Music Notes Recogniser using Digital Signal Processing

Mann Raval
19BEC109
Dept. of Electronics and Communication,
Institute of Technology,
Nirma University
Ahmedabad, India
19bec109@nirmauni.ac.in,
ORCID: 0000-0003-3280-9683

Maarjani Sanghavi
19BEC117
Dept. of Electronics and Communication,
Institute of Technology,
Nirma University
Ahmedabad, India
19bec117@nirmauni.ac.in
ORCID:0000-0002-5616-5838

***Abstract*—Songs play an important role in our daily lives. A song is made up of two parts: vocals and backing music. Where the vocalist's voice qualities are determined by the singer, and backing music is composed of a variety of musical instruments such as piano, guitar, drums, and so on. The ability to extract a song's characteristics is becoming increasingly useful for a variety of purposes, including learning, teaching, and creating. This project takes a song as input and extracts the characteristics before detecting and identifying the notes, each of which has a length. The music is first recorded, then the qualities are identified using digital signal processing techniques. The experiment is carried out using multiple piano songs in which the notes have previously been recognised, and the identified notes are compared to the original notes until a greater detection rate is achieved. The experiment is then repeated with piano tunes containing unknown notes using the suggested method.**

Keywords*—*Time Frequency Analysis, Musical Notes, Musical Notes, Sampling Frequency, Recording, Extraction

I. Introduction

For an experienced listener, extracting meaningful musical information from a live or recorded performance is very simple, but for a learner or machine, it is far more difficult. It would be nice to collect this information in a speedy, error-free, automated manner for a variety of practical applications. This thesis examines the development of a software system that takes as input a digitised waveform representing an acoustic music signal and attempts to extract notes from the signal in order to generate a musical score. Event detection, or precisely where within the signal the individual notes really begin and stop, and pitch extraction, or the identification of the pitches being played in each interval, are both part of this signal processing technique. The temporal domain analysis of the data is used to detect events when the problem occurs at various speeds. Because of a phenomenon known as harmonic ambiguity, which occurs when one pitch's fundamental frequency is an integer multiple of another pitch, pitch detection (nothing but frequency identification) is more difficult. Careful signal processing in both the time domain and frequency domain signals solves the problem.

The major goal of this project is to provide a learning aid for musicians, producers, composers, DJs, remixers, teachers, and students of music. This project may be thought of as a box that accepts any music as input and outputs the song's features. The goal of this research is to develop methods for analysing and describing a signal so that musical parameters may be retrieved quickly and objectively. A recurrent problem in the musical literature is that the method for obtaining such parameters is intuitively satisfying but, in our opinion, not particularly sound in terms of signal processing.

II. Literature Survey

A. Sound

The following three parameters can be used to describe a sound: Pitch is the first thing that comes to mind.

- Pitch
- Quality
- Loudness

Pitch is the perceived frequency of a sound by the human ear. A note with a high frequency has a high pitch, whereas a note with a low frequency has a low pitch. A pure tone is a sound produced by a tuning fork or an electronic signal generator that has only one frequency. Because of its greater intensity, the fundamental note has the highest amplitude and is heard the most. Overtones or harmonics are the other frequencies that define the sound quality, such as $2f_0$, $3f_0$, $3f_0$ and so on. The sense of being too loud is a physiological one.

B. Musical Notes

Signal frequencies between 20 and 20 kHz are audible to humans. Some of this large variety is taken into account. Piano is related with it. The ranges of different pianos are diverse. Each piano tone has an own character. As seen in picture 1, the fundamental frequency is represented by a note such as C, D,...etc. The latter C is divided over 12 half steps. Distance from the preceding one and with a fundamental frequency that is twice as high. As a result, this section (from one C)One octave is the interval between C and the next C. C1, C2, and so on distinguish distinct octaves.

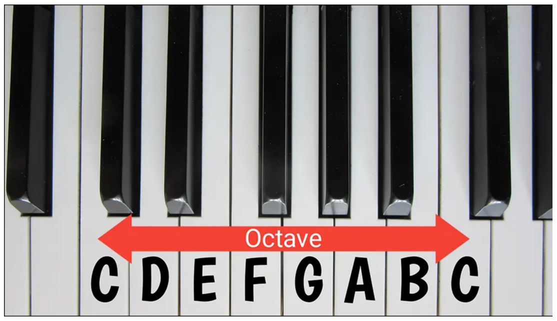


Fig. 1. Octave of Piano

C. Equation of Frequency

The basic formula of the notes of the tempered scale is given by

$$f_n = f_0 \times (a)^n \tag{1}$$

Here, f_0 the frequency of a single fixed note that needs to be determined. A common choice is setting the A above middle C(A_4) at $f_0 = 440$ Hz, n = number of half steps away from the fixed note you are, fn= the frequency of the notes n half steps away, $a = (2)^{1/12}$. Equation of n^{th} key of the keyboard or piano is given by following equation,

$$f(n) = 2^{\frac{n-49}{12}} \times 440 \text{ Hz} \tag{2}$$

D. Sampling

When your voice (or a musical instrument) produces a sound wave, it's an analog wave of changing air pressure. A computer, on the other hand, must record discrete values at discrete time intervals in order to store a sound wave. Therefore the analog signal needs to be converted into discrete values. Sampling is the method of recording discrete time values, and quantizing is the process of capturing discrete pressures. The normal sampling frequency in recording studios is 48 KHz, while CDs have a rate of 44.1 KHz. The minimum sampling frequency of the signal should atleast twice of the maximum frequency of the audio. Humans can hear frequencies between 20 and 20 KHz, which explains why sampling frequencies in the 40 KHz range are so popular. Sampling is done at higher rate so that none of the information is lost and the quantization noise is reduced.

$$f_s \geq f_{max} \tag{3}$$

For example, as the human range to hear audio is 20 Hz to 20 KHz, the audio should be sampled at minimum 40 KHz as 2×20 KHz $= 40$ KHz

E. Frequency and Fourier Transforms

A Fourier transform is used to decompose a complex signal, such as a musical tone, into its component sinusoids. Many integrals and a continuous signal are used in this approach. We need to utilise the Discrete Fourier Transform instead of the Continuous Fourier Transform to conduct a Fourier transform on a sampled (rather than continuous) data. |To compute the DFT of an N-point sequence using the definition

$$A_k = \sum_{n=0}^{N-1} W_N^{kn} a_n \tag{4}$$

The index corresponding to the highest amplitude represents

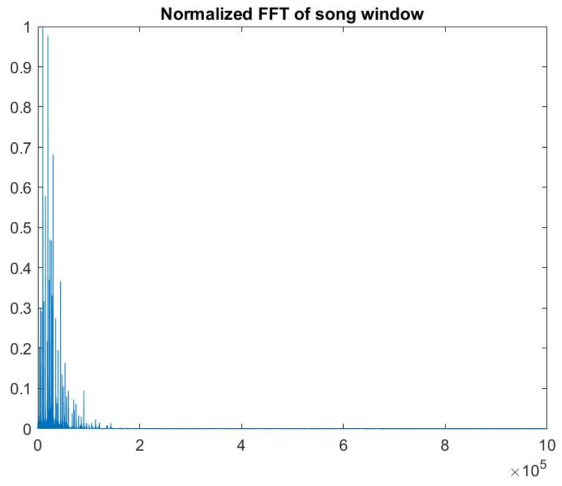


Fig. 2. FFT of the Music Signal

the most significant frequency component, as shown in fig 2, which may be calculated using the method.

$$\mathrm{f} = \frac{i}{T} \times \mathrm{f_s} \tag{5}$$

Where i = Index at which the maximum amplitude exists, T = Total samples of FFT at a time, f_s is the sampling frequency.

F. Padding with Zeros

Although padding with zeroes is a common approach and effective in many applications, it does not appear to improve our data in this case. We have twice the frequency resolution, but the data isn't any better. Quick tests padding with a lot more zeroes show that, while the peaks get rounder due to better frequency resolution, the difference between the highest point on the original sample and the highest point on the padded sample is only 1 Hz at most, indicating that it's probably not worth the effort, even at low frequencies. Padding of zeros is to make the size of the input sequence in the power of two. it is done to make the total number of samples equal to the next higher power of two.

III. Methodology

The audio signal file is made with different frequency components. Those frequencies are obtained through Fast Fourier Transform Algorithm. The audio signal is preprocessed by Averaging, Thresholding, Selecting the window operations. Then the audio signal is padded with zeros, computation through Fast Fourier Transform Algorithm and lastly the assignments of the frequencies takes place to the respective notes. This whole process is divided into detection and identification which is summarised into a flowchart in Fig. 3. |In time domain, the signal is visible as per waveform in Fig. 4

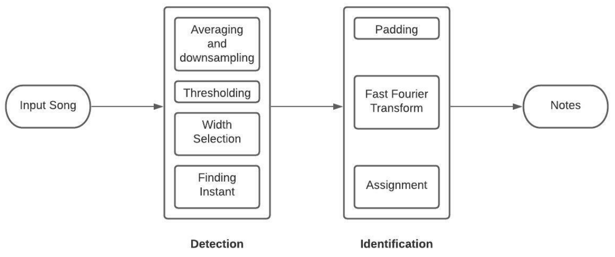


Fig. 3. Flowchart of the algorithm proposed

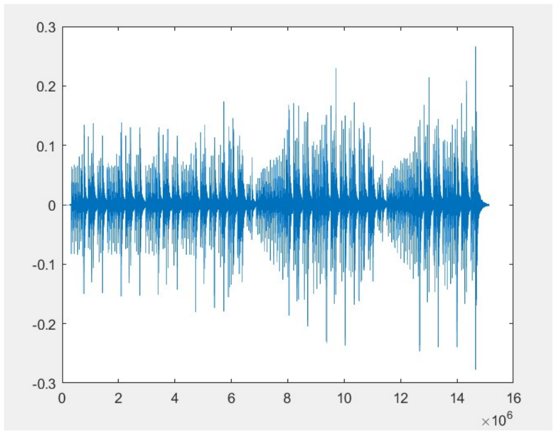


Fig. 4. Time Domain representation of Audio Signal

IV. Detection

A. Averaging and Downsampling

A song has large number of samples. Averaging is done to get the general idea of the sample values. Here, the first stage is averaging in which the average is found for every 100 samples and the value is allocated to the first sample, and the average value is assigned to the second sample for the next 100 samples. This reduces the number of samples and removes the fluctuations present. When the signal decays slowly, the averaged signal becomes more dense. However, in the case of fast decaying signals, the averaged signal represents the original signal's envelope. Then the audio signal is downsampled to reduce the number of samples and to make the computations faster. Downsampling is done by reducing the sampling rate. It is performed to reduce the complexity of the system at higher frequencies.

B. Thresholding

Constant Thresholding: After averaging, we must identify the averaged signal's peaks. Constant thresholding, as the name implies, determines one ideal value for which we may obtain the greatest number of peaks.
Adaptive Thresholding: When we choose a constant threshold value, it's possible that for some notes it'll be larger than the maximum value of the note, while for others it'll be low enough that two peaks of the note will be merged and only one peak (one note) will be identified. The notion of adaptive thresholding was introduced to solve this problem.

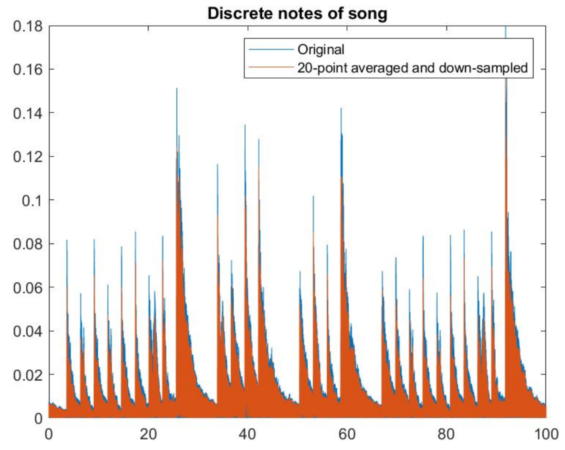


Fig. 5. Averaged and Downsampled Respresentation of Audio Signal

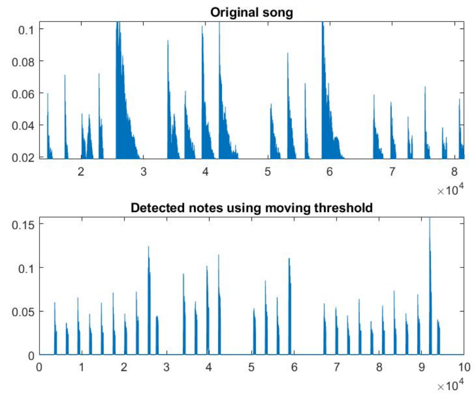


Fig. 6. Original Signal and Signal after Thresholding

V. Results

Happy Birthday, Jingle Bell, Twinkle-Twinkle, Fur Elise, and other songs were used in the experiment. Fur Elise's findings, which are the audio sample's notes, are presented below.

Therefore, The notes on the piano are detected for the Audio Signal based on Fur Elise is,

29 notes were detected by the recognition algorithm. The fundamental frequencies are plotted usThe reconstructed audio signal reconstructed from the detected audio signals has following waveform

Acknowledgment

Dr. Bhupendra Fataniya and Dr. Mehul Naik's invaluable direction, thorough advise, encouragement, and assistance were much appreciated by the authors over the course of their work. Their support aided us in effectively completing

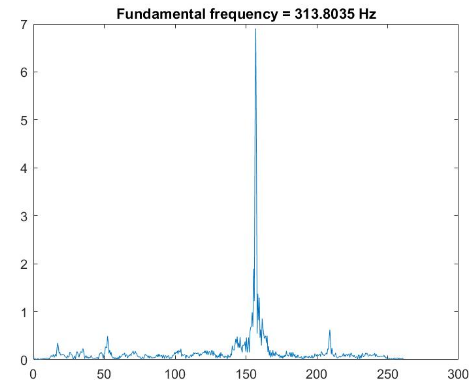


Fig. 7. Fundamental Frequency = 313.8035 Hz

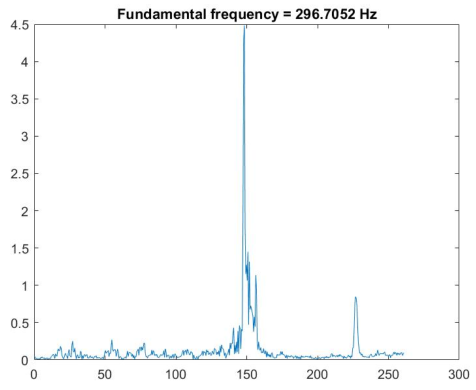


Fig. 10. Fundamental Frequency = 296.7052 Hz

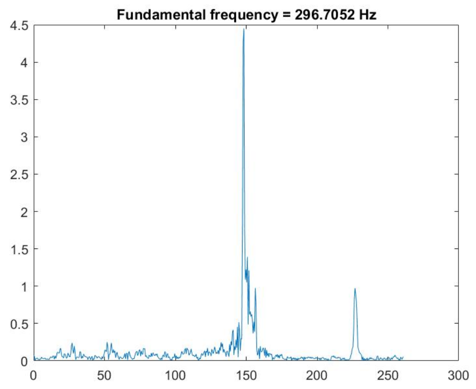


Fig. 8. Fundamental Frequency = 296.7052 Hz

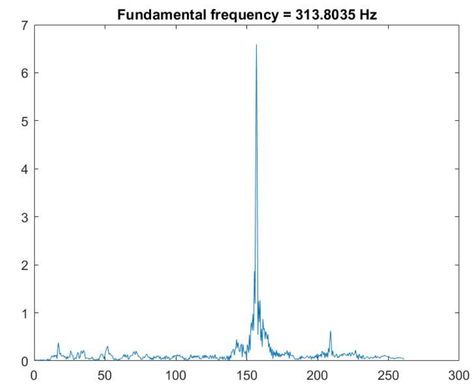


Fig. 11. Fundamental Frequency = 313.8035 Hz

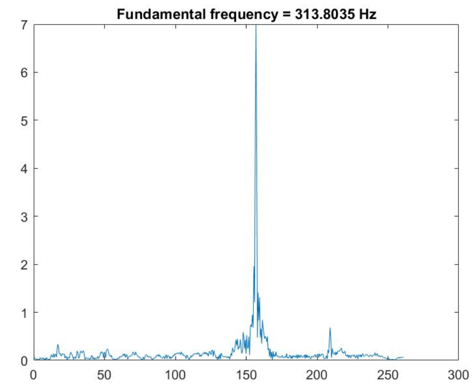


Fig. 9. Fundamental Frequency = 313.8035 Hz

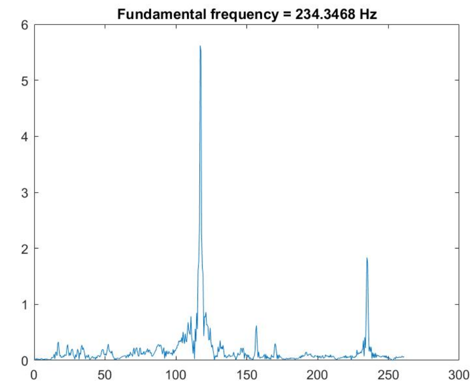


Fig. 12. Fundamental Frequency = 234.3468 Hz

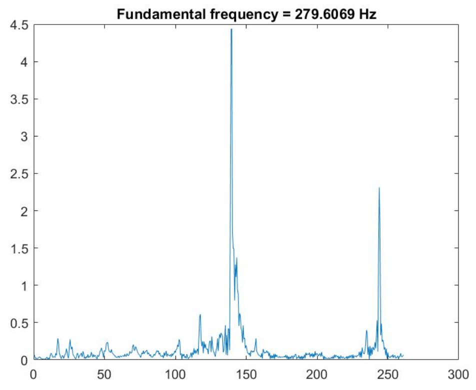


Fig. 13. Fundamental Frequency = 279.6069 Hz

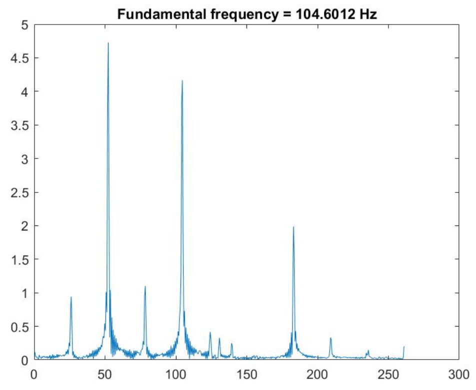


Fig. 16. Fundamental Frequency = 104.6012 Hz

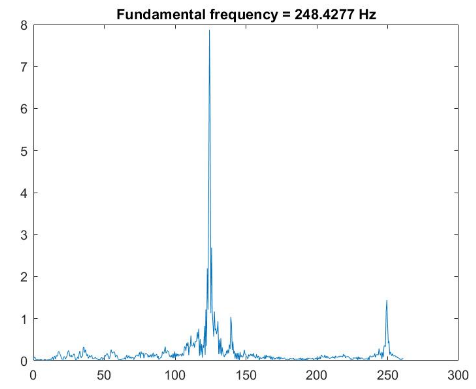


Fig. 14. Fundamental Frequency = 248.4277 Hz

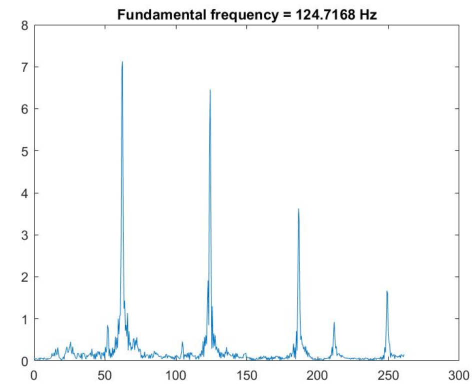


Fig. 17. Fundamental Frequency = 124.7168 Hz

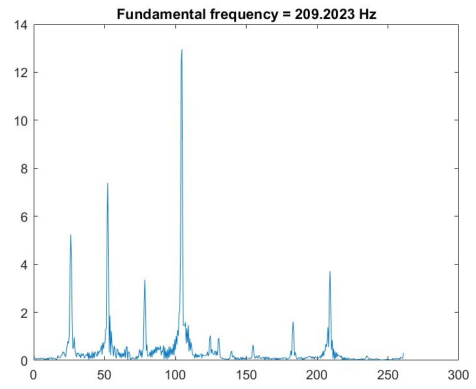


Fig. 15. Fundamental Frequency = 209.2023 Hz

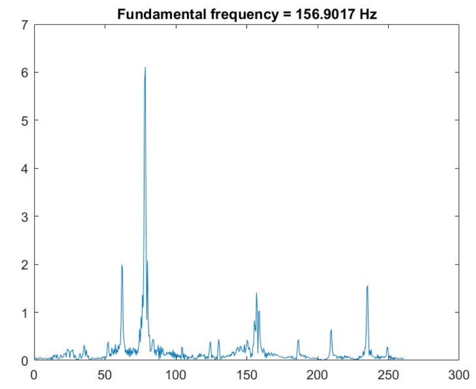


Fig. 18. Fundamental Frequency = 156.9017 Hz

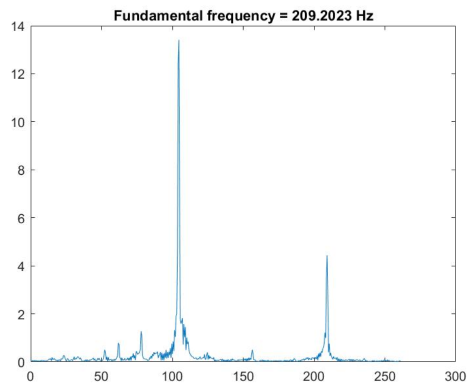


Fig. 19. Fundamental Frequency = 209.2023 Hz

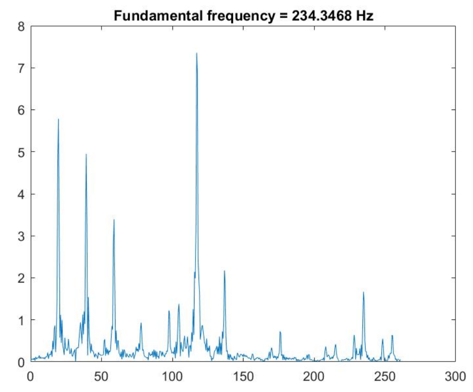


Fig. 20. Fundamental Frequency = 234.3468 Hz

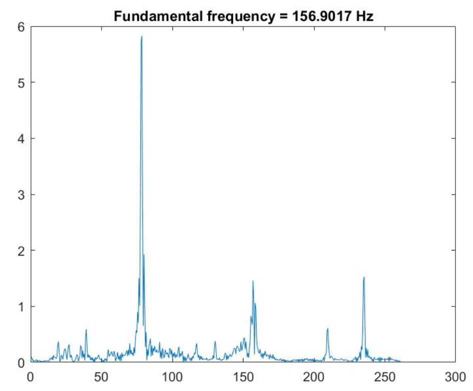


Fig. 21. Fundamental Frequency = 156.9017 Hz

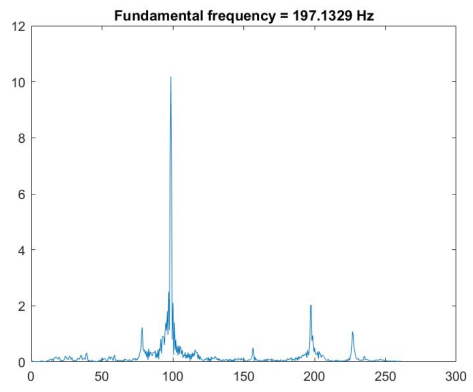


Fig. 22. Fundamental Frequency = 197.1329 Hz

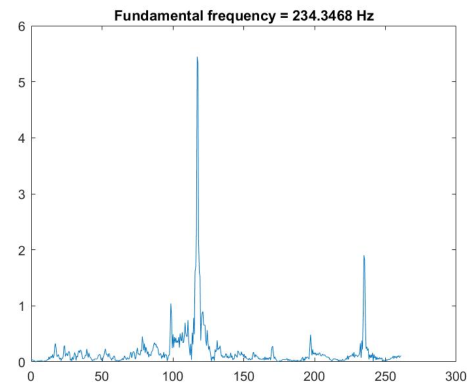


Fig. 23. Fundamental Frequency = 234.3468 Hz

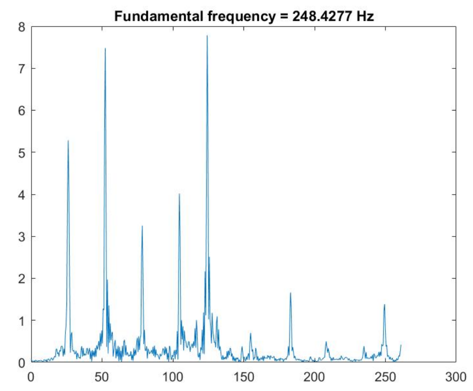


Fig. 24. Fundamental Frequency = 248.4277 Hz

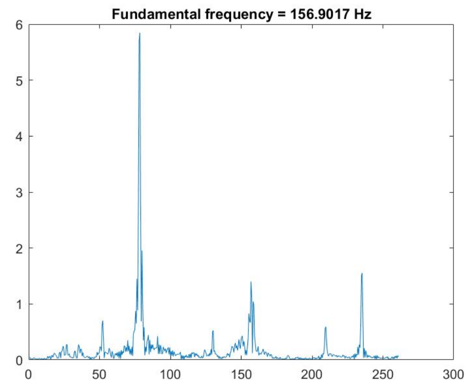


Fig. 25. Fundamental Frequency = 156.9017 Hz

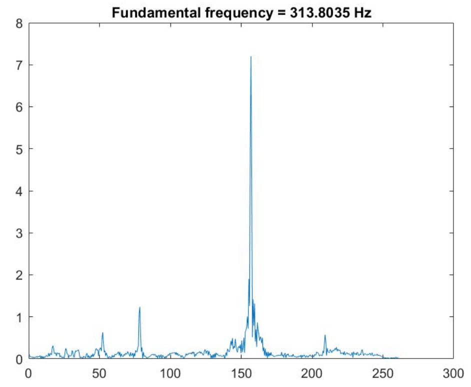


Fig. 26. Fundamental Frequency = 313.8035 Hz

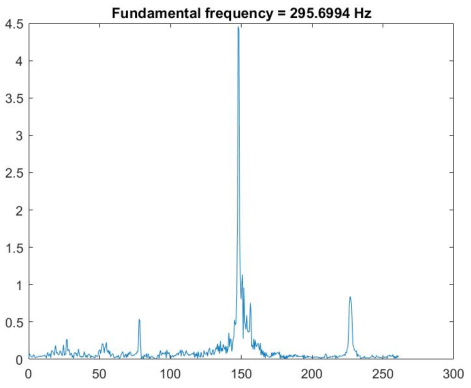


Fig. 27. Fundamental Frequency = 295.6994 Hz

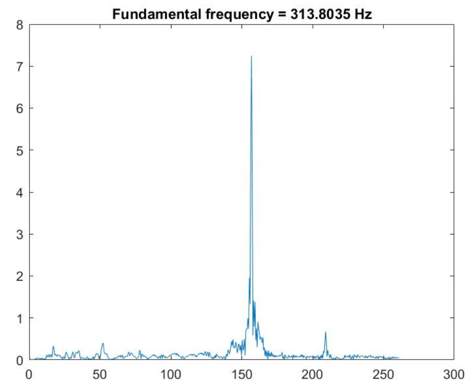


Fig. 28. Fundamental Frequency = 313.8035 Hz

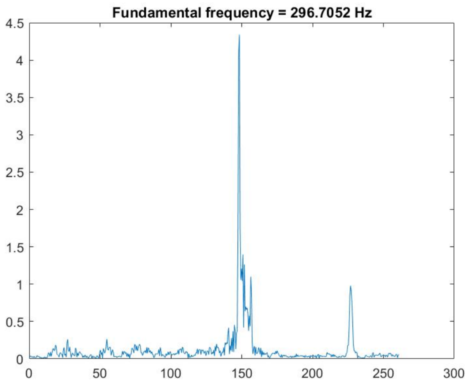


Fig. 29. Fundamental Frequency = 296.7052 Hz

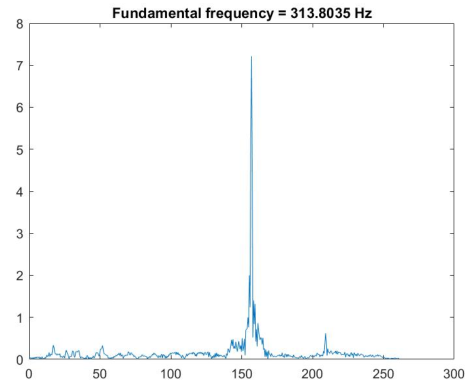


Fig. 30. Fundamental Frequency = 313.8035 Hz

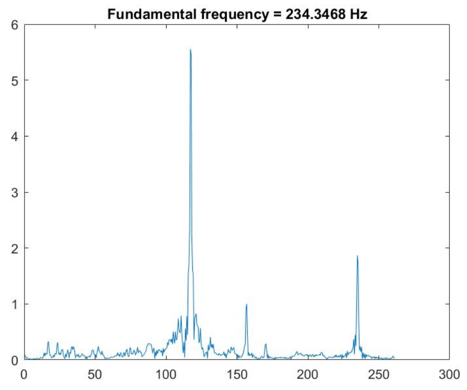


Fig. 31. Fundamental Frequency = 234.3468 Hz

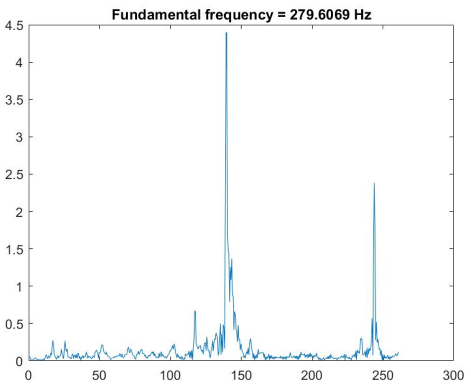


Fig. 32. Fundamental Frequency = 279.6069 Hz

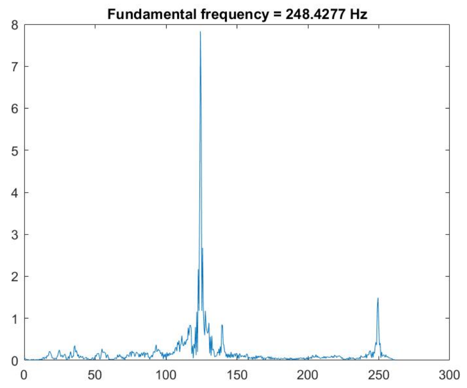


Fig. 33. Fundamental Frequency = 248.4277 Hz

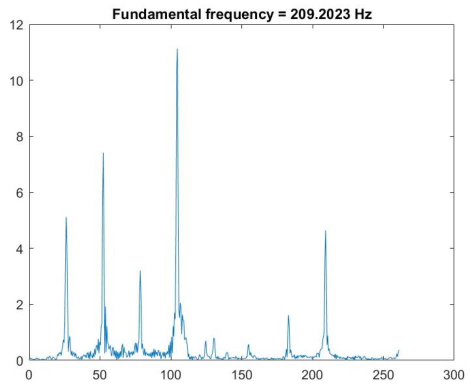


Fig. 34. Fundamental Frequency = 209.2023 Hz

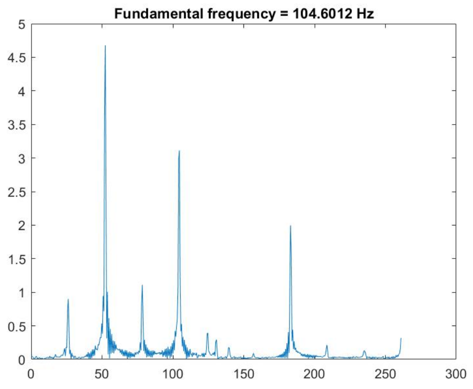


Fig. 35. Fundamental Frequency = 104.6012 Hz

this work. Their imaginative and intelligent drive to take on this work and execute it effectively is acknowledged by the authors. The authors would like to express their gratitude to the Department of Electronics and Communication, as well as all of its teaching and support staff. The authors wishes to express their gratitude to their parents, siblings, friends, and colleges for their unwavering support during this trip. The authors prays to the Almighty for blessings on everyone who helped to accomplish this work, whether directly or indirectly.

References

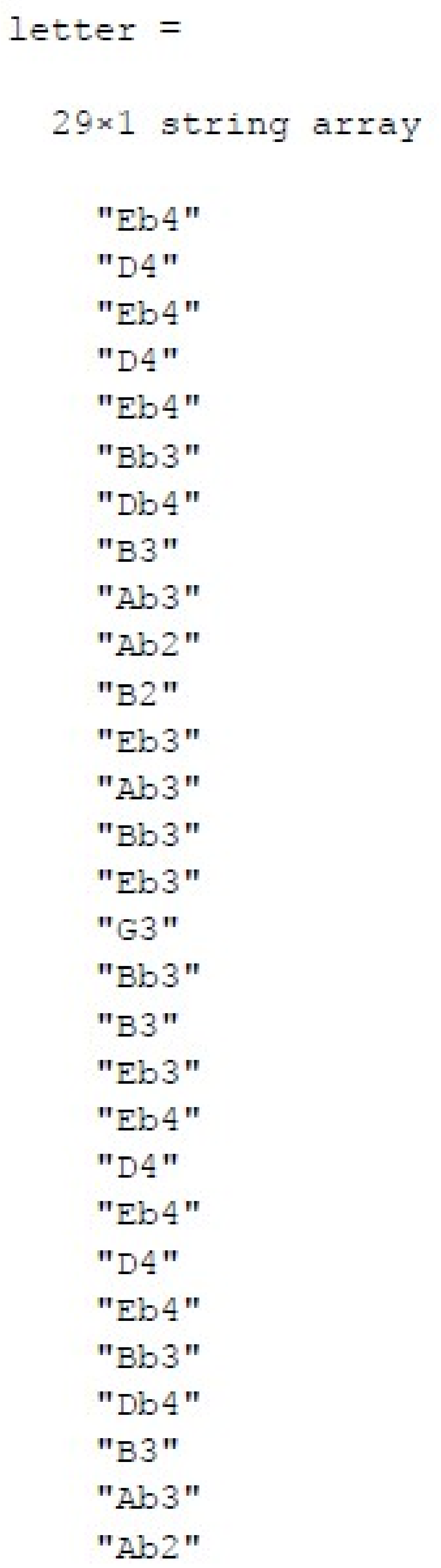
[1] Jay K. Patel, E.S. Gopi, Musical Notes Identification using Digital Signal Processing, Procedia Computer Science, Volume 57, 2015, Pages 876-884, ISSN 1877-0509,

[2] Eric Tarr,"Hack Audio", Routledge, Edition I, 2019

[3] Vijay Madisetti, " The Digital Signal Processing Handbook", Second Edition, CRC Press, 2010

[4] M. Muller, D. P. W. Ellis, A. Klapuri and G. Richard, "Signal Processing for Music Analysis," in IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 6, pp. 1088-1110, Oct. 2011, doi: 10.1109/JSTSP.2011.2112333.

[5] Sanjit Mitra,"Digital Signal Processing: A Computer based Approach", 4^{th} Edition, McGraw Hill, 2013



```
letter =

  29×1 string array

    "Eb4"
    "D4"
    "Eb4"
    "D4"
    "Eb4"
    "Bb3"
    "Db4"
    "B3"
    "Ab3"
    "Ab2"
    "B2"
    "Eb3"
    "Ab3"
    "Bb3"
    "Eb3"
    "G3"
    "Bb3"
    "B3"
    "Eb3"
    "Eb4"
    "D4"
    "Eb4"
    "D4"
    "Eb4"
    "Bb3"
    "Db4"
    "B3"
    "Ab3"
    "Ab2"
```

Fig. 36. Output Piano Notes

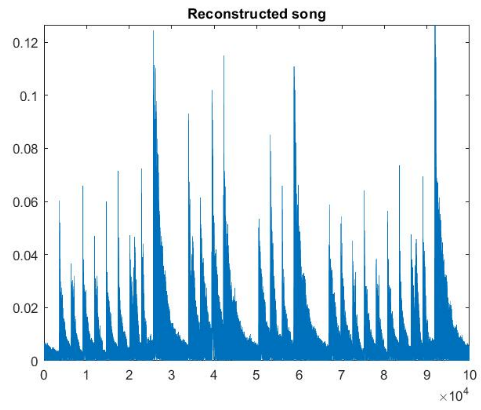


Fig. 37. Reconstructed Audio Signal

```
% Mann Raval (19BEC109) Maarjani Sanghavi (19BEC117)
% Digital Signal Processing
% Main program is used to extract notes from audio file
% Input file: FurElise_Slow.mp3
% relies on plotsound.m file for one (optional) function

%% set up song
clear;
clc;
clf;
close all;
mute = true; % set this to false to hear audio throughout program
             % useful for debugging
[song,Fs] = audioread('FurElise_Slow.mp3');
Fs = Fs*4;   % speed up song (original audio file is very slow)
figure, plot(song(:,1)), title('Fur Elise, entire song')

%% set parameters (change based on song)
% t1 and t2 are the start points and the points of the audio signal which
% is used for analysis
t1 = 2.9e6; t2 = 4.9e6;
% analyze a window of the song
y = song(t1:t2);
[~,n] = size(y);
t = linspace(t1,t2,n);
if ~mute, plotsound(y,Fs); end
audiowrite('fur_elise_window.wav',y,Fs);

%% FFT of song
 Y = fft(y);
 Y_norm = abs(Y./max(Y));
 figure, plot(Y_norm), title('Normalized FFT of song window'), xlim([0
floor(n/2)])

%% downsample by m
clc
m = 20;
Fsm = round(Fs/m);
p = floor(n/m);
y_avg = zeros(1,p);
for i = 1:p
    y_avg(i) = mean(y(m*(i-1)+1:m*i));
end
figure, plot(linspace(0,100,n),abs(y)), hold on
        plot(linspace(0,100,p),abs(y_avg))
        title('Discrete notes of song')
        legend('Original', '20-point averaged and down-sampled')
if ~mute, sound(y_avg,Fsm); end

%% threshold to find notes
close all
y_thresh = zeros(1,p);
i = 1;
while (i <= p)
    thresh = 5*median(abs(y_avg(max(1,i-5000):i)));
    if (abs(y_avg(i)) > thresh)
        for j = 0:500
            if (i + j <= p)
                y_thresh(i) = y_avg(i);
                i = i + 1;
            end
        end
        i = i + 1400;
    end
    i = i + 1;
end


figure, subplot(2,1,1), plot(abs(y_avg)), title('Original song'), ylim([0
1.1*max(y_avg)])
        subplot(2,1,2), plot(abs(y_thresh)), title('Detected notes using
moving threshold')

if ~mute, sound(y_thresh,round(Fsm)); end

%% find frequencies of each note
clc; close all


i = 1;
i_note = 0;
while i < p
    j = 1;
    end_note = 0;
    while (((y_thresh(i) ~= 0) || (end_note > 0)) && (i < p))
        note(j) = y_thresh(i);
        i = i + 1;
        j = j + 1;
        if (y_thresh(i) ~= 0)
            end_note = 20;
        else
            end_note = end_note - 1;
        end
        if (end_note == 0)
           if (j > 25)
               note_padded = [note zeros(1,j)]; % pad note with zeros to
double size (N --> 2*N-1)
               Note = fft(note_padded);
               Ns = length(note);
               f = linspace(0,(1+Ns/2),Ns);
               [~,index] = max(abs(Note(1:length(f))));
               if (f(index) > 20)
                   i_note = i_note + 1;
                   fundamentals(i_note) = f(index)*2;
                   figure, plot(f,abs(Note(1:length(f))))
                           title(['Fundamental frequency =
',num2str(fundamentals(i_note)),' Hz'])
                           %plot(note_padded)
               end
```

```
                i = i + 50;
            end
            clear note;
            break
        end

    end
    i = i + 1;
end

%% play back notes
amp = 1;
fs = 20500;  % sampling frequency
duration = .5;
recreate_song = zeros(1,duration*fs*length(fundamentals));
for i = 1:length(fundamentals)
    [letter(i,1),freq(i)]= FreqToNote(fundamentals(i));
    values = 0:1/fs:duration;
    a = amp*sin(2*pi*freq(i)*values*2);
    recreate_song((i-1)*fs*duration+1:i*fs*duration+1) = a;
    if ~mute, sound(a,fs); pause(.5); end
end
letter
audiowrite('fur_elise_recreated.wav',recreate_song,fs);
```