MELODY EXTRACTION FROM POLYPHONIC MUSIC SIGNALS BASED ON SYNCHROSQUEEZED WAVE PACKET TRANSFORM

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

(With specialization in Instrumentation and Signal Processing)

By

BHADRECHA BHANESH BABUBHAI

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)
June 2016

CANDIDATE'S DECLARATION

I hereby declare that this thesis report entitled **Melody Extraction from Polyphonic Music Signals Based on Synchrosqueezed Wave Packet Transform**, submitted to the Department of Electrical Engineering, Indian Institute of Technology, Roorkee, India, in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Electrical Engineering with specialization in Instrumentation and Signal Processing is an authentic record of the work carried out by me during the period June 2015 through May 2016, under the supervision of **Dr. Ambalika Sharma** and **Dr. P. Sumathi**, Department of Electrical Engineering, Indian Institute of Technology, Roorkee. The matter presented in this thesis report has not been submitted by me for the award of any other degree of this institute or any other institutes.

Date:

Place: Roorkee

BHADRECHA BHANESH

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

Dr. Ambalika Sharma
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology Roorkee

Dr. P. Sumathi
Associate Professor
Department of Electrical Engineering
Indian Institute of Technology Roorkee

ABSTRACT

Since the development in digital technology, music has a lead role in the leading technological evolution. Music can be thought as basic human need. As huge amount of data is required to store and transfer audio signal, challenges arise as how efficiently one can store audio signal without almost no changes in its quality. With the use of cloud backup one can have own music collection, but this leads to wastage of bandwidth as the same music content can repeat for persons. So, to index and search these vast amounts of data becomes a challenging task. In melody extraction, the melody (pre-dominant fundamental frequency of a music signal) is extracted from the music which then gives salient applications like indexing and searching music content, query by humming (QBH), voicing detection, voice separation, genre classification (like Rock, Pop, Indian etc...), music teaching (by extracting different beat frequencies), games, security systems, karaoke system and more. So many melody extraction algorithms have been developed, but most of the algorithms depend on a type of music where for specific music it can work effectively. Melody extraction can be vocal or instrumental. Vocal melody extraction algorithms extract the human voice (Voicing detection) whilst instrumental includes detection of particular instrument sound like tabla, guitar, flute etc. As synchrosqueezed wave packet transform gives very accurate time-frequency representation, the need for extra steps to calculate melody is eliminated and directly giving final melody. Proposed melody algorithm shows an improved accuracy compared some existing methods.

Acknowledgements

I would like to express my deep sense of gratitude and sincere thanks to my guides **Dr. Ambalika Sharma** and **Dr. P. Sumathi**, Department of Electrical Engineering, Indian Institute of Technology Roorkee, for their valuable guidance and support. I am highly indebted to them for their encouragement and constructive criticism throughout the course of this project work. In spite of their hectic schedule, they were always there for clarifying my doubts and reviewed my dissertation progress in a constructive manner. Without their help, this thesis would not have been possible.

I would like to extend my gratitude to my friends and classmates for their support throughout the past two years.

I owe a lot to my family who unremittingly supported me during my years of study. I would like to pay high regards to my parents for being a true motivator throughout my life. Their relentless backing made this work possible and I would like to dedicate it to them.

Also I would like to thank Yogesh Sariya and Ksh. Milan, Research Scholars, Department of Electrical Engineering for their help and support in understanding the project.

BHADRECHA BHANESH BABUBHAI

Contents

List of Figures

List of Tables

Abbreviations

STFT	**S**hort **T**ime **F**ourier **T**ransform
SSWPT	**S**ynchro **S**queezed **W**ave **P**acket **T**ransform
MIR	**M**usic **I**nformation **R**etrieval
TFR	**T**ime **F**requency **R**epresentation
std	**St**andard **D**eviation
QBH	**Q**uery **B**y **H**umming
SNR	**S**ignal **N**oise **R**atio
DFT	**D**iscrete **F**ourier **T**ransform
DSP	**D**igital **S**ignal **P**rocessing
OPA	**O**verall **P**itch **A**ccuracy
RPA	**R**aw **P**itch **A**ccuracy
BPF	**B**and **P**ass **F**ilter
IIR	**I**nfinite **I**mpulse **R**esponse
RCA	**R**aw **C**hroma **A**ccuracy
VU	**V**oicing **U**nvoicing
HPF	**H**igh **P**ass **F**ilter

To my teachers and my parents

Chapter 1

Introduction

Music is everywhere around us. "Music expresses which cannot be put into words and which can not remain silent (Victor Hugo)". Music information retrieval is related to music technology, cognitive science, signal processing and computer engineering. A huge demand in music signal processing has arisen due to technological advancements in music and so music information retrieval (MIR) has become a separate field of study. Most MIR systems work in two steps : 1. Extract intelligent music information 2. Use this information in broad area applications like searching, storage, security, compression and analysis. Music features can be extracted in three levels based on their mathematical complexity. Low level (spectrum coefficient), medium level (melody and rhythm) and high level (genre, album and mood etc.). The most fundamental frequency of any signal is called as pitch of that signal. The sine wave of frequency 50 Hz can be heard with the proper beeps tune. So this 50 Hz is pitch frequency for sine wave. For a non-stationary signal like music its pitch varies at every time fraction. Pitch also represents a quality of sound perceived by humans. Arranging each fractional pitch in a timely order gives a smooth melody. Melody plays big role in human life, starting with an infants first cry to daily human social communications [1]. We often reproduce a melody of a song in our daily life. Most people remember singers voice to identify particular song and they try to replicate the same voice in own humming style [14]. As melody directly resembles the voice of any person, melody extraction is the first step for many music information retrieval applications. Most algorithms developed so far depend on the type of music. Building a robust melody extraction algorithm independent of any music type is a challenging task.

Most algorithms developed so far are based on a single resolution transforms like Short Time Fourier Transform (STFT). In the proposed scheme synchrosqueezed wave packet transform (SSWPT) is implemented which is multi resolution transform. Multi resolution transforms give a more detailed analysis for time-frequency representation of music signals which results in improved accuracy for melody extraction algorithms.

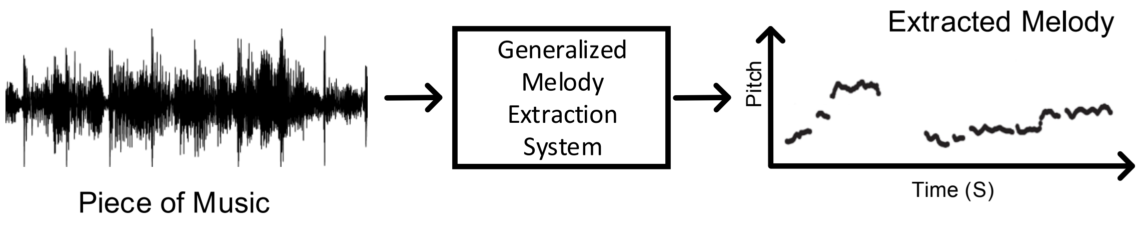


FIGURE 1.1: Generalized melody extraction system

Figure 1.1 shows a generalized melody extraction system. Given a music signal as an input, melody extraction system will give the melody as an output. We want such a system that can extract automatic melody from music signals. Melody extraction can be done from monophonic audio signal or it can be from polyphonic music signal. As monophonic music signal itself represents one indirect form of melody, the main challenge remains only in extracting melody from polyphonic music content. There can be on line and off line melody extraction system. On line melody extraction system can deal with large amount of data by using the cloud computing. Using cloud computing and cloud storage, indexing and storage of millions of songs become very easy. Melody extraction plays key role in applications like query by humming (QBH), karaoke systems, voice based security systems, music transcription, audio fingerprinting, detection of cover songs, genre classification, pattern analysis etc.

1.1 Scope of melody extraction

Text based searching and indexing of data is very easy compared to the database which is a feature based but it can not give detailed analysis. One can design feature based indexing and storage where data is managed based on its features. For example music files all over the internet can be indexed and stored based on music type, genre etc. Music and video signals acquire huge amounts of storage space. For a cloud computing system efficient use of storage is an essential need. Text based indexing of data creates redundancy in storage and so wastage of large amounts of data. So the solution is to index this data based on their features. Melody is one of those features of music signal. As there are more features for any music signal melody is the key feature that helps in deriving other features. In query by humming systems relevant music to the queried one is searched based on music features. In QBH systems, user hums particular part of music and that respective music will be the searched output. In music teaching extracted melody is used to derive features like pitch, rhythm, score and beats to train new musicians. In music plagiarism melodies of songs are compared to check their originality. Source separation is used to separate singing voice and music. Source separation is also used to extract particular source from the different sources played at the same time. When a user demands an exact match for the music, then audio fingerprinting is used. Pattern analysis is also exploiting the use of melody to extract patterns.

There is a major difference between melody extraction and source separation. As melody extraction does not separate vocal tract from the music signal, but it simply extracts main melody line which is the fundamental frequency of the singer. Melody extraction helps in the source separation algorithm. The goal of source separation is to separate any particular voice among multiple voices in any music signal.

1.2 Melody extraction algorithms

Melody extraction algorithms can be mainly classified into two types, 1. time domain algorithms and 2. Frequency domain algorithms. Time-domain algorithms detect fundamental frequency from the periodicity of the music signal. A simple method is to use zero crossing rate of music signal and by calculating number of time the signal has crossed zero we can have period of the signal and inverse of that period gives fundamental frequency. ZCR based methods are more sensitive to noise so not used in practice. Some methods use autocorrelation function to extract the melody. For a music piece its autocorrelation is calculated and maximum of autocorrelation gives the fundamental frequency f_0. ACF based methods mostly give errors where frequency is detected as half of the fundamental frequency. YIN algorithm [13] uses modified ACF to detect pitch of the music signal. Some methods employ human auditory system [3] model and then use this model in development of algorithm.

Frequency domain algorithms are based on single resolution and multi resolution spectral transform. Some uses only FFT to detect the pitch. Some algorithms also use cepstrum based techniques which uses inverse Fourier transform of log power spectrum. Some algorithms use harmonic summation based techniques where first peak of the spectrum is calculated and then harmonics are calculated based on this detected peak. Harmonic summation is calculated based on exponential rule to give less weightage to harmonics. Some source separation methods also employ frequency domain approach where the entire spectrum is divided into pieces and based on some statistical rules each piece is determined as voiced piece or unvoiced piece. Unvoiced pieces are made zero and finally inverse transform is calculated to get back separated source.

1.3 Basic terminologies related to melody extraction

1.3.1 Pitch

Pitch is the most fundamental frequency of the signal. So basically pitch is measured as inverse of the time period of the signal which is $f_0 = 1/T_0$. More generally pitch is defined as the number of oscillations per second. Most time domain algorithms use the correlation function to determine the fundamental frequency of the signal. For most applications, fundamental frequency is estimated by fast Fourier transform (FFT).

1.3.2 Melody

Primary purpose of this thesis is to extract melody from given music. So we must know what a melody is. After listening to a particular music if listener is asked to reproduce the same music in humming form then it is melody. Melody is closely related to pitch. Melody can be defined as pitch values arranged in timely sequence. Melody can be extracted for human voice or instruments like piano, violin, guitar etc. For every spoken word the pitch is different and it varies from person to person. For most humans the vocal pitch frequency range is 150 Hz to 1750 Hz [1]. Lower values of pitch are created by instruments like bass and higher pitches are created by instruments like guitar[1].

1.3.3 Octave

In music theory one octave is double the change in frequency when the base in logarithm is 2. Any music is created base on standard pitch notes. The difference between two pitch is one octave. For example if frequency of one note is 440 Hz then the next note will have frequency of 880 Hz that is one octave ahead of previous note. The lowest octave frequency range is from 16 Hz to 32 Hz.

1.3.4 Semitone

Semitone is half the value of tone or called the smallest music interval. After simulating any melody extraction algorithm the detected pitch should be half semitone away from the original pitch else it is considered as false pitch value.

1.3.5 Monophonic music

When any individual instrument is played without any human voice it creates monophonic music. Music played only by guitar can be considered as monophonic music. A single person singing a song without any external instrument is a monophonic type singing. Most of people in daily life used to hum for their favorite song and this hum is also type of monophonic music. An opera singer sings monophonic music.

1.3.6 Polyphonic music

Music created by playing simultaneously multiple instruments makes polyphonic music. polyphonic music signal contains multiple melodies. Melody extraction is totally selective process.

1.4 Literature survey

So many melody extraction algorithms have been proposed till date. Most algorithms use the pre-processing part in which the audio signal is first filtered with some techniques like band pass filtering and Equal loudness filtering. After pre-processing part pitch detection part is carried out where either time domain or frequency domain operations are employed to detect pitch candidates. Finally post-processing part smooths these pitch values to remove sharp transitions. Methods that use filtering of audio signals at beginning of algorithm mostly employ pre-processing part. General overview of some melody extraction algorithms is given here.
Equal loudness filter (ELF) employed in [1] is a type of band pass filter (BPF) that enhances frequencies more perceptual to human ear. After Equal loudness filter, STFT is

computed followed by IF peak correction based on instantaneous frequency calculations. Finally harmonic summation calculates strong melody candidates. From all melody candidates single pitch is detected by determining statistical parameters. Comparison of most melody extraction techniques are presented in [2]. Harmonic cluster based methods [3] used STFT and log spectrum followed by IFT. Each salient feature is determined and stored as a comb which finally used to decide the pitch values. Comparison of time and frequency based algorithms is presented in[4]. Tandem algorithm used in [5] roughly estimates melody lines using spectral transform and then accurates these lines by harmonicity and temporal continuity.

DFT based salience peaks calculation is carried out in [6]. TWM errors are calculated to decide main melody and finally voicing detection is used to remove unvoiced part. REPET (Repeating Pattern Extraction Technique) discussed in [7] finds a small repeating frame in entire music and then removes similar frames from the music file to remove unvoiced part. Beat pattern is used to estimate the repeating pattern. Wavelet transform based pitch detection is available in [8]. Cepstrum based pitch detection is presented in [9] where voicing/unvoicing is based on statistical parameters like zero crossing rate (ZCR), median, mean and short time energy measurement. In [9] melody extraction is performed in two levels. First pass is for voicing detection and second pass for pitch calculations. Klapuri [10] used harmonicity and spectral smoothness techniques to estimate melody lines. Several frequency regions are separated using MFCC filter bank. PREFEST algorithm[11] calculated bass line and vocal melody line separately using probabilistic methods. In [12] melody extraction is first determined by STFT and then using spectral peak and sub-harmonic summation pitch values are extracted. Method in [12] is similar to [14]. Tian cheng [14] used onset and offset detection of music signals to detect unvoiced part which is then removed making this unvoiced frames to zero. Sinusoidal extraction and salience function design methods are explained in [15] where melody is selected by determining multiple parameters. Application of singing voice separation is presented in [16] where a large database is tested for voicing detection[23]. Modified pitch detection method [17] calculates pitch based temporal correlation and spectral similarity measure. Application of synchrosqueezed wave packet transform is to decompose various signals [18].

1.5 Goal of dissertation work

The objectives of this dissertation work include:

1. Melody extraction using existing algorithms
2. To propose a synchrosqueezed wave packet transform based melody extraction system.
3. Testing proposed algorithm for test database (mir-1k)
4. Creation of GUI for the proposed algorithm.
5. Comparison of proposed algorithm with existing methods.

1.6 Thesis structure

This thesis is structured as:

Chapter 1: Introduction : gives basic idea of melody and melody extraction system. Scope for melody extraction is also explained with generalized block diagram of melody extraction system and its various application area. Scope of melody extraction system and literature survey.

Chapter 2: Proposed Scheme: gives detailed idea of SSWPT based melody extraction algorithm.

Chapter 3: Results and Discussions: Simulation of mir-1k dataset and various results considered out using proposed algorithm.

Chapter 4: Conclusion and Scope for Future Work: Explains future scope for proposed technique.

Chapter 2

Proposed scheme

Some melody extraction algorithms use time domain approaches whilst others use frequency domain approaches. Time domain algorithms mainly uses statistical parameters like energy, standard deviation (std), zero-crossing rate (ZCR). Frequency domain algorithms exploit the properties of time-frequency representations using spectral transforms.

Proposed method is based on frequency domain properties. In proposed method synchrosqueezed wave packet transform is used for time-frequency representation. As being multi resolution transform SSWPT gives accurate representation compared to single resolution transforms like STFT. We have divided algorithm in three stages. starting with pre-processing followed by time-frequency representation and finally post processing part. Each stage is explained in details in subsequent sections of this chapter. Figure 2.1 shows block diagram of synchrosqueezed wave packet transform based melody extraction system.

2.1 Pre-processing

Most of the time pre-processing part does filtering and voicing/un-voicing detection of audio signal. Pre-processing stage is combination of equal loudness filter followed by voicing/un-voicing detection. As human voice melody range is between 100 Hz to 1250 Hz[1] ,some techniques employ band pass filter with frequency range 150 Hz to 1250

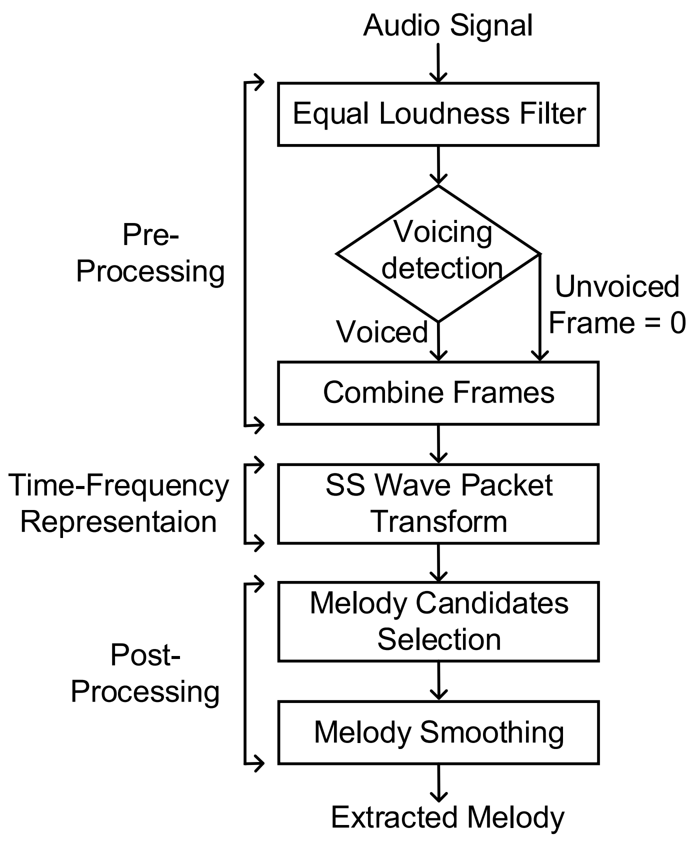


FIGURE 2.1: Block diagram of SSWPT based melody extraction system

Hzinstead of equal loudness filter. Overall accuracy of any melody extraction system mainly depends on how accurate voicing/un-voicing detection is determined. We have combined vocal (only singer voice) and background music (music accompaniment) with different SNR values. Music accompaniment is considered as noise signal.

2.1.1 Equal loudness filter

Audio signal is first filtered through equal loudness filter which enhances the frequencies of the music signal where the probability of human voice is maximum. Enhancing the vocal regions help in better detection of melody lines. Equal loudness filter is created by taking inverse of averaged equal loudness curves [22]. We designed equal loudness filter by using 10^{th} order IIR filter cascaded with 2^{nd} order butter worth high pass filter with the cut-off frequency of 150 Hz. The filter co-efficient data is available at [22]. The filter structure for 10^{th} order IIR filter is given by equation 2.1 and for 2^{nd} order HPF is given by equation 2.2.

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1\,z^{-1} + b_2\,z^{-2} + b_3\,z^{-3} + ... + b_9\,z^{-9} + b_{10}\,z^{-10}}{1 + a_1\,z^{-1} + a_2\,z^{-2} + a_3\,z^{-3} + ...a_9\,z^{-9} + a_{10}\,z^{-10}} \tag{2.1}$$

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1\, z^{-1} + b_2\, z^{-2}}{1 + a_1\, z^{-1} + a_2\, z^{-2}} \tag{2.2}$$

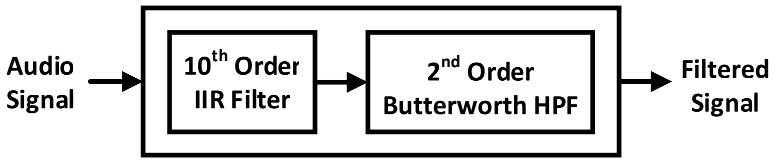


FIGURE 2.2: Block diagram of equal loudness filter

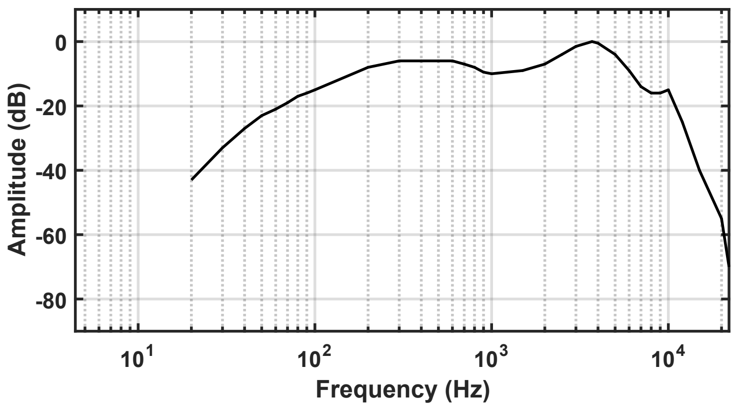


FIGURE 2.3: Equal loudness filter frequency response

Figure 2.3 represents frequency response obtained by the equal loudness filter with using filter co-efficients given by[22]. Lower frequency regions are mostly covered by instruments like bass and high frequency region with instruments like guitar. As shown in figure 2.3 it clearly shows higher and lower frequency regions are attenuated. Music signal is combination of singer voice and background music. Equal loudness filter will enhance the voice part mainly.

2.1.2 Voicing detection

Some algorithms use voicing detection after spectral transform while others use before it. We have used voicing detection before spectral transform. After filtering process music signal is divided into equal length frames. Smaller frame lengths of 30 ms with overlapping of 5 ms are used while framing. Smaller size of frame length and overlap size is considered to reduce data lose during thresholding. We have tested our algorithm on mir-1k dataset. Given the sampling frequency for mir-1k dataset to be 16 kHz, frame length is calculated as $N = f_s * W$. N is frame length in samples and W is frame length in time. Calculating parameters using this formula we get N=480 samples and hop (overlapping window) size H=80 samples.

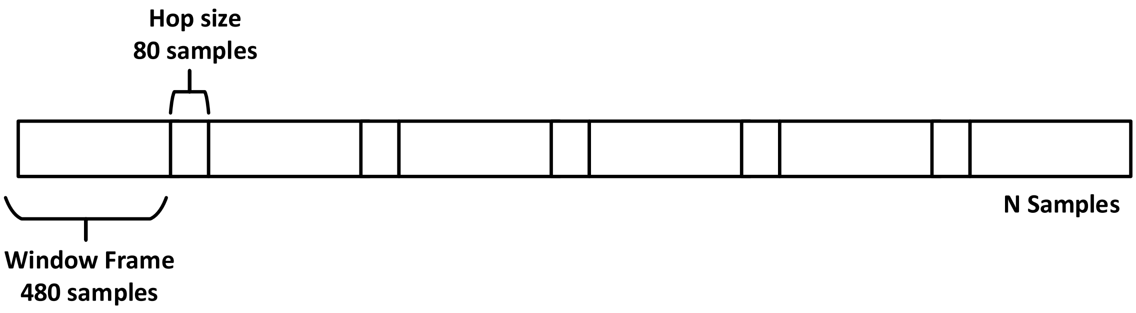


FIGURE 2.4: Framing of music signal with overlapping

We used voicing detection based on statistical parameters. Squared energy measure of a frame is employed to decide voicing detection. The energy of voiced frame is generally larger than the unvoiced frame. The squared energy of a frame with N samples is defined by

$$E_s = \sum_{n=1}^{N} |x(n)|^2 \tag{2.3}$$

After dividing the signal into frames. Energy in each frame is calculated. Energy is calculated for each frame by using equation 2.3. The reference threshold is considered as 20% of average of all frames energy. Smaller value of threshold (20%) is considered as to remove loss of smaller energy voiced frames in music signal.

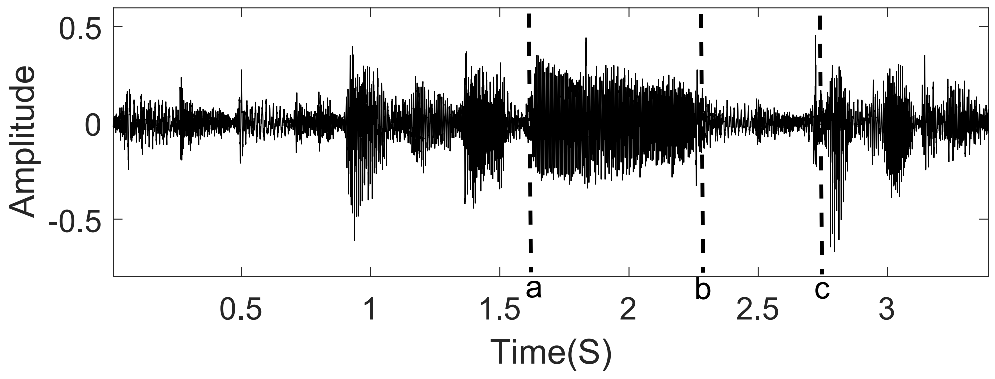


FIGURE 2.5: Piece of music showing voiced and unvoiced frame

Figure 2.5 shows piece of music signal with 3.5 second duration. The frame between a and b is voiced frame where the energy is higher and the frame between b and c is unvoiced frame. Unvoiced frame is mainly due to music accompaniment. Some algorithms have used voicing detection based on onset and offset detection as used in [14].

Figure 2.6 shows music signal of 12 second duration and its spreaded energy over time. Threshold is considered as 20% of average energy so frames having energy less than threshold are made equal to zero and frames having energy higher than threshold are

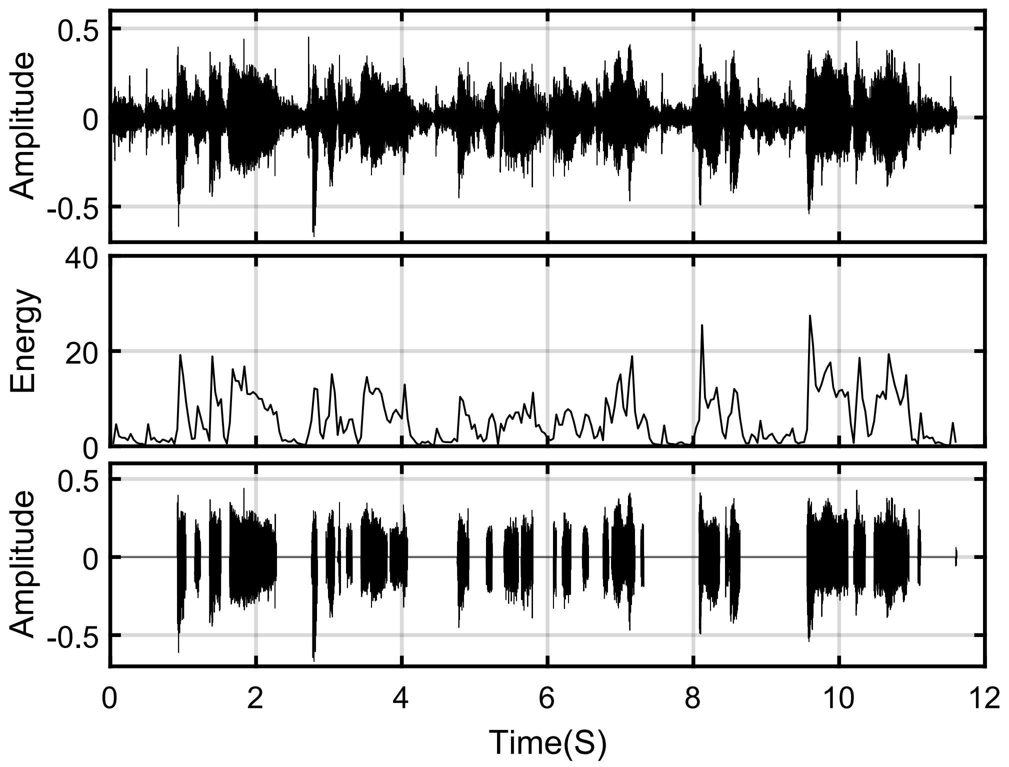


FIGURE 2.6: Voicing detection based on energy of signal

retained. As unvoiced frame does not contain any melody information, after voicing detection these frames are made equal to zero.

2.2 Time-frequency representations

Time-frequency representations are obtained using spectral transforms. Some melody extraction algorithms use single resolution spectral transforms and other uses multi resolution spectral transforms. Time-frequency representation is backbone to select salient melodic candidates from the entire music signal.

2.2.1 Single resolution transform: short time fourier transform

For comparison of proposed algorithm the simulation of existing methods which use STFT as spectral transform are considered. Melody extraction algorithms used by [1],[14],[12] and [3] are using short time fourier transform as spectral transform. STFT of a given signal x(n) is defined by the equation

$$\begin{aligned} X_l(k) &= \sum_{n=0}^{N-1} w(n)x(n+lH)e^{-j\frac{2\pi}{N}kn}, \\ l &= 0,1...,and\ \ k = 0,1,...,N-1. \end{aligned} \tag{2.4}$$

where x(n) is music signal in time domain. w(n) is window used for STFT and hanning window with window size of 2048 samples are implemented while simulation. Hop size used in STFT is 160 samples. Selection of hop size is based on the dataset given. In mir-1k dataset the pitch values are calculated using hop size of 10 ms. Smaller hop size gives more accurate pitch detection as it involves more number of samples to be overlapped. Given all these parameters the STFT of a given signal is X(k) where l is the frame number.

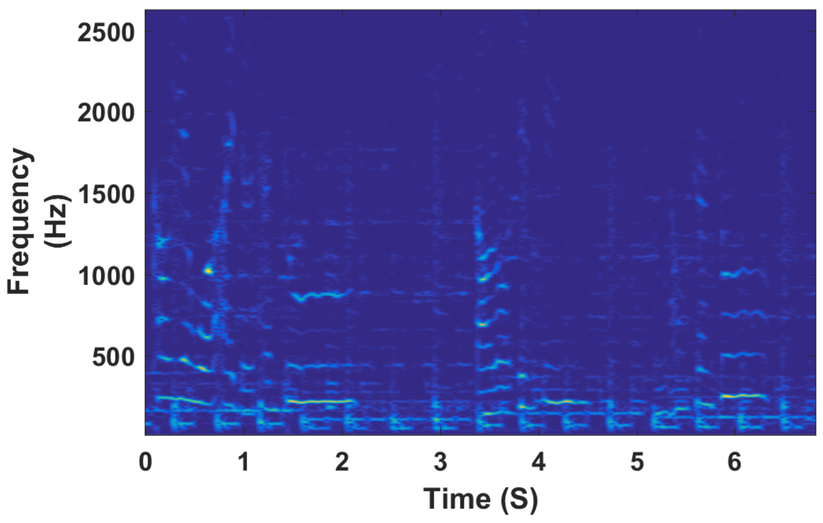


FIGURE 2.7: Time frequency representation using STFT

Figure 2.7 is a time-frequency representation of a 7 second music file abjones_1_02.wav using STFT. Blue color represents lower intensity values whereas yellow color represents higher intensity values.

2.2.2 Multi resolution transform: SSWPT

For a signal $f(t)$ its wave packet transform[18] is defined as

$$W_f(a,b) = \langle w_{ab}, f \rangle = \int \overline{w_{ab}(t)} f(t)dt \tag{2.5}$$

where $w_{ab}(t)$ is family of wave packets. The $\{w_{ab}(t) : |a| \geq 1, b \in \mathbb{R}\}$ is defined by the equation

$$w_{ab}(t) = |a|^{s/2}\, w(|a|^{s}\,(t-b))e^{2\pi i(t-b)a} \tag{2.6}$$

where parameter $s \in (1/2, 1)$. The fourier equivalent of $w_{ab}(t)$ is

$$\widehat{w_{ab}}(\zeta) = |a|^{-s/2}\, e^{2\pi i b\zeta}\widehat{w}(|a|^{-s}\,(\zeta - a)) \tag{2.7}$$

The instantaneous frequency information is derived from the $W_f(a,b)$ as

$$v_f(a,b) = \begin{cases} \frac{\partial_b W_f(a,b)}{2\pi i W_f(a,b)}, for\ |W_f(a,b)| > 0; \\ \infty, otherwise \end{cases} \tag{2.8}$$

By squeezing $W_{ab}(t)$ based upon the instantaneous frequency information $v_f(a,b)$ the accurate time-frequency representation of signal $f(t)$ is obtained by

$$T_f(a,b) = \int_{\mathbb{R}} |W_f(a,b)|^2\, \delta(\Re v_f(a,b) - v)da, \quad for \quad v, b \in \mathbb{R} \tag{2.9}$$

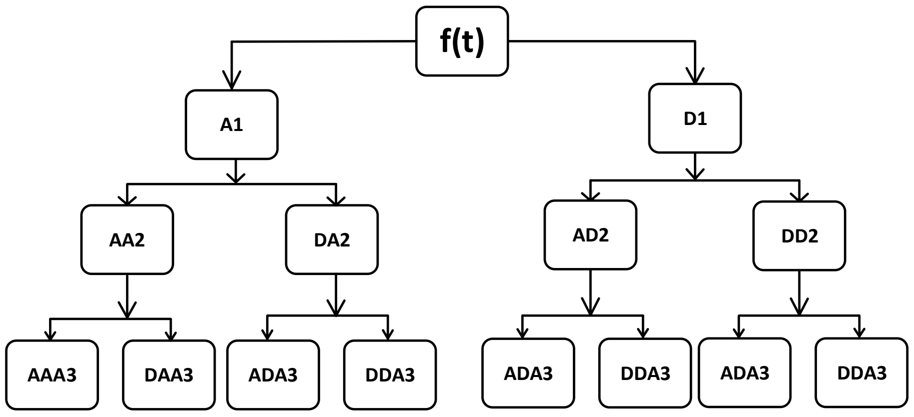


Figure 2.8: Decomposition of a function using wave packet transform

The synchrosqueezed Wave Packet Transform decomposes a signal into its higher and lower frequency components. A three level decomposition of such a signal $f(t)$ is as shown in Figure 2.8. Most melody extraction algorithm uses fast fourier transform (FFT) or Short Time Fourier Transform (STFT) for time-frequency representation.

SSWPT mainly depends on the threshold value of its wavelet packet. We have checked different threshold values and found that the optimum threshold value for audio signals is $1e^{-2}$. Some important parameters used in SSWPT and its optimum values are given in table 3.1.

TABLE 2.1: Parameters for SSWPT

Parameter (dB)	Default Value	Significance
epsl	$1e^{-2}$	Threshold for SST
res	1	Visualization resolution parameter in frequency
N	Signal length	Signal length
NG	round(N/32)	Number of subsampling points for each frame
is_real	1	type of real or complex signals
R_high	$N/2Hz$	Higher frequency bound for analysis
R_low	$0Hz$	Lower frequency bound for analysis
fs	16000	Sampling rate of the music file

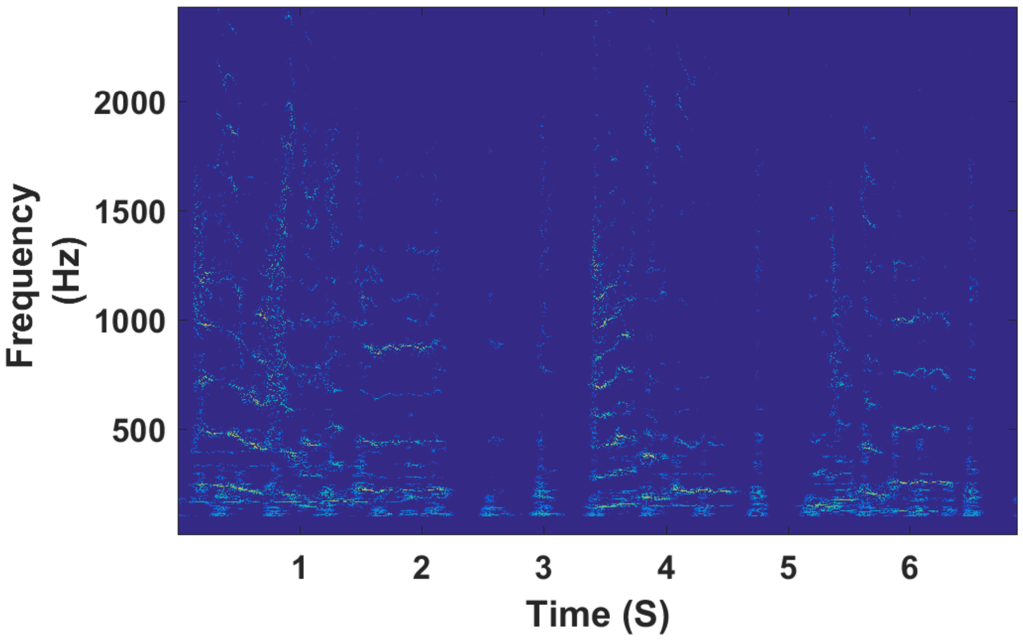


FIGURE 2.9: Time-frequency representation using SSWPT

Figure 2.9 shows a SSWPT for a 7 second music file. For each 32 samples in time domain, SSWPT gives one frame in time-frequency representation. Smaller value of frame gives more accurate time information in spectral representation which improves melody detection in later stages. Frequency information from the given SSWPT distribution is given by

$$f_p = l * \frac{f_s/2}{N/2} \tag{2.10}$$

where f_p is the frequency value of the corresponding location of spectrum, f_s is the sampling frequency and N is the length of the signal in samples.

2.3 Post-processing

Post-processing stage is further divided into median filtering and salience function creation.

2.3.1 Median filtering

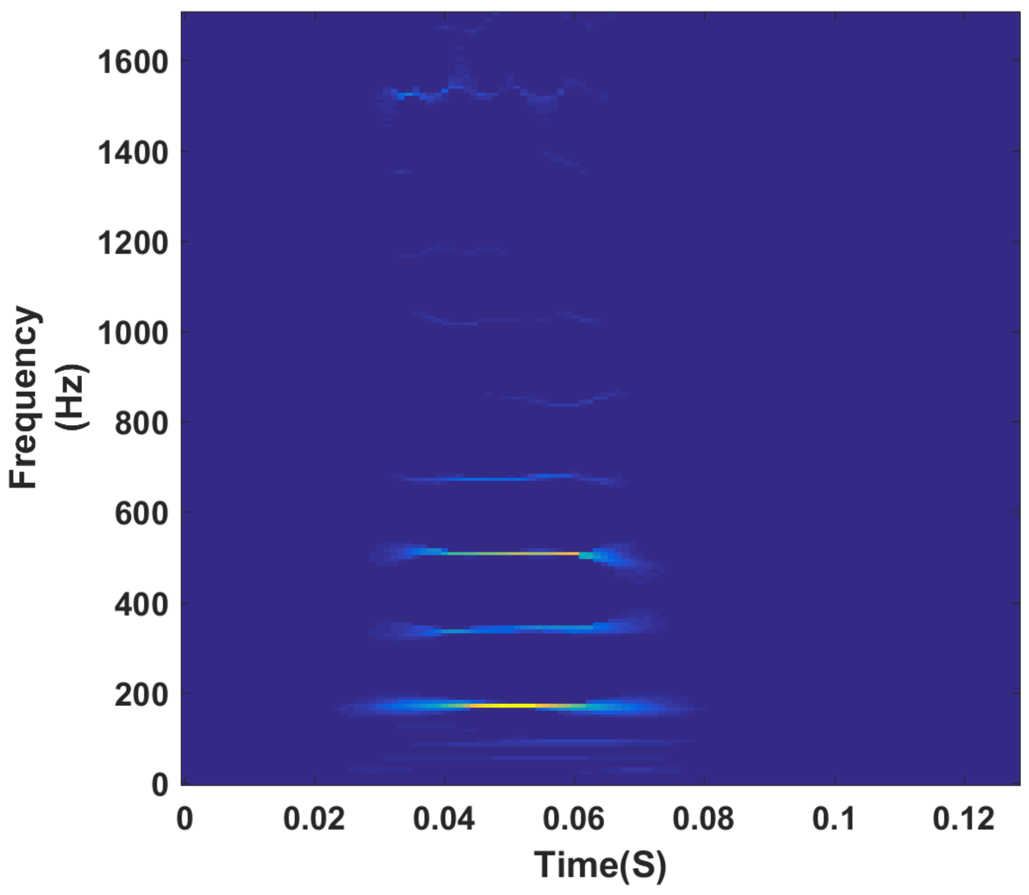
The result after SSWPT for a music signal is $M * N$ matrix. The unwanted music part is lower intensity pixels in this matrix. Applying median filter will smooth out this noise pixels and so main melody which is higher intensity edges is not affected. By default 2D median filter kernel size is 3x3. Melody detection is considered as edge detection problem. The high intensity edges are extracted as an edges which becomes the fundamental candidates for melody detection. After simulation of proposed algorithm it is found that median filer improves the raw pitch accuracy (RPA) and overall pitch accuracy also to a great extent.

2.3.2 Salience function creation

After applying median filtering the mean of every single column of time-frequency matrix is computed. Mean of this columns is a row vector of length equal to number of columns of this matrix. This row vector is used later for thresholding purpose.

Melody candidates are those which are having higher intensity values in the entire spectrum. As shown in figure 2.10 the maximum intensity value of the SSWPT spectrum is the first strong candidate for final melody which is 171.9 Hz. Lower intensity harmonics are considered in calculation for final melody in harmonic summation part.

From a given time-frequency representation global maximum at every frame is computed to decide the strong candidate for melody. From this candidate harmonic candidates are selected as 2^{nd}, 3^{rd}, 4^{th}, 5^{th} harmonics. FFT of the given frame shown in figure 2.10 shows fundamental and its harmonics which are considered as candidates for harmonic



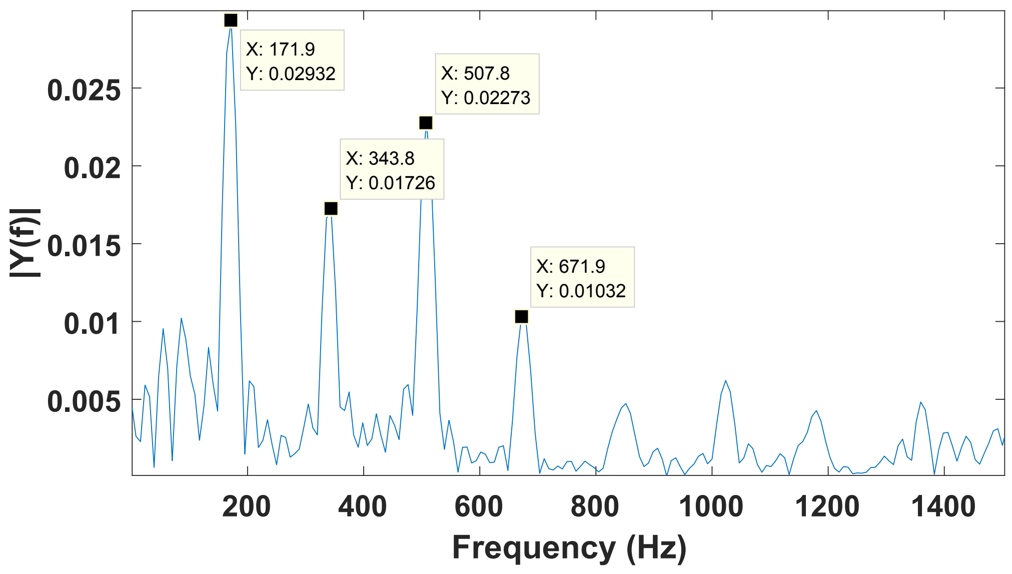


FIGURE 2.10: Strong melody candidate and its harmonics shown with FFT

function. Harmonic function is decided by the equation

$$S(F) = \sum_{n=1}^{n} h_n A(nF) \tag{2.11}$$

where h_n is weight provided to each harmonic candidates. h_n is defined as $h_n = h^{n-1}$

where $h < 1$ and $A(nF)$ is the amplitude of n^{th} harmonic. Lower values of h gives more weightage to fundamental and less weightage to its harmonics.

At last final melody candidates are selected by maximum of the function $S(F)$.

$$F_0 = \max(S(F)) \tag{2.12}$$

The mean of the raw vector calculated previously is used to obtain threshold. Threshold is considered as mean of the raw vector. If the selected final pitch lies below the calculated threshold value then it is made zero else retained.

2.3.3 Frequency smoothing

After final melody candidates selection frequency smoothing is done to remove peaks at certain points in time-frequency representation. These peaks occurs at the window edges of the SSWPT. If the pitch values before and after the current pitch are zero then we make present pitch as zero. If the pitch value of before and after the current pitch value are lower then we make current pitch as minimum of previous and next pitch value. Frequency smoothing helps in removing octave errors which in turn gives final smooth melody curve.

Chapter 3

Results and discussions

The proposed algorithm is implemented in $MATLAB^{®}$ software.

In our evaluation we have used mir-1k [23] dataset with 4 different SNRs. Melodies of only few songs are computed instead of whole dataset. This dataset consists of 1000 different songs having separate music track and accompaniment track. The duration of all the songs is 2 hours with total size of 488 MB. To test the dataset music accompaniment and singing voice are mixed with SNR value of +5 dB, 0 dB and -5 dB. SNR is computed using the equation $Y(n) = a\,X(n) + b\,N(n)$ where Y is mixed signal, X is singing voice and N is music accompaniment which is considered as noise. Different values of **a** and **b** are calculated by the equation

$$SNR = \frac{a^2 * \sum |X(n)|^2}{b^2 * \sum |N(n)|^2} \tag{3.1}$$

TABLE 3.1: SNR calculation

SNR (dB)	a Value	b Value
+5	1.7783	1
0	1	1
-5	0.577	1

3.1 Performance parameters

The quality of extracted melody is defined by performance parameters. Overall pitch accuracy, raw pitch accuracy, standard deviation and time complexity are performance parameters.

- **Overall Pitch Accuracy (OPA)**

  Overall pitch accuracy is the ratio of correct pitch frames to the all pitch frames. Large amount of overall pitch accuracy gives better performance. If the percentage error between evaluated pitch frame and given original pitch frame is 3% then it is considered as correct pitch frame.

$$OPA = \frac{Number\ of\ correct\ pitch\ frames}{Total\ number\ all\ frames} \tag{3.2}$$

- **Raw Pitch Accuracy (RPA)**

  The raw pitch accuracy is the ratio of correct pitch values in voiced frames to that total number of voice frames. A larger number of raw pitch accuracy shows that voicing detection scheme used in algorithm is accurate. Raw pitch accuracy (RPA) mainly depends on voicing/unvoicing detection.

$$RPA = \frac{Number\ of\ correct\ pitch\ frames\ in\ voiced\ frames}{Total\ number\ voice\ frames} \tag{3.3}$$

- **Raw Chroma Accuracy (RCA)**

  If a extracted pitch value lies in the given octave then it is detected as correct pitch for chroma accuracy. Raw chroma accuracy is defined as number of extracted pitches that are in the given octave range to the total number of true pitch values.

- **Standard Deviation**

  The standard deviation (σ_e) is defined as

$$\sigma_e = \sqrt{\frac{1}{N}\sum(p_s - p'_s)^2 - e^2} \tag{3.4}$$

  where p_s is standard pitch value, p'_s is detected pitch value and N is the total number of correct pitch frames. e is the mean where e is defined as

$$e = \frac{1}{N}\sum(p_s - p'_s) \tag{3.5}$$

- **Time Complexity**

  Time complexity can be considered as major parameter for melody extraction systems as melody extraction system can be thought as on line system rather than off line. Most music information retrieval application are well suited for on line applications. Time complexity is measured as how much time melody extraction algorithm takes to extract melody for one single music file. In proposed method the time considered is 33 minutes for 52 music files which is on an average 38 seconds for a single music file.

3.2 Evaluation results

The proposed algorithm is evaluated for 3 different SNR values of -5 dB, 0 dB and +5 dB. The dataset used for evaluation are as per table below.

Table 3.2: Raw pitch accuracy and overall pitch accuracy for abjones

Sr.No	Singer Name	Total Songs	Total Duration (s)	Vocal length (s)
1	Abjones	43	327	229
2	Kenshin	52	430	335
3	Annar	42	290	186
4	Heycat	40	313	235
5	Geniusturtle	69	583	475
6	Fdps	48	397	280

For comparison TWMDP [6] method is used.

Raw pitch accuracy and overall pitch accuracy for all the songs in mir-1k dataset are computed. The results show that proposed algorithm is more accurate than the compared method. For sake of simplicity only 10 songs from each singer are shown in results. The overall accuracy calculation data is shown in appendix section.

3.2.1 Accuracy calculation for +5 dB

Table 3.3: Raw pitch accuracy and overall pitch accuracy for +5 dB

Sr.No	Song Name	RPA (%)	OPA (%)
1	abjones_1_02	69.02	71.65
2	abjones_1_03	67.92	66.23
3	abjones_2_01	81.51	82.62
4	abjones_2_02	76.92	81.16
5	abjones_2_04	72.43	76.66
6	abjones_2_05	79.11	82.53
7	abjones_2_06	70.65	69.31
8	abjones_2_07	74.55	70.45
9	abjones_2_08	69.3	72.78
10	abjones_5_01	80.31	74.64
11	kenshin_1_01	83.91	79.86
12	kenshin_1_02	74.3	68.19
13	kenshin_1_03	82.26	75.98
14	kenshin_1_06	85.56	77.52
15	kenshin_1_09	73.58	72.68
16	kenshin_1_11	85.84	76.3
17	kenshin_2_01	76.8	73.04
18	kenshin_2_02	79.75	76.12
19	kenshin_2_04	82.74	76.43
20	kenshin_3_01	79.2	74.84
21	annar_1_01	77.44	67.59
22	annar_1_03	76.08	64.23
23	annar_1_04	81.2	86.01
24	annar_1_06	77.00	62.2
25	annar_1_08	75.95	73.63
26	annar_2_04	77.38	79.43
27	annar_2_07	90.66	81.39
28	annar_2_08	75.24	71.26
29	annar_3_01	82.01	79.45
30	annar_4_01	87.56	84.83

Table 3.4: Raw pitch accuracy and Overall pitch accuracy for +5 dB

Sr.No	Song Name	RPA (%)	OPA (%)
1	heycat_1_04	88.69	84.87
2	heycat_1_06	85.29	84.44
3	heycat_1_07	84.25	85.29
4	heycat_2_02	74.59	66.28
5	heycat_2_04	72.09	64.93
6	heycat_3_02	85.6	82.69
7	heycat_3_03	70.11	69.73
8	heycat_3_05	82.68	81.69
9	heycat_3_08	89.84	87.76
10	heycat_4_01	95.92	77.38
11	fdps_1_02	46.63	49.06
12	fdps_1_03	53.66	53.43
13	fdps_1_04	37.61	46.73
14	fdps_1_12	32.14	42.07
15	fdps_2_01	55.32	60.36
16	fdps_2_02	47.53	55.13
17	fdps_3_04	60.86	54.93
18	fdps_3_05	50.44	53.38
19	fdps_3_06	51.61	45.98
20	fdps_3_07	57.3	54.96
21	geniusturtle_1_04	44.58	52.68
22	geniusturtle_1_06	40.72	48.33
23	geniusturtle_1_08	36.27	39.33
24	geniusturtle_1_09	40.57	47.8
25	geniusturtle_2_07	42.75	42.44
26	geniusturtle_3_02	63.5	63.09
27	geniusturtle_3_04	59.67	54.1
28	geniusturtle_4_01	53.12	53.19
29	geniusturtle_4_02	56.56	58.93
30	geniusturtle_4_04	69.93	69.51

3.2.2 Accuracy calculation for 0 *dB*

Table 3.5: Raw pitch accuracy and overall pitch accuracy for 0 dB

Sr.No	Song Name	RPA (%)	OPA (%)
1	abjones_1_02	63.65	52.76
2	abjones_1_03	58.16	50.56
3	abjones_2_01	67.57	41.00
4	abjones_2_02	68.42	48.51
5	abjones_2_04	59.93	34.15
6	abjones_2_05	71.83	43.42
7	abjones_2_06	55.79	41.97
8	abjones_2_07	57.14	41.88
9	abjones_2_08	55.69	38.16
10	abjones_5_01	69.321	56.32
11	kenshin_1_01	67.82	65.41
12	kenshin_1_02	53.14	47.56
13	kenshin_1_03	61.52	53.24
14	kenshin_1_06	64.78	53.70
15	kenshin_1_09	50.20	50.54
16	kenshin_1_11	63.97	50.43
17	kenshin_2_01	57.36	47.68
18	kenshin_2_02	64.17	57.16
19	kenshin_2_04	67.25	58.63
20	kenshin_3_01	55.69	52.50
21	annar_1_01	44.89	41.12
22	annar_1_03	53.47	44.76
23	annar_1_04	61.34	51.14
24	annar_1_06	71.53	42.40
25	annar_1_07	50.57	44.09
26	annar_1_08	56.03	43.46
27	annar_2_04	71.19	45.52
28	annar_2_07	79.42	68.23
29	annar_2_08	65.62	49.12
30	annar_3_01	65.60	60.03

Table 3.6: Raw pitch accuracy and Overall pitch accuracy for 0 dB

Sr.No	Song Name	RPA (%)	OPA (%)
1	heycat_1_04	74.56	54.20
2	heycat_1_06	75	62.50
3	heycat_1_07	80.42	70.75
4	heycat_2_02	56.40	47.20
5	heycat_2_04	55.57	41.15
6	heycat_3_02	67.12	61.24
7	heycat_3_03	41.18	33.61
8	heycat_3_05	70.62	60.86
9	heycat_3_08	76.09	65.37
10	heycat_4_01	79.63	55.38
11	fdps_1_02	31.60	35.31
12	fdps_1_03	27.29	29.84
13	fdps_1_04	26.57	31.86
14	fdps_1_12	21.05	28.20
15	fdps_2_01	20.32	19.76
16	fdps_2_02	33.40	37.22
17	fdps_3_04	40.86	33.38
18	fdps_3_05	31.69	27.60
19	fdps_3_06	33.15	26.50
20	fdps_3_07	40.96	30.20
21	geniusturtle_1_04	43.95	44.81
22	geniusturtle_1_06	37.45	45.18
23	geniusturtle_1_08	34.11	33.21
24	geniusturtle_1_09	35.71	36.65
25	geniusturtle_2_07	31.54	28.57
26	geniusturtle_3_02	54.91	51.30
27	geniusturtle_3_04	41.60	33.87
28	geniusturtle_4_01	37.32	33.88
29	geniusturtle_4_02	47.81	46.66
30	geniusturtle_4_04	54.89	52.40

3.2.3 Accuracy Calculation for -5 *dB*

Table 3.7: Raw pitch accuracy and overall pitch accuracy for -5 dB

Sr.No	Song Name	RPA (%)	OPA (%)
1	abjones_1_02	49.02	30.30
2	abjones_1_03	35.45	29.69
3	abjones_2_01	43.93	24.69
4	abjones_2_02	52.22	31.94
5	abjones_2_04	38.46	18.54
6	abjones_2_05	43.67	22.59
7	abjones_2_06	32.78	24.13
8	abjones_2_07	40.17	29.54
9	abjones_2_08	39.10	25.44
10	abjones_5_01	51.24	42.09
11	kenshin_1_01	41.27	41.38
12	kenshin_1_02	37.02	34.00
13	kenshin_1_03	30.14	24.57
14	kenshin_1_06	30.63	24.76
15	kenshin_1_09	28.42	31.86
16	kenshin_1_11	34.37	27.02
17	kenshin_2_01	35.26	27.43
18	kenshin_2_02	35.51	31.10
19	kenshin_2_04	35.14	30.54
20	kenshin_3_01	27.72	31.71
21	annar_1_01	15.10	19.08
22	annar_1_03	29.34	26.16
23	annar_1_04	31.56	27.20
24	annar_1_06	58.75	33.00
25	annar_1_07	21.56	19.00
26	annar_1_08	37.50	25.84
27	annar_2_04	56.90	31.59
28	annar_2_07	55.02	45.11
29	annar_2_08	52.16	35.04
30	annar_3_01	30.68	30.89

Table 3.8: Raw pitch accuracy and Overall pitch accuracy for -5 dB

Sr.No	Song Name	RPA (%)	OPA (%)
1	heycat_1_04	49.78	32.66
2	heycat_1_06	49.08	37.63
3	heycat_1_07	63.19	51.96
4	heycat_2_02	27.80	22.86
5	heycat_2_04	30.16	22.40
6	heycat_3_02	34.63	32.39
7	heycat_3_03	18.77	14.78
8	heycat_3_05	49.22	38.24
9	heycat_3_08	46.21	35.97
10	heycat_4_01	48.66	33.38
11	fdps_1_02	9.58	18.43
12	fdps_1_03	11.00	14.53
13	fdps_1_04	13.06	18.30
14	fdps_1_12	8.64	16.92
15	fdps_2_01	8.33	9.74
16	fdps_2_02	19.28	20.97
17	fdps_3_04	16.73	13.98
18	fdps_3_05	14.73	13.55
19	fdps_3_06	11.29	9.02
20	fdps_3_07	17.69	12.86
21	geniusturtle_1_04	41.87	37.23
22	geniusturtle_1_06	26.64	32.22
23	geniusturtle_1_08	28.43	25.87
24	geniusturtle_1_09	29.34	24.34
25	geniusturtle_2_07	14.95	13.25
26	geniusturtle_3_02	32.10	30.32
27	geniusturtle_3_04	19.16	15.54
28	geniusturtle_4_01	24.13	20.27
29	geniusturtle_4_02	32.29	31.06
30	geniusturtle_4_04	36.82	35.13

3.2.4 Comparison of proposed method with TWMDP method

TABLE 3.9: RPA and OPA for TWMDP method

Sr.No	Song Name	RPA(5 dB)	RPA(0 dB)	RPA(-5 dB)
1	abjones	61.90	51.10	35.00
2	kenshin	73.10	63.70	49.70
3	annar	73.20	67.60	56.10
4	heycat	78.90	74.30	61.50
5	fdps	70.20	59.50	44.00
5	geniusturtle	67.50	54.50	38.80

TABLE 3.10: RPA and OPA for Proposed Method

Sr.No	Song Name	RPA(5 dB)	RPA(0 dB)	RPA(-5 dB)
1	abjones	63.98	54.32	39.60
2	kenshin	83.96	68.65	33.50
3	annar	82.42	71.63	57.13
4	heycat	79.70	65.80	63.22
5	fdps	44.11	30.00	46.43
5	geniusturtle	55.65	54.50	42.16

Comparing accuracy data for TWMDP and proposed method, the proposed method is showing higher accuracy even for SNR values of 0 dB and -5 dB. Figure 3.1 shows comparison chart for accuracy calculated when SNR value is +5 dB.

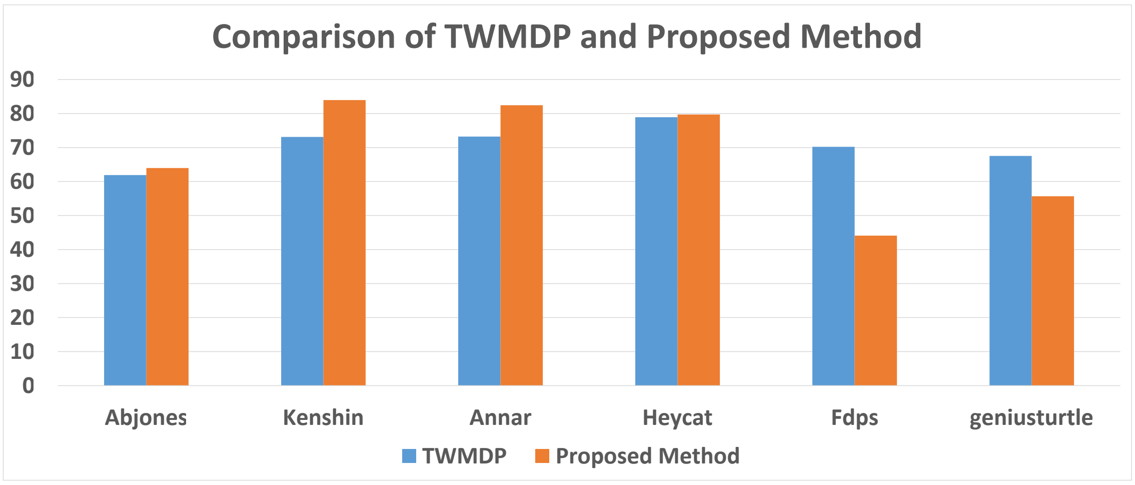


FIGURE 3.1: Comparison of TWMDP and proposed method

3.2.5 Extracted melody of some selected songs

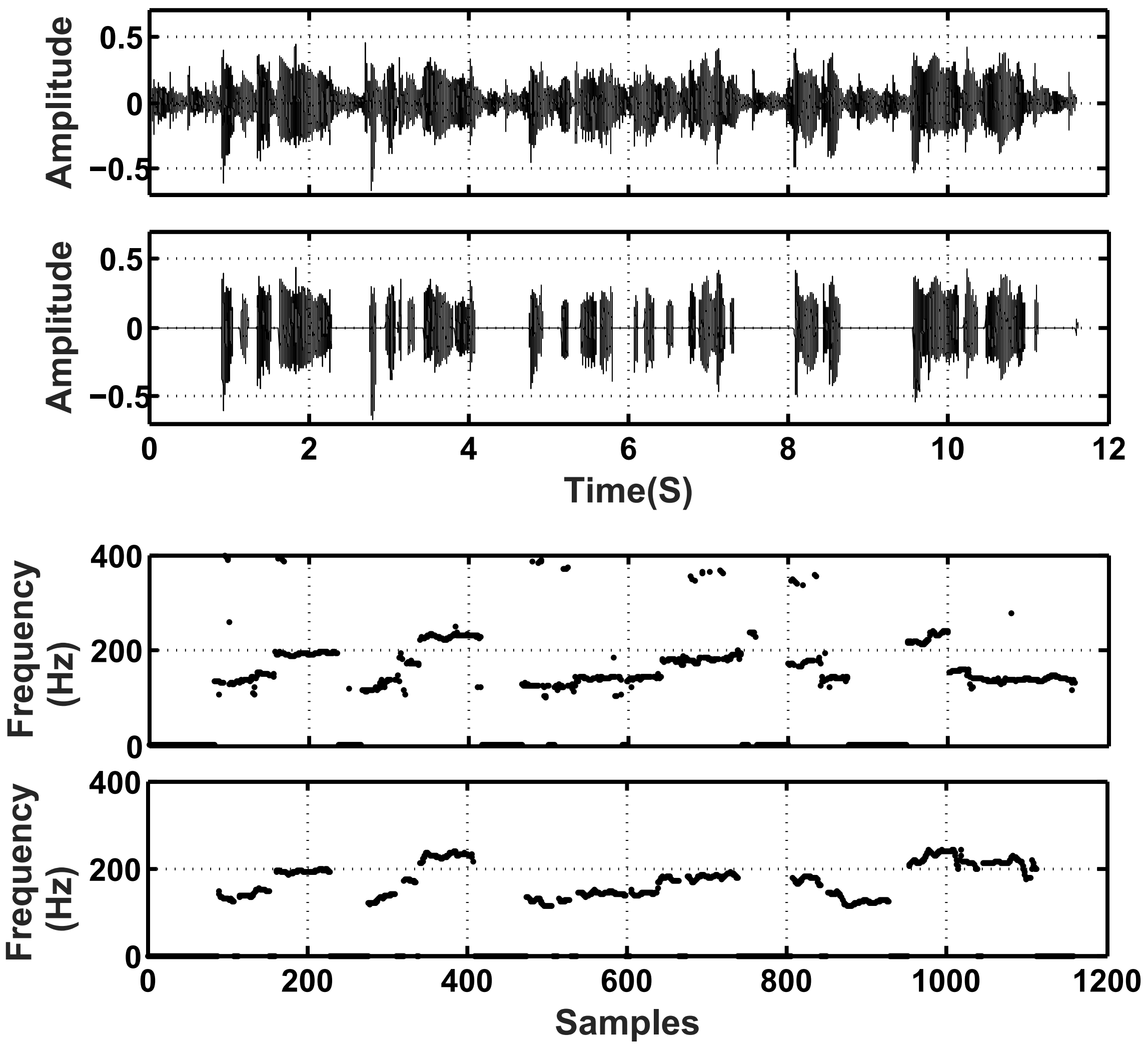


FIGURE 3.2: Melody for song abjones_1_01

Figure 3.2 is an extracted melody for song abjones_1_01.wav. Trace one of figure 3.2 shows time domain plot for music file. Trace two is music file after voicing detection where unvoiced part is almost cleared. Trace three is extracted melody lines using proposed algorithm and trace four is original pitch which is given with the database. Visual comparison shows that extracted melody is nearly equal to the original melody lines.

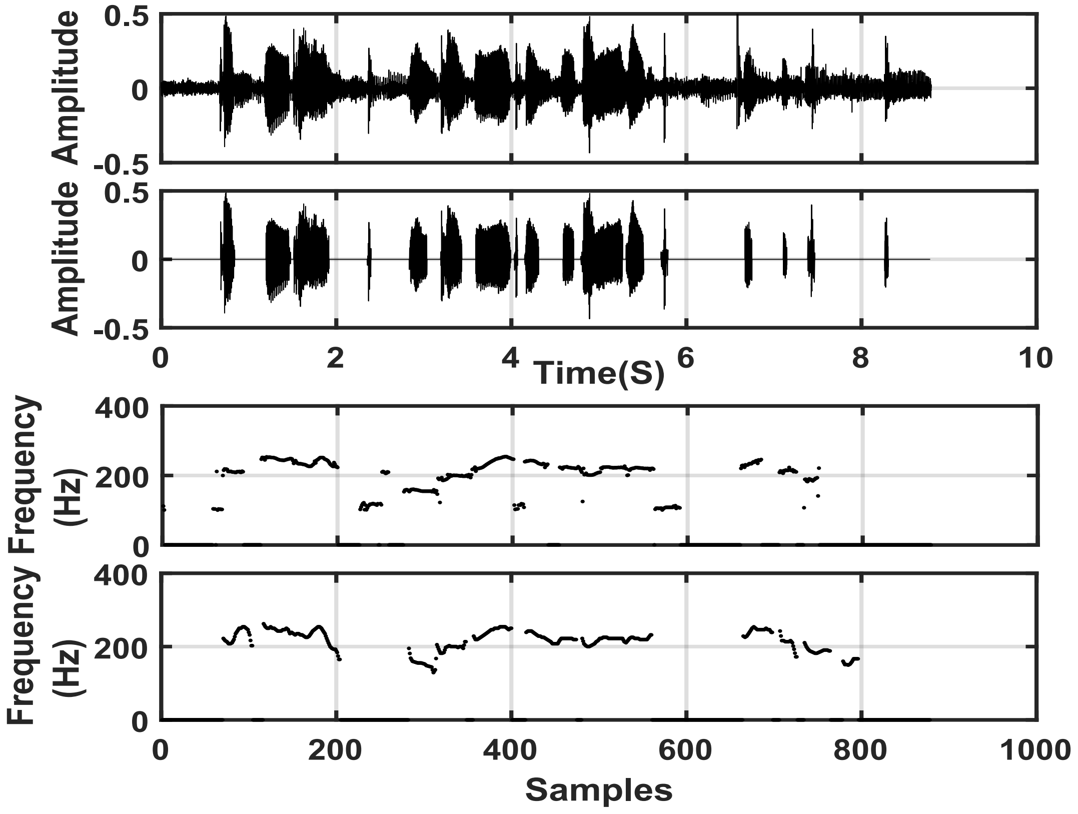


Figure 3.3: Melody for song ani_1_01

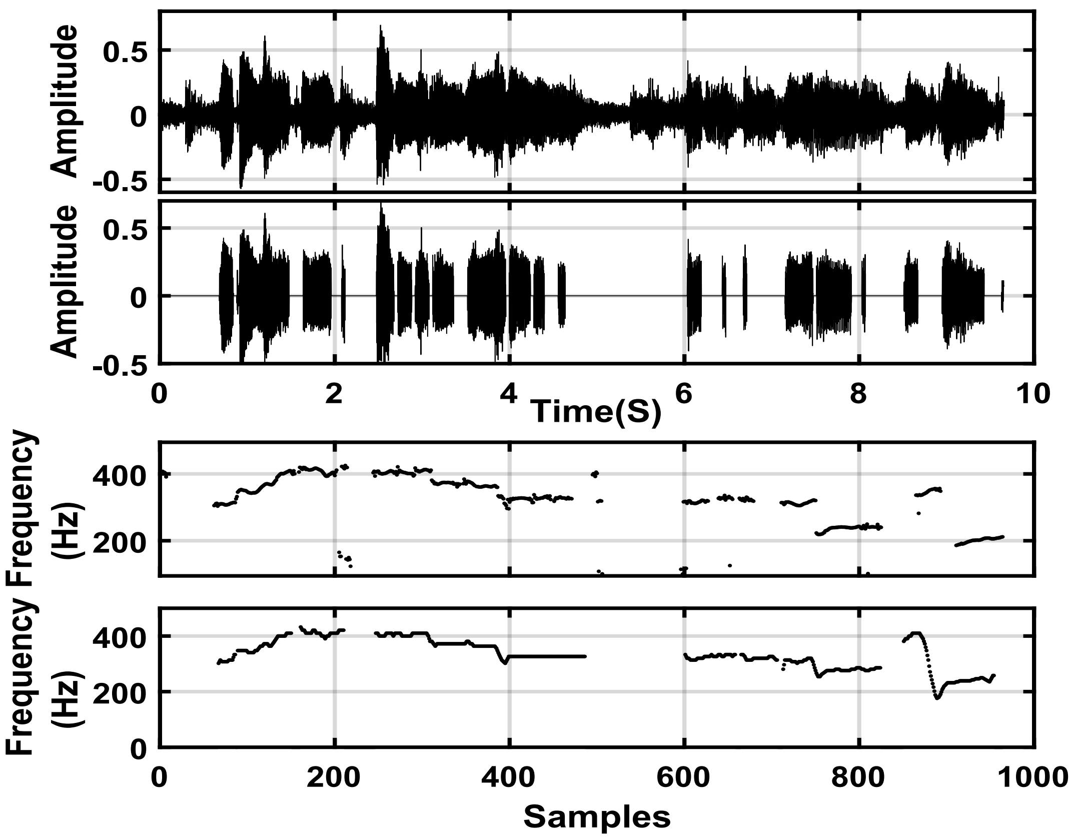


Figure 3.4: Melody for song heycat_1_03

3.3 Developed GUI

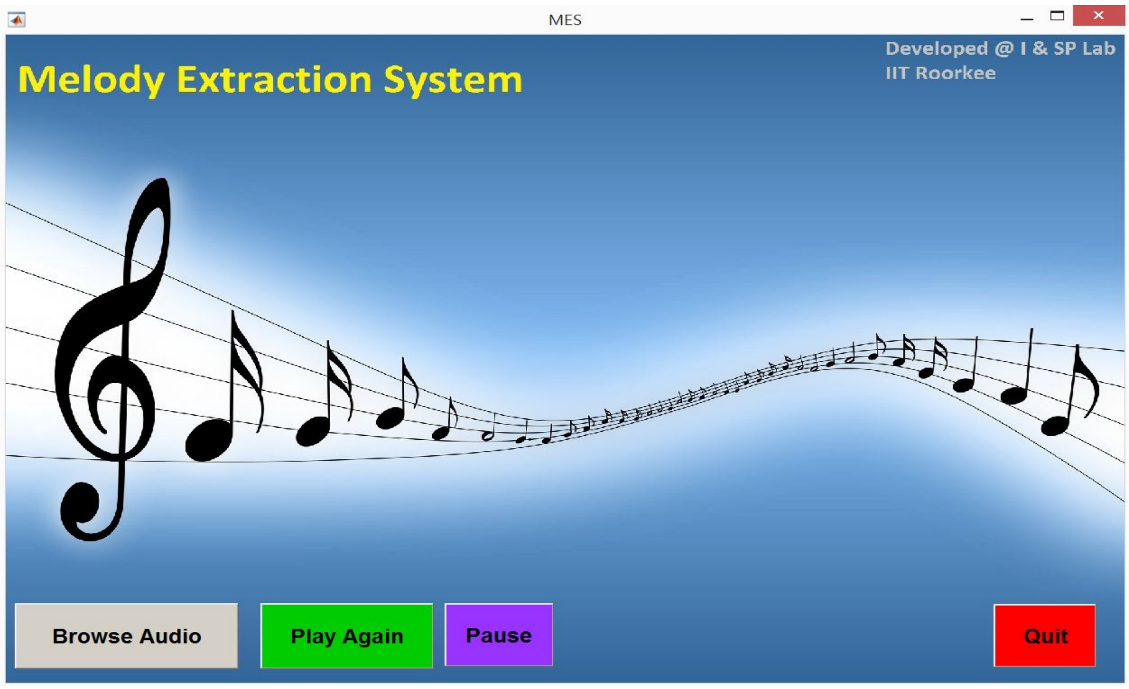


FIGURE 3.5: Developed GUI front end

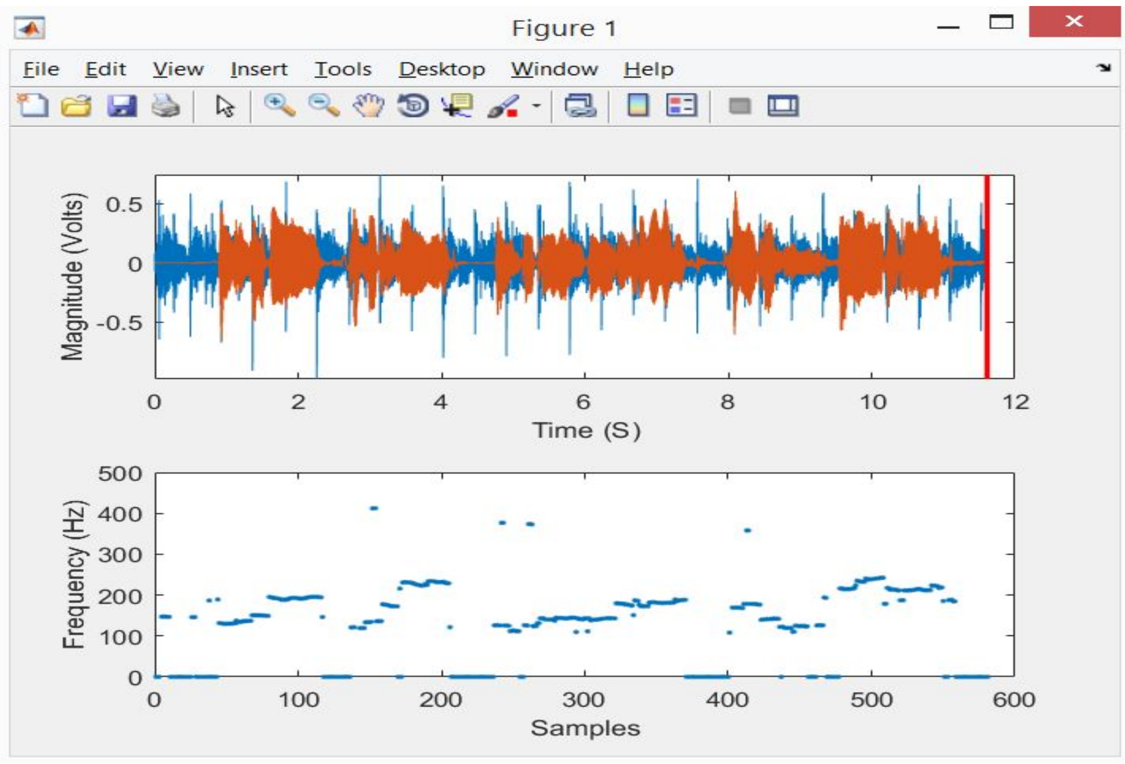


FIGURE 3.6: Extracted melody for the song abjones.wav using the developed GUI

Chapter 4

Conclusion and scope for future work

4.1 Conclusion

4.2 Scope for future work

As synchrosqueezing wave packet transform is giving nearly very accurate time-frequency distribution, it can be future of many signal processing algorithms. Till now only statistical methods and basics of signal processing are used to extract main melody from the music signal. To use synchrosqueezing transform to its full potential one needs higher processing power.

Proposed method can be extended via implementing adaptive algorithms that can improve maximum possible accuracy. Techniques like machine learning and neural nwtwork can be implemented combined with proposed method. By using machine learning we can train algorithm to first detect the type of music signal first and then based on this type we can select proper voicing/unvocing detection techniques. As threshold for voicing detection technique can vary based on music type, use of machine learning can be highly useful for voicing detection stage used in melody extraction algorithms. As U/V detection is the backbone behind any melody extraction system, we believe that use of machine learning will improve algorithm accuracy to its fullest.

Figure 4.1 show a basic block diagram of melody extraction algorithm using higher level computational techniques.

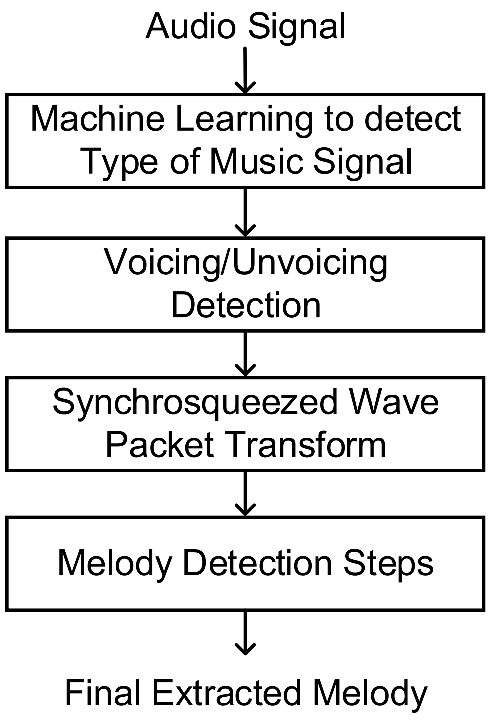

Audio Signal
Machine Learning to detect Type of Music Signal
Voicing/Unvoicing Detection
Synchrosqueezed Wave Packet Transform
Melody Detection Steps
Final Extracted Melody

Figure 4.1: Melody extraction algorithm employing machine learning techniques

Bibliography

[1] J. Salamon, E. Gomez, ”Melody extraction from polyphonic music signals using pitch contour characteristics”, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp.1759-1770, Aug. 2012.

[2] J. Salamon, E. Gomez, D. Ellis, G. Richard, ”Melody extraction from polyphonic music signals: Approaches, applications, and challenges”, *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118-134, March 2014.

[3] Vipul Arora and Laxmidhar Behera, “On-Line Melody Extraction From Polyphonic Audio Using Harmonic Cluster Tracking,” *IEEE Trans. Audio, Speech, Language Process,* vol. 21, no. 3, pp. 520-530, Mar. 2013.

[4] Lawrence R. Rabiner, Michaelj. Cheng, Aaron E. Rosenberg and Carol A. McGonegal, “A Comparative Performance Study of Several Pitch Detection Algorithms,” *IEEE Trans. Audio, Speech, Language Process,* vol. 24, no. 5, pp. 399-418, Oct. 1976.

[5] Guoning Hu and DeLiang Wang, “A Tandem Algorithm for Pitch Estimation and Voiced Speech Segregation,” *IEEE Trans. Audio, Speech, Language Process,* vol. 18, no. 8, pp. 2067-2079, Nov. 2010.

[6] Vishweshwara Rao and Preeti Rao, “Vocal Melody Extraction in the Presence of Pitched Accompaniment in Polyphonic Music,” *IEEE Trans. Audio, Speech, Language Process,* vol. 18, no. 8, pp. 2145-2154, Nov. 2010.

[7] Zafar Rafii and Bryan Pardom, “REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation,” *IEEE Trans. Audio, Speech, Language Process,* vol. 21, no. 1, pp. 73-84, Jan. 2013.

[8] Shubha Kadambe and G. Faye Boudreaux-Bartels, “Application of the Wavelet Transform for Pitch Detection of Speech Signals,” *IEEE Trans. Audio, Speech, Language Process,* vol. 38, no. 2, pp. 917-924, Mar. 1992.

[9] Sassan Ahmadi and Andreas S. Spanias, “Cepstrum-Based Pitch Detection Using a New Statistical V/UV Classification Algorithm,” *IEEE Trans. Audio, Speech, Language Process,* vol. 7, no. 3, pp. 333-338, May. 1999.

[10] Anssi P. Klapuri, “Multiple Fundamental Frequency Estimation Based on Harmonicity and Spectral Smoothness,” *IEEE Trans. Audio, Speech, Language Process,* vol. 11, no. 6, pp. 804-816, Nov. 2003.

[11] Masataka Goto, “A real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals,” *Speech Communication,* vol. 43, no. 4, pp. 311-329, Sep. 2004.

[12] D. J. Hermes, ”Measurement of pitch by subharmonic summation”, *The Journal of the acoustical society of America*, vol. 83, issue no. 1, pp. 257-264, Sep. 1988.

[13] De Cheveigné, Alain and Kawahara, Hideki, “YIN, a fundamental frequency estimator for speech and music”, *The Journal of the acoustical society of America*, vol. 111, issue no. 4, pp. 1917-1930, Apr. 2002.

[14] T. Cheng, W. Xu, Y.Tian, X. Hei, ”Extracting singing melody in music with accompaniment based on harmonic peak and subharmonic summation”, *Proceedings of 4th IET International Conference on Wireless, Mobile and Multimedia Networks (ICWMMN)*, pp. 200-205, Nov. 2011.

[15] Justin Salamon, Emilia Gmez and Jordi Bonada, “Sinusoid Extraction and Salience Function Design for Predominant Melody Estimation,” *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 1685 -1688, Sep. 2011.

[16] Chao-Ling Hsu and Jyh-Shing Roger Jang, “On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset,” *IEEE Trans. Audio, Speech, Language Process,* vol. 18, no. 2, pp. 310-319, Jun. 2009.

[17] Jianling Hu, Sheng Xu, and Jian Chen, “A Modified Pitch Detection Algorithm,” *IEEE Commun. Lett,* vol. 5, no. 2, pp. 64-66, Feb. 2001.

[18] Haizhao Yang, “Synchrosqueezed wave packet transforms and diffeomorphism based spectral analysis for 1D general mode decompositions,” *Applied and Computational Harmonic Analysis,* vol. 39, no. 1, pp. 33-66, Jul. 2015.

[19] Thanatphom Kuntharose and Kam Patanukhom, "Modified Sub-Harmonic Summation Method for Time-Frequency Analysis in Melody Extraction," *International Symposium on Intelligent Signal Processing and Communication Systems,* pp. 1-5, Dec. 2011.

[20] Chuan Cao, Ming Li, Jian Liu and Yonghong Yan, "Singing Melody Extraction in Polyphonic Music by Harmonic Tracking," *International Conference on Music Information Retrieval,* pp. 373-374, Sep. 2007.

[21] Emilia Gmez, "Tonal description of polyphonic audio for music content processing," *INFORMS Journal on Computing,* vol. 18, no. 3, pp. 294-304, Sep. 2006.

[22] Apr. 2016, Equal loudness filter, [Online].
Available: $http://replaygain.hydrogenaud.io/proposal/equal_loudness.html$

[23] mir-1k (MIREX 2009) dataset, [Online].
Available: $http://mirlab.org/dataSet/public/$

Appendix A: Codes

.1 Equal loudness filter and voicing detection

```
clc;
clear all;
close all;

% Equal Loudness Filter Parameters design
b1=[0.0542   -0.0291   -0.0085   -0.0085   -0.0083    0.0225   -0.0260    0.0162
      -0.0024    0.0067  -0.0019];
a1=[ 1.0000   -3.4785    6.3632   -8.5475    9.4769   -8.8150    6.8540   -4.3947
        2.1961   -0.7510  0.1315];
b2 =[    0.9850   -1.9700    0.9850];
a2 =[    1.0000   -1.9698    0.9702];
[x,fs] = audioread('abjones_1_01.wav'); % read the audio file with sampling
    frequency
x=filter(b1,a1,x);  % filtered by 10th order IIR filter
x=filter(b2,a2,x);  % filtered by 2nd order butterworth HPF with cutoff 150 Hz
x = 1.77*x(:,2)+x(:,1);
l=length(x);dt=1/fs;t=0:dt:(l/fs)-dt;
subplot(4,1,1);plot(t,x);grid on;xlabel('Time(S)');ylabel('Amplitude');

% Voicing Detection starts from here
fsize=0.04*fs;fram=[];
end1=fix(length(x)/fsize);i=1;
for j=1:end1
    fram(:,j)=x(i:i+fsize-1);
    i=i+fsize;
end
[r,c]=size(fram);energy=[];
for i=1:c
    energy(i)=sumsqr(fram(:,i));
end
th=0.98*mean(energy);
for i=1:c
   if energy(i)<th
       fram(:,i)=0;
end
```

```
end
% Combine all the frames after voicing detection
fram=fram(:);fram=[fram;x((length(fram)+1):length(x))];
% clear x;
fram = fram(:).';
subplot(4,1,2);plot(t,fram);grid on;xlabel('Time(S)');ylabel('Amplitude');
% fram = fram/max(fram);
```

.2 Time-frequency representation and post-processing

```
% parameters for Synchro-Squezed wave packet transform
N = 40000;
x1 = [0:N-1]/N;
epsl = 1e-2;         % threshold for SST
res =1;              % visualization resolution parameter in frequency
NG = round(N/32);    % number of subsampling points in space
is_real = 1;         % 1: real signals, 0: complex signals
rad = 1.5;           % rad in [0,2] to contral the size of supports of wave
    packets in the frequency domain
t_sc = 1/2 + 1/8;
R_high = N/2       % range of interest is [R_low, R_high]
R_low = 0;
is_cos = 1;
is_unif = 1;
typeNUFFT = 1;
red = 32;
sampleRate = fs;
timeEnd = N/sampleRate;
upBound = round(sampleRate/2);
lowBound = 0;%max(1,round(R_low/res));
end1=floor(length(fram)/40000);
k=1;i=1;out=[];std1=[];m=1;v(1,:)=250:500;v(2,:)=500:750;v(3,:)=750:1000;v(4,:)
    =1000:1250;v(5,:)=1250:1500;v(6,:)=1500:1750;v(7,:)=1750:2000;
for p=1:end1
    sig=fram(i:i+39999);
    [ss_energy coefTensor InsFreq] = ss_wp1_fwd(sig,is_real,is_unif,typeNUFFT,x1,
    NG,R_high,R_low,rad,is_cos,t_sc,red,epsl,res,0);
    [r,c]=size(ss_energy);
    ss_energy(1:250,:)=0;ss_energy(2000:r,:)=0;
    out1=zeros(r,c);
    for j=1:c
        std1(1,m)=std(ss_energy(250:500,j));
        std1(2,m)=std(ss_energy(500:750,j));
        std1(3,m)=std(ss_energy(750:1000,j));
        std1(4,m)=std(ss_energy(1000:1250,j));
        std1(5,m)=std(ss_energy(1250:1500,j));
        std1(6,m)=std(ss_energy(1500:1750,j));
        std1(7,m)=std(ss_energy(1750:2000,j));
        [ma,la]=max(std1(:,m));
        va(m,:)=v(la,:);

        [mm,ll]=max(ss_energy(:,j));
        out1(ll,j)=mm;
        freq1(k)=ll;
        freq(k)=ll*((fs/2)/r);
        k=k+1;
        if ismember(ll,va(m,:))
        else
            [ma,la]=max(ss_energy(va(m,:),j));
            freq(k)=la*((fs/2)/r);
        end
        m=m+1;
       end
    out=[out out1];
    clear ss_energy;
    p=p+1;
   i=i+40000;
end
l=length(fram);
sig=fram(i:l);
[ss_energy coefTensor InsFreq] = ss_wp1_fwd(sig,is_real,is_unif,typeNUFFT,x,NG,
    R_high,R_low,rad,is_cos,t_sc,red,epsl,res,0);
    [r,c]=size(ss_energy);
    out1=zeros(r,c);
    ss_energy(1:250,:)=0;ss_energy(2000:r,:)=0;
 for j=1:c
        std1(1,m)=std(ss_energy(250:500,j));
        std1(2,m)=std(ss_energy(500:750,j));
        std1(3,m)=std(ss_energy(750:1000,j));
        std1(4,m)=std(ss_energy(1000:1250,j));
        std1(5,m)=std(ss_energy(1250:1500,j));
        std1(6,m)=std(ss_energy(1500:1750,j));
        std1(7,m)=std(ss_energy(1750:2000,j));
        [ma,la]=max(std1(:,m));
        va(m,:)=v(la,:);

        [mm,ll]=max(ss_energy(:,j));
        out1(ll,j)=mm;
        freq1(k)=ll;
        freq(k)=ll*((fs/2)/r);
        k=k+1;
        if ismember(ll,va(m,:))
        else
            [ma,la]=max(ss_energy(va(m,:),j));
            freq(k)=la*((fs/2)/r);
        end

        m=m+1;
```

```
end
 out=[out out1];
 l=length(freq);
    for i=2:l-1
        if freq(i-1)==0
        elseif freq(i-1)>freq(i) && freq(i+1)>freq(i)
            freq(i)=min(freq(i-1),freq(i+1));
        end
    end
    load('abjones_1_01.pv');
    ans=abjones_1_01(:,2);
    l=length(ans);
    freq=freq(1:5:length(freq));
    freq=freq(1:l);
    subplot(4,1,3);plot(freq,'.');grid on;ylim([0 400]);xlabel('Samples');ylabel(
    'Frequency (Hz)');

    subplot(4,1,4);plot(ans,'.','LineWidth',2,...
                'MarkerEdgeColor','k');grid on;xlabel('Samples');ylabel('
    Frequency (Hz)');
figure();imagesc(std1);set(gca,'YDir','normal');
```