JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

IDENTIFICATION AND REMOVAL OF NON-SPEECH REGIONS IN SPEECH SIGNALS

Neha Shanbhag, Shreyashi Samaddar, Shashank Gajbhiye

SCTR's Pune Institute of Computer Technology, Pune-43.

ABSTRACT

Communication is a very important aspect of our day-to-day life, from chatting withour friends to national safety. Noise is an unavoidable factor in communication. Noise contaminates the characteristics of the speech signal and degrades its quality. This project identifies and removes non-speech components from speech signal. It wasobserved that the old techniques for noise suppression had some drawbacks such as when the energy of speech signal is lower than the threshold the speech signal is considered as noise and is subtracted, and the output is given. There is a need to design an algorithm which will not suppress the speech signal even when its energy is low. This project is set out to detect speech signal present in the audio signal contaminatedby noise and remove non-speech components. The project starts with understanding thecharacteristics of an audio signal, performing basic operations such as sampling on the audio signal. These characteristics were obtained using Voice Activity Detection methodology. This in turn created a realization of voice identification in a speechsignal. Further we take the live input and using our formulated algorithm we removed the silent parts from the input audio file.

List of Acronyms & Symbols

WAV	Waveform Audio File Format
ZCR	Zero- Crossing Rate
PSD	Power Spectral Density
VAD	Voice Activity Detector
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform

CHAPTER 1

Introduction

1.1 Background/context

It can be said that all acoustic environment contains noise. The speech signal produced or recorded by any audio device usually consist of noise. This leads to distorted output of the audio signal with degraded quality. Such distortion may be tackled by passing the audio through a filter preferably a digital filter. Hence, the needarises to design an optimal filter which can suppress noise with minimum distortion. Noise suppression is an old topic in speech processing, dating back to at least the 70s. The idea is to take a noisy signal and remove as much noise as possible while causing minimum distortion to the speech of interest. Today, major advances have been made in the area of noise suppression, as opposed to the older noise cancellation technology. These improvements provide for more effective communication and situational awareness while still protecting the user's hearing.

1.2 Relevance

During day-to-day digital communication we often face distortion in speech.

Various methods such as Energy thresholding were implemented but they had their own disadvantages. Since we had the presence of Digital signal Processing in our curriculum along with Audio and speech related elective subjects, we planned to implement our learning through this project. The goal of this project is to identify and remove non speech components from speech signals.

activity detection (VAD) refers to the task of determining whether a signal contains speech or not. With sensitive VAD we can classify the signals components into speech and non-

1.3 Literature Survey

Various traditional methods like analysis of the signal to the energy it produces, bandwidth shutdown, comparing the signal which prerecorded noise etc. have 3 shortcoming of their own. This is where sensitive VAD comes into picture. Voice speech by comparing the false positives and false negatives. Autocorrelation can be used as a sensitive VAD, with the limitation to positive correlations only. Zerocrossing can

also be used as a sensitive VAD, while comparing the signal components with human speech frequency, with the only drawback of not able to detect those speech signals which have different speech frequency on purpose to that of normal human range.

According to Tom Bäckström, in his paper "Voice Activity Detection", Introduction to Speech Processing, Aalto University Wiki [4] we can implement VAD using Energy Thresholding Method. From the figure below the first section show the real time activity detection where the red line indicates the active region of the signal. Second section shows the energy of speech signal where the red line shows the threshold that we have set above which the real time speech signal will be recognized as positive pulse and below that will be 0. Third section shows keywords from the input speech signal. For this experiment we have set the sampling frequency to 16kHz, screen width to show the signal to 2 seconds and threshold energy for removal to 100.

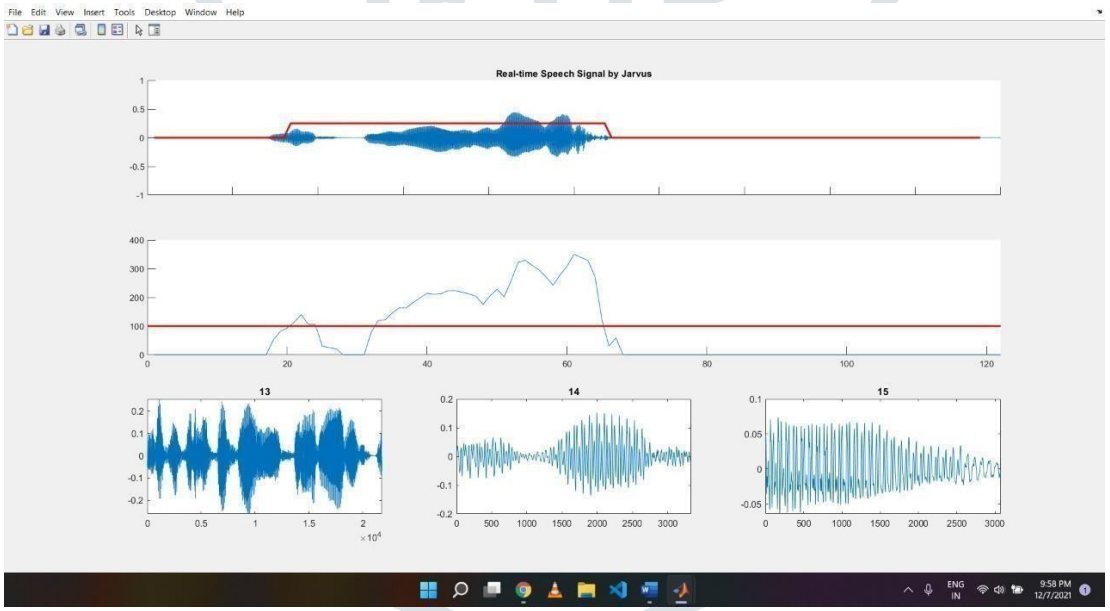

Fig. 1.1: Real time voice activity detector

While performing Energy Thresholding we see that Some of the Speech signal which has lower energy gets classified as non-speech. Hence, we need a better approach to suppress noise.

When we analyze the speech using different parameters of the speech it assumes that the distribution of the input features for speech and non-speech signals are linearly separable. Hence it cannot take any non-linear shapes of the distribution, nor can it consider since the speech signal is represented with voiced and non- voiced signal.

There are multiple techniques for noise suppression currently. Among them, the implementation based on Zero Crossing Rate (ZCR) and Power Spectral Density (PSD) not only reduces the processing speed but also enables the extraction of Silent

parts of the speech signal recorded even in an environment with highly degrading noise. The implementation of such concepts not only helps us reducing processing time, lower data losses but also profits us in saving space as the file containing background noise would occupy a larger space than the file that had undergone noise suppression.

The reduction in space is produced with the help of spectral subtraction which includes techniques such as zero crossing rate, power spectral density for identifying the noises present in the given file and then deducting the noise part from the file, thusreducing the amount of the total space used. For example, it was found that in a 100mb audio file more than one fourth of the file size consisted of unvoiced signal. When the data being handled is in the gigabyte scale, these noise sections result in a huge amount of wasted space.

Hence in this project, we will implement these two techniques and analyze their performance with respect to each other.

1.3.1 Zero-Cross Rate (ZCR) Detection

The Zero-Crossing Rate (ZCR) of an audio frame is the rate of sign-changes of the signal during the frame. In other words, it is the number of times the signal changes value, from positive to negative and vice versa, divided by the length of the frame. ZCR can be interpreted as a measure of the noisiness of a signal. For example, it usually exhibits higher values in the case of noisy signals.[1]

$$Z(i) = \frac{1}{2W} \sum^{W_L} |sgn[x_i(n)] - sgn[x_i\,(n-1)]| \qquad (1.1)$$

$$L_{n=1}$$

Where sgn(•) is the sign function, i.e.

$$sgn[x_i(n)] = \begin{cases} 1, & x_i(n) \geq 0, \\ -1, & x_i(n) < 0. \end{cases}$$

1.3.2 Power Spectral Density (PSD) Detection

Power spectral density function (PSD) shows the strength of the variations (energy) as a function of frequency. In other words, it shows at which frequencies variations are strong and at which frequencies variations are weak. The unit of PSD is energy per frequency (width) and you can obtain energy within a specific frequency range by integrating PSD within that frequency range. Computation of PSD is done directly by the method called FFT or computing autocorrelation function and then transforming it.

1.4 Motivation

It can be said that all acoustic environment contains noise. This noise infected speech signal results in various changes in the signal and degrades the quality and intelligibility of the speech signal. This may cause damage to human-machine communication systems. Suppression of such noise is vital for clear communication. Various speech processing algorithms are resource are vigorous and require significant computing power or transmission bandwidth. Yet speech is discontinuous such that we often have pauses between sentences and breaks even within sentences. Consequently, there is great potential in saving resources by deactivating advanced speech processing methods whenever the input signal does not contain speech. The limitation of the previous method is that the threshold value(energy) is constant hence if energy of noise is more than the threshold then it will be considered as speechsignal and if the speech signal is less than the threshold value the speech will be considered as noise and will be eliminated. Hence, we plan to design an advanced filter that can identify and remove the noisy regions in the speech signal.

1.5 Aim of the Project

To identify and remove non-speech regions in speech signals.

1.6 Scope and Objectives

The Main objective of this project is to detect speech signal in audio file which contains both noise and audio signal in analog format, converting it to digital signal and dividing it into respective time intervals and further differentiating into speech and non-speech components.

The scope of the project is to take live input from the user and return compressed audio file. Time and frequency domain plots have been plotted for the corresponding audio files which facilitate the user to visually distinguish between the files.

1.7 Technical Approach

For this project we equipped ourselves with the knowledge of python. We started with learning basic python programming. Then, we studied various python libraries that
can help us build this project such as librosa, numpy, matplotlib.

Further, we studied how to deal with audio signal. We performed various basic operations on the audio signal such as playing the audio file using python, digitized the
audio signal, sampled the audio signal, and converted the audio signal to its frequency domain. We studied various techniques related to our project such as voice activity detection (VAD).

There are multiple techniques for noise suppression currently. Among them, the implementation based on Zero Crossing Rate (ZCR) and Power Spectral Density (PSD) not only reduces the processing speed but also enables the extraction of Silent parts of the speech signal recorded even in an environment with highly degrading noise. The implementation of such concepts not only helps us reducing processing time, lower data losses but also profits us in saving space as the file containing background noise would occupy a larger space than the file that had undergone noise suppression.

CHAPTER 2

Theoretical Description of Project

2.1 Theoretical Background

Noise suppression is an old topic in speech processing, dating back to at least the 70s. As the name implies, the idea is to take a noisy signal and remove as much noise as possible while causing minimum distortion to the speech of interest. Today, major advances have been made in noise suppression, as opposed to the older noise cancellation technology. These improvements provide for more effective communication and situational awareness while still protecting the user's hearing. Despite significant improvements in quality, the high-level structure has remained mostly the same. Some form of spectral estimation technique relies on a noise spectral

estimator, itself driven by a voice activity detector (VAD) or similar algorithm. The extraction of voiced, unvoiced and silent parts of a speech is a problem in thepresence of natural background noise which is an important step of speech enhancement. Speech enhancement usually requires adaptive filtering and statistical processing of speech signal which requires high processing time and memory requirements. If extraction of voiced, unvoiced and silent parts also precedes in the same manner the entire process will consume a lot of time and which will not be reliable for real time processing. Using Python, we will implement two techniques which will detect and remove silent parts.

2.2 Technical specifications of the project, resources required

2.2.1 Technical specification

We will detect speech signal in audio signal which contains both noise and audio signal. We will convert an audio signal to digital signal and then further divide it into respective time intervals. After converting the audio signal into time intervals, in each region we will be able to differentiate between speech and non-speech signal. We will input an audio file containing speech and non-speech components, the processed output will consist of speech and suppressed non-speech components.

2.2.2 Resources required

We used the following resources for our project:

a) ANACONDA PYTHON 3.8.8: We have used Anaconda Python 3.8.8 as our python distribution. We decided to go with Anaconda python as it brings many of the tools used in data science and machine learning with just one install, so it's great for having short and simple setup.

b) AUDIO FILE: We will take a live input from the user and save it in our system in order to perform various operations on it.

c) Jupyter Notebook: Jupyter Notebook is a web based application which is used for creating and sharing computational files. It provides the user with a simple, streamline, document-centric experience. We will write our python code in Jupyter Notebook.

We have used the following python libraries for implementation in our project:

a) NumPy

What is NumPy? NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape

manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

i. numpy.array(): Creates an array.
ii. numpy.fft.fft(): This function computes the one-dimensional n-point discrete Fourier Transform (DFT) with the efficient Fast Fourier Transform (FFT) algorithm [CT].
iii. numpy.arange(): Return evenly spaced values within a given interval.
iv. numpy.ceil(): Return the ceiling of the input, element-wise.
v. numpy.log10(): Return the base 10 logarithm of the input array, element wise.
vi. numpy.where(): numpy.where(condition, [x, y,]/) returns elements chosen from x or y depending on condition.

b) matplotlib.pyplot

matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that “axes” here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

i. matplotlib.pyplot.plot() : Plot y versus x as lines and/or markers.
ii. imatplotlib.pyplot.title() : Set one of the three available axes titles. The available titles are positioned above the axes in the center, flush with the left edge, and flush with the right edge.
iii. matplotlib.pyplot.show() : Display all open figures.
iv. matplotlib.pyplot.figure(): It is used to create a new figure.
v. matplotlib.pyplot.gca(): Using given arguments the function creates or gets current axes instance.
vi. matplotlib.pyplot.xlabel(): It sets the label for the x-axis of the plot.
vii. matplotlib.pyplot.ylabel(): It function sets the label for the y-axis of the plot.

c) scipy

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy, an interactive

Python session becomes a data-processing and system-prototyping environment rivaling systems, such as MATLAB, IDL, Octave, R-Lab, and SciLab.

i. scipy.io.wavfile.read(): Opens a WAV file. Return the sample rate (in samples/sec) and data from an LPCM WAV file.

ii. scipy.signal.welch(): Welch's method computes an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms

d) wave

The wave module provides a convenient interface to the WAV sound format. It does not support compression/decompression, but it does support mono/stereo.

i. wave.open(file, mode=None) : If file is a string, open the file by that name, otherwise treat it as a file-like object. mode can be:

'rb' : Read only mode.'wb'

: Write only mode

e) sounddevice

This Python module provides bindings for the PortAudio library and a few

convenience functions to play and record NumPy arrays containing audio signals.

i. sounddevice.rec(): To record audio data from your sound device into a NumPy array, use sounddevice.rec()

ii. sounddevice.wait(): Checks whether sounddevice.rec() has finished recording or not.

f) soundfile

SoundFile is used to read and write sound files.

i. soundfile.write(): To write data to the file.

g) IPython.display

IPython is a Public API for display tools in IPython. It is tools for interactive and parallel computing in Python.

i. IPython.display.Audio(): Creates an audio object. When this object is returned by an input cell or passed to the display function, it will result in Audio controlsbeing displayed in the frontend. It works in the notebook only.

h) OS Path Module

OS module in Python provides functions for interacting with the operating system.

i. os.path.getsize(): It returns a integer value which representing the size of specifiedpath in bytes.

i) ipywidgets

Widgets are eventful python objects that have a representation in the browser,often as a control like a slider, textbox, etc. In order to use the widget framework, ipywidgets needs to be imported.

i. widgets.Button(): button widget is used to handle mouse clicks. The on_click method of the Button can be used to register function to be called when the button is clicked.

j) librosa

librosa is used for music and audio analysis. It provides the necessary components

to create music information retrieval systems.

i. librosa.load(): Load an audio file as a floating point time series. Audio will be automatically resampled to the given rate (default sr=22050).
ii. librosa.display.waveplot(): Plots the amplitude magnitude of a waveform.
iii. librosa.feature.zero_crossing_rate():Compute the zero-crossing rate of an audio time series.

k) math

This module provides access to the mathematical functions defined by the Cstandard.

i. math.floor(): math.floor approximates the decimal output to its largest integer.

2.3 Block Diagram

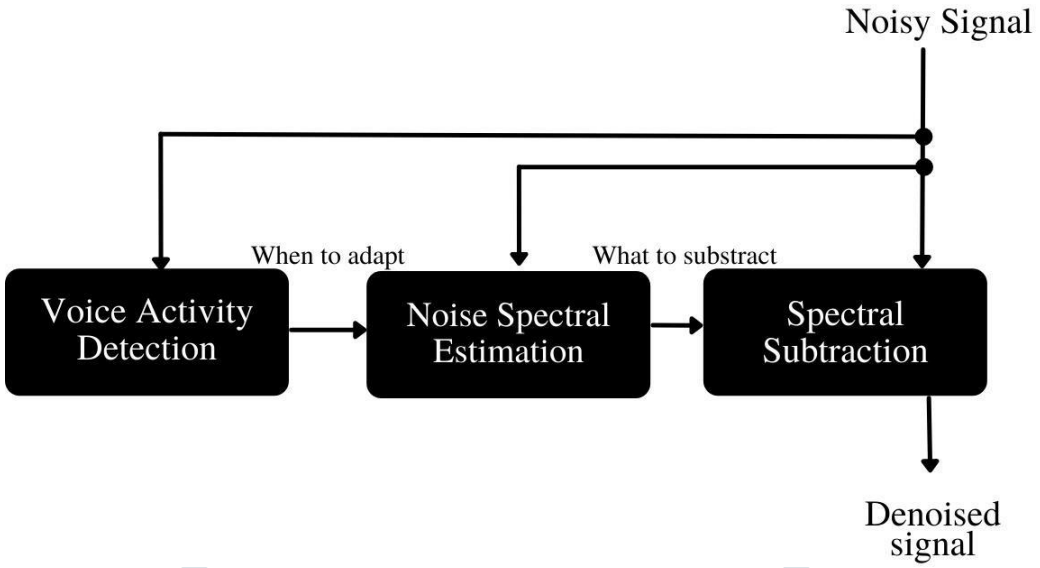

Fig. 2.1: Noise Removal Block Diagram

2.4 Algorithm

2.4.1 Zero-Crossing Rate

The input audio file can be split into individual frames. Thus, an algorithm can be created which calculates the ZCR of frames and then compares the ZCR of a frame iteratively with a threshold value. If the ZCR value is higher than the threshold, then that frame corresponds to noise and then can be deleted.

The steps of the algorithm are as described below:

i. Make necessary imports.
ii. Input recording from user and save the recording as .wav file.
iii. Load the input recording.
iv. Display the input audio file in time domain.
v. Set framelength to 2000 and framestart to 0. The hoplength is framelength/4.
vi. Create array ‘zcr’ which computes the ZCR of the audio signal using librosa.
vii. Plot the computed ZCR.
viii. Set ZCR threshold to 0.1 .
ix. Create ‘data’ array containing the input file.
x. Set framestart and frameend to 0.
xi. Use for loop to iterate through ‘zcr’ array with ‘index’ as the Iterator.
xii. If ‘zcr’ value at ‘index’ is less than the threshold, then increment framestart and frameend by hoplength.
xiii. If frameend value exceeds length of ‘data’ array break the loop.
xiv. Iterate through the range [framestart, frameend – hoplength] and set values in ‘data’ array to 2.
xv. End the for loop.
xvi. Iterate through the ‘data’ array and list the values that are not 2 in ‘output’ array
xvii. Plot the ‘output’ array, this is the output waveform.
xviii. Save the output audio file.
xix. Play both input and output audio file’s using widgets.
xx. Display the file size of the audio files in kilobyte.
xxi. Plot the frequency domain plot of the output and input audio files.

2.4.2 Power Spectral Density (PSD)

The algorithm will mask the frames corresponding to frequencies less than the threshold value (120 Hz) and later remove them.
The steps of the algorithm are as described below:

i. Make necessary imports.
ii. Input recording from user and save the recording as .wav file.
iii. Load the input recording.
iv. Display the input audio file in time domain.
v. Set framelength to 1200 and framestart to 0. The hoplength is frame length/4.
vi. Create 'output' array containing the input file.
vii. Start a while loop with the condition frameend < length of 'output' array.
viii. Find PSD using welch's function.
ix. At each iteration of the while loop find the maximum value of the PSD and append it into an array 'max_psd'.
x. Increment framestart and frameend by hoplength.
xi. End the while loop.
xii. Plot the array 'max_psd'.
xiii. Copy the 'max_psd' array into 'gpsd'.
xiv. Set PSD threshold to 120Hz
xv. Create 'data' array containing the input file.
xvi. Set framestart and frameend to 0.
xvii. Use for loop to iterate through 'gpsd' array with 'index' as the Iterator.
xviii. If 'gpsd' value at 'index' is less than the threshold, then Increment framestart and frameend by hoplength.
xix. If frameend value exceeds length of 'data' array break the loop.
xx. Iterate through the range [framestart, frameend – hoplength] and set values in 'data' array to 2.
xxi. End the for loop.
xxii. Iterate through the 'data' array and list the values that are not 2 in 'dx' array
xxiii. Plot the 'dx' waveform, this is the output waveform.
xxiv. Save the output audio file.
xxv. Play both input and output audio file's using widgets.
xxvi. Display the file size of the audio files in kilobyte.
xxvii. Plot the frequency domain plot of the output and input audio files.

CHAPTER 3

System Design

3.1 Block Wise Design

With reference to Fig. 2.1, it can be seen that it is a conceptual view of a conventional noise suppression algorithm.

3.1.1 Voice Activity Detection(VAD)

This module detects when the signal contains voice and when it's noise.

3.1.2 Noise spectral detection

The voice activity detection output is used by a noise spectral estimation module to

figure out the spectral characteristics of the noise (how muchpower at each frequency).

3.1.3 Spectral Subtraction

Knowing how the noise looks, it can be “subtracted” from the input audio.

3.2 Equations

3.2.1 ZCR

ZCR is the number of times the signal changes value, from positive to negative and vice versa, divided by the length of the frame. Hence,

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]| \quad (3.1)$$

Where sgn(•) is the sign function, i.e.

$$sgn[x_i(n)] = \begin{cases} 1, & x_i(n) \geq 0, \\ -1, & x_i(n) < 0. \end{cases} \quad (3.2)$$

CHAPTER 4

Implementation, Testing and Debugging

It can be said that all acoustic environment contains noise. The speech signal producedor recorded by any audio device usually consist of noise. This leads to distorted output of the audio signal with degraded quality. Such distortion may be tackled by passing the audio through a filter preferably a digital filter. Hence, the need arises to design an optimal filter which can suppress noise with minimum distortion.

Removal of silent parts from the audio file is an important pre-processing step in speech processing. Major firms that specialize in speech processing receive huge amount of data which is in terms of several terabytes. Handling such large data is very difficultfor the machine to process. It will increase the load on the device and computational time.Hence, it is important to remove silent parts from the audio file. This allows the audio file to be compressed to a great extent.

There are multiple techniques for noise suppression currently. After going through several sources such as books and research papers we decided on the implementation based on Zero Crossing Rate (ZCR) and Power Spectral Density (PSD) which not only reduces the processing speed but also enables the extraction of Silent parts of the speech signal recorded even in an environment with highly degrading noise. The implementation of such concepts not only helps us reducing processing time, lower data losses but also profits us in saving space as the file containing background noise would occupy a larger space than the file that had undergone noise suppression.

We formulated two algorithms to remove silent parts from the audio file one by using ZCR and the other by using PSD. We computed the ZCR using the librosa library. librosa.feature.zero_crossing_rate() was the function used. In this function there is a parameter which is the audio time series. This is basically the input audio file. So, to this audio time series we added a constant ‘0.001’. This constant is added because during the silent parts, ZCR is high, and the silence oscillates quietly around zero. Hence, the constant was added before computing the ZCR in order to go about this issue. The threshold for both ZCR and PSD methods was determined by trial-and-error method. Thethreshold for ZCR was found to be 0.1 and 120 for PSD.

Finally, we formulated a program. The program displays the time domain and frequency domain waveforms of the input and output audio files along with the file sizein kilobytes. The program also enables the user to listen to the audio files. We tested various audio files and recordings and observed the output.

CHAPTER 5
Results and Discussion

We formulated an algorithm based on ZCR and PSD to remove the silent parts. We were able to significantly compress the original output audio file. The input and audio file can be played. Additionally, the amplitude and frequency plot of the input and outputsignal have been displayed allowing to differentiate between the input and output audio files visually.

1. Using the record button, live the input can be taken from the user. The input audio file is saved in wav format.

Fig. 5.1: Button for live input

2. After we input the audio file from the user and save it in wav format, we plot the input audio signal in time-domain.

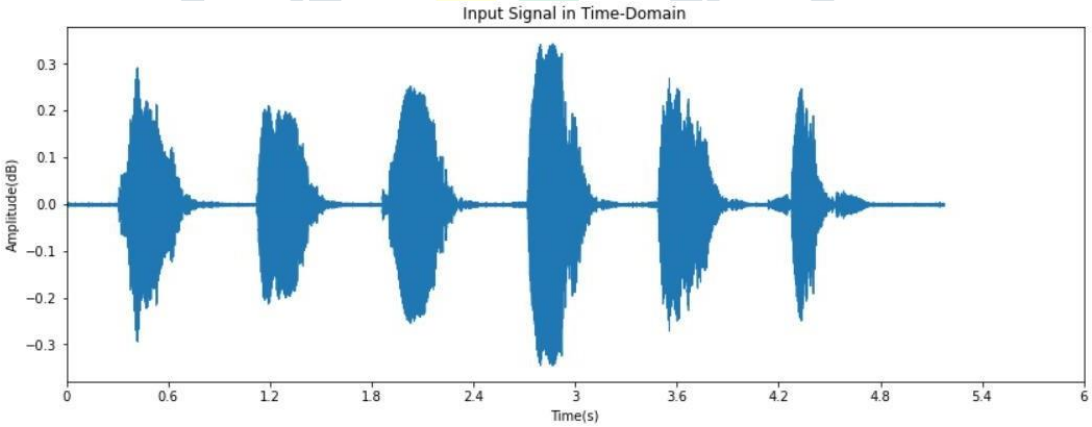

Fig. 5.2: Input signal in time domain

3. We calculate the zero-crossing rate at each point and plot it.

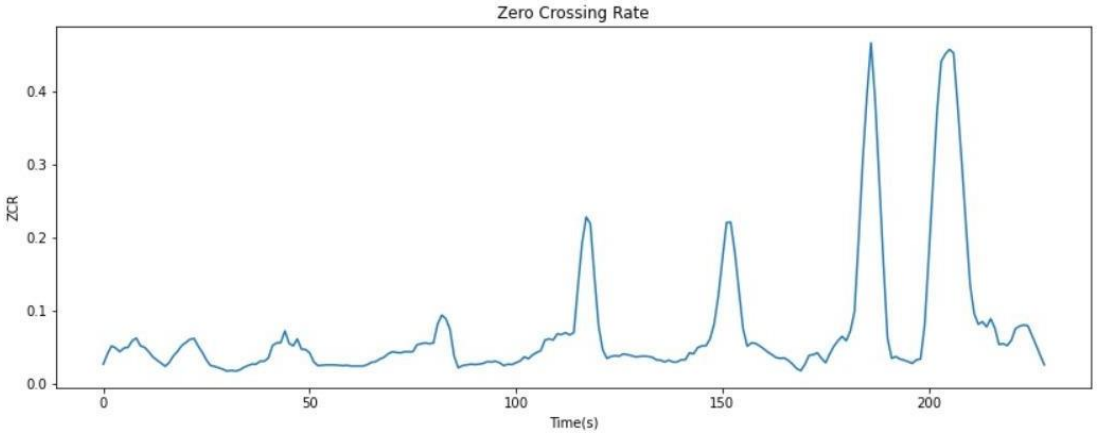

Fig. 5.3: Zero Crossing Rate

4. We calculated the maximum power spectral densities and plotted it.

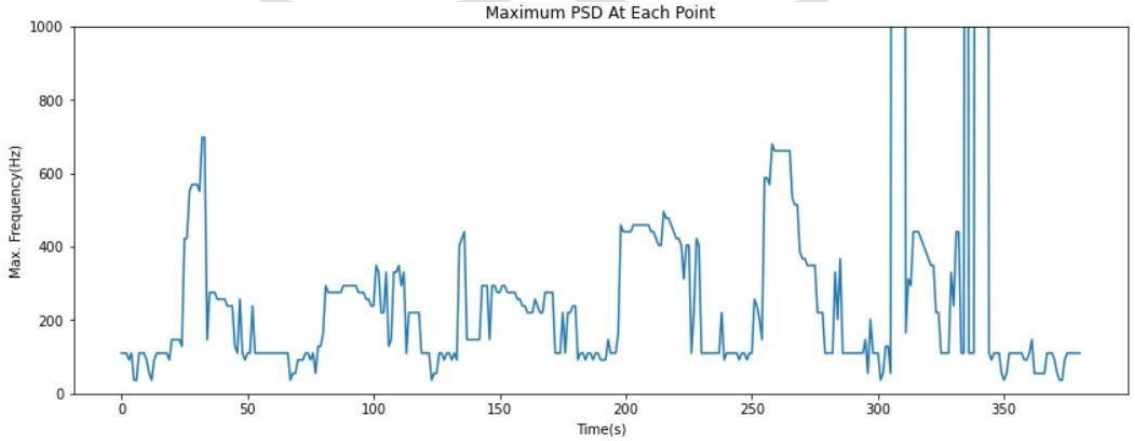

Fig. 5.4: Maximum Power Density

5. We formulated an algorithm to remove silent parts from the speech signal based on zcr and psd. We plotted the time domain signal of both the outputs. It was observed that the ZCR algorithm resulted in a signal which had abrupt cutsand numerous points of data losses, as evident by looking at the waveform. Conversely, the PSD algorithm resulted in an output that had smooth transitions and the data loss was minimal.

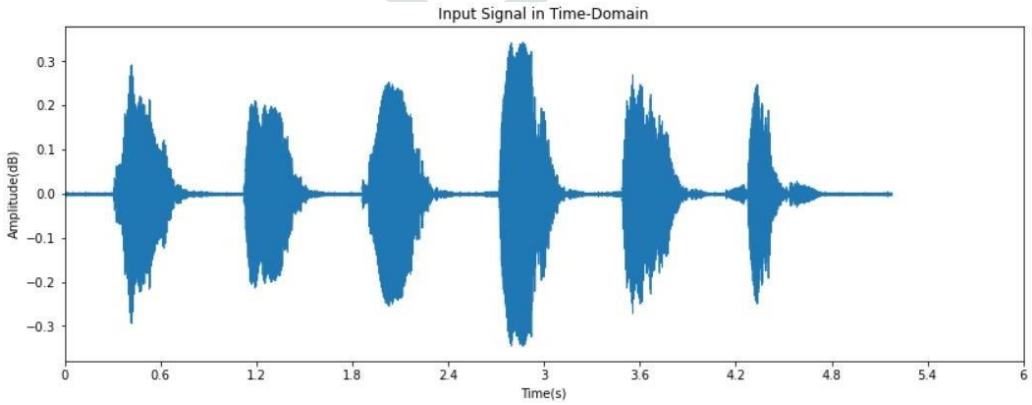

Fig. 5.5: Input signal in Time Domain

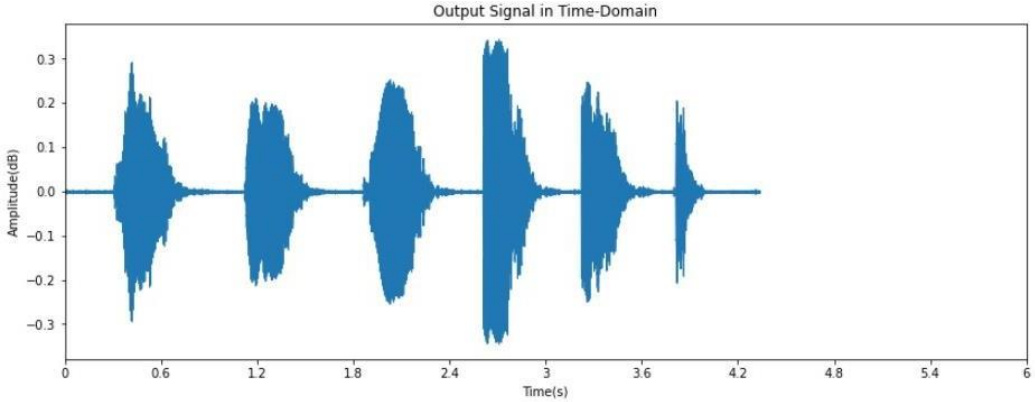

Fig. 5.6: ZCR output in Time Domain

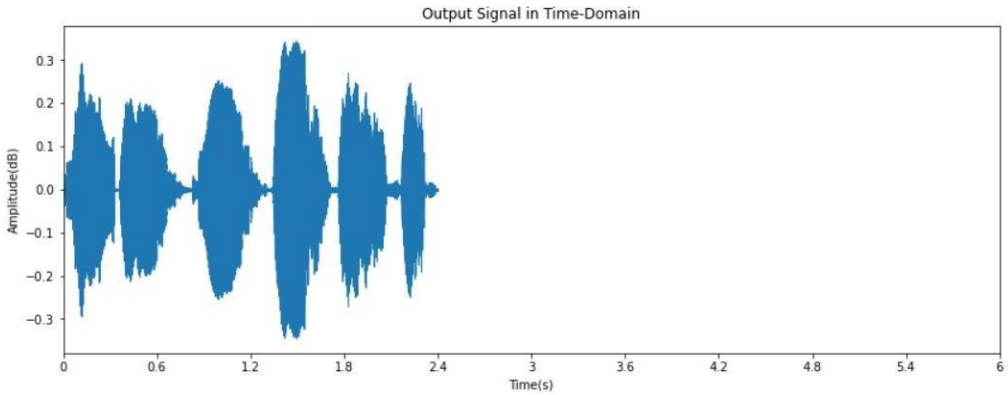

Fig. 5.7: PSD output in Time Domain

6. To allow listening to the audio files widgets have been added. Apart from listening to the audio we can also observe the difference in duration of audio. It can be seen that the PSD based algorithm compresses the audio file more efficiently.

Fig. 5.8: Input audio file

Fig. 5.9: ZCR output audio file

Fig. 5.10: PSD output audio file

7. Here we have displayed the size of the audio files in Kilobytes.

```
Size of input audio file: 532 KB
Size of output audio file: 187 KB
```

Fig. 5.11: Size of ZCR output

```
Size of input audio file: 532 KB
Size of output audio file: 103 KB
```

Fig. 5.12: Size of PSD output

8. Here we have plotted the frequency domain plot of the input and output of ZCR and PSD respectively.

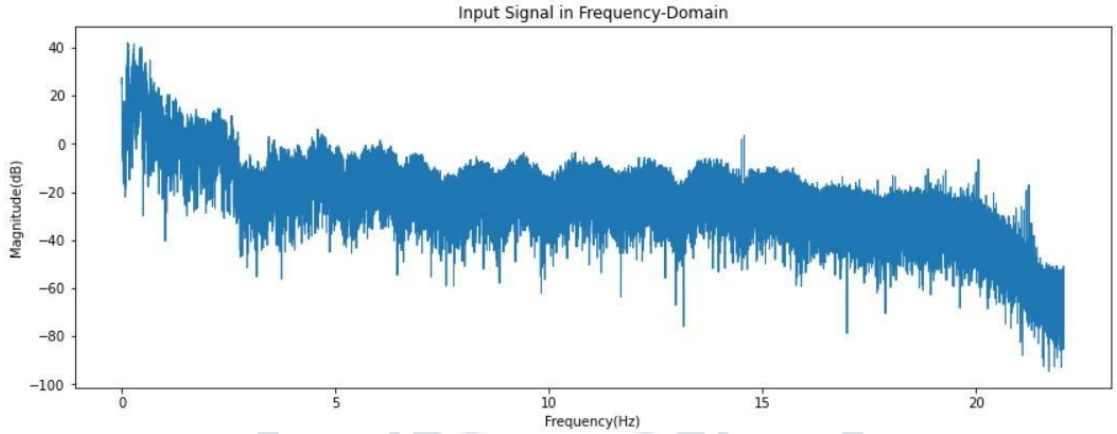

Fig. 5.13: Input Signal in Frequency Domain

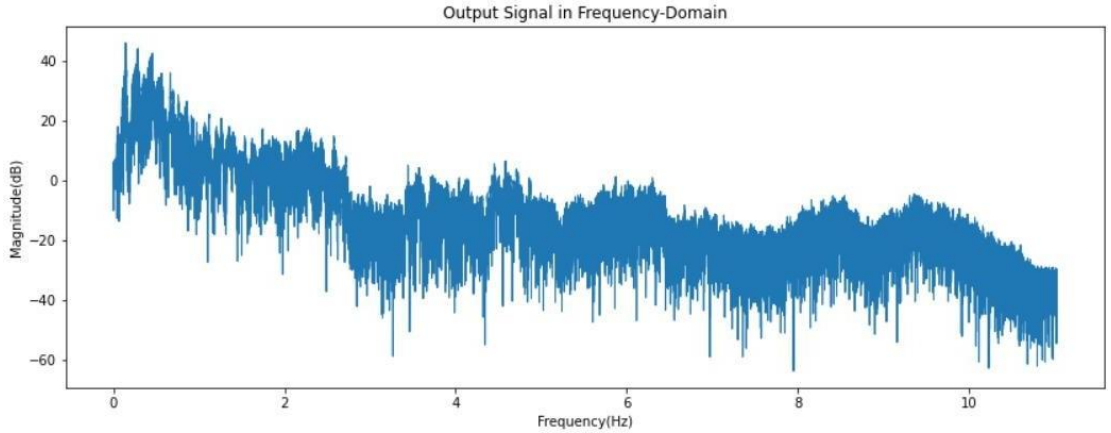

Fig. 5.14: ZCR Output Signal in Frequency Domain

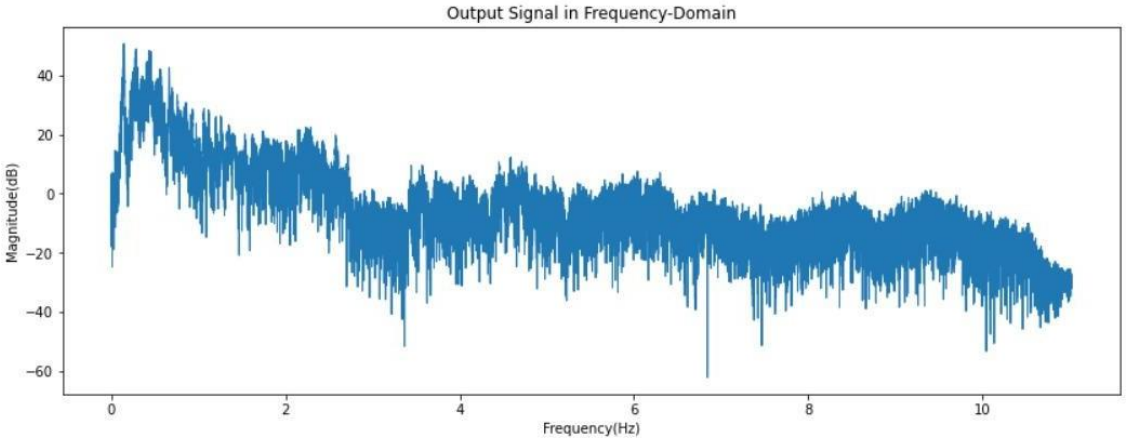

Fig. 5.15: PSD Output Signal in Frequency Domain

CHAPTER 6

Conclusions

Noise contaminating a speech signal is an inevitable phenomenon. It causes distortions in the output of the audio signal and also degradation of the quality. Removal of silent parts from the audio file is an important pre-processing step in speech processing. Major firms that specialize in speech processing receive huge amount of data which is in terms of several terabytes. Handling such large data is verydifficult for the machine to process. It will increase the load on the device and computational time. Hence, it is important to remove silent parts from the audio file, allowing the audio file to be compressed. There are multiple techniques available for noise suppression presently. After going through several sources such as books and research papers we decided on the implementation based on Zero Crossing Rate (ZCR) and Power Spectral Density (PSD) which reduces the processing speed and enables the extraction of Silent parts of the speech signal recorded even in an environment with highly degrading noise. The implementation of such concepts helps us to reduce the processing time, lower data losses and also profits us in saving space as the file containing background noise. We formulated an algorithm based on ZCR and PSD to remove the silent parts. We were able to significantly compress the original input file to about one-fourth it's size. The input and audio file can be played. Additionally, the amplitude and frequency plot of the input and output signal have been displayed allowing to differentiate between the input and output audio files visually. It was observed that the results obtained using PSD were superior and had lower data loss with a better defined power spectrum when compared to the results of ZCR algorithm.

CHAPTER 7

Future Scope

We implemented algorithms based on ZCR and PSD. The accuracy of the above implemented techniques can be further improved. New techniques in Artificial Intelligence like Recurrent Neural Networks (RNN), Hidden Markov Model (HMM), etc. are emerging which have proven to increase the accuracy and further decrease data loss. Hence, a lot of future scope is present for this topic, open for further advancement.

References

[1] Theodoros Giannakopoulos, Aggelos Pikrakis, *Introduction to Audio Analysis*, Academic Press, 2014, pp 59-103

[2] Simon Graf, Tobias Herbig, Markus Buck & Gerhard Schmidt, "Features for voice activity detection: a comparative analysis," "*EURASIP Journal on Advances in Signal Processing",* 2015

[3] Thomas Drugman, Member, IEEE, Yannis Stylianou, Senior Member, IEEE, Yusuke Kida, Masami Akamine, Senior Member, IEEE, "Voice Activity Detection: Merging Source and Filter-based Information "

[4] Tom Bäckström, "Voice activity detection (VAD)", Introduction to Speech Processing, Aalto University Wiki, Aug 12, 2020

[5] Javier Ramirez, Jose C. Segura 1, Carmen Benitez, Angel de la Torre, Antonio Rubio 2, " Efficient voice activity detection algorithms using long-term speech information", 8 October 2000

[6] McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python." In Proceedings of the 14th python in science conference, pp. 18-25. 2015.

[7] freeCodeCamp, "Learn Python – Full Course for Beginners [Tutorial]: https://www.youtube.com/watch?v=rfscVS0vtbw"

[8] Hiroshi Matsuura ."Cygnus research international. Internet: https://www.cygres.com.