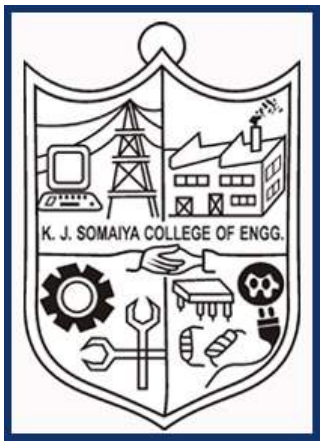


K. J. SOMAIYA COLLEGE OF ENGINEERING

2UXC-601

DIGITAL SIGNAL PROCESSING

Submitted To:
Dr. Jagannath Nirmal
Head of Department
Department of Electronics
Prof. Bhargavi Kaslikar
Professor
Department of Electronics

Submitted By :
Meet Doshi
1812078
Annas Khan
1812080
Rohit Singh
1812083
Semester VI

CONTENTS

[1]

[1]

Date: February 28, 2021

Corresponding authors:
Meet Doshi(email: meet10@somaiya.edu)
Annas Khan(email: annas.khan@somaiya.edu)
Rohit Singh(email: rohit.ss@somaiya.edu)

CASE STUDY: MUSIC NOTE EXTRACTION

Meet Doshi[1], Annas Khan[2], Rohit Singh[3],
Department of Electronics Engineering
K.J. Somaiya College of Engineering, Vidyavihar, 400077, Maharashtra, India

[1]

Index Terms

IEEE, IEEEtran, Digital Signal Processing, Audio Extraction, Audio to Text, Signal Processing for Music Analysis,Simultaneous Estimation of Chords and Musical Context From Audio, Estimating Note Intensities in Music Recordings.

I. ABSTRACT

Music signal processing may appear to be the junior relation of the large and mature field of speech signal processing, not least because many techniques and representations originally developed for speech have been applied to music, often with good results. However, music signals possess specific acoustic and structural characteristics that distinguish them from spoken language or other non-musical signals. This paper provides an overview of some signal analysis techniques that specifically address musical dimensions such as melody, harmony, rhythm, and timbre.We will examine how particular characteristics of music signals impact and determine these techniques, and we highlight a number of novel music analysis and retrieval tasks that such processing makes possible. Our goal is to demonstrate that, to be successful, music audio signal processing techniques must be informed by a deep and thorough insight into the nature of music itself.

II. INTRODUCTION

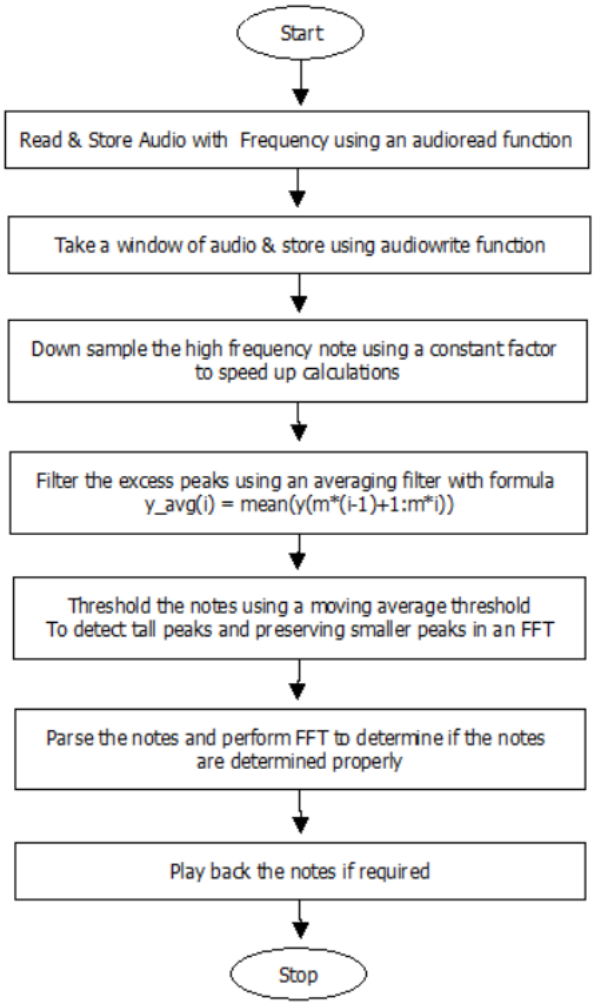
Music is a ubiquitous and vital part of the lives of billions of people worldwide. Musical creations and performances are among the most complex and intricate of our cultural artifacts, and the emotional power of music can touch us in surprising and profound ways. Music spans an enormous range of forms and styles, from simple, unaccompanied folk songs, to orchestras and other large ensembles, to a minutely constructed piece of electronic music resulting from months of work in the studio. The revolution in music distribution and storage brought about by personal digital technology has simultaneously fueled tremendous interest in and attention to the ways that information technology can be applied to this kind of content. From browsing personal collections, to discovering new artists, to managing and protecting the rights of music creators, computers are now deeply involved in almost every aspect of music consumption, which is not even to mention their vital role in much of today's music production[1]. This paper concerns to study a methodology to make raw music extraction code to using sophisticated signal processing techniques to make the process less expensive in terms of time and processing power. We first take a look into a algorithm which can be used to convert any audio file to its respective musical notes. Then we show what methods can be used in this process to make the application faster and smoother whilst also preserving musical notes.

[1]

Date: February 28, 2021
Corresponding authors:
Meet Doshi(email: meet10@somaiya.edu)
Annas Khan(email: annas.khan@somaiya.edu)
Rohit Singh(email: rohit.ss@somaiya.edu)

III. PROCEDURE

A. *FLOWCHART*



B. *ALGORITHM*

Now we will define the exact algorithm which is used for our application. The application is based on Für Elise Bagatelle by Giscard Rasquin and Ludwig van Beethoven audio.

Steps:

1) Use MATLAB's audioread function as follows
   [song,Fs] = audioread('Name of the file');
   This will store the sampled audio file data in 'song' and the sample rate in Fs.
2) Speed up the song by a constant factor is the song is too slow
   Fs=Fs*4;
3) Window a part of the song for analysing like this
   y = song(t1:t2);
   Now we have a sampled data of the real audio file from time t1 up to t2. Save the windowed audio file for future debugging.
4) Down sample the audio file to compute FFT and process audio faster
   Fsm = round(Fs/20); Down Sampling factor 20.
5) Create a new vector yavg and initialize to 0's for filtering.
   Use a for loop to use averaging filter

for i = 1:p
yavg(i) = mean(y(m*(i-1)+1:m*i));
Equation for the averaging filter

$$y[n] = \frac{1}{N}\sum_{i=0}^{N-1} x[n-i]$$

6) Calculate a threshold using the formula

$$thresh = 5 \times Median[|\ yavg(\max(1, i-5000) : i)\ |]$$

7) Threshold captures high frequency peaks whilst saving smaller important ones too.
if $yavg[i] > thresh$
for $j = 0 : 500$
if $i + j <= p$
$ythresh[i] = yavg[i]$
i = i+1
8) Create a function FreqToNote(f) which converts frequencies to their respective notes.
9) Now go through each note which was threshold and find their frequency using FreqToNote(f) function.
10) Play back each note and plot their FFT to make sure the notes are correctly mapped.

IV. CODE

```
% 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('C:\Users\meetd\Down-
loads\FurElise_Slow.mp3');
Fs=Fs*4; %Speed up the song
%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('C:\Users\meetd\Downloads\fur_elise_win-
dow.wav',y,Fs);

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

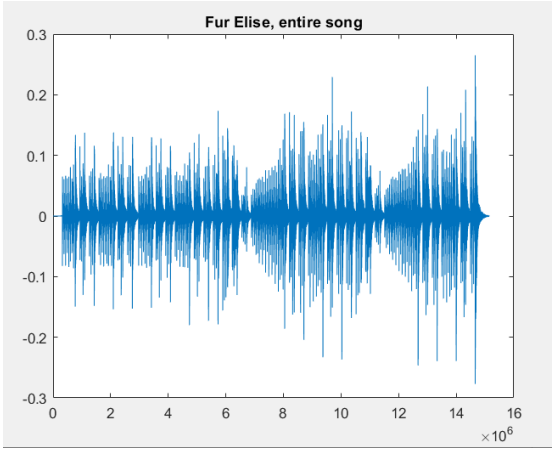
```
%% downsample by m
clc
m = 20;
Fsm = round(Fs/m);
p = floor(n/m);
y_avg = zeros(1,p);
for i = 1:p
    y_avg(i) = mean(y(m*(i-1)+1:m*i));
end
%figure, plot(linspace(0,100,n),abs(y)), hold on
        %plot(linspace(0,100,p),abs(y_avg))
       % title('Discrete notes of song')
        %legend('Original', '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('De-
tected 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
```

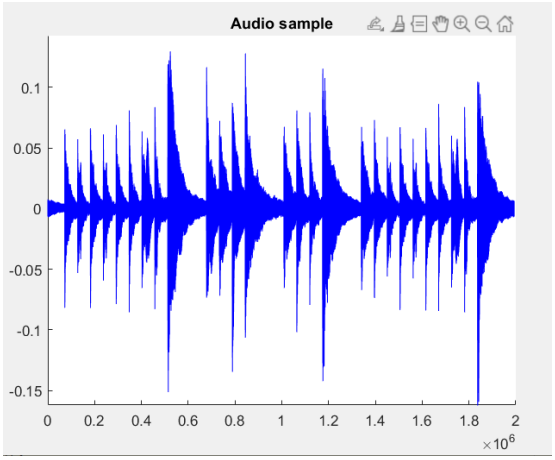
```
audiowrite('C:\Users\meetd\Downloads\fur_elise_recre-
ated.wav',recreate_song,fs);


%%helper functions
function [note,frequency] = FreqToNote(f)
    index = round(17.31232*log(f) - 47.37620);
    frequencies = [16.35 17.32 18.35 19.45 20.6 21.83 23.12
24.5 25.96 27.5 29.14 30.87 32.7 34.65 36.71 38.89 41.2
43.65 46.25 49 51.91 55 58.27 61.74 65.41 69.3 73.42 77.78
82.41 87.31 92.5 98 103.83 110 116.54 123.47 130.81 138.59
146.83 155.56 164.81 174.61 185 196 207.65 220 233.08
246.94 261.63 277.18 293.66 311.13 329.63 349.23 369.99 392
415.3 440 466.16 493.88 523.25 554.37 587.33 622.25 659.25
698.46 739.99 783.99 830.61 880 932.33 987.77 1046.5
1108.73 1174.66 1244.51 1318.51 1396.91 1479.98 1567.98
1661.22 1760 1864.66 1975.53 2093 2217.46 2349.32 2489.02
2637.02 2793.83 2959.96 3135.96 3322.44 3520 3729.31
3951.07 4186.01 4434.92 4698.63 4978.03 5274.04 5587.65
5919.91 6271.93 6644.88 7040 7458.62 7902.13];
    notes = ["C0" "Db0" "D0" "Eb0" "E0" "F0" "Gb0" "G0"
"Ab0" "A0" "Bb0" "B0" "C1" "Db1" "D1" "Eb1" "E1" "F1" "Gb1"
"G1" "Ab1" "A1" "Bb1" "B1" "C2" "Db2" "D2" "Eb2" "E2" "F2"
"Gb2" "G2" "Ab2" "A2" "Bb2" "B2" "C3" "Db3" "D3" "Eb3" "E3"
"F3" "Gb3" "G3" "Ab3" "A3" "Bb3" "B3" "C4" "Db4" "D4" "Eb4"
"E4" "F4" "Gb4" "G4" "Ab4" "A4" "Bb4" "B4" "C5" "Db5" "D5"
"Eb5" "E5" "F5" "Gb5" "G5" "Ab5" "A5" "Bb5" "B5" "C6" "Db6"
"D6" "Eb6" "E6" "F6" "Gb6" "G6" "Ab6" "A6" "Bb6" "B6" "C7"
"Db7" "D7" "Eb7" "E7" "F7" "Gb7" "G7" "Ab7" "A7" "Bb7" "B7"
"C8" "Db8" "D8" "Eb8" "E8" "F8" "Gb8" "G8" "Ab8" "A8" "Bb8"
"B8"];
    note = notes(index);
    frequency = frequencies(index);
end

function plotsound(y,Fs)
waitTime = .1;
samples = ceil(Fs*waitTime);
figure, hold on, xlim([0 length(y)]), ylim([.9*min(y)
1.1*max(y)])
        title('Audio sample')
sound(y,Fs)
    for i = 1:samples:length(y)-samples
        plot(i:i+samples,y(i:i+samples),'b')
        pause(waitTime)
    end
end
```

End of document

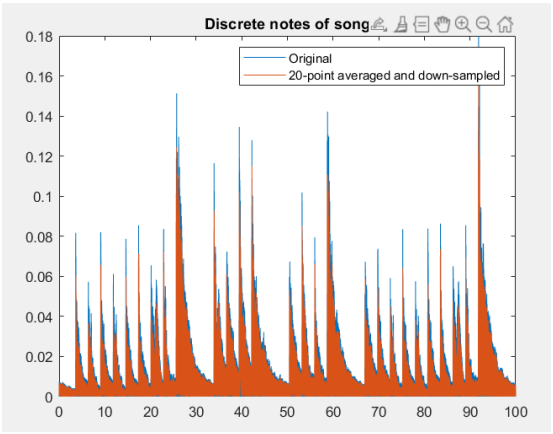
V. SIMULATION



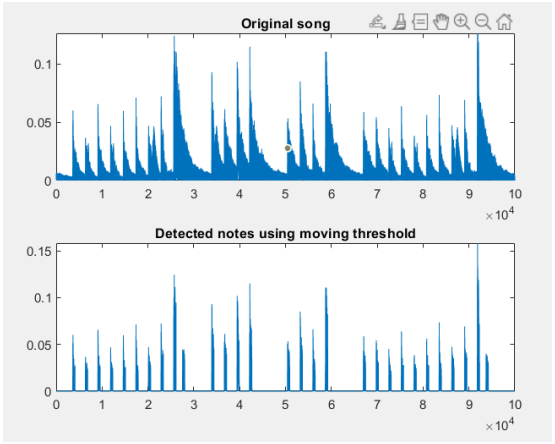
(a) Whole Audio



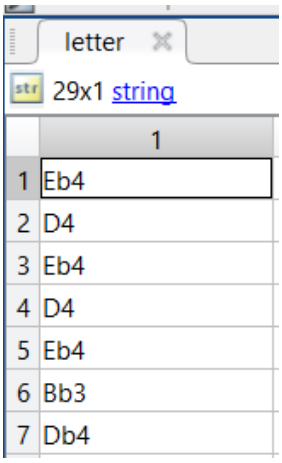
(b) Windowed Audio



(c) Discrete Notes of the audio



(d) Detected Notes



(e) Frequency converted to Notes

VI. **RESULTS**

First 19 notes from the produced array *letter* from the program.

Sr. No	*Actual Notes*	*Notes Generated*
1	*E*	*Eb4*
2	*Eb*	*Eb4*
3	*E*	*D4*
4	*Eb*	*Eb4*
5	*E*	*D4*
6	*B-*	*Bb3*
7	*D*	*Db4*
8	*C*	*B3*
9	*A-*	*Ab3*
10	*C-*	*Ab2*
11	*E-*	*B2*
12	*A-*	*Eb3*
13	*B-*	*Ab3*
14	*E-*	*Bb3*
15	*Ab-*	*Eb3*
16	*B-*	*G3*
17	*C*	*Bb3*
18	*E-*	*B3*
19	*E*	*Ab2*

We have used numeric notation for tones in replacement of '+' & '-' symbols. The actual notes are derived from rhythmic notes from piano, whilst the notes generated are a function of time divided into equal duration of 2 seconds from the song. When we detect notes from the windowed file & compare it to the normalized FFT of the window we see that all the peaks detected in the FFT are saved and their highest frequencies are noted. These noted frequencies are the basis of the note conversion. We refer the large peaks for notes calculation but also preserve the smaller ones for precision. This way the notes calculated are an accurate function of time for the *letter* array generated which consists of the notes required.

VII. **APPLICATIONS**

Here are some of the applications which are useful today.

1) Tonal representations
   Especially chroma features—have been used for a wide range of music analysis and retrieval tasks in which it is more important to capture polyphonic musical content without necessarily being concerned about the instrumentation. Such applications include chord recognition, alignment, "cover song" detection, and structure analysis.
2) Creating new audio notes for new songs.
   Everyday hundreds of songs are released, artists have to keep up with instruments to practice that song. But creating notes for every song with such a pace takes enormous time. Instead this program can be modified to each artist's needs for an instrument. This way artist can concentrate on practising more rather than creating notes.
3) Deaf culture.
   Due to inability for hearing, deaf people communicate via sign language. This program makes their job easier to communicate notes of music to other people. If combined with a sign language generator this program can even help them communicate entire songs with a great pace just by refering to the output of sign language generator.
4) Google Search for Music
   People often tend to forget some songs which they have heard before but have trouble remembering them. We can split entire songs into smaller windows of notes and with the help of audio processor we can find a song just by a person trying to humm a rhythm.

VIII. CONCLUSION

[2] Signal processing for music analysis is a vibrant and rapidly evolving field of research, which can enrich the wider signal processing community with exciting applications and new problems. Music is arguably the most intricate and carefully constructed of any sound signal, and extracting information of relevance to listeners therefore requires the kinds of specialized methods that we have presented, able to take account of music specific characteristics including pitches, harmony, rhythm, and instrumentation. It is clear, however, that these techniques are not the end of the story for analyzing music audio, and many open questions and research areas remain to be more fully explored. Some of the most pressing and promising are listed below.

- Decomposing a complex music signal into different components can be a powerful preprocessing step for many applications. For example, since a cover song may preserve the original melody but alter the harmonization, or alternatively keep the same basic chord progression but devise a new melody (but rarely both), a promising approach to cover version recognition is to separate the main melody and accompaniment, and search in both parts independently
- A different decomposition, into harmonic and percussive components, has brought benefits to tasks such as chord recognition, genre classification, and beat tracking. Note that such decomposition need not be perfect to yield benefits—even a modest improvement in the relative level between components can give a significant improvement in a subsequent analysis.
- Improved recognition and separation of sources in polyphonic audio remains a considerable challenge, with great potential to improve both music processing and much broader applications in computational auditory scene analysis and noise-robust speech recognition. In particular, the development, adaptation, and exploitation of sound source models for the purpose of source separation seems to be required in order to achieve an accuracy comparable to that of human listeners in dealing with polyphonic audio.
- In conjunction with the appropriate signal processing and representations, machine learning has had some great successes in music signal analysis. However, many areas are limited by the availability of high-quality labeled data. In chord recognition, for instance, the entire field is using the same corpus of 200 tracks for which high-quality manual chord transcripts have been prepared. However, while special-purpose human labeling remains the gold standard, it is interesting to note that a given piece of music may have multiple, closely related sources of information, including alternate recordings or performances, partial mixes derived from the original studio multi tracks, score representations including MIDI versions, lyric transcriptions, etc. These different kinds of information, some available in large quantities, present opportunities for innovative processing that can solve otherwise intractable problems such as score-guided separation, generate substitutes for manual ground-truth labels using music synchronization techniques or use multi-perspective approaches to automatically evaluate algorithms.

REFERENCES

[1] Signal Processing for Music Analysis Meinard Müller, Member, IEEE, Daniel P. W. Ellis, Senior Member, IEEE, Anssi Klapuri, Member, IEEE, and Gaël Richard, Senior Member, IEEE

[2] Simultaneous Estimation of Chords and Musical Context From Audio Matthias Mauch, Student Member, IEEE, and Simon Dixon, Member, IEEE

[3] ESTIMATING NOTE INTENSITIES IN MUSIC RECORDINGS Sebastian Ewert University of Bonn Bonn, Germany Meinard Muller Saarland University and MPI Informatik Saarbrucken, Germany

[4] https://mypianonotes.com/fur-elise/

[5] https://in.mathworks.com/help/audio/index.html

[6] Identifying Missing and Extra Notes in Piano Recordings Using Score-Informed Dictionary Learning Siying Wang, Student Member, IEEE, Sebastian Ewert, Member, IEEE, and Simon Dixon

[7] Piano Transcription in the Studio Using an Extensible Alternating Directions Framework

[8] Music Conversion from Synthetic Piano to Chinese Guzheng Using Image-based Deep Learning Technique

[2]

Date: February 28, 2021

Corresponding authors:
Meet Doshi(email: meet10@somaiya.edu)
Annas Khan(email: annas.khan@somaiya.edu)
Rohit Singh(email: rohit.ss@somaiya.edu)