by clicking the **SHIFT PITCH** button. Once the user presses this button the pitch plot of modified audio is plotted and user can directly move the straight part of pitch (single note) up or down basedon the need.User needs click and drag to the desired pitch level. Figure 8.4 shows the original and modified (pitch shifted) pitch plot, blue being the modified and purple the original audio.

User can check for the scale present in a particular part of audio. So the user needs to select a part of pitch using **SELECT A PART OF PITCH** button and then press the **ESTIMATE** button. If there is any scale present in the selected part, corresponding name of the scale is obtained in the text box next to estimate button. Figure 8.5 shows the scale detection, selected region is highlighted in red and CMAJOR is the scale present

Figure 8.3: App showing uploading a recorded audio

Figure 8.4: App showing pitch shifted audio and original audio

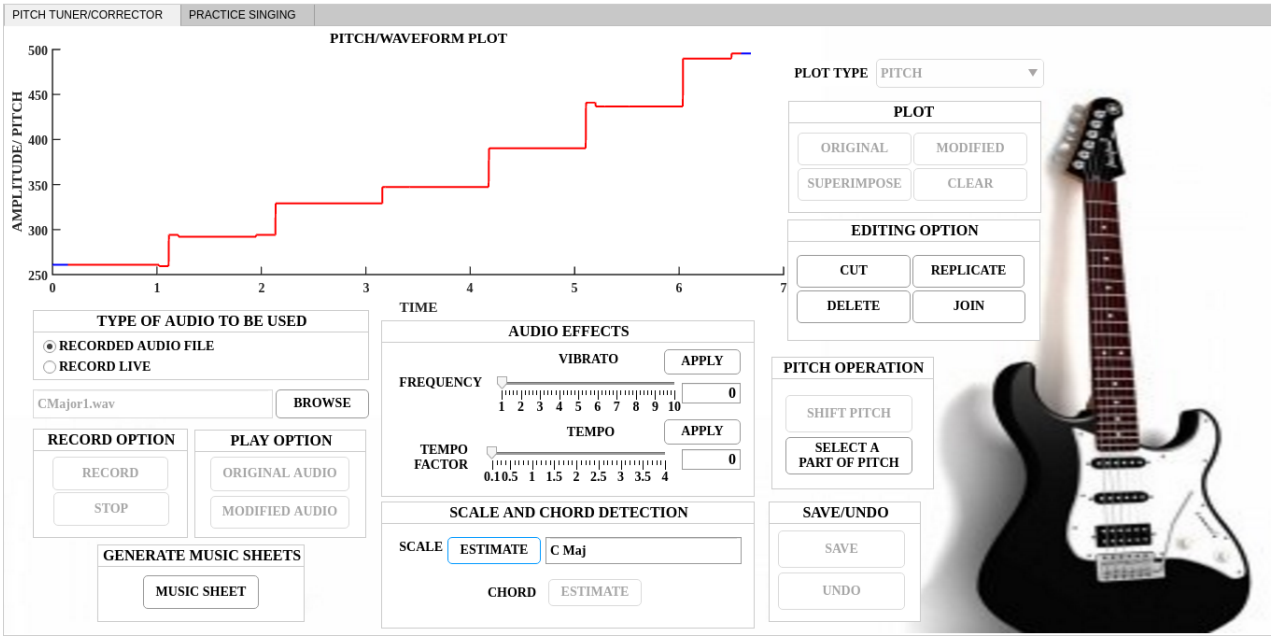


Figure 8.5: App illustrating the extraction of scale information

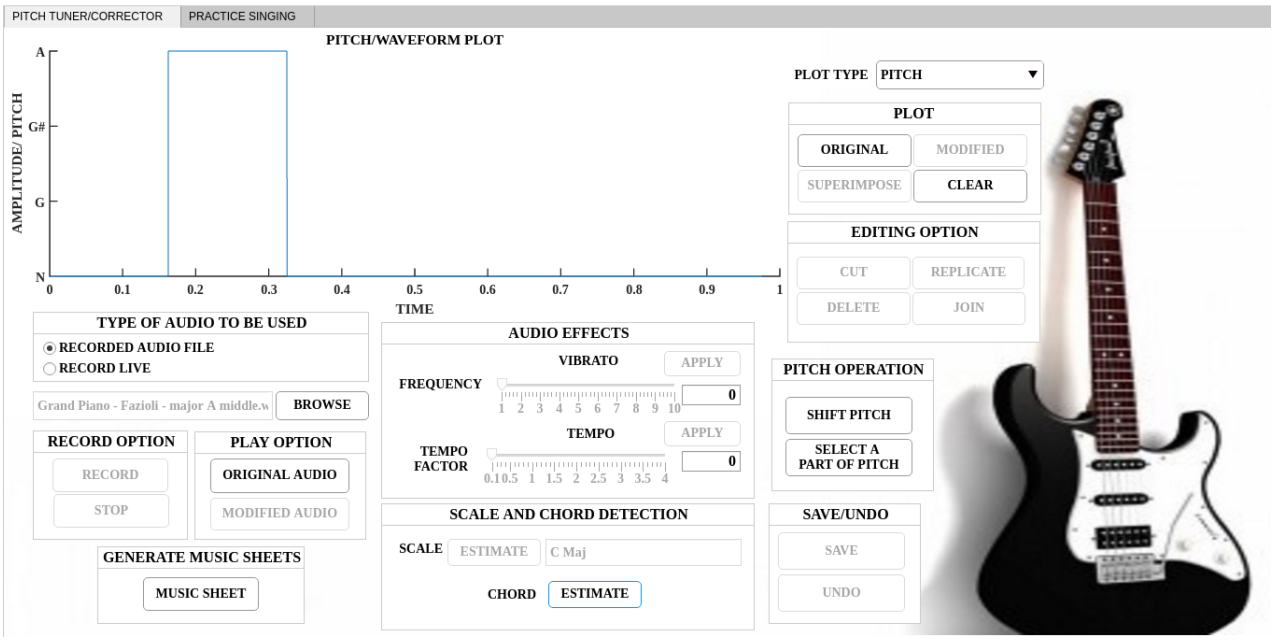


Figure 8.6: App demonstrating Chord detection

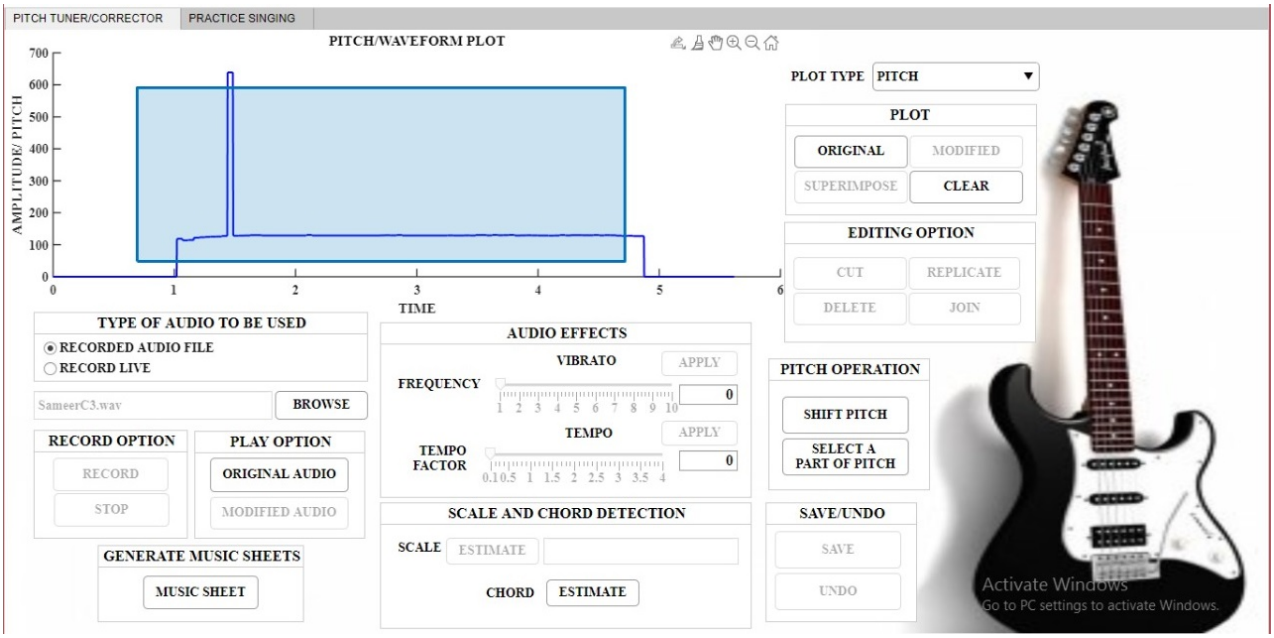


Figure 8.7: App demonstrating selection of a part of pitch

Figure 8.8: App showing the usage of editing options

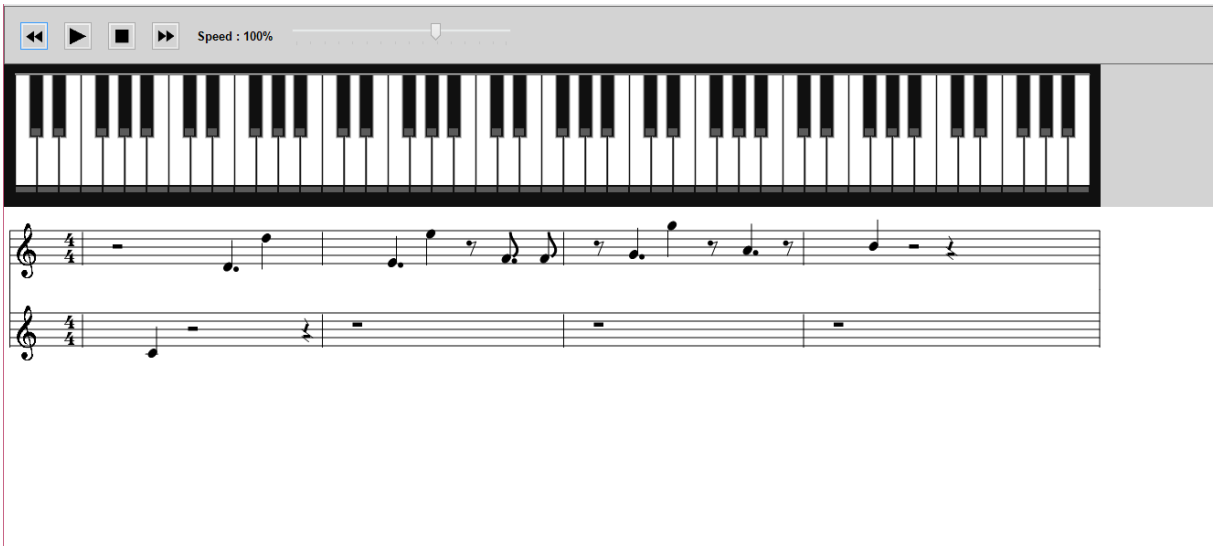


Figure 8.9: App showing generation of music sheets

in the audio and the is displayed in the textbox. For Chord detection we have ploted presence of various pitch wrt time. When **ESTIMATE** button next to Chord label is clicked a plot appears on the Axes which is chord versus time. Figure 8.6 depicts the chord detection. An audio which has A MAJOR chord was uploaded and when ESTIMATE button is pressed corresponding plot is obtained on the Axes.

When user clicks **SELECT A PART OF PITCH** button he will given an option to draw a rectangle and enclose the ROI within the rectangle. Figure 8.7 shows the drawing of rectangle and enclosing the ROI. After rectangle is drawn if it is properly selected, the ROI in pitch plot turn red which can be seen in Figure 8.5. He can perform any operations like scale detection, audio effects, editing options can be performed. Figure 8.8 shows the editing option implemented on the audio. We deleted the middle part of audio using **DELETE** button and replicated the starting partof the audio again at the end using the option **REPLICATE**. Figure 8.8 shows the superimposition plot of original audio and audio undergone editing changes.

We generated music sheet for an audio. This was done by converting the audio into **.mid** file if it was in **.wav or .mp3** file format. A .mid file format of audio is also stored in the system once the user clicks **MUSIC SHEET**. When user clicks **MUSIC SHEET** buttona window pops out throwing the music sheet of the audio used. Many options like **PLOT TYPE, SAVE, UNDO** can be explored from the app. If User wishes to see a waveform plot he can select it from the **TYPE PLOT** drop down list and use any of the plot option. User can save his modification by clicking the **SAVE** button or remove all the changes using **UNDO** button.

8.2.2 PRACTICE SINGING

In this part as stated earlier user can upload a reference audio to follow or append the table and follow accordingly. Figure 8.10 shows appending of the table with pitch duration

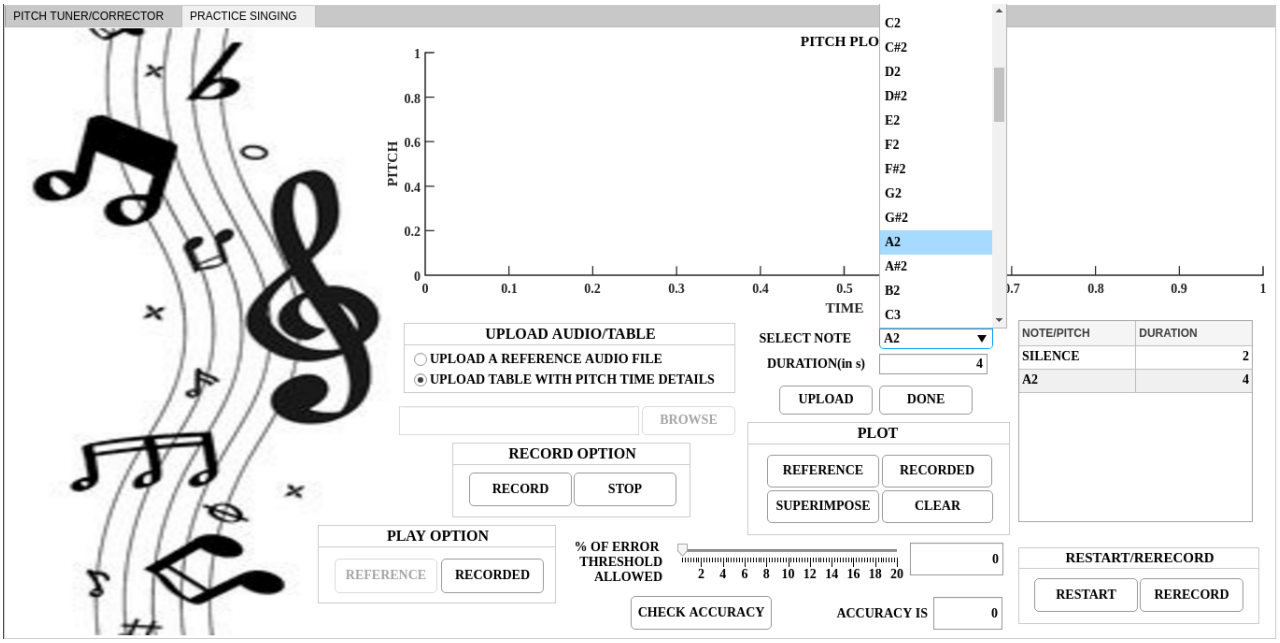


Figure 8.10: App showing appending of table as a reference audio

details. User must select suitable pitch from the **SELECT NOTE** drop down list and enter duration of that pitch in the Textbox. In our example we chose SILENCE first for 4 seconds so that user gets to know the recording environment and later A_2 note for 4 seconds. User must select corresponding Note and Duration and must press UPLOAD button and corresponding data is fed to the table. Once user is done uploading the table he needs to press the DONE button and no further changes or addition of data will not be possible.

Once user has fixed the reference audio he must record the audio. Figure 8.11 shows the recording of an audio and following a reference.User starts off his recording by pressing RECORD button. Plot has 2 lines one being the reference audio(plotted in purple) which the user is trying to follow and his recorded audio is being plotted in blue. For user convenience other than STOP button other button are disabled and user stops the recording by pushing STOP button.

Once both reference and recording is ready user can check his accuracy. First he needs to set the % OF ERROR THRESHOLD ALLOWED or rather how much error is acceptable for correctness in pitch. As the user slides the slider corresponding value is noted in the textbox and he can choose any value between 0.25% to 20%. We have chosen 20% error threshold which can be seen in Figure 8.12. This means in our example when A_2 note is present, frames are considered to correct if the frequency is in the range 88Hz to 132Hz i.e. ranging from **Reference Frequency -** %20 **of Reference Frequency to Reference Frequency +** %20 **of Reference Frequency**. After setting the slider one need to press the ACCURACY button. In our case we got an accuracy of 62.55% which means 62.55% of the frame were in the range and rest out of range.

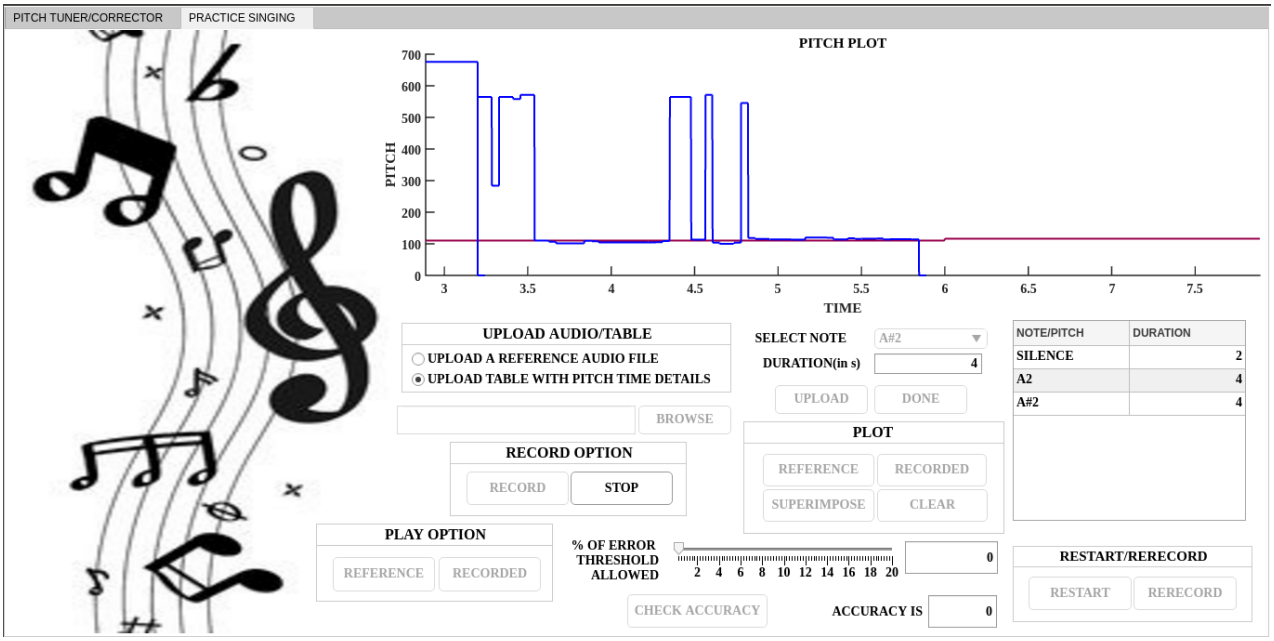


Figure 8.11: Following a reference audio and recording the same

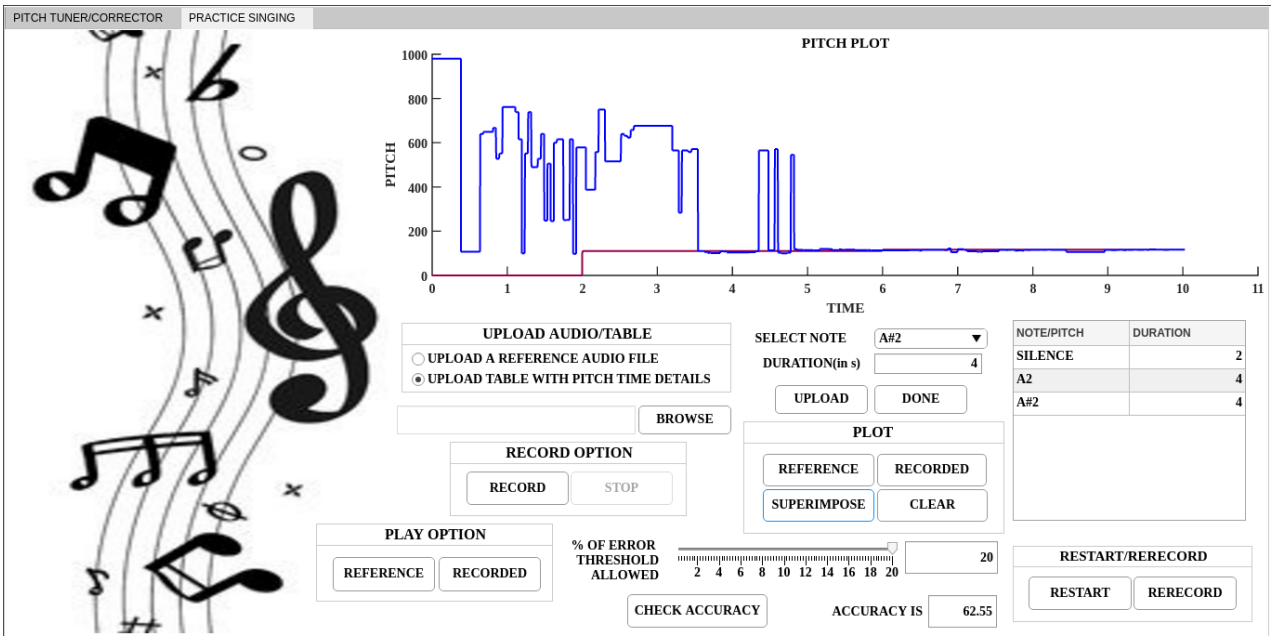


Figure 8.12: App showing plots of reference and recorded audio and percentage of accuracy

Chapter 9

Results and Discussion

We have implemented our project in MATLAB software. Functions for pitch detection algorithms, scale detection, chord detection, pitch shifting and audio effects were created and called at necessary instances. In this chapter we discuss the efficiency(in terms of computation time and error in calculation) of algorithm to detect pitch and chord, comparision of pitch shifting algorithms.

9.1 Results

9.1.1 Time Complexity/Computation Time for pitch detection algorithms

The time required for calculation of pitch was calculated using MATLAB function tic-toc. Data set for checking pitch included standard notes used in music played by various instruments .Musical notes like E6, E4, E3, G5, B2, A4, C1 corresponds to 1295Hz, 324Hz, 162Hz, 770Hz, 121Hz, 440Hz and 32Hz respectively. Instruments like Violin, Trumpet, Bass, Flute, Guitar, and Oboe were used.

Modified Autocorrelation Method

The time complexity of Modified Autocorrelation method is **O(n)** since it just involves traversing through the windowed segment once i.e just running through the audio length. Major amount of time is used in determining the peaks of the auto correlated audio.

Average Magnitude Difference Function

The time complexity of Average Magnitude Difference Function method is $\mathbf{O}(n^2)$ since it involves 2 nested traversals. Outermost traversal is required for running through the audio signal and inner traversal is for circular shifting windowed sample. Not much significant time is used for subtraction of 2 sequences.

Yin Method

The time complexity of Yin method is $\mathbf{O}(n^2)$ since it involves 2 nested traversals. Outermost traversal is required for running through the audio signal and inner traversal

is for circular shifting windowed sample. Significant time is used for multiplication of 2 sequences. Hence when compared to other methods Yin method takes more time because the above two reasons.

Cepstrum Method

The time complexity of cepstrum method is **O(n)** since it just involves traversing through the windowed segment once i.e. just running through the audio length. Major amount of time is used in determining the peaks of the processed audio.

INSTRUMENT SAMPLES	AUTO	AMDF	YIN	CEPS
VIOLIN-E6	0.267	0.769	1.373	0.125
VIOLIN-E4	0.215	0.323	0.507	0.115
TRUMPET-E3	0.307	1.396	2.432	0.134
OBOE-G5	0.250	0.643	0.968	0.120
GITAR-B2	0.279	0.991	1.748	0.137
FLUTE-A4	0.366	2.261	3.672	0.142
BASS-C1	0.327	1.057	1.855	0.130

Table 9.1: Comparison of time required for Pitch computation in seconds.

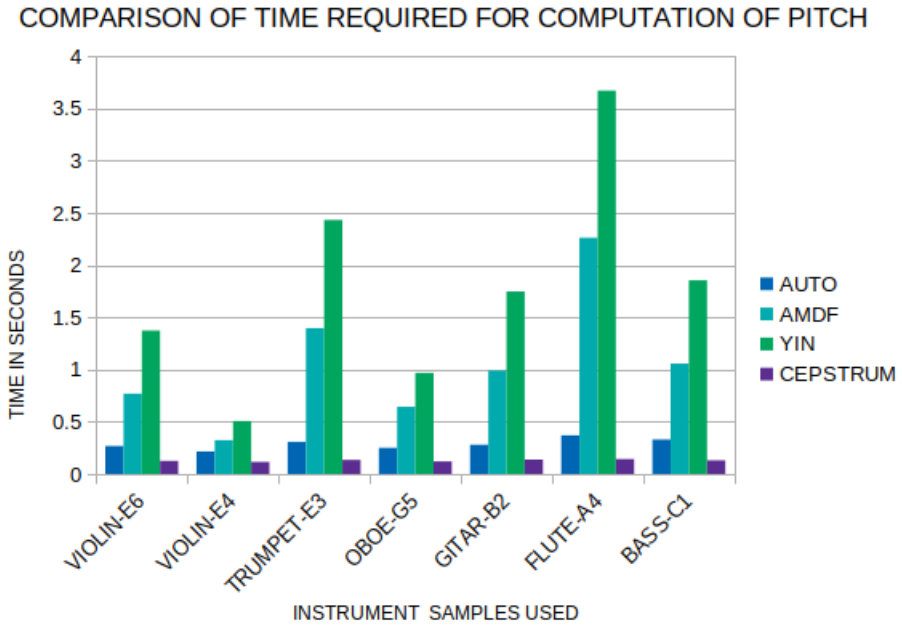


Figure 9.1: Comparison of time required for Pitch computation.

Table 9.1 and Figure 9.1 shows the amount of time required for pitch computation using modified autocorrelation (AUTO), average magnitude difference function (AMDF),

yin (YIN) and cepstrum (CEPS) method for various instrument samples. It can be seen that on an average Yin algorithm takes the maximum time and cepstrum the least time to compute pitch. Although average magnitude difference function and yin methods both have a time complexity of $\mathbf{O}(n^2)$ Yin method involves multiplication of 2 signals which takes more time for computation when compared to subtraction as in average magnitude difference function.

9.1.2 Error in calculation of pitch

Error while calculating can arise due to any reasons. We calculated the Gross Error (GER) using the Equation 9.1.

$$GER = \sum_{m=1}^{N} \frac{\mid ActualFrequency - DetectedFrequency \mid}{ActualFrequency} \times 100\% \tag{9.1}$$

Where the summation runs from 1 to total number of frames. We take absolute sum of the error to avoid cancelling. In auto correlation proper peak may not be detected due to improper thresholding as a result which wrong pitch will be detected. Similar error can be expected In case of cepstrum method. If fundamental frequency is not dominant when compared to its harmonics cepstrum method fails. This can be seen when B2 note is played using guitar. Amplitude value at 121Hz is much lesser when compared to amplitude at 242Hz (first harmonic of B1 note) in spectral domain. As a result of which we got a GER of 187%. All the pitch detection algorithms used performs good when the pitch is less than 1000Hz or low frequency components. So when E6 note is played chances of getting error is more and is evident when compared to others.

INSTRUMENT SAMPLES	AUTO	AMDF	YIN	CEPS
VIOLIN-E6	9.64	6.65	5.80	6.64
VIOLIN-E4	2.78	1.37	1.56	4.55
TRUMPET-E3	2.82	1.45	1.32	4.75
OBOE-G5	4.61	1.25	1.25	9.42
FLUTE-A4	1.79	1.02	1.21	4.62
BASS-C1	9.74	5.42	5.25	6.92

Table 9.2: Comparison of percentage error in Pitch computation

Figure 9.2 and Table 9.2 shows comparison of error percentage for multiple pitch detection algorithms. It can be seen that on an average Yin and average magnitude difference function algorithms has maximum accuracy or low error and cepstrum and autocorrelation method has least accuracy or maximum error in pitch computation.

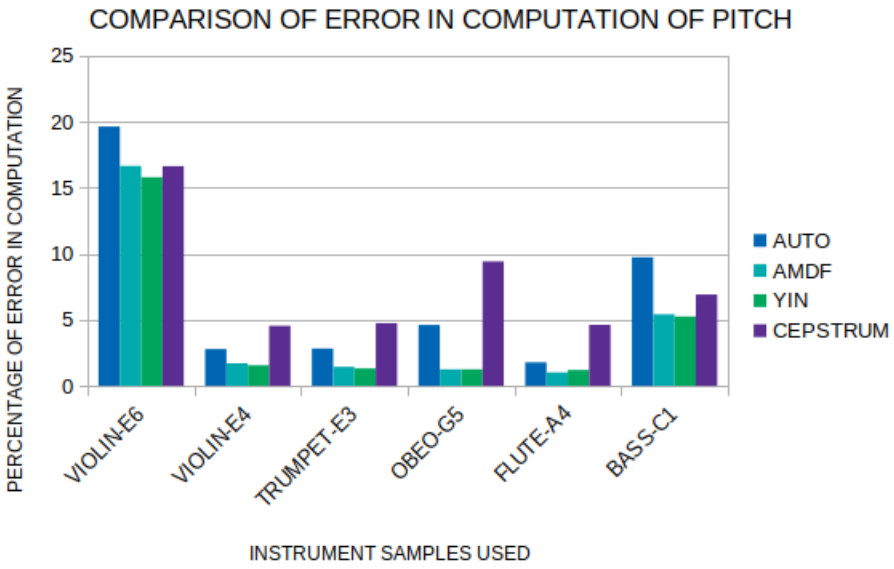


Figure 9.2: Comparison of percentage error in Pitch computation

9.1.3 Accuracy of various Machine Learning algorithms for chord detection

Most techniques proposed in the literature for chord recognition do not use machine learning methods. We have chosen a simple but powerful method to recognise chords.

We modelled the extracted features on several machine learning algorithms like SVM, KNN, logistic regression, random forest, neural networks etc. On training the model and comparing their test accuracies, we chose a feed-forward neural network using a classical gradient descent algorithm with a negative log-likelihood as cost function.

k-NN

K-Nearest Neighbours algorithm (k-NN) is a non-parametric method used for classification and regression. In either cases k-closest training examples are considered in the feature space. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbours, in our KNN, a neighbour factor of 7 was chosen as the cluster size to get the best performance.

Logistic regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labelled ”0” and ”1”. For the logistic

regression units, Relu was used as the activation function and Adam as optimizers to obtain non linearity is prediction classes based on inputs.

SVM

Support-Vector Machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of gap on which they fall.

Naive Bayes

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernel density estimation and achieve higher accuracy levels. Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Decision trees

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. It is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Evaluation

All the models were trained with the extracted PCP feature set and tested on some sample files chosen. Hyperparameters were also tuned for each classifier to achieve maximum performance.

A negative sample should not be considered as positive sample and this ability of the classifier is given by precision. All the positive samples are found with the ability of the classifier defined by recall. A weighted harmonic mean of the precision and recall is given by F1 score. Using these evaluation metrics, the results for all classifiers were consolidated as show in Table 9.3.

MACHINE LEARNING ALGORITHM	ACCURACY	PRECISION	RECALL SCORE	F1 SCORE
LOGISTIC REGRESSION	93.75	94.44	93.75	93.63
KNN	95.21	96.33	96.22	96.18
SVM	94.82	93.77	92.43	90.134
NAIVE BAYES	91.79	92.32	91.78	91.61
DECISION TREE	91.98	92.41	91.98	91.86
RANDOM FOREST	94.54	93.48	92.16	92.30
NEURAL NETWORK	96.79	97.12	96.88	95.89

Table 9.3: Comparison of accuracy in Chord detection using various ML algorithms

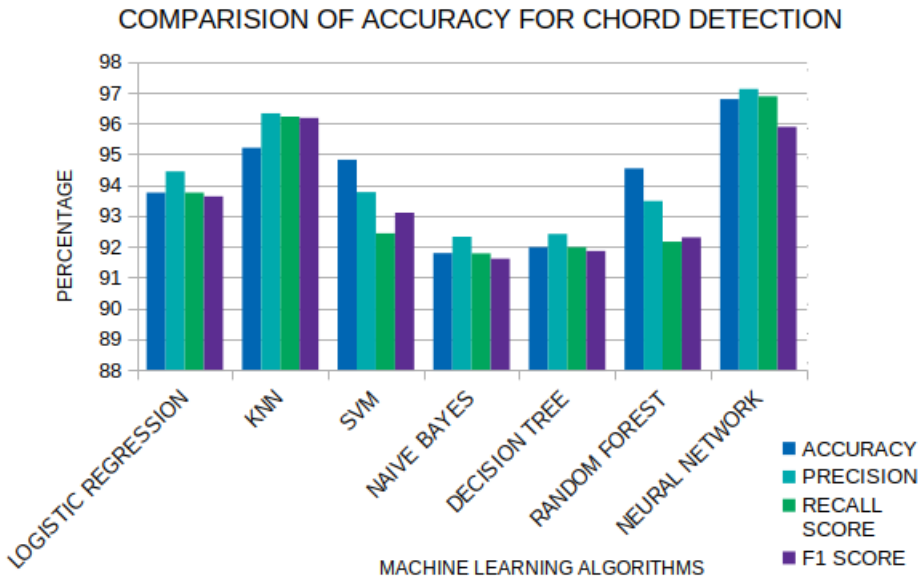


Figure 9.3: Comparison of accuracy in Chord detection using various ML algorithms

From Table 9.3 and Figure 9.3, we see that the Neural Network offers the best performance. This is because of the huge dataset we used for training and tuning made to the hyperparameters, increasing the performance of the model drastically.

A confusion matrix was also obtained to visualize the number of true and false labels from the predicted samples of the test dataset. Where the rows represent the True Label and columns represent the Predicted value.

	A	Am	Bm	C	D	Dm	E	Em	F	G
A	43	1	0	0	0	0	0	0	0	0
Am	0	42	0	0	0	0	0	0	0	0
Bm	0	0	41	0	0	0	0	0	0	0
C	0	0	0	37	0	0	0	0	0	0
D	0	0	0	0	28	10	0	0	0	0
Dm	0	0	0	0	3	37	0	0	0	0
E	0	0	0	0	0	0	42	0	0	0
Em	0	0	0	0	0	0	2	32	0	0
F	0	1	0	0	0	0	0	0	43	0
G	0	0	0	0	0	0	0	0	0	38

From the above confusion matrix, we see that most of the samples have been categorized correctly. The model performance can further be improved by implementing newer architectures of neural networks like LSTM or reinforcement learning approaches. Transfer learning has also been used widely nowadays due to their robustness from previously trained parameters and are found to be really suitable for speech processing and music related applications.

9.2 Discussion

9.2.1 Pros and Cons of different Pitch detection algorithms

Modified Autocorrelation method

Advantage:

⋆ This method of pitch detection is easy and has a faster computation.

⋆ Easy to understand the concept since its mathematical modeling is easy.

Disadvantage:

⋆ The level of peak to be chosen is challenging. Peak picking may give all the peaks before the first local maxima, but that is not the actual peak. Hence we make use of Adaptive thresholding Autocorrelation function.

⋆ Error in calculating the pitch is moderate.

Average Magnitude Difference Function

Advantage:

⋆ Mathematical modelling behind this method is too simple.

⋆ Error in calculation of pitch is minimum

Disadvantage:

⋆ Time required for computation of pitch is more.

⋆ It has a harder harware implementation.

Yin method

Advantage:

⋆ This is comparitively an easy method that can be implemented with low latency and the efficiency is also more, and this can be extended to handle other forms of aperiodicity that occur in different applications.

⋆ Error in calculation of pitch using this algorithm is least.

Disadvantage:

⋆ Time required for computation of pitch is more.

⋆ It has a harder harware implementation.

Cepstrum method

Advantage:

⋆ Cepstrum analysis, once understood is a very simple way to estimate spectral component since it does not deal with phase.

⋆ Time computation is moderate since it involves FFT and IFFT repeatedly.

Disadvantage:

⋆ Performing FFT and IFFT are computationally expensive and there is a chance of losing some data.

⋆ Cepstrum analysis is essentially a low-pass filter and filters the spectral components which is averaging the spectral components.

⋆ For cepstrum method the fundamental must be dominant compared to its harmonics else chances of getting error are high. This is explained in the GER part.

9.2.2 Comparision between Phase Vocoder and Harmonic Model for Pitch shifting

In Chapter 6 two important, highly used and efficient pitch shifting algorithms were discussed. In this section, the comparison of the two algorithms on different samples are enumerated.

Basis of comparison – Timbre

Classification of a pitch shifting algorithm for a particular audio sample into good and bad depends mostly on the timbre of the modified audio, because after all we are changing the pitch to make the audio sound better. Timbre is described in Section 3.3.

Methodology

To obtain the most suitable algorithm for a particular type of audio, the modified audio was subjected to timbre testing by comparing the timbre with the original audio. In this test, the harmonic spectra of many audio samples that included vocal and instrumental were compared before and after pitch shifting. Quantitatively, the following were compared:

- ⋆ Ratio of spacing between the harmonics in the spectrum
- ⋆ Magnitudes of the harmonics
- ⋆ Envelope of the harmonics.

The timbre comparison of the audio samples before and after modification resulted in the following:

From the Figure 9.4 and Figure 9.5, it is clear that phase vocoder works better for instrumental music samples while the harmonic transformation works well for vocal music samples as compared to instrumental music samples.

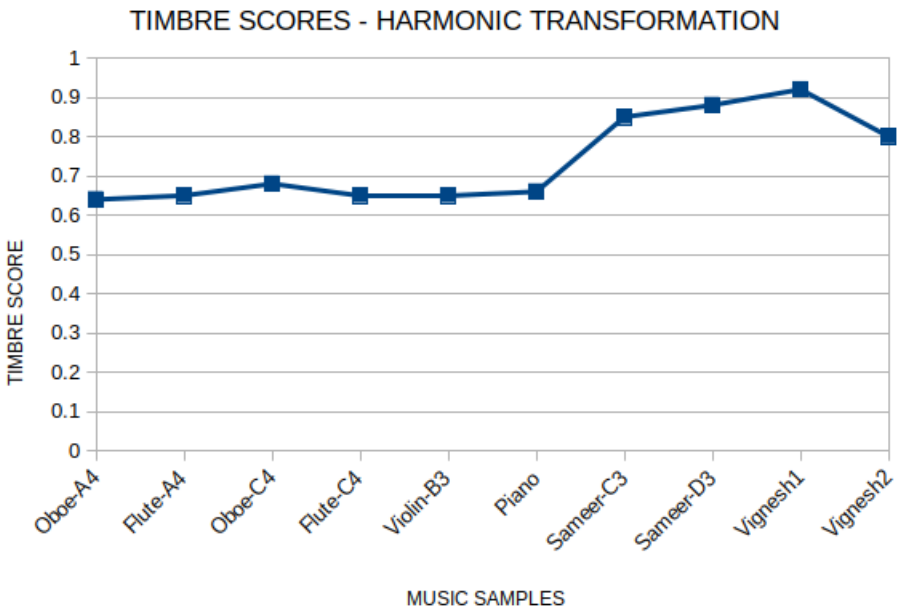


Figure 9.4: Timbre score for Harmonic Transformation

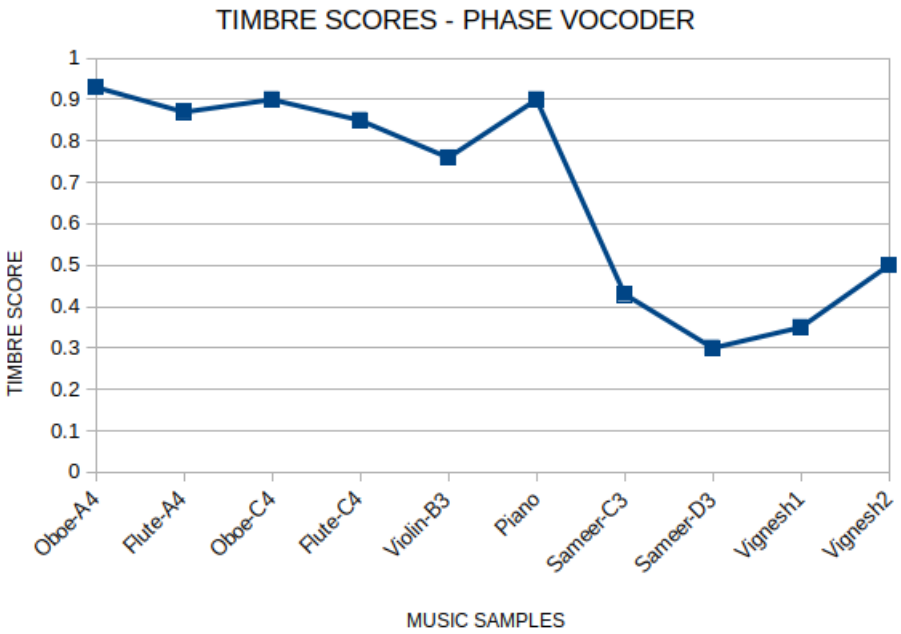


Figure 9.5: Timbre score for Phase Vocoder

Chapter 10

Conclusion and Future scope

10.1 Conclusion

In the previous chapter we saw the advantages and drawbacks of various pitch detection algorithms. So we can conclude that we can make use of appropriate pitch detection algorithm for appropriate application. Say for live pitch plot and recording we can make use of algorithms like autocorrelation and cepstrum method where we must have less computation time and moderate error. For application like pitch shifting, scale detection, chord detection where the error has to minimum irrespective of time computation we make use of algorithms like Yin and AMDF.

In chord detection we discussed 2 methods chromagram method and machine learning approach both seemed to be efficient in terms of time and accuracy. When compared to both machine learning approach stands a little high but both are of equal prominance and efficient methods.

The method we have used in scale detection is pretty straightforward and does not include much of a technical operations. Note seperation and later comparision of notes detected is pretty easy and uncomplicated.

Pitch shifting is the key in our project. Phase vocoder and harmonic model are the two proposed models. Phase vocoder has a good efficiency for all instrumental samples but considering pitch shifting for human voice phase vocoder is not prefered. When Human voice pitches are shifted using phase vocoder it results in robotic voice (artificially synthesized) which is also known as **Donald Duck effect**. So Harmonic model is more prefered as chances of preserving the formant frequencies is high there by not destroying these formant frequencies and avoiding Donald Duck effect. Harmonic model is applicable for both instrument samples as well as human voices but the only criteria for synthesis of audible (without much of damage to formant frequencies) sound is that the difference between the original pitch and shifted pitch should not be large.

10.2 Future scope

Although the project has a high success rate there are a few drawbacks which we need to correct and try implementing them in future. Few of the drawbacks are listed below:

- ⋆ Pitch detection algorithms like autocorrelation and cepstrum has more error compared to methods like Yin and AMDF, we need to reduse the errors by selecting appropriate peaks in the post process. Error in determining the pitch must be minimum since many operations like pitch shifting, chord shifting, scale shifting are the resultant of the pitch detected.

- ⋆ Since we haven‘t used technical approach towards scale detection chances of error is pretty high so we need to explore more effective scale detection algorithms and use the same.

- ⋆ More pitch shifting models must be discovered which has the ability to preserve the formant frequencies even if the difference between the original pitch and shifted pitch is large.

- ⋆ As of now pitch correction is manual i.e. a person detects the fault in the pitch and corrects it,we need to build a system which detects the fault in the pitch autonomously and correct the pitch to appropriate levels without any human inference.

- ⋆ MATLAB isn‘t a free software so many people cannot access the product we developed, so we need to convert it an open source like a website or an android / iOS application.

References

[1] Oppenheim, A. V., Schafer, R. W. and Buck, J. R. (1999). Discrete-Time Signal Processing(Prentice Hall, New Jersey).

[2] Rabiner,L.R and R.W.Schafer "Digital Processing Of Speech Signals"

[3] Rabiner, L.R. (1977),"On the Use of Autocorrelation Analysis for Pitch Detection," IEEE Trans. Acoust., Speech, Signal Process. 25 , 24-33

[4] De Cheveigné, A., Kawahara, H. (2002), "YIN, a fundamental frequency estimator for speech and music," J. Acoust. Soc. Am. 111 , 1917-1930.

[5] Noll, A. M. (1967). "Cepstrum pitch determination," J. Acoust. Soc. Am. 41 , 293-309.

[6] Oudre, Laurent & Grenier, Yves & Févotte, Cédric. (2009). Chord recognition using measures of fit, chord templates and filtering methods. 9-12. 10.1109/ASPAA.2009.5346546.

[7] Oudre, Laurent, Yves Grenier and Cédric Févotte. "Template-based Chord Recognition : Influence of the Chord Types." ISMIR (2009).

[8] J. Osmalskyj, J-J. Embrechts, S. Piérard, M. Van Droogenbroeck "NEURAL NETWORKS FOR MUSICAL CHORDS RECOGNITION", Actes des Journées d'Informatique Musicale (JIM 2012), Mons, Belgique, 9-11 mai 2012.

[9] David Temperly (1999), " The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered", Music Perception, Vol.17, No.1, 65-100.

[10] Flanagan J.L. and Golden, R. M. (1966). "Phase vocoder". Bell System Technical Journal.

[11] Serra, Xavier. An Environment for the analysis, transformation and resynthesis of music sounds. Stanford: University of Stanford, Center for Computer Research in Music and Acoustics; 1988.

[12] Serra,X and Smith.J "Spectral Modeling Synthesis: A Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition"

[13] Audio Signal Processing for Music Applications by University at Pompeu Fabra of Barcelona and Stanford University

B.M.S COLLEGE OF ENGINEERING – 560019

DEPARTMENT OF TELECOMMUNICATION ENGINEERING

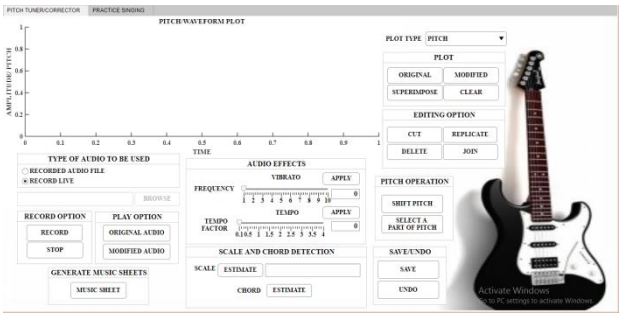
TITLE	**Pitch Detection and Pitch Correction**		
STUDENT (NAME, USN, MOBILE NO.)	KHOUSHIKH S	1BM16TE018	8553954933
	PRAJWAL S RAO	1BM16TE029	8277399317
	R ADVAITH ANANTH KRISHNAN	1BM16TE031	9342976091
	SRIRAM R	1BM16TE046	8197401142
GUIDE	Dr. Balachandra K		
SIGNATURE OF HOD			
VENUE	Department of Telecommunication Engineering, BMSCE Bangalore- 560019		
PROJECT COMMENCEMENT DATE	27/01/2020	**PROJECT COMPLETION DATE**	25/06/2020

ABSTRACT

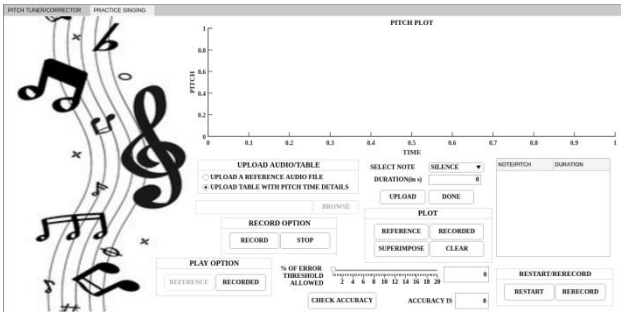
This project is aimed to develop a software that is used in pitch detection and correction of musical instrument samples and human voices. Pitch correction is a widely used and controversial form of music editing. Audio Signal Processing can be defined as a field of engineeringthat focuses on computational methods for intentionally altering sound.A chord, in music, is any harmonic set of pitches consisting of multiple notes that are heard as if sounding simultaneously.Chords and sequences of chords are frequently used in modern Western classical music. For live recording or as studio tools audio effects play an important role.

Pitch correction is an electronic effects unit or audio software that changes the intonationof an audio signal so that all pitches will be notes from the equally tempered system. Classification of a pitch shifting algorithm for a particular audio sample into good and bad depends mostly on the timbre of the modified audio.

PHOTOGRAPHS



Pitch Tuner/Corrector



Sing with Reference

CONCLUSION

Pitch detection and correction are extremely challenging tasks as they should not disrupt the music itself. Pitch correction for human voices and musical instruments using Harmonic model yields good results compared to phase vocoder technique. Implementation of Chord detection and scale detection using template matching approach. Adding audio effects to music files and generation of MIDI files. The biggest challenge is to detect and correct the pitch in real time which would be very useful in live concerts. Detection of chords using machine learning approach is quite challenging since we need to get datasets from different instruments.

PROGRAM OUTCOME

Our project satisfies PO-1, PO-2, PO-3, PO-4, PO-5, PO-6, PO-7, PO-8, PO-9, PO-10, PO-11 and PO-12.

ABOUT THE PROJECT TEAM

KHOUSHIKH S is pursuing graduate degree in Telecommunication Engineering from BMS College of Engineering. His areas of interests are Signal processing and electronics.

PRAJWAL S RAO is pursuing graduate degree in Telecommunication Engineering from BMS College of Engineering. His areas of interests are Mathematical modelling, Electronics and Signal Processing.

R ADVAITH ANANTHKRISHNAN is pursuing graduate degree in Telecommunication Engineering from BMS College of Engineering. His areas of interests are Electronics and Signal Processing.

SRIRAM R is pursuing graduate degree in Telecommunication Engineering from BMS College of Engineering. His areas of interests are Machine Learning and Signal Processing.

APPENDIX I

SOFTWARE PLATFORM USED

The platform that is chosen for developing this project is MATLAB. MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. MATLAB is a very helpful tool for audio processing applications. It also is popularly used to implement Machine Learning problems.

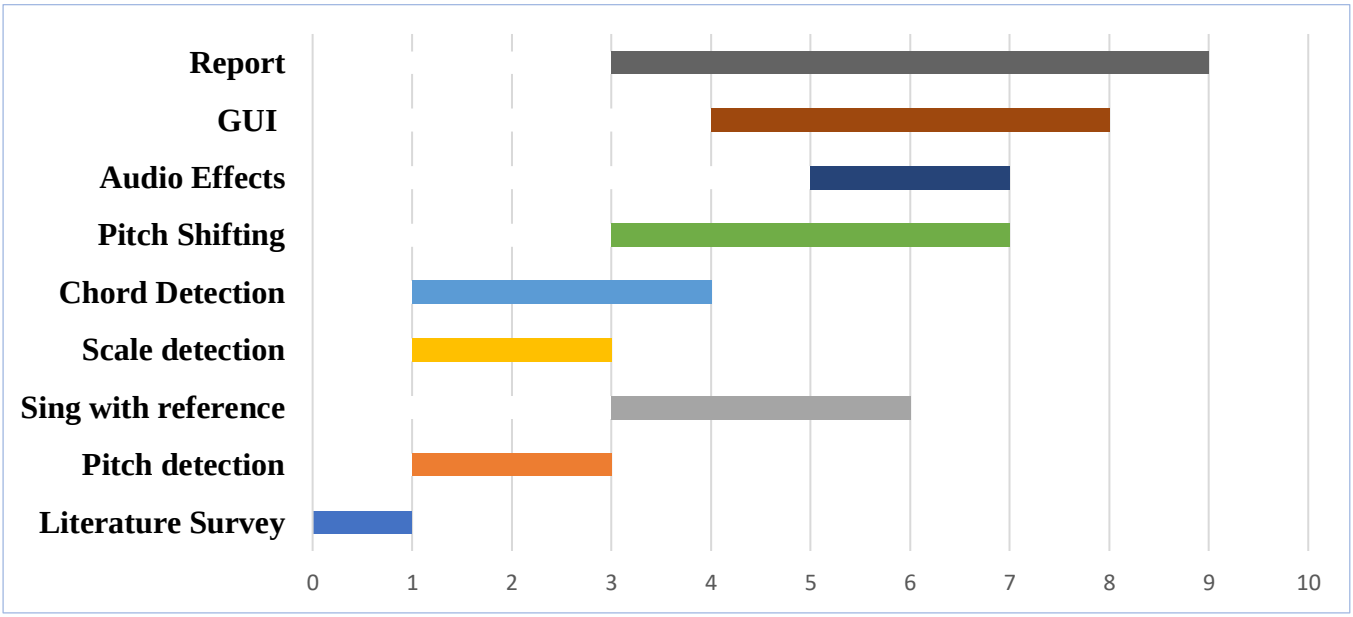
Another important feature in MATLAB is the ability to develop apps with Graphical User Interface. It is possible to develop this interface from the written code which enables implementing the program code in an easier way.

APP DESIGNER

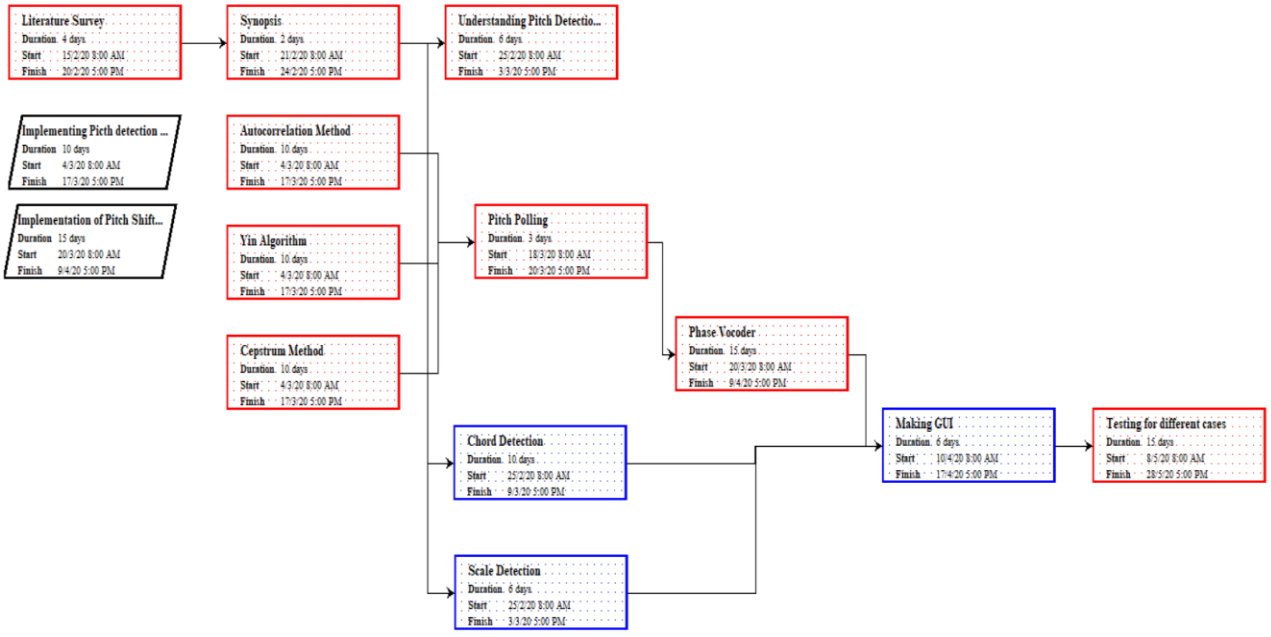
App designer is an application designer that is used to create professional apps in MATLAB. The user interface can be created and a code can be written to control the functions and working of the interface. It has a component library containing a huge variety of components like buttons, axes, drop-down boxes, sliders, radio buttons and many more. Each of these can be coded to perform the required function. The code for the action of each component in the user interface is written in their respective call-back functions. These push-button call-back functions are executed when the user clicks the respective component in the user interface when the app is running. The different varieties of components available in App Designer library are used in such a way that the app is user friendly and has an optimized user interface.

APPENDIX II

GANTT CHART



NETWORK DIAGRAM



APPENDIX III

ENGINEERING CONCEPTS USED

1. Discrete Signal Processing

It is an important and essential study in the field of electronics and telecommunication engineering. This is vast topic with does a deep dive into processing of signals. In the signal processing domain, we come across to types of signals- Continuous-Time Signals and Discrete-Time Signals. A Continuous-Time signal is continuous both in time and amplitude and is usually represented by a continuous variable. A Discrete-Time Signal on the other hand is defined at discrete intervals of time.

Various operations can be performed on the time domain signal. These operations include, Signal Shifting, Scaling, Reversal, Differentiation, Integration, Convolution etc. Along with these operations various system properties also come into place like Causal and on- Causal Systems, Linear and Non-Linear Systems, Time-Variant and Time-Invariant System, Stable and Unstable System, Static and Dynamic System etc. The time domain signal can be applied with various transforms like the Z-Transform, Discrete Fourier Transform, and Fast Fourier Transform etc.

Pitch detection using modified auto correlation, AMDF, Yin and Cepstrum methods can be used.

Modified Fourier series like Harmonic transformation can be used for shifting the fundamental frequency of a signal and also changing the tempo of the audio signal.

With all the above-mentioned processes, signal processing is of great importance in the electronics and telecommunication industry.

2. Machine learning

Machine learning concepts such as PCA, logistic regression, Soft max layer, ReLU (Rectified Linear Unit), Adam optimizer were used in our project.