Digital Signal Processing Using PBL Approach

23EECC303

“ MATLAB Based Music Notes Extraction Using Fast Fourier Transform ”

Submitted by:

Sl.No.	Name	SRN	Signature
1	Praveen Magadum	02FE21BEC064	
2	Chidambar patil	02FE21BEC026	
3	Komal Melavanki	02FE21BEC042	
4	Laxmi Kammar	02FE21BEC044	

Submitted for the partial fulfillment of the course

Mentored by:

Prof. S.B Kulkarni

Prof/Assistant Professor,

Dept. of E&CE,

KLE Technological University, Dr. M S Sheshgiri Campus, Belagavi.

Academic Year 2023-24

Contents

1 Problem Statement 3
2 Abstract 3
3 Introduction 3
4 Literature Survey 4
5 Characteristics of a signal 4
6 Data set used for the project 5
7 Block Diagram 5
8 Frequency and Fourier transform 6
9 Implementation steps 7
10 Results 8
11 Conclusion 9
12 References 9
13 Appendex
Appendex A1 1
Appendex A2 3

1. Problem Statement

Develop a System that extracts musical notes from an audio signal using the Fast Fourier Transform (FFT) technique.

2. Abstract

Signals and sound waves are part of our everyday life. However, music is a distinct type of signal. Musical sounds each have a certain pitch that we can differentiate as notes. A song contains basically two things, vocal and background music. Reading a song on sheet music and then playing it on an instrument is an easy task for any musician. In this century, computer software has also been designed to do just this. Programs can create audio files (music we can hear) from sheet music very effectively for a whole range of instruments. A major problem is that the reverse task, listening to or recording audible music and then generating the sheet music for that piece, is much more difficult to complete for both computers and talented musicians alike. Extracting the characteristics of a song becomes more important for various objectives like learning, teaching, and composing. The idea of this project is to develop a program that would take an audio input (a song) and process it, in order to give musical notes as an output.

Keywords:
Time -frequency analysis, Musical Note, Sampling frequency.

3. Introduction

Songs play a vital role in our day-to-day life. A song contains two things, vocal and background music. Where the characteristics of the voice dependent the singer and in the case of background music, it involves a mixture of different musical instruments like piano, guitar, drum, etc. Extracting the characteristic of a song becomes more important for various objectives like learning, teaching, and composing.

This project takes the 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 are used to identify the characteristics. The event detection is carried out using the time domain analysis of the signal, this occurs when one pitch whose fundamental frequency is an integer multiple of another pitch. This project aims to propose methods to analyze and describe a signal, from which the musical parameters can be easily and objectively obtained, in a sensible manner. A common limitation found in the musical literature is that how such parameters are obtained is intuitively satisfactory but, to our view, not very sound from a signal processing perspective.

4. Literature Survey

[1] – This Paper provides a comprehensive overview of digital signal processing techniques for music synthesis and recognition. It covers topics such as Fourier analysis, filter design, and audio coding.

[2]- This Paper provides a foundational introduction to signal processing concepts with a focus on applications in sound and vibration. It covers topics such as sampling, filtering, and spectral analysis.

[3] - This Paper explores the intersection of music and signal processing, covering topics such as music perception, music representation, and music analysis. It provides a comprehensive overview of the field and discusses the latest research advances.

5. Characteristics of Audio signals

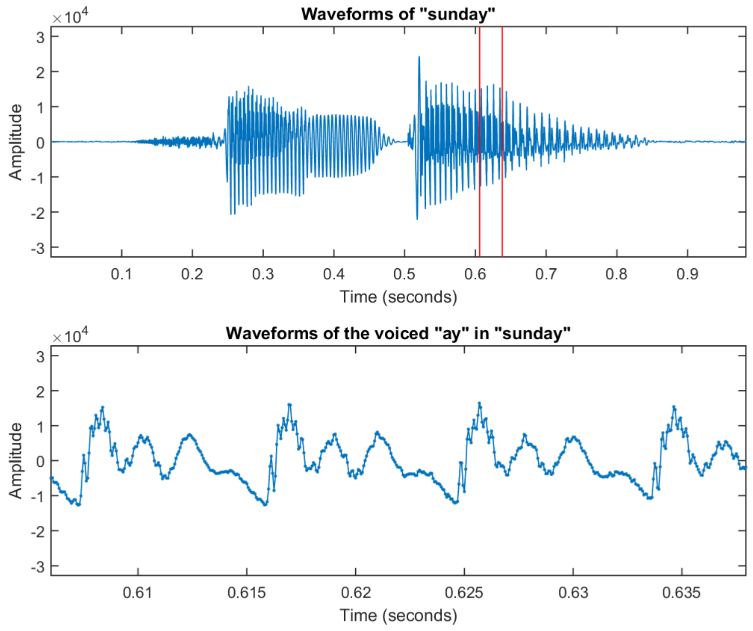


Figure 1: waveforms of Audio signal

1. **Time-varying:** Audio signals represent sound pressure variations over time, constantly changing with the sound itself.

2. **Frequency content:** Human hearing typically perceives frequencies between 20 Hz and 20 kHz. DSP techniques often analyze and manipulate specific frequency ranges within this spectrum.

3. **Non-stationary behavior:** The frequency content and amplitude of audio signals often change over time, requiring time-frequency analysis techniques in DSP.

4. **Non-linearity:** Real-world sounds often exhibit non-linear behavior, requiring specialized DSP algorithms for accurate manipulation and synthesis.

6. Data set used for our project

The data used in this project is mentioned below:

1. Fur Elise Slow Piano Dataset
   https://github.com/nayirik/MusicNoteExtraction/blob/master/FurElise_Slow.mp3
2. Raw Music file
   https://www.kaggle.com/datasets/kishanj/music-notes-datasets

7. Block Diagram

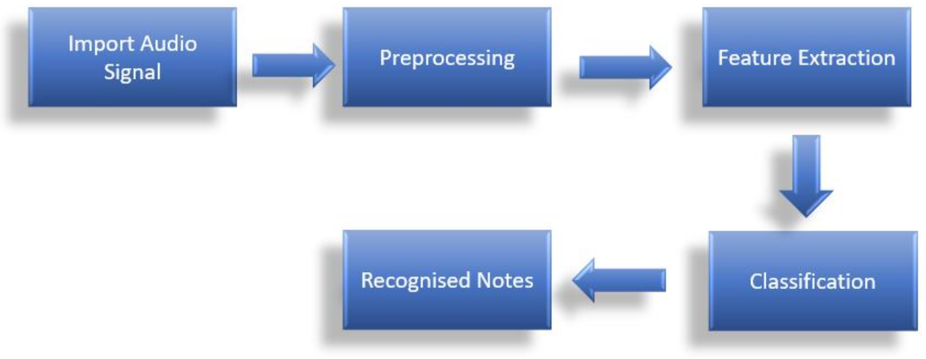


Figure 1 : Block diagram

- Import Audio Signal:
    - Acquire the audio signal from a digital source (e.g., a file).
    - Load the signal into the processing environment.
- Preprocessing:
    - Prepare the signal for analysis:
    - Sampling Rate Adjustment: Ensure consistent sampling rate for accurate frequency measurements.
    - Noise Reduction: Minimize unwanted noise using techniques like filtering or noise estimation.
    - Windowing: Apply window functions (e.g., Hann, Hamming) to reduce spectral leakage.
- Feature Extraction (FFT):
    - Apply the Fast Fourier Transform (FFT) to each frame:
    - Decomposes the time-domain signal into its constituent frequency components.
- Recognized Notes:
    - Identify dominant frequencies in the FFT output
- Result will be stored in an array
-

8 . Frequency and Fourier transforms (FFT)

A Fourier transform provides the means to break up a complicated signal, like musical tone, into its constituent sinusoids. This method involves many integrals and a continuous signal. We want to perform a Fourier transform on a sampled (rather than continuous) signal, so we have to use the Discrete Fourier Transform instead. The "Fast Fourier Transform" (FFT) is an important measurement method in the science of audio and acoustics measurement.

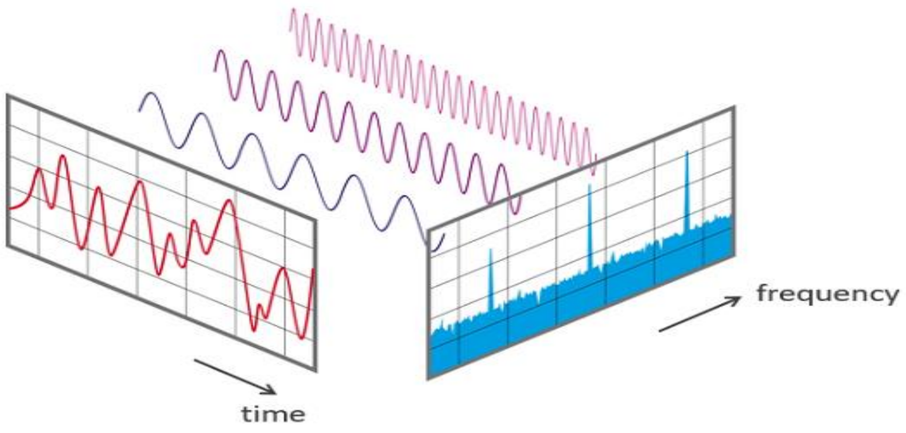
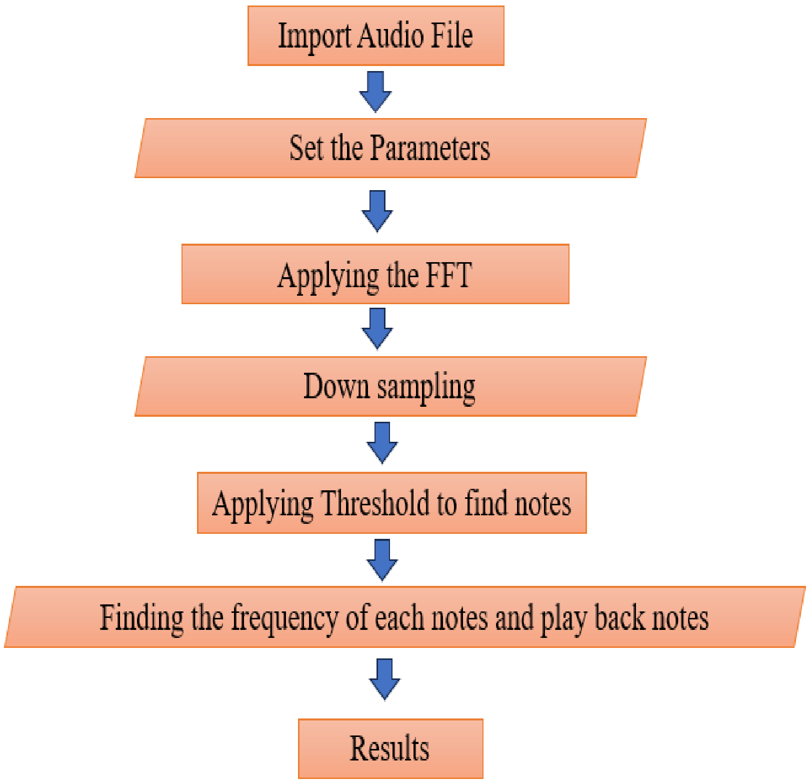


Figure 2 : View of frequency domain and time domai

It converts a signal into individual spectral components and thereby provides frequency information about the signal. FFTs are used for fault analysis, quality control, and condition monitoring of machines or systems. This article explains how an FFT works, the relevant parameters, and their effects on the measurement result. Strictly speaking, the FFT is an optimized algorithm for the implementation of the "Discrete Fourier Transformation" (DFT). A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT) .Fourier analysis converts a signal from its original domain (often time or space) to are presentation in the frequency domain and vice versa.

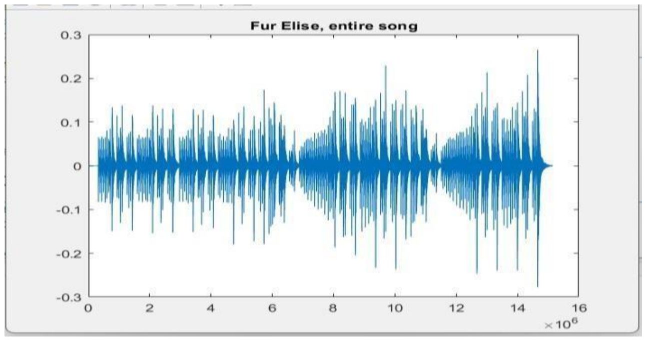
The audio signals stored in the computer are different in format, sampling rate,the number of bits, or the original audio signal containing sharp noise, which can affect the processing effect. At the same time, the unit of audio processing is in a frame, so before the feature extraction, the original audio data needs to be pre-processed: into a unified format, pre-emphasis, segmentation, and windowing framing.

8. Implementation Steps


Import Audio File
Set the Parameters
Applying the FFT
Down sampling
Applying Threshold to find notes
Finding the frequency of each notes and play back notes
Results

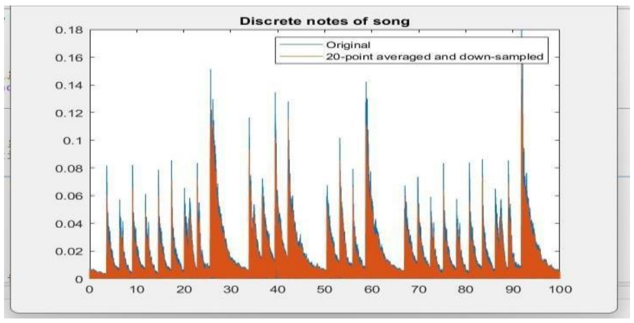
8. Results

1. Imported Audio File (Fur elise dataset)



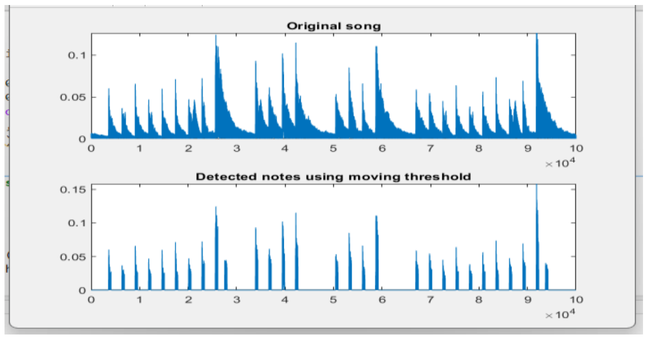
Loads and analyzes a specific window of the Fur Elise song.

2. Results of Down sampling



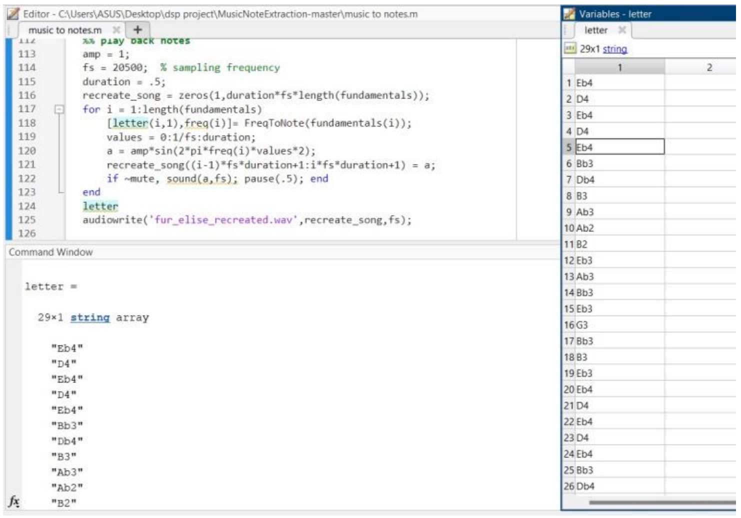
Down samples the data to focus on prominent notes**.**

3. Results of moving threshold



- Applies a dynamic threshold to identify potential notes.
- Extracts the fundamental frequency of each detected note using FFTanalysis.

4. Converted Notes



Editor - C:\Users\ASUS\Desktop\dsp project\MusicNoteExtraction-master\music to notes.m

```
%% 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);
```

Command Window

```
letter =

  29×1 string array

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

Variables - letter

29x1 string

	1	2
1	Eb4	
2	D4	
3	Eb4	
4	D4	
5	Eb4	
6	Bb3	
7	Db4	
8	B3	
9	Ab3	
10	Ab2	
11	B2	
12	Eb3	
13	Ab3	
14	Bb3	
15	Eb3	
16	G3	
17	Bb3	
18	B3	
19	Eb3	
20	Eb4	
21	D4	
22	Eb4	
23	D4	
24	Eb4	
25	Bb3	
26	Db4	

10. Conclusion

The application of Fast Fourier Transform (FFT) in digital signal processing for music notes extraction has demonstrated its efficacy in dissecting audio signals into their constituent frequency components. Through this project, we have successfully employed FFT alongside signal preprocessing techniques to extract musical notes from audio files.

The process involved several key steps: loading the audio file, preprocessing to enhance signal quality, applying FFT for frequency analysis, identifying peaks corresponding to musical notes, and mapping these frequencies to note representations.

References

[1] J. C. Stein, "Digital Signal Processing for Music Synthesis and Recognition", Cambridge University Press, 2003.

[2] D. G. Stork and M. E. Henneke, "Fundamentals of Signal Processing for Sound and Vibration", Cambridge University Press, 2007.

[3] X. Serra, "Music and Signal Processing", Cambridge University Press, 2012.

Appendex A

Appendex

A.1 MATLAB Code

```
%% To set up song parameters
    mute = true;
    [song,Fs] = audioread('FurElise_Slow.mp3');
    Fs = Fs*4;
    figure, plot(song(:,1)), title('Fur Elise, entire song')

%% 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);

%% To Find the 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)])

%% downsampling by m
    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
```

%%Applying threshold to find notes

```
    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
```

%% To 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)
```

%%To Plot the waveplot

```
 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
```

A.2 Additional Information

%Syntax To convert the frequency to notes

```
function [note,frequency] = FreqToNote(f)
index = round(17.31232*log(f) - 47.37620);
 frequencies=[];
 notes=[];
 note = notes(index);
 frequency = frequencies(index); end
```

Overall, the code follows these steps:

- Loads and analyzes a specific window of the Fur Elise song.
- Downsamples and smooths the data to focus on prominent notes.
- Applies a dynamic threshold to identify potential notes.
- Extracts the fundamental frequency of each detected note using FFT analysis.
- Converts frequencies to musical notes and recreates each note as a sine wave.
- Combines the recreated notes into a sequence and plays the entire melody.
- This code provides a basic example of music analysis and note extraction techniques using MATLAB. It can be further refined and extended to handle different songs, instruments, and musical features.