Netal Asawa (2019130003)
Sejal Gurkhe (2019130017)
Ishita More (2019130039)
TE Comps (A)
Subject: FOSP Lab

Exp No. 9
FOSP Mini Project

Aim: To identify any application based on Signal Processing Domain.

Theory:

Musical Note Extraction:

Songs play a vital role in our day to day life. A song contains basically two things, vocal and background music. Where the characteristics of the voice depend on the singer and in case of background music, it involves mixture of different musical instruments like piano, guitar, drum, etc. To extract the characteristic of a song becomes more important for various objectives like learning, teaching, composing. This project takes song as an input, extracts the features and detects and identifies the notes, each with a duration. First the song is recorded and digital signal processing algorithms used to identify the characteristics. The experiment is done with the several piano songs where the notes are already known, and identified notes are compared with original notes until the detection rate goes higher. And then the experiment is done with piano songs with unknown notes with the proposed algorithm.

Musical tones have three identifying characteristics; volume, pitch and timbre. Volume is power, or the amplitude of the corresponding wave, and it is measured in decibels. Frequency is the measure of how "high" or "low" a tone is, which is measured in hertz (Hz). The piano, for example, has notes as low as 28 Hz and as high as 4,000 Hz. The third identifying feature, timbre, stems from the fact that musical sounds are made up of many different sine waves (as opposed to a sound that is made of just one sine wave). Each instrument has a characteristic pattern of sine waves, which is what makes it possible to distinguish between an oboe and an electric guitar playing the same note.

Sampling

When a sound wave is created by your voice (or a musical instrument), it's an analog wave of changing air pressure. However, in order for a computer to store a sound wave, it needs to record discrete values at discrete time intervals. The process of recording discrete time values is called *sampling*, and the process of recording discrete pressures is called *quantizing*. Recording studios use a standard sampling frequency of 48 kHz, while CDs use the rate of 44.1 kHz. Signals should be sampled at twice the highest frequency present in the signal. For example, if the highest frequency present in the signal was 100 Hz, you'd have to use a sampling frequency of at least 200 Hz. Humans can hear frequencies from approximately 20-20,000 Hz, which explains why common sampling frequencies are in the 40 kHz range. Sampling at less than twice the highest frequency present leads to distortion during processing

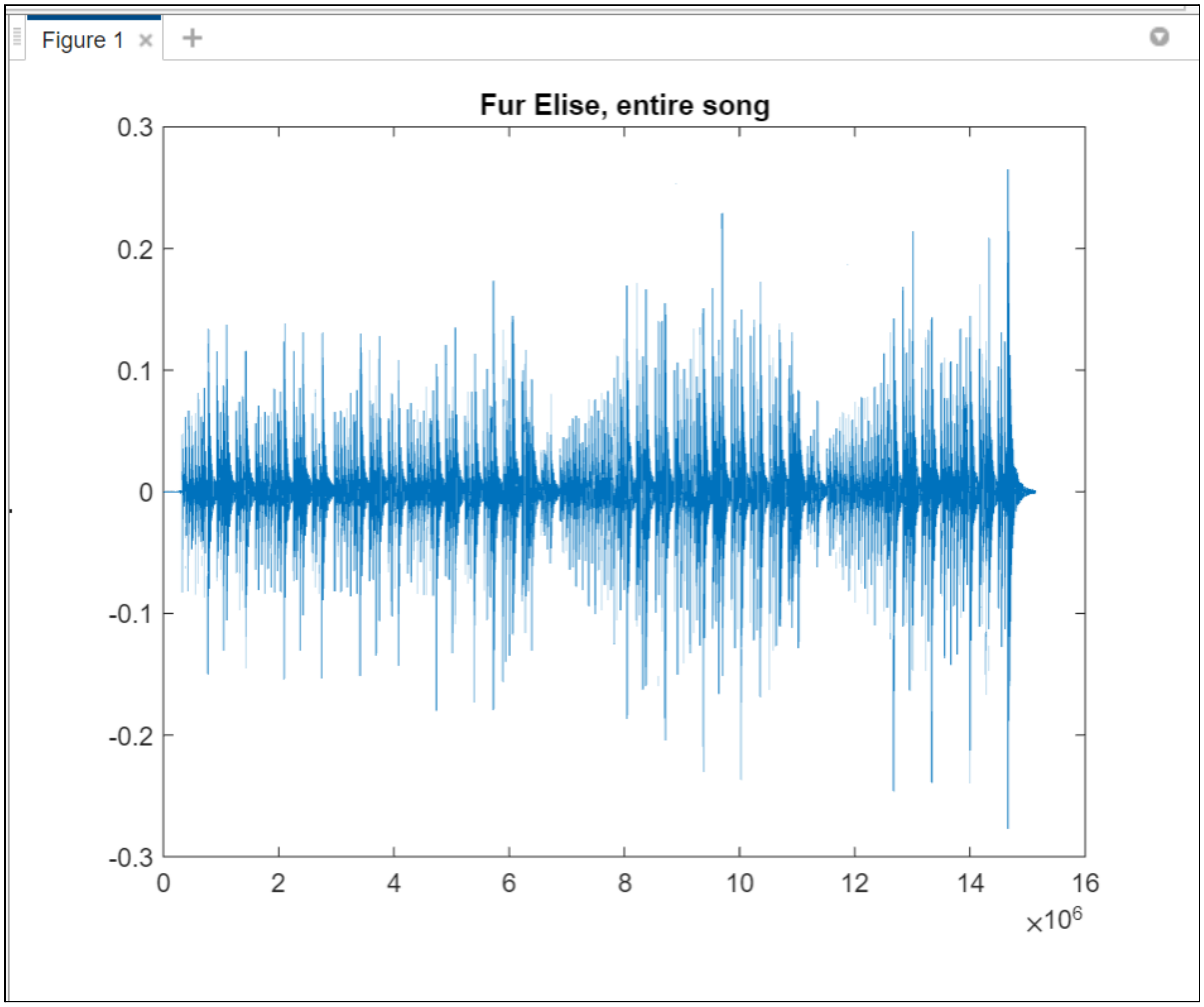
Matlab Code:

```
% Netal Asawa - 2019130003
% Sejal Gurkhe - 2019130017
% Ishita More - 2019130039
% main program 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 = 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);
```

Output:



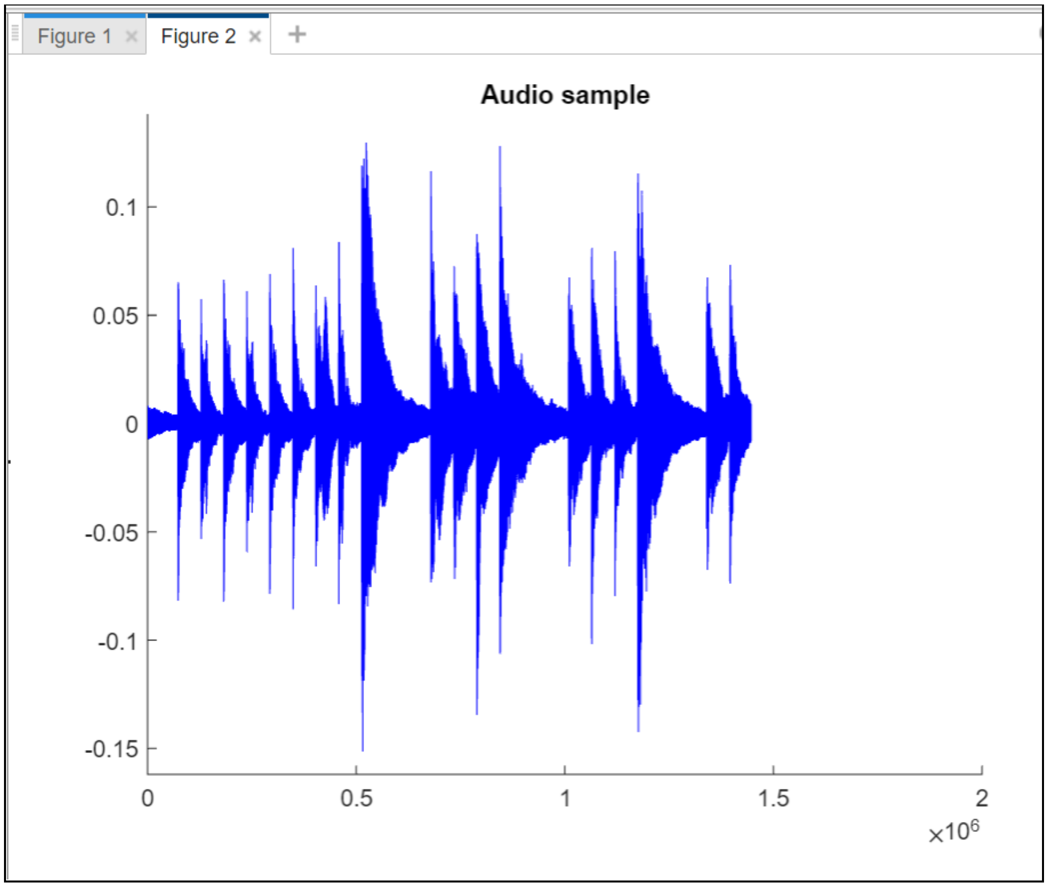


Figure 1
Figure 2
Audio sample
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
$\times 10^6$

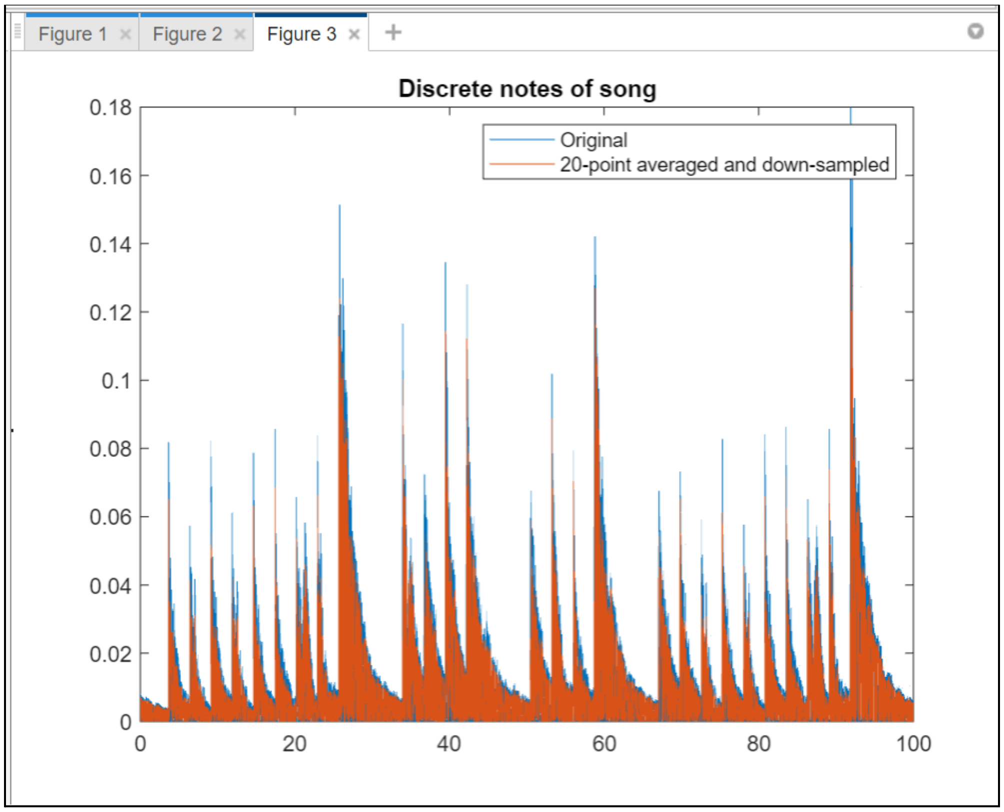


Figure 1
Figure 2
Figure 3
Discrete notes of song
Original
20-point averaged and down-sampled
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
20
40
60
80
100

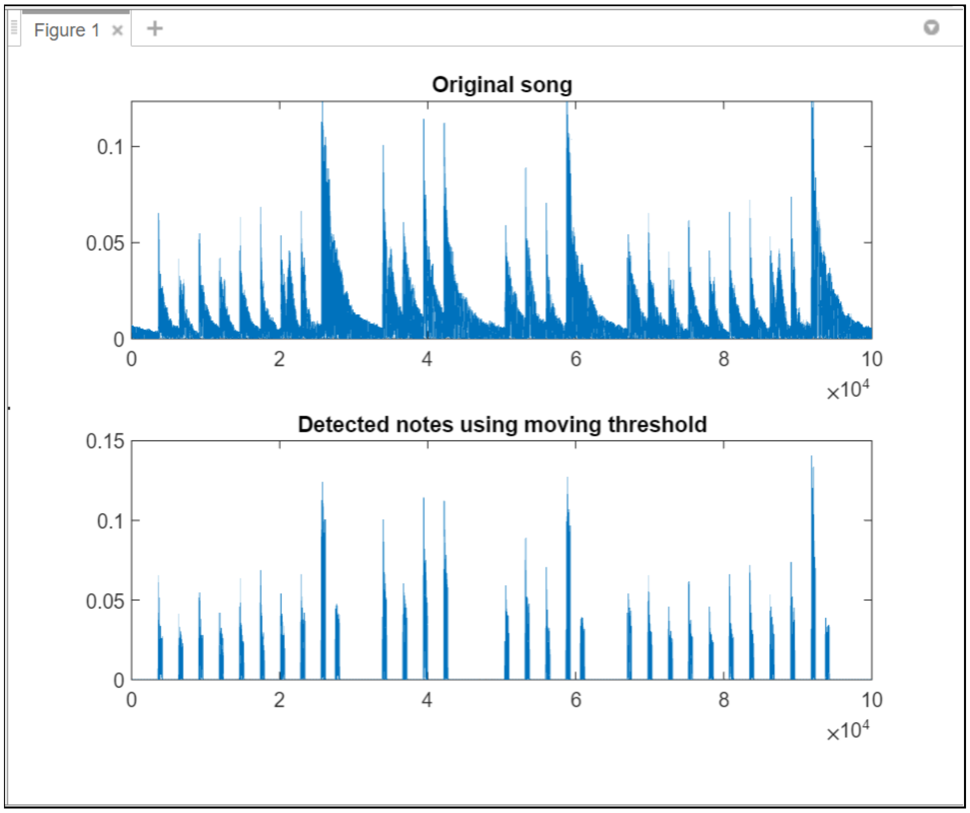
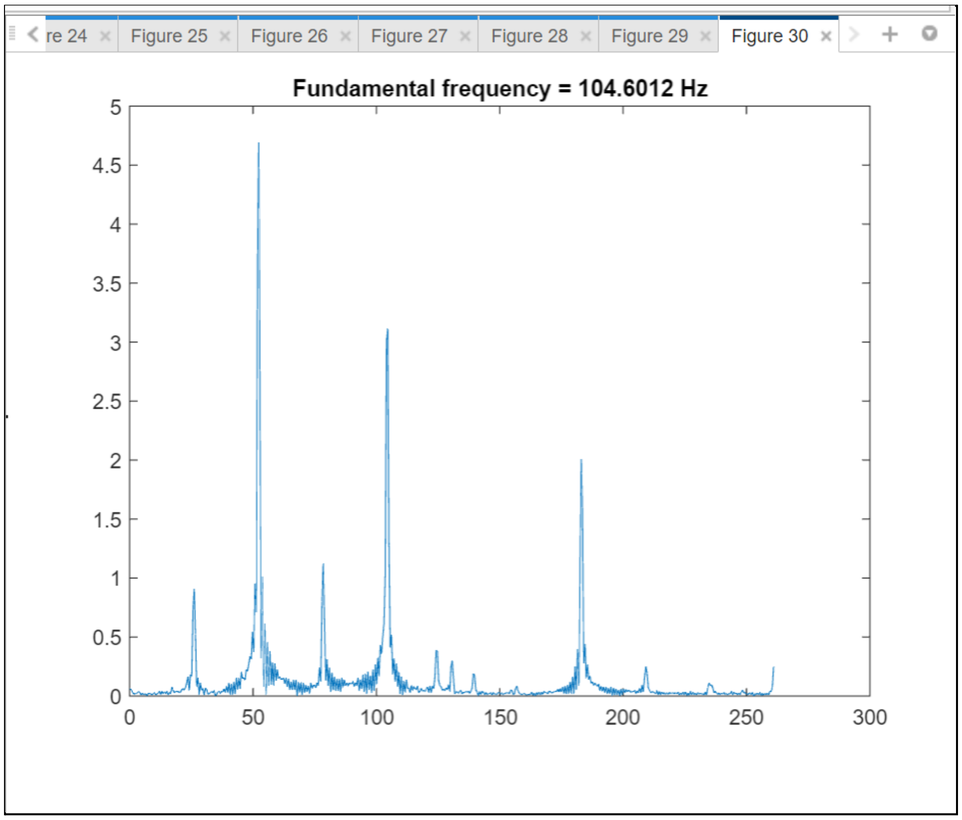
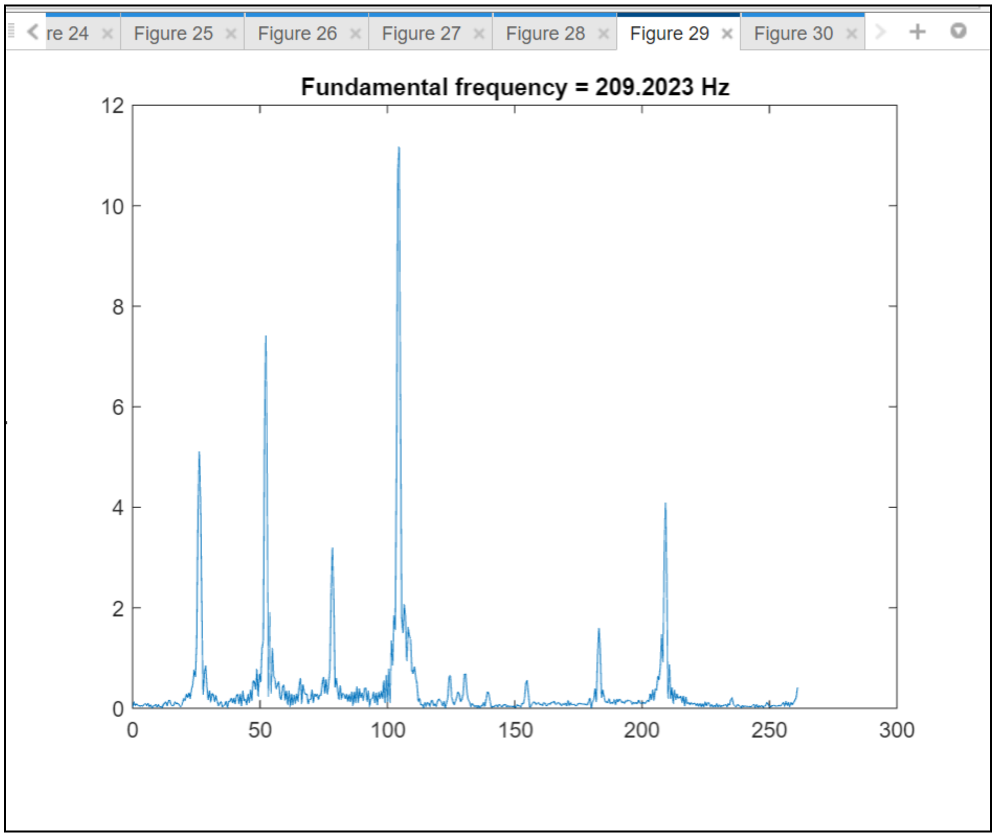
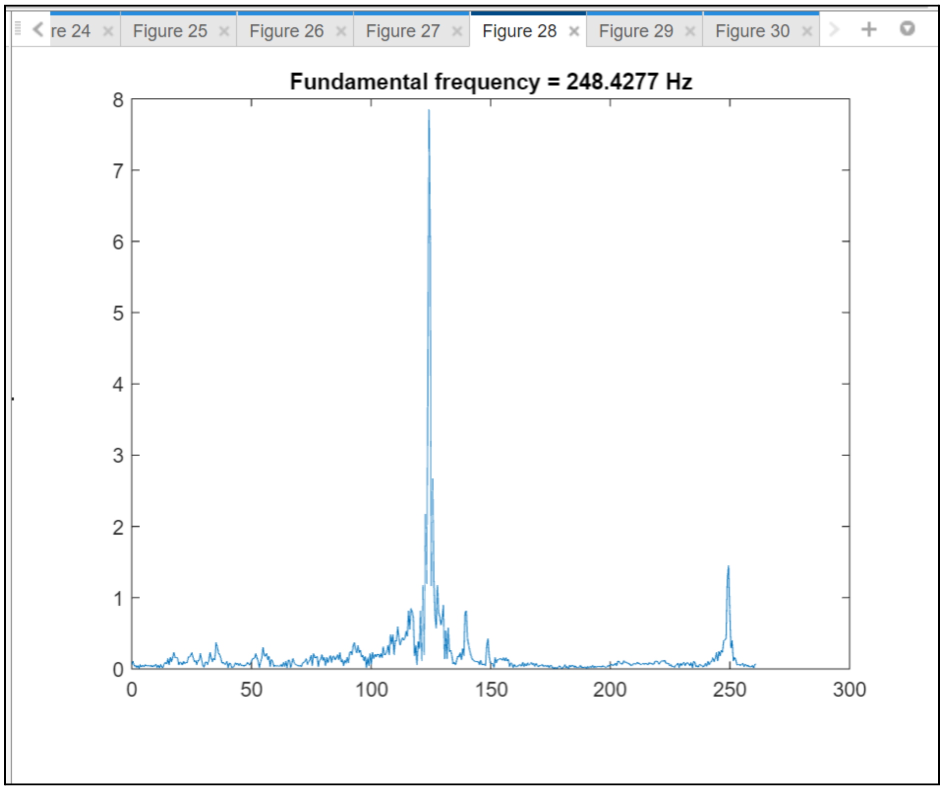


Figure 1
Original song
Detected notes using moving threshold
$\times 10^4$


re 24
Figure 25
Figure 26
Figure 27
Figure 28
Figure 29
Figure 30
Fundamental frequency = 104.6012 Hz


re 24
Figure 25
Figure 26
Figure 27
Figure 28
Figure 29
Figure 30
Fundamental frequency = 209.2023 Hz
0
2
4
6
8
10
12
0
50
100
150
200
250
300


re 24
Figure 25
Figure 26
Figure 27
Figure 28
Figure 29
Figure 30
Fundamental frequency = 248.4277 Hz
0
1
2
3
4
5
6
7
8
0
50
100
150
200
250
300

Reference:

- https://www.cs.hmc.edu/~kperdue/MusicalDSP.html
- Research Paper