Musical Note Recognition Using Matlab

Akshunn Trivedi
(19BEC0038)
Vellore Institute of Technology
Vellore,India
akshunn.trivedi2019@vitstudent.ac.in

Nikhil M Jain
(19BEC0046)
Vellore Institute of Technology
Vellore,India
nikhilm.jain2019@vitstudent.ac.in

P. Vivek Sagar
(19BEC0010)
Vellore Institute of Technology
Vellore,India
polaki.viveksagar2019@vitstudent.ac.in

***Abstract*— Musical sounds each have a certain pitch that we can differentiate as notes also known as musical notes. This project describes a way to create a system that recognizes these musical sound waves and determines the note's pitch and octave according to the standard piano middle C scale.**

***Keywords*—**musical notes, MATLAB, frequency, amplitude, *Fast Fourier Transform*

I. Introduction

Signals and sound waves are part of our everyday life. These sound waves resonate through our ears as we carry a conversation or simply listen to the everyday sounds of life. However, music is a distinct type of signal.

Notes represent the pitch and duration of a sound in musical notation. A note also represents a pitch class. The use of notes and its analysis helps us categorise different varieties of sound and obtain information about it. Notes are something that help the music or song to maintain a flow. They make up most of the melodies, harmonies, etc.

II. Objective

To classify the segments of an audio into different musical notes using the concepts of Fast Fourier Transform.

III. Musical Notes and its Frequency Calculation

The basic formula for the calculating the frequency of musical notes of the equal tempered scale is given below

$$F_n=F_0*(a)^n$$

Where,
F_0 = the frequency of a reference note. (which must be chosen) In this report we have chosen 'A' above middle 'C' '(A_4)' for which F_0 = 440 Hz.

n = the number of half steps away from the reference note for a particular specified note. If you are at a higher note, 'n' is positive. If you are on a lower note, 'n' is negative.

F_n = the frequency of the note n half steps away.

$a = (2)^{1/12}$

The following table depicts the position of musical notes n value (w.r.t. to A4 note) and its respective frequency.

Note	'n' value	Frequency(Hz)
C0	-57	16.35
C#0	-56	17.32
D0	-55	18.35
D#0	-54	19.45
E0	-53	20.60
F0	-52	21.83
F#0	-51	23.12
G0	-50	24.50

G#0	-49	25.96
A0	-48	27.50
A#0	-47	29.14
B0	-46	30.87
C1	-45	32.70
C#1	-44	34.65
D1	-43	36.71
D#1	-42	38.89
E1	-41	41.20
F1	-40	43.65
F#1	-39	46.25
G1	-38	49.00
G#1	-37	51.91
A1	-36	55.00
A#1	-35	58.27
B1	-34	61.74
C2	-33	65.41
C#2	-32	69.30
D2	-31	73.42
D#2	-30	77.78
E2	-29	82.41
F2	-28	87.31
F#2	-27	92.50
G2	-26	98.00
G#2	-25	103.83
A2	-24	110.00
A#2	-23	116.54
B2	-22	123.47
C3	-21	130.81
C#3	-20	138.59
D3	-19	146.83
D#3	-18	155.56
E3	-17	164.81
F3	-16	174.61
F#3	-15	185.00
G3	-14	196.00
G#3	-13	207.65
A3	-12	220.00
A#3	-11	233.08
B3	-10	246.94
C4	-9	261.63
C#4	-8	277.18
D4	-7	293.66
D#4	-6	311.13
E4	-5	329.63
F4	-4	349.23
F#4	-3	369.99
G4	-2	392.00
G#4	-1	415.30
A4	0	440.00
A#4	1	466.16
B4	2	493.88
C5	3	523.25
C#5	4	554.37
D5	5	587.33
D#5	6	622.25
E5	7	659.26
F5	8	698.46
F#5	9	739.99
G5	10	783.99
G#5	11	830.61
A5	12	880.00
A#5	13	932.33
B5	14	987.77
C6	15	1046.50
C#6	16	1108.73
D6	17	1174.66
D#6	18	1244.51
E6	19	1318.51
F6	20	1396.91
F#6	21	1479.98
G6	22	1567.98
G#6	23	1661.22
A6	24	1760.00
A#6	25	1864.66
B6	26	1975.53
C7	27	2093.00
C#7	28	2217.46
D7	29	2349.32
D#7	30	2489.02
E7	31	2637.02
F7	32	2793.83
F#7	33	2959.96
G7	34	3135.96
G#7	35	3322.44
A7	36	3520.00
A#7	37	3729.31
B7	38	3951.07
C8	39	4186.01
C#8	40	4434.92
D8	41	4698.64
D#8	42	4978.03
E8	43	5274.04
F8	44	5587.65
F#8	45	5919.91
G8	46	6271.93
G#8	47	6644.88
A8	48	7040.00
A#8	49	7458.62
B8	50	7902.13

IV. Platform and Important Inbuilt Functions Used for Project

The project is developed on MathWorks's MATLAB platform as it has many required built-in functions. Moreover, it is far more user friendly and has many support libraries. It can also take audio as input in any format (wav, mp3, aac, flac, AIFF, etc).

The important inbuilt functions used in MATLAB for our project are in the following.

audioread(filename): It reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

length(X): It returns the length of the largest array dimension in X. For vectors, the length is simply the number of elements.

fft(X): It computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

num2str(X): It converts a numeric array into a character array that represents the numbers.

max(X): It returns the maximum elements of an array X.

plot(X,Y): It creates a 2-D line plot of the data in Y versus the corresponding values in X.

V. High Level Working

The sound data is read and is taken as input. Window size is defined to evaluate the sound. Then the number of notes present in sound data is checked. For each evaluation window, Fast Fourier Transform is taken. Later, the note which has the frequency closest to the maximum amplitude is found out. At last, all notes are defined and are assigned to its corresponding frequency values.

VI. Code

```
% noteparse.m

function divs = noteparse(data)
len = length(data);
threshup = .2 * max(data);  %
threshdown = .04 * max(data);
quiet=1;
j=1;
for i=51:len-50
   k=i-50:i+50;
  if quiet == 1
    if (max(abs(data(k))) > threshup)
      quiet = 0;
      divs(j) = i;
      j=j+1;
    end
   else
    if (max(abs(data(k))) < threshdown)
      quiet = 1;
      divs(j) = i;
      j=j+1;
    end
  end
end



% notewindow.m

function w = notewindows(data)
divs = noteparse(data);
d2(1) = 0;
for i=1:length(divs)
   d2(i+1)=divs(i);
end
d2(length(divs)+2) = length(data);
for i=1:length(d2)/2
   w(i) = (d2(2*i-1) + d2(2*i))/2;
end



% analyse.m

clc
clear all
file='tuning_fork_A4.wav';
[x,fs]=audioread(file);
w = noteparse(x);
num_notes=length(w)-1
Ns = length(x);
t = (1/fs)*(1:Ns);
Xk = abs(fft(x));
Xk= Xk(1:Ns/2);
f = fs*(0:Ns/2-1)/Ns;
xk=Xk/max(Xk);
figure(1)
subplot(2,1,1);
```

```
plot(t, x,'color','blue')
xlabel('Time (s)')
ylabel('Amplitude')
title('Input Audio Signal');
subplot(2,1,2);
plot(f, xk)
hold on
xlim([0 (max(f)+100)])
ylim([0,1.1])
xlabel('Frequency (Hz)')
ylabel('Normalisied Amplitude')
title('Magnitude spectrum of FFT');
mainNames = char('C', 'C#', 'D', 'D#', 'E', 'F', 'F#',
'G', 'G#', 'A', 'A#', 'B');
names = char('A0', 'A#0' ,'B0');
A0 = 27.5;
ind = 4;
for i = 0:87
   data(i+1) = A0 * (2^(1/12))^i;
   if i>2
       a = [mainNames( rem((ind - 4), 12)+1,:)
num2str(fix((ind - 4)/12)+1)];
       names = char(names, a);
       ind = ind + 1;
   end
end
bands(1) = 20;
for i = 1:87
   bands(i+1) = +7+(data(i) + data(i+1))/2;
end
bands(89) = 4500;
   for i=1:88
      freqBand(i) = mean(Xk(
round(bands(i)/(fs/Ns)
):round(bands(i+1)/(fs/Ns))))^2;
       freqband(i)=freqBand(i);
   end
   for i=1:num_notes
   m = find(freqband == max(freqband(:)));
   disp(names(m,:))
 freqband=freqband(freqband~=max(freqband));

   end
hold off
```

Output Results and Conclusion

Different audio sounds which consists of musical notes were kept as an input for this code and for each sound data it has displayed number of notes and the notes present. For our refence, input audio data (Amplitude vs time) and its magnitude spectrum of Fourier transform (Normalised Amplitude vs Frequency) are also plotted which gave us a broader perspective of audio sound data.

Given below is a displayed for an input audio sound of a song we took as input. In Fig.1 its graph of input audio signal (Amplitude vs time) as well as its magnitude spectrum of its Fourier transform (Normalised Amplitude vs Frequency) is depicted. In Fig. 2 the actual output i.e., number of notes and notes present is shown.

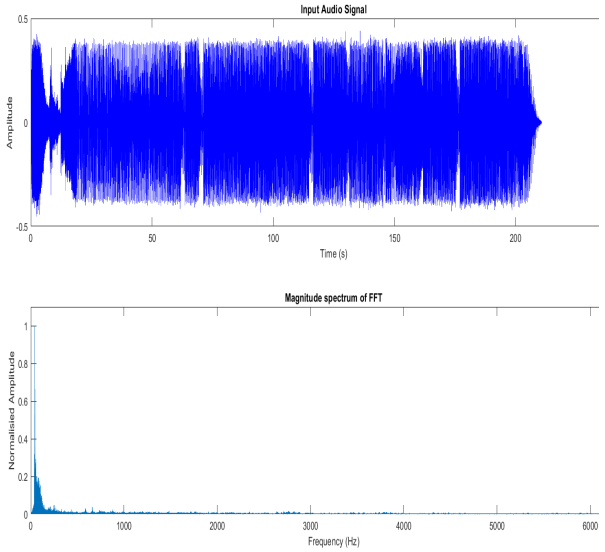
Fig.1

```
len =

     9294960


num_notes =

    40

D 1
C#1
C 1
F#1
G#1
E 1
D 1
C 1
C 2
C#1
B0
C#1
B0
B0
B0
E 1
D#1
F 1
B0
C#1
```

Fig.2

```
C#1
B0
B0
B0
E 1
D#1
F 1
B0
C#1
D 1
C 1
C 1
B0
A#0
C 1
C 3
A#0
G 1
D 1
C 3
E 1
G 1
D 1
G 2
E 1
C 1
E 1
A#0
F#2
```

Fig 3



Thus, the working of our code is verified.

Attachments:
Link for our presentation:
https://docs.google.com/presentation/d/1B6qFbp5Raiiw786l8_bvXOMAQIdJ0t6kyTgbd1ZQAqE/edit?usp=sharing

Link for the code explanation:

https://www.youtube.com/watch?v=3-Ik5tBDrfE

ACKNOWLEDGMENT

We would like to take this opportunity to express our special thanks to our teacher Prof. Saranya KC who gave us this golden opportunity to do this project. Along with her active guidance she has also provided us immense support without which we couldn't have completed our project.