DSP PROJECT

Karthik Hegde - IMT2018509
Karthik Udupa - IMT2018510
Keshav Singhal - IMT2018511
Sriram - IMT2018526

November 5, 2020

CONTENTS

LIST OF FIGURES

1 FILTERING

Filtering is the process of removing unwanted components or features or frequencies from our signal.
We implemented both fir and iir filters using various python functions of scipy library. But we just used fir filters in our experiments and for our further use of signal as it was sufficient.

1.1 Implementation

We designed fir filter with a 5Hz transition width and using Kaiser window method.
We got kaiser window parameters and then finally used them in firwin to create a fir filter.
Here we can send in the cutoff frequency as a parameter and choose the filter i.e. low-pass, bandpass, bandstop or highpass according to our use.

1.2 Experiments

- Starting with, we recorded the keyboard notes with a high frequency note. Below is the frequency spectrum of our recorded audio. We can observe that there are high frequency peaks at above 2000Hz, these are due to the high frequency notes and are not of our interest.

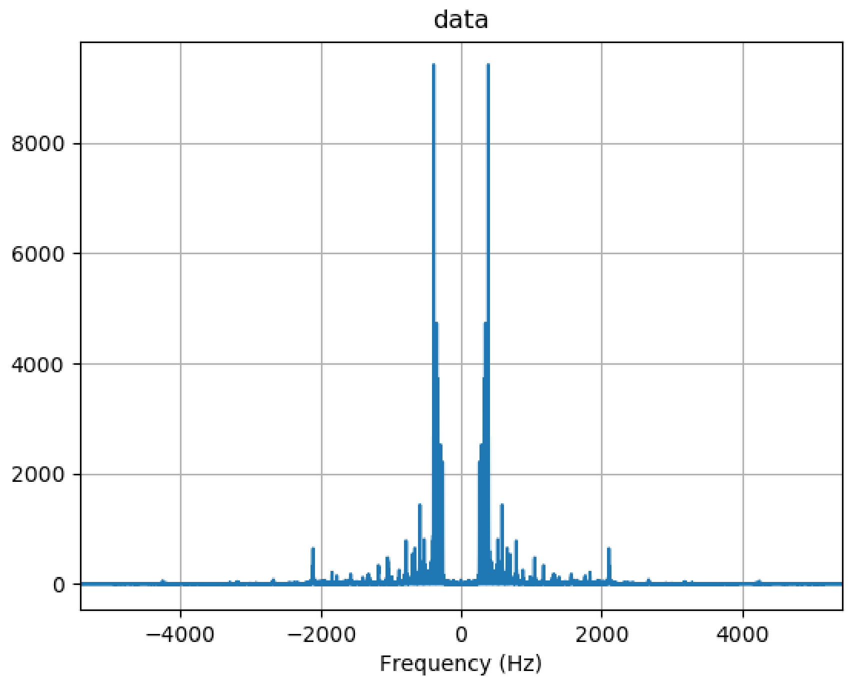

Figure 1: Spectrum of our recorded signal

- Further added 2 electrical noises(sin and cos waves of different amplitudes and frequencies) of low frequency.

- Also added a static noise(random noise generated by a random number generator).
  Below is the plot showing noisy signal.

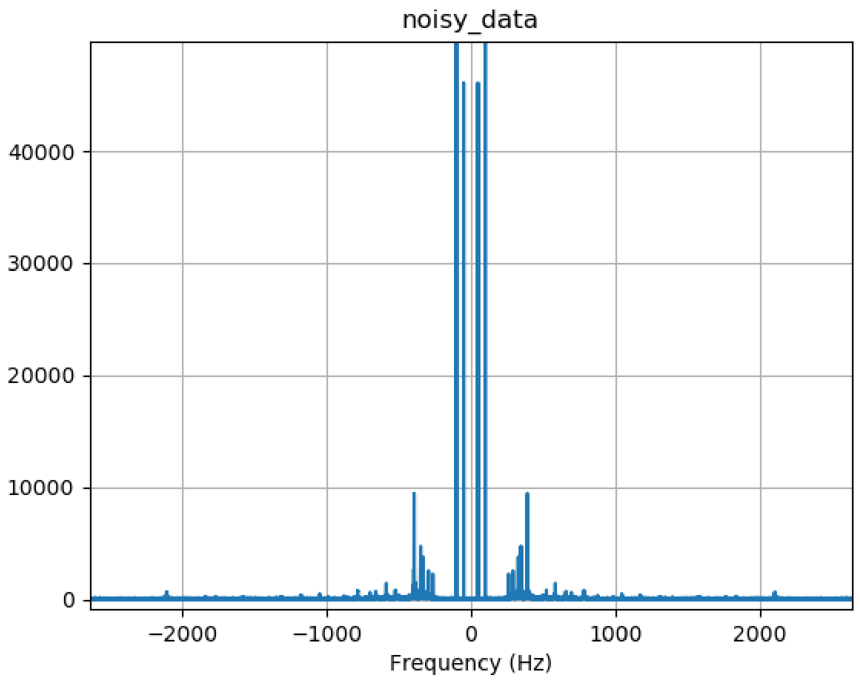

Figure 2: Spectrum of noisy signal

- Now we can observe that our required signal lies in frequency range from around 150-200Hz to 2000Hz.
  Hence using a bandpass filter will do our work and we can get back our signal.
  Below shown is the frequency response of our bandpass filter and the filter coefficients received from kaiser window.

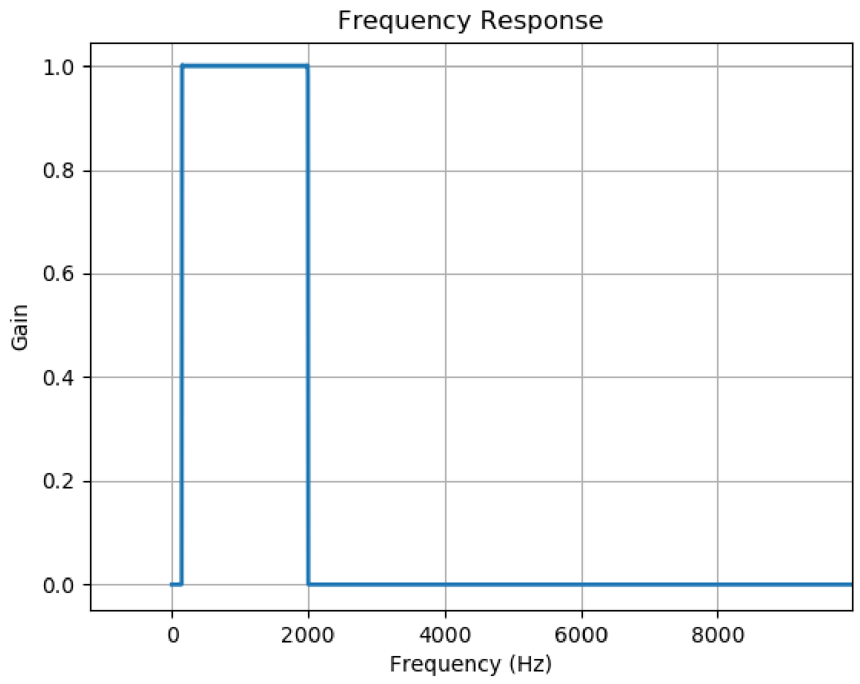

Figure 3: Frequency response of bandpass filter.

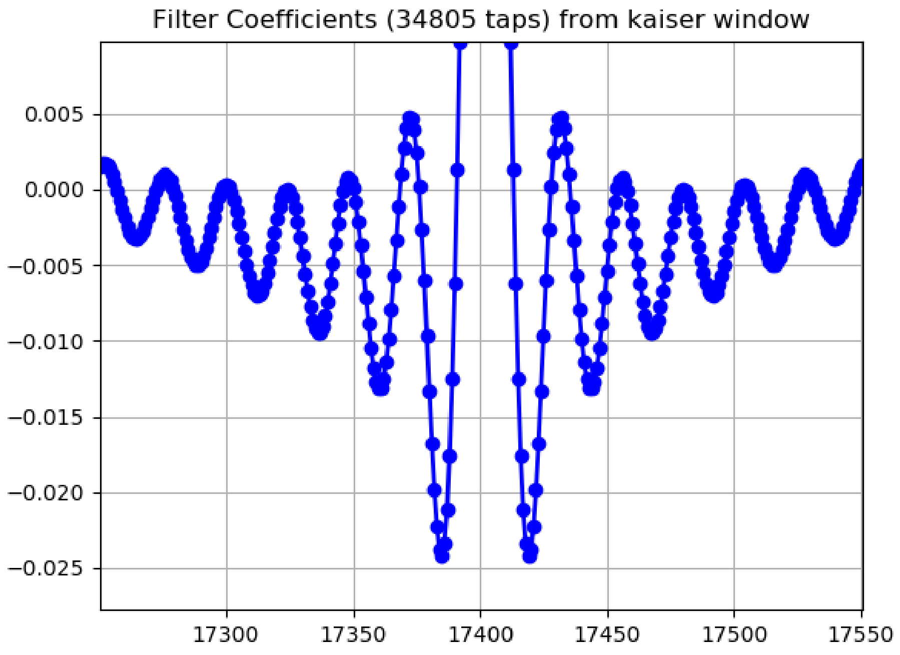

Figure 4: Filter coefficients from kaiser window.

- Finally showing below our filtered output signal.

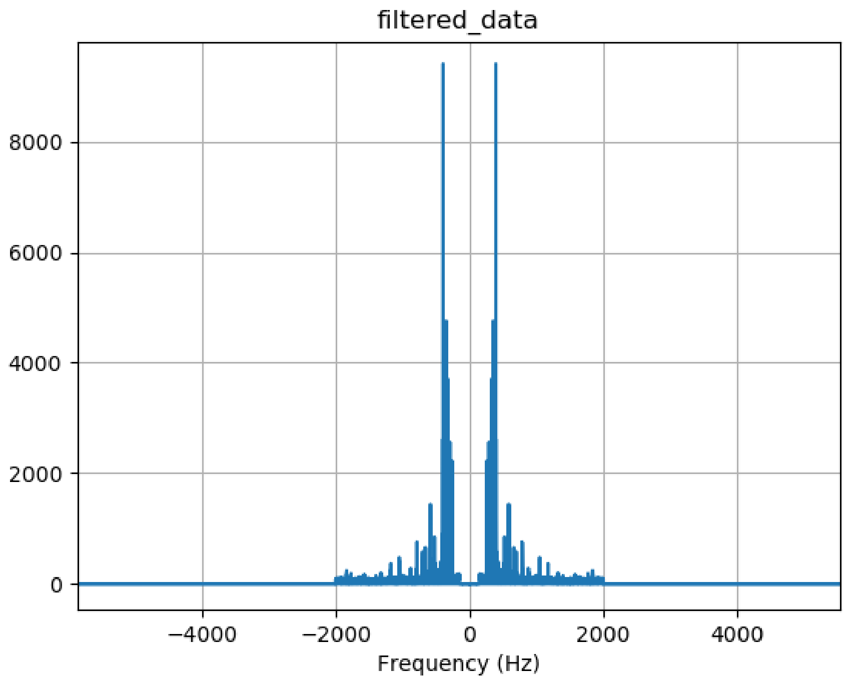

Figure 5: Spectrum of filtered signal.

2 SEGMENTATION

2.1 Introduction

What is Audio Segmentation?
Segmenting the audio stream into homogeneous regions to reveal semantically meaningful temporal segments.
Homogeneity is based on the task that we want to perform on the signal.

1. Music / Noise
2. Speech / Non-speech
3. Male voice / Female voice

For our problem statement, segmentation is the individual note separation from the audio signal.
The problem of segmentation is split into two components.

- Time domain analysis and split.
- Frequency domain analysis and split

Both techniques when combined yields decent accuracy in segmenting the audio signal.

2.2 Algorithm

Algorithm 1: Audio Segmentation : Time domain

```
Result: Note separation from audio
    1. Obtain Amplitude Envelope
    2. Obtain a weighted mean curve
    3. Pass the curve through low pass filter
    4. Find the inflections points
    5. Find the split points based on the inflection points
```

1. Amplitude Envelope and Weighted Mean Curve.
   The Amplitude Envelope is computed to find the Weighted Mean Curve(WMC). The WMC allows us to look at the general trend in the signal in time domain to spot the occurance of new note.

$$avg[i] = LagWeight * avg[i-1] + s[i] \tag{1}$$

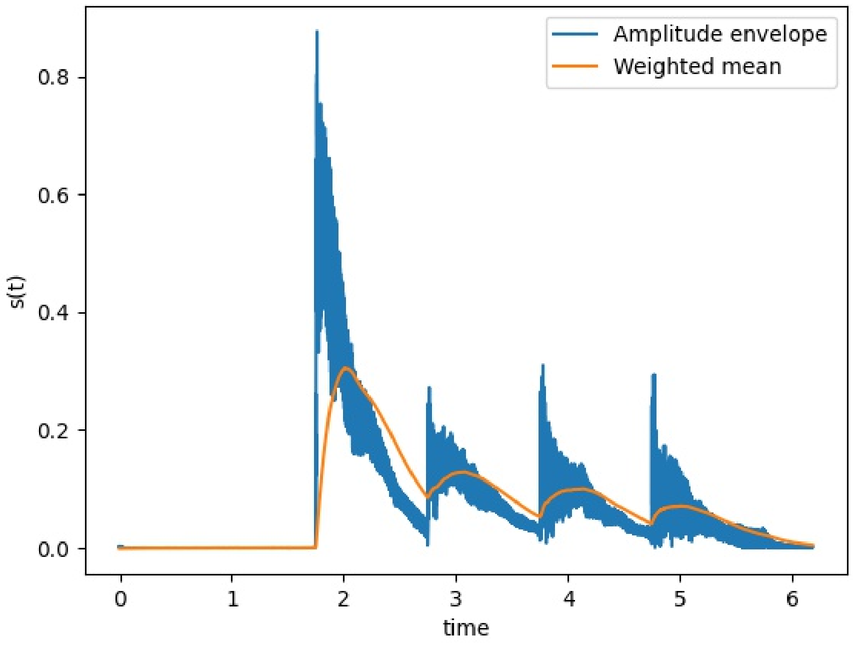

Figure 6: Amplitude Envelope and WMC

2. Low pass filter output.
   WMC in way is performing the task of low pass filter.
   But a low pass filter with low threshold is much more elegant in removing minor vibrations. This is useful because having a smooth and continuous functions helps in the calculation of inflection points.

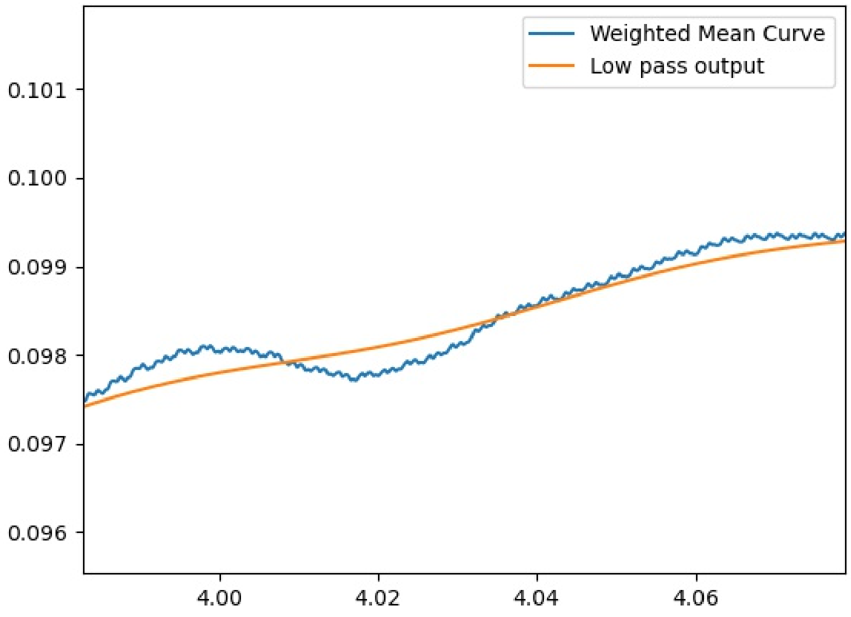

Figure 7: Difference after low pass filter

3. Inflection points and segment points.
   Inflection point refers to points where the curvature of the signal changes from concave to convex or vice-versa.
   The inflection points can be computed by analysing the second derivative of the signal.
   The following example is for the youtube(piano) recording.

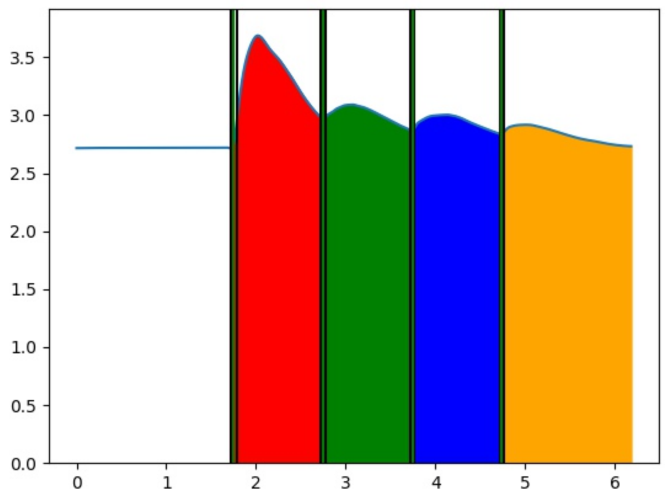

Figure 8: Segmentation

The following 2 plots are for a more generalized audio signal with higher temp and different sized notes.
The first plot contains the inflection points.
The second plot contains the final segment split locations.

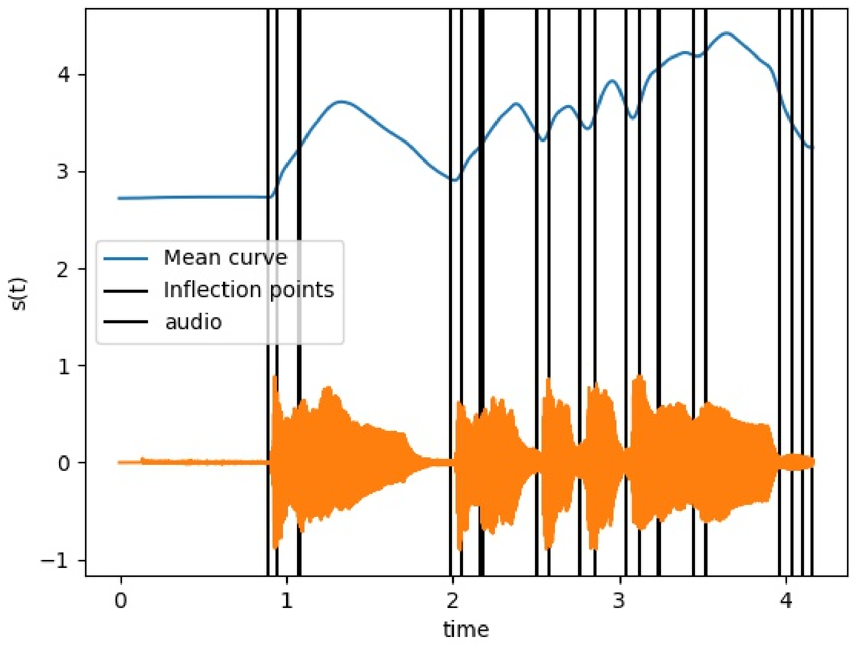

Figure 9: Inflection points

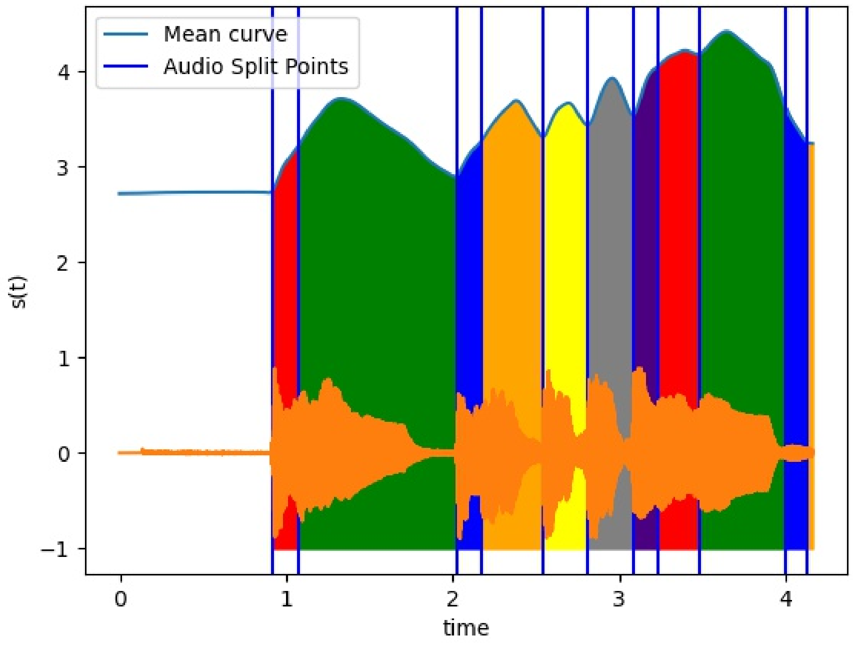

Figure 10: Audio segmenting points

Algorithm 2: Audio Segmentation : Frequency domain

Result: Note separation from audio

1. Analyse the frequency spectrum of consecutive segments.
2. Let f1 and f2 be the fundamental frequencies of the two segments.
3. By considering smaller windows in the segments we observe the shift from pure f1 to mixture of f1 and f2 to pure f2.
4. The previous step will enable us to split signals which are not split by the time domain analysis.

Currently, we have looked(implemented) at consecutive segments and determined if they belong to the same note by observing their fundamental frequencies.

2.3 Deep learning

Deep Learning Methods for Segmentation
Just like image segmentation is achieved with YOLO (2-D convolution), a simple 1-D convolution neural network can be used to perform state-of-the-art audio segmentation. There is a requirement of large amount of labelled training data for this method. Here's a blog about heart beat segmentation(AnalyticsVidhya)

3 BLIND SOURCE SEPARATION USING ICA

Blind source separation (or source separation) is the process of separating different source components of a signal be it music or an image, from a set of mixed signals. Independent component analysis (ICA) has proven to be useful in this problem of source separation especially for music or sound signals.

If x1, x2 are mixed signals and s1, s2 are the source signals, then we can mathematically represent this as

$$x1(t) = a * s1(t) + b * s2(t) \tag{2}$$

$$x2(t) = c * s1(t) + d * s2(t) \tag{3}$$

In vector and matrix notation this can be generalized as

$$X = AS \tag{4}$$

In simple mathematical terms, ICA algorithm finds A^{-1} using the method of non-gaussianity under the assumption that the sources are statistically independent (uncorrelated).

The implementation uses scikit-learn's FastICA method, which is applied on the linear mixture (here mixed via code) of piano and mridangam signals and the results are shown in the figure 13

3.1 Limitations of ICA

ICA outputs the scalar multiple of the source signal which sometimes could be negative i.e the output signal could be attenuated form of the original signal or could even show inversion in the amplitude values. However, for the sound signals the inversion or the attenuation is of least significance. Observe and compare the piano plots in the figures 11 and 13 for inversion in the peaks. But they still sound the same.

The serious requirement of the ICA or BSS is that it requires the same number of mixed signals as there are source signals. In addition ICA algorithm fails on convoluted mixtures.

Our purpose was to use this concept in the process of deriving the piano signal from a signal involving mixture of other unwanted signals. Although BSS using ICA is the first step towards achieving this feat, the requirements such as using linear mixers and a set of mixed signals of unknown number, if one is unaware of the actual number of components to be separated, leads us to look for other better methods.

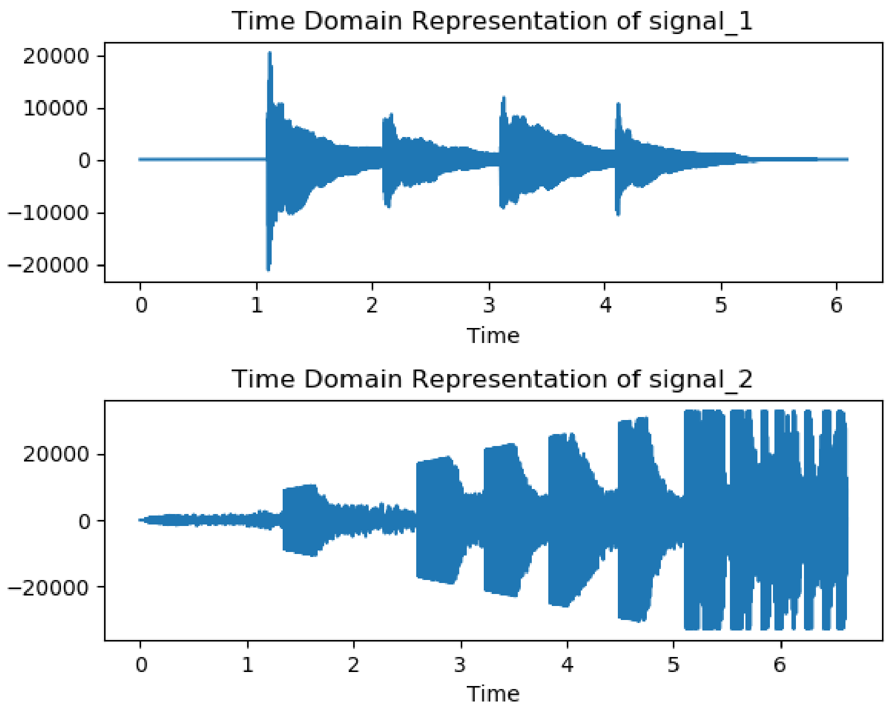

Figure 11: Original signals (top to bottom - piano & mridangam)

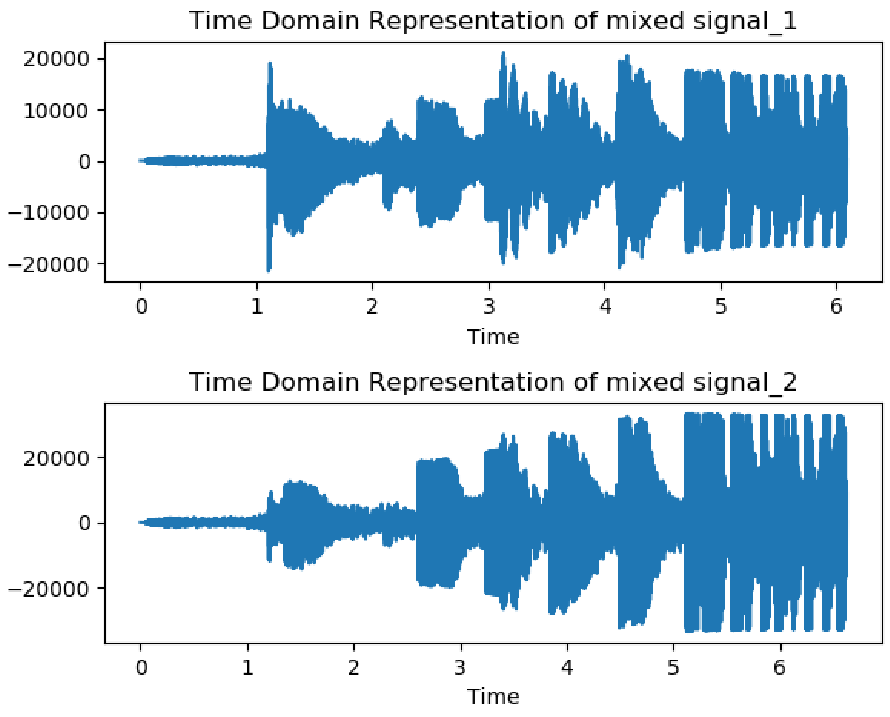

Figure 12: Linear combination of piano and mridangam signals in different proportions

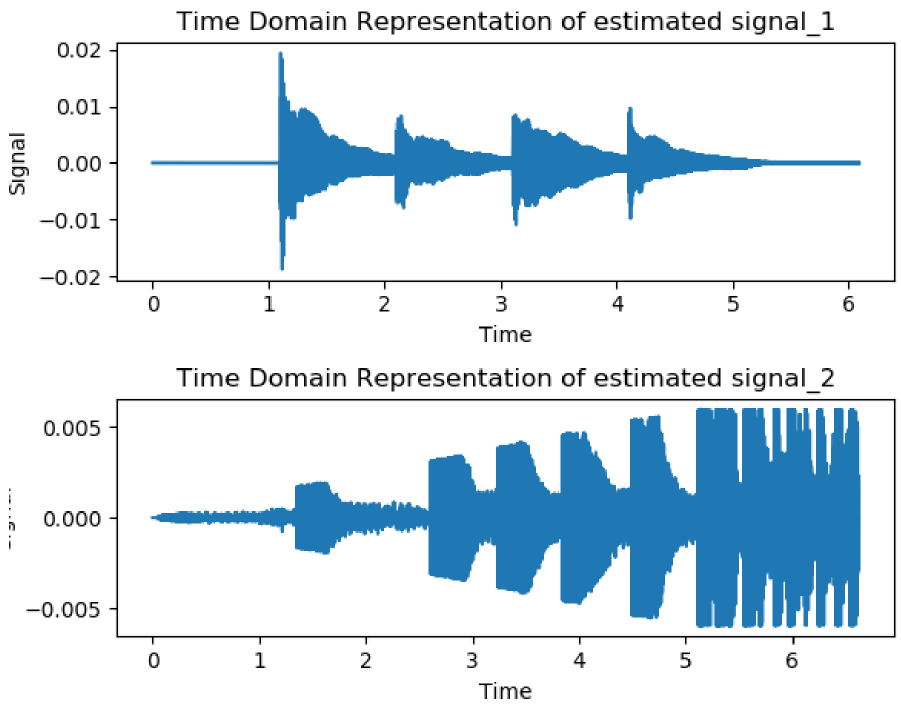

Figure 13: Separated signals from the linear mix (top to bottom - piano & mridangam)

4 TRANSCRIPTION

Finding the fundamental frequency present and the corresponding note in an audio segment is transcription. In the amplitude spectrum, the fundamental frequency will have the maximum amplitude. By using that method we can easily find the fundamental frequency and closest corresponding note is found from the table.

4.1 Multiple fundamental frequencies transcription

The above mentioned method can detect one fundamental frequency given an audio signal. It works for our use case. But in real life it is very much possible that one signal contains multiple fundamental frequencies mixed. We can solve this problem by finding all the peaks in the amplitude spectrum using scipy.signal.find_peaks.

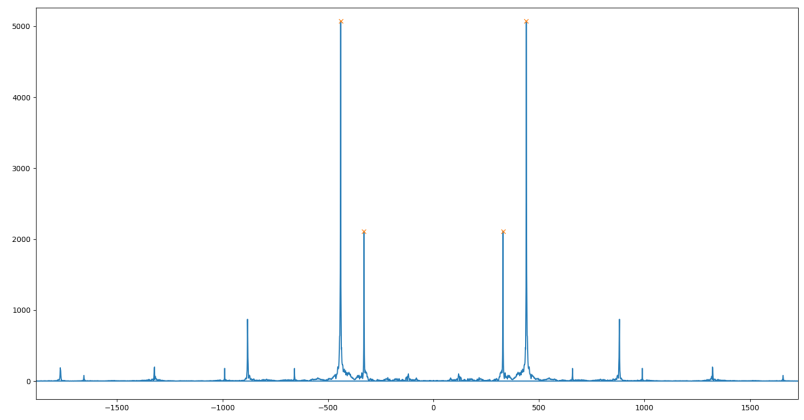

Figure 14: Peaks in signal with signals A4 (440 Hz) and E4(329.63 Hz)

5 EVALUATION

We were able to predict the notes for melodies from the YouTube video with 100% accuracy.
For generalization, we have recorded few audio samples from electronic keyboard. The duration of each note as well as the tempo was varied to test the algorithm.

SAMPLE OUTPUT:

1. Youtube audio 1 : (F)ABF
   Prediction :
   F4 369.99 Hz
   A4 440.0 Hz
   B5 987.77 Hz
   F4 349.23
   Accuracy = 100%

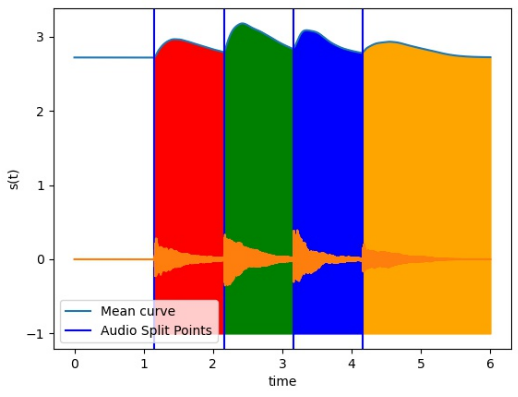

Figure 15: Evaluation audio 1

2. Youtube audio 2 : AEFC
   Prediction :
   A4 440.0 Hz
   E4 329.63 Hz
   F4 349.23 Hz
   C5 523.25 Hz
   Accuracy = 100%

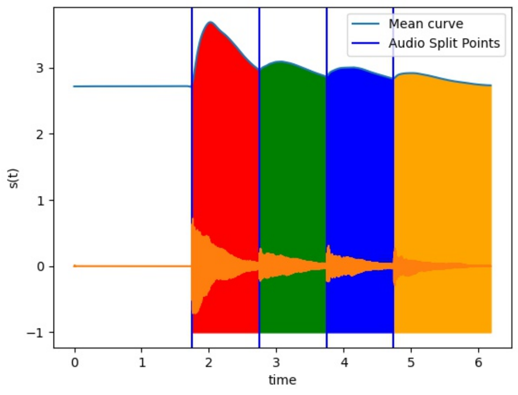

Figure 16: Evaluation audio 2

3. Piano recording 1 : CDEFG
   Prediction :
   G 783.99 Hz (Incorrect)
   D 293.66 Hz (Correct)
   D 587.33 Hz (Algorithm has split one note twice)
   E 329.63 Hz (Correct)
   F 349.23 Hz (Correct)
   G 392.0 Hz (Correct)
   Accuracy : 4/5 notes predicted accurately.

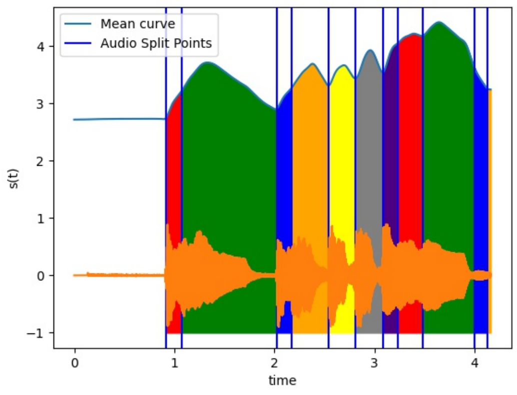

Figure 17: Evaluation audio 3

REFERENCES

Blind Source Separation via ICA: Math Behind Method

Segmentation