TOOLS FOR ANALYZING SPEECH SIGNALS

Göktuğ Yıldırım, yildirimg@mef.edu.tr

MEF University Electrical and Electronics Department, Istanbul, Turkey

***Abstract*— The goal of this project is to develop a GUI (Graphical User Interface) that can analyze the given speech signals using digital signal processing techniques. Signal processing is one of the most important topics in systems engineering and electronics engineering. In general, analysis on analog and digital signals can be defined as application to various systems by detecting temporal and spatial changes. In relation to that, speech processing is one of the subtitles of the signal processing. Although MATLAB is one of the most popular signal analyzing tools available today and it provides a detailed environment for general signal processing and analyzing tasks, it does not have a user-friendly interface for sound analyzing because MATLAB is not just a customized tool for speech processing. On the other hand, WaveSurfer is a widely used audio editor for acoustic-phonetic studies, but according to my observations, it has not appropriated enough user-friendly interface for basic speech analyzing tasks. It has lots of settings properties. So, a large amount of settings may be overwhelming to a novice user. In this project, loudness, pitch, formant frequencies were used as acoustic features to develop this tool. Thanks to this developed tool, novice users will be able to analyze the speech signals by observing waveform, Fast Fourier Transform (FFT), Short-Time Log Energy, Short-Time Zero Crossing, Short-Time Fourier Transform, Short-Time Real Cepstrum, spectrogram, formant frequencies, pitch changing easily. As a result of this project, the developed tool will provide an efficient speech analysis environment for the novice users especially undergraduate students in electrical and electronics engineering.**

***Keywords:* Speech Analyzing, Speech Analyzing GUI, Speech Feature Analysis**

I. Introduction

Signal processing is a research subject that electrical electronics and computer engineering students especially should care about throughout their education life. Because they take many courses including signal processing techniques in order to understand what a signal is, and how to process and analyze it. However, the related courses include the mathematical methods that are difficult to understand. For this reason, the students have difficulty understanding the visual structure of a signal according to my observations. In relation to that, MATLAB provides a satisfactory environment to analyze the signals, but it has not suitable and a user-friendly environment for the basic signal analyzing operations. This situation causes that students lose their interest regarding related courses.

Speech processing is one of the subtitles of the signal processing. Especially electrical and electronics engineering students may be related to speech processing and analyzing in undergraduate life. However, they may not be able to use MATLAB easily in order to analyze speech signals. For this reason, I developed a speech analyzing GUI that has a user-friendly interface for novice users especially students.

PyQt5 framework has been used to develop the GUI by using the Python programming language. However, all the signal processing algorithms have been implemented without using any Python module except the Fast Fourier Transform (FFT). The interface that meets the user when the program is started is shown in the Figure 1.

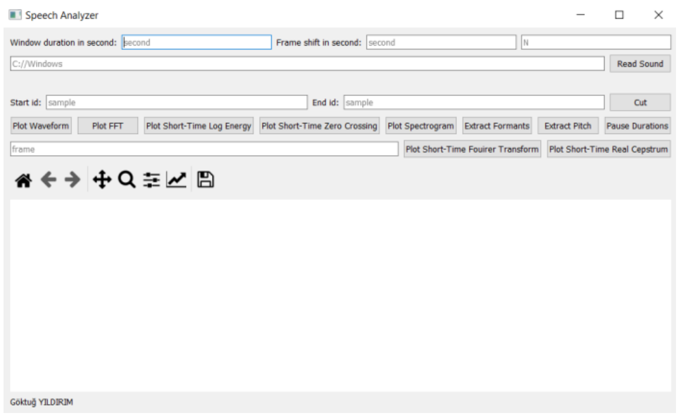


Fig. 1. Speech Analyzer GUI

A. How to Use GUI

When the program is started, user must the enter four important parameters into the interface. The first one is the window duration in seconds, the second one is the frameshift duration in seconds, the third one is the FFT N parameter, and the last one is the path of the sound in your computer. After the user enters the required parameters in the interface, the program automatically calculates, then shows the sound sample length, total frame count, sampling rate, and window length in terms of sample count. The frames are related to the short-time analysis of the speech signal. Meanwhile, the program detects the voiced, unvoiced, and silence frames automatically in the background in order to formant frequency and pitch analysis. After reading the sound, the user can the cut the sound by entering the sampling intervals, if the user wants to cut the sound.

Thanks to this developed tool, the user can plot the waveform, Fast Fourier Transform (FFT), Short-Time Log Energy, Short-Time Zero Crossing, Short-Time Fourier Transform, Short-Time Real Cepstrum, spectrogram, formant frequencies, the pitch of a sound by clicking related buttons. At this point, the user must enter the frame id to only plot Short-Time Fourier Transform and Short-Time Real Cepstrum. Also, users can download the plot as an image by clicking the download button, and zoom the plot.

II. Methodology

A. *Windowing and Frameshift*

Windowing operation is the basis of the short-time analysis of a signal. The window uses in the short time analysis, is a mathematical function that applies a weighting (often between 0 and 1) to each discrete time series sample in a finite set [1]. In this project Hamming Window (see Figure 2) was used as a window and its formula is shown in Equation 1. T refers to window length in terms of sample. The Hamming window weights the samples in the related frame into the middle of the frame. Also, frameshift is a method of preventing the discontinuities in the spectral domain. In relation to that, the user can define the Hamming window length as a second and frameshift as a second in the GUI.

$$u(t) = 0.54 - 0.46 * cos\left(\frac{2\pi t}{T}\right) \qquad (1)$$

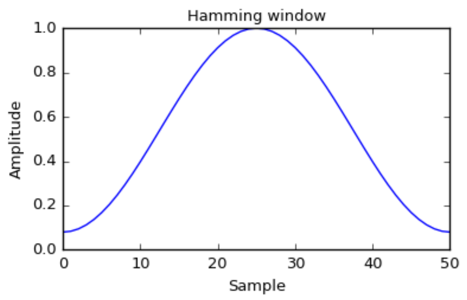


Fig. 2. Hamming Window

B. *Short-Time Log Energy*

The Short-Time Log Energy is a measure of the loudness of a signal. It is generally used to identify the loudness of a signal and classify whether the related frame is voiced or unvoiced frame. Because if the frame is voiced its Short-Time Log Energy value is high, but in the unvoiced frames, Short-Time Log Energy value of frame is lower. This feature was used in voiced unvoiced classification part. Also, Short-Time Log Energy formula is shown in Equation 2 and Equation 3. L refers to window length. R refers to shift length. r refers to frame id. m refers to sample id in a frame. Also, Short-Time Log Energy plot of a signal is shown in Figure 3.

$$E_r = \sum_{m=0}^{L-1} (s[(rR+m] \,.\, w[m])^2 , \qquad r = 0,1,2, \dots \quad (2)$$

$$\dot{E}_r = 10log_{10}E_r - max_r(10log_{10}E_r) \qquad (3)$$

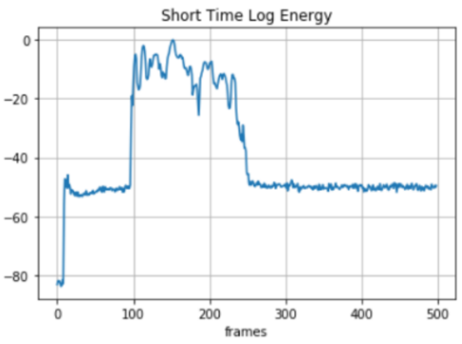


Fig. 3. Short-Time Log Energy

C. *Short-Time Zero Crossing*

Short-Time Zero Crossing is a simple measure of the frequent content of a signal. It has a higher amplitude in the unvoiced region and has a lower amplitude in the voiced region. Because signals in the unvoiced frames are more frequent than signals in the voiced frames. So, it can be used to classify whether the related frame is voiced or unvoiced frame. Short-Time Zero Crossing formula is shown in Equation 4. L refers to window length. R refers to shift length, r refers to frame id, m refers to sample id in a frame. Also, the Short-Time Zero Crossing plot of a signal is shown in Figure 4.

$$Z_r = \frac{R}{2L} \sum_{m=0}^{L-1} |sgn(s[rR+m]) - sgn[(rR+m-1])| \quad (4)$$
$$r = 0,1,2, \dots$$

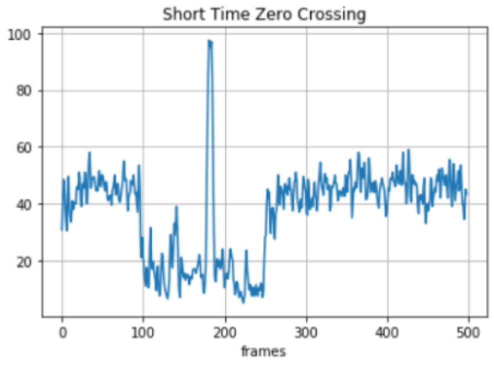


Fig. 4. Short-Time Zero Crossing

D. Spectrogram

The spectrogram representation is one of the most powerful short time signal analysis methods. It is formed by combining lots of Short-Time Fourier Transform frames. The x-axis of spectrogram refers to frame ids, y-axis refers to frequencies of a signal in the Short-Time Fourier Transform. In fact, the spectrogram is a 3D tensor. However, the third dimension of a spectrogram indicates that the amplitudes of signals is represented with colors. The higher amplitudes are represented with more intense colors. Also, the spectrogram used in GUI was implemented without using any framework. If the length of window used in spectrogram is increased in the time axis, it will be a narrowband spectrogram. It provides a more detailed frequency axis. The length of window used in spectrogram is decreased in the time axis, it will be wideband spectrogram. It provides a more detailed time axis. Figure 5 shows the wideband and narrowband spectrogram.

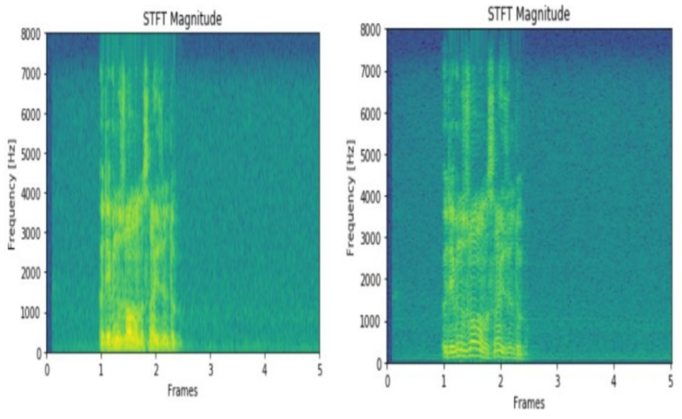


Fig. 5. The left one is wideband spectrogram. The right one is narrowband spectrogram.

E. Voiced/Unvoiced Frame Classification

As I noticed that in the part of expressing how to use GUI, when the user runs the program, voiced/unvoiced frame classification is carried out automatically. Because users may want to plot pitch contour of the signal. Since the pitch feature is defined only for voiced sounds, I had must need to write a pitch extraction algorithm that works on the only voiced frames. Also, users may want to extract the formant frequencies. At this point, I need to detect silence frames. Because formant frequencies are defined both voiced and unvoiced sounds. In relation to that, I wrote an algorithm that works on both voiced and unvoiced frames expect the silence frames, to extract formant frequencies. I just detected voiced and unvoiced frames. So, I stated the rest of the frames as a silence frame.

I classified voiced and unvoiced sound using Short-Time Log Energy and Short-Time Zero Crossing Rate (ZCR). Since, I know that voiced sounds have high energy and low ZCR, while unvoiced sounds are on the opposite: low energy but high ZCR, Short-Time Energy and ZCR was calculated to classify voiced and unvoiced sounds (see Figure 6). Short-Time Log Energy formula is shown in Equation 3 above. Also, ZCR formula is shown in Equation 4 above.

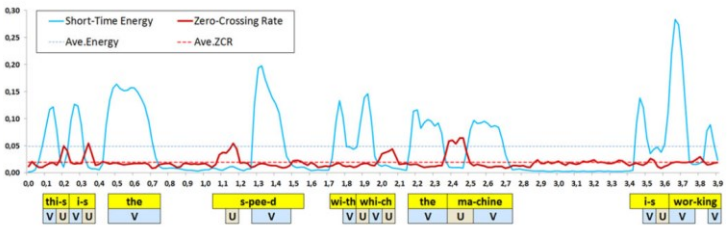


Fig. 6 Voiced/Unvoiced classification with short time analysis, image published on [2], copyright belongs to original author(s).

To classify voiced and unvoiced sounds, I followed a heuristic-adaptive decision scheme [2]. Then the average energy value of the whole sound E and average ZCR value of the whole sound Z are calculated. For i_{th} frame in the whole frames, if $E_i > E$ and $E_i > Z_i$, it is taken as a voiced frame. To detect unvoiced frames in the whole frames, I followed the conditions below:

If the frame is not a voiced sound
$E_i < \max(E, Z_i)$
$Z_i > Z \times 1.5$ or $(E_i + Z_i) < E$

If all conditions are satisfied, related frame is taken as an unvoiced sound.

F. Interpolation in the Voiced and Unvoiced Frame Lists

After detecting voiced and unvoiced framed ids according to algorithm expressed above, I stored these ids into two different list. Since the classification algorithm does not work well, I observed some discontinuities in these lists. For instance, assume that voiced and unvoiced frame id lists shown in below:

- List of voiced frame ids: [122 123 124 discontinuity 129 130]
- List of unvoiced frame ids: [321 322 323 discontinuity 330 331 332]

Since these list ids are unreasonable, I decided to write an algorithm that can interpolate the lists appropriately. Therefore, I defined the N parameter. If the discontinuities difference between two frame ids is lower than the N parameter, my algorithm interpolates the list. After the interpolation new lists are formed as shown below.

- List of voiced frame ids: [122 123 124 125 126 127 128 129 130]
- List of unvoiced frame ids: [321 322 323 324 325 326 327 328 329 330 331 332]

If the discontinuity is higher than N parameter, I assumed that it is really silence region and my algorithm does not interpolate them.

Also, this algorithm runs automatically when the user reads the sounds.

G. Short-Time Real Cepstrum and Pitch Detection

After providing interpolated voiced frame ids, I used Short-Time Real Cepstrum to extract pitch of each voiced frames. So, the real cepstrum of voiced frames were calculated. Calculation steps of real cepstrum of voiced frames are shown in Figure 7 below.

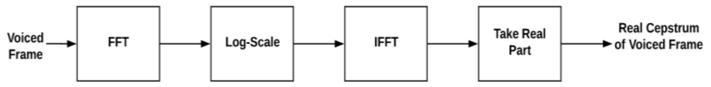


Fig. 7 Calculation Steps of Real Cepstrum of Voiced Frames

According to Figure 7, firstly Fast Fourier Transform (FFT) of voiced frames is calculated. Secondly, I took the logarithm of the FFT. Then, I took the Inverse Fourier Transform of the spectrum. At the last step, I took the real parts of the cepstrum. Therefore, the real cepstrum of the voiced frame was provided. Real cestrum of a voiced frame shown in Figure 8.

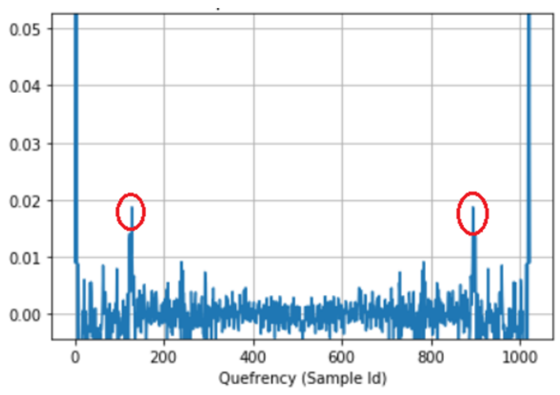


Fig. 8 Real Cepstrum of a Voiced Frame

After providing the real cepstrum of the voiced frame, strong peaks in the high quefrency were detected. After that, the frame id of the strong peak was divided into the sampling rate. Therefore, the pitch period of the voiced frame was provided. Therefore, pitch frequency was provided for each voiced frame with $1/T_s$. Also, GUI can plot the real cepstrum of any frame with the method explained above.

H. Formant Frequency Extraction with Low-time Liftering

Since formant information comes from the vocal tract filter, the formant frequency feature is defined both voiced and unvoiced sounds. Because vocal tract filter shapes the excitation comes from voiced and unvoiced sounds. If we take FFT of any frame, we cannot observe smooth enough to detect formant frequencies especially for voiced frames. Because impulse train excitation and formant information are mixed with each other. To deal with it and provide smoother spectral envelope, I applied low-time liftering on the both voiced and unvoiced frames after providing real cepstrum of the frame. Figure 9 shows us the real cepstrum of voiced frame before low-time liftering and real ceptstrum of the same frame after low-time liftering.

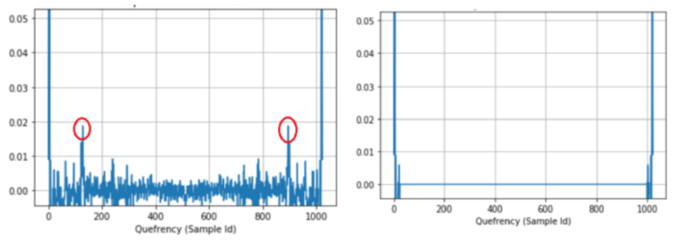


Fig. 9 Left one is before liftering, right one is after liftering.

Therefore, I deleted the excitation information of the sound after the calculation of low-time liftered real cepstrum of frame. Also, Figure 10 shows us spectrum of the frame before liftering and after liftering.

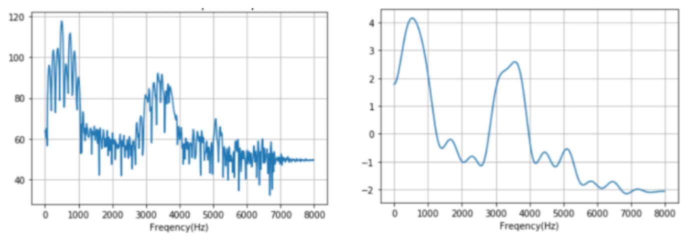


Fig. 10 Left one is FFT of frame before low-time liftering, right one is FFT of low-time liftered real cepstrum.

According to Figure 10, I provided a smoother envelope to detect formant frequencies. After low-time liftering operations, I took FFT of real cepstrum of all voiced and unvoiced frames. At this point, the problem becomes finding maxima ids. To deal with it, I used the SciPy framework in order to find maxima ids of the envelope. However, this framework finds a different number of maxima for each frame. So, I deleted maximas, if the detected maxima count is higher than 4 in order to show just only F1, F2, F3, F4. After this operation I provided the maxima FFT ids shown in below as an example:

FFT (N) parameter is 1024, but I aim to show formants in spectrogram with fs/2 and N/2 mapping.

- Frame x1: [124 252 380 480]
- Frame x2: [140 270 350 470]
- Frame x3: [110 280 360 460]
- Frame x4: [133 240 370 450]

I observed that if I used these ids directly, formants will not be shown in the spectrogram smoothly. To deal with it I defined a window length for example 3. I took the average of first 3 formant id vectors, then I assigned the result to Frame x1. After that I shifted this window one step. At this time, I took average of x2, x3, x4 vectors, then I assigned the result Frame x2. This operation pursues until the end of the list. Therefore, I provided smoother maxima ids. If I increased the window length, I provided much more smooth maxima ids. This operation acts like a low pass filter defined as linear constant-coefficient difference equations in the time domain. I also used this technique to plot smoother pitch contour.

After providing smoother maxima ids, I created zero vectors that have the length of FFT (N) and I add strong peaks into zeros vectors at the related maxima ids in order to show formant frequencies on the spectrogram. Generally, my algorithm works well. However, window length defined by me for smoothing, may not be appropriate for every sound.

I. Pause Duration

Pause duration refers to there is no speech in the related region in the speech signal. I assumed that the regions that are excepted the voiced and unvoiced regions, are silence regions. However, the algorithm cannot work well because it cannot distinguish the unvoiced region and silence regions properly. I also show the paused regions into spectrogram. Purple color refers to there is speech, yellow color refers to there is no speech in the related frames. I created zeros vector and I added strong amplitudes at the N/4 FFT id. Also, these amplitudes are different for speech regions and pause regions in order to provide different colors.

III. Results

I recorded “o” sound to test my GUI.

A. Plotting Waveform of Speech Signal

Figure 11 shows us waveform of a speech signal.

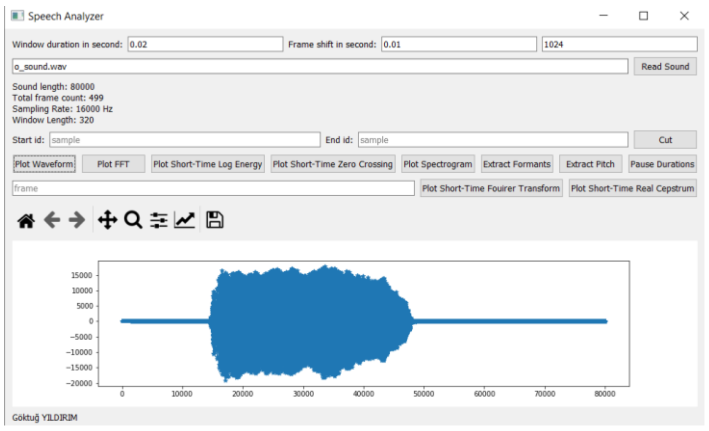

Speech Analyzer
Window duration in second: 0.02
Frame shift in second: 0.01
1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample
End id: sample
frame
Göktuğ YILDIRIM

Fig. 11 Waveform of a speech signal

B. Plotting FFT of Speech Signal

Figure 12 shows us FFT of a speech signal.

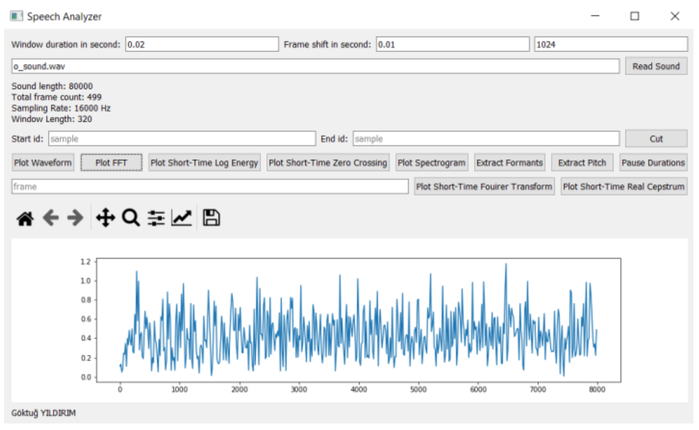

Speech Analyzer
Window duration in second: 0.02
Frame shift in second: 0.01
1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample
End id: sample
frame
Göktuğ YILDIRIM

Fig. 12 FFT of a speech signal

I cropped the sound in order to get rid of high frequencies comes from noise. Figure 13 shows the FFT of cropped speech signal signal.

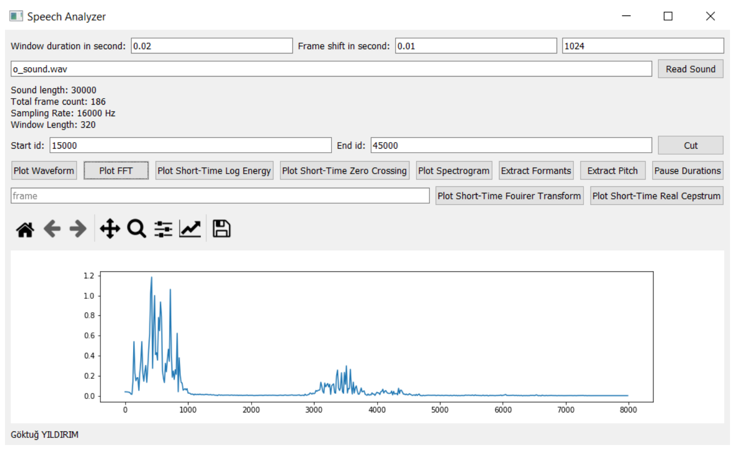

Speech Analyzer
Window duration in second: 0.02
Frame shift in second: 0.01
1024
o_sound.wav
Sound length: 30000
Total frame count: 186
Sampling Rate: 16000 Hz
Window Length: 320
Start id: 15000
End id: 45000
frame
Göktuğ YILDIRIM

Fig. 13 FFT of a cropped speech signal

C. Plotting Short-Time Log Energy

Figure 14 shows the plotting of Short-Time Log Energy.

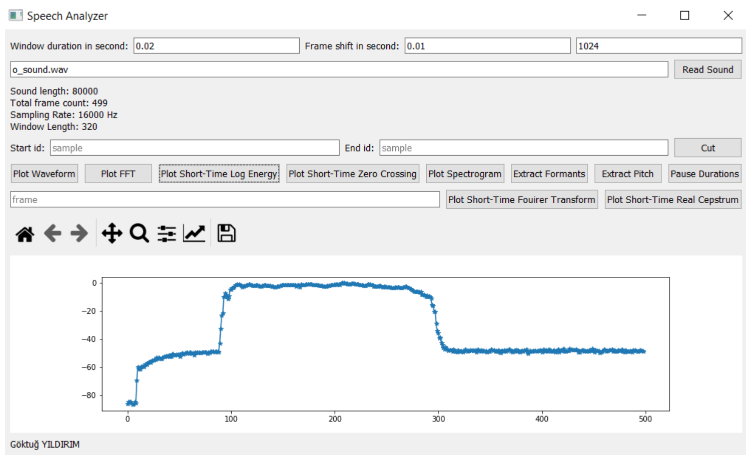

Speech Analyzer
Window duration in second: 0.02
Frame shift in second: 0.01
1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample
End id: sample
frame
Göktuğ YILDIRIM

Fig. 14 Short-Time Log Energy of a speech signal

D. *Plotting Short-Time Zero Crossing*

Figure 15 shows the plotting of Short-Time Zero Crossing.

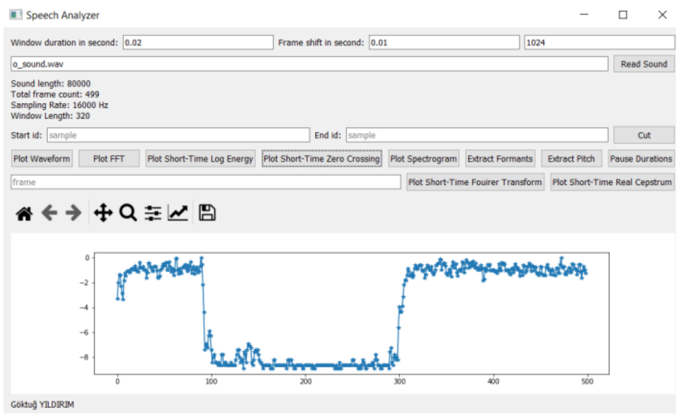

Window duration in second: 0.02 Frame shift in second: 0.01 1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample End id: sample
frame
Göktuğ YILDIRIM

Fig. 15 Short-Time Zero Crossing of a speech signal

E. *Plotting Spectrogram*

Figure 16 shows the plotting of spectrogram.

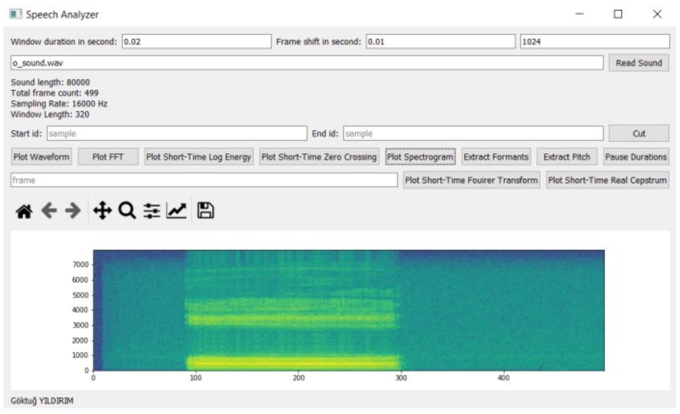

Window duration in second: 0.02 Frame shift in second: 0.01 1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample End id: sample
frame
Göktuğ YILDIRIM

Fig. 16 Spectrogram

F. *Plotting Formant Frequencies*

Note that: In fact, there is no discontinuities, when I zoom the plot, I can see them. I cannot understand this issue. I will explain this in the demonstration. Figure 17 shows the plotting of formant frequencies.

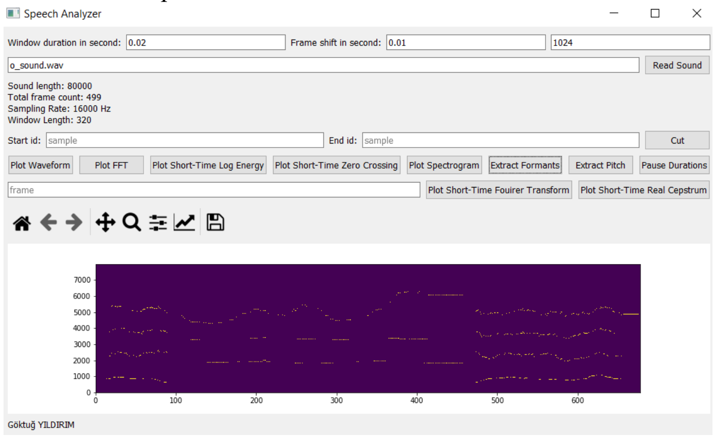

Window duration in second: 0.02 Frame shift in second: 0.01 1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample End id: sample
frame
Göktuğ YILDIRIM

Fig. 17 Formant Frequencies

G. *Plotting Pitch Contour*

Figure 18 shows the plotting of pitch contour.

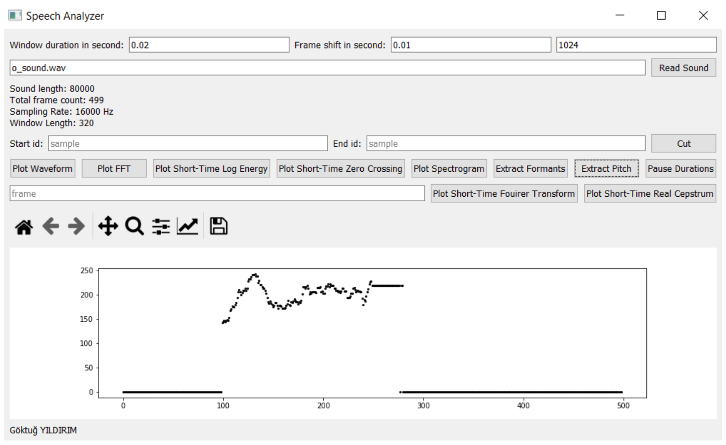

Window duration in second: 0.02 Frame shift in second: 0.01 1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample End id: sample
frame
Göktuğ YILDIRIM

Fig. 18 Pitch Contour

H. *Plotting Short-Time Fourier Transform*

Figure 19 shows the plotting of Short-Time Fourier Transform.

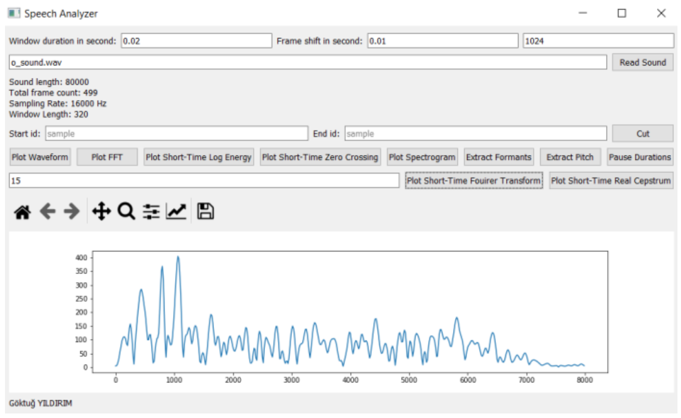

Window duration in second: 0.02 Frame shift in second: 0.01 1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample End id: sample
15
Göktuğ YILDIRIM

Fig. 19 Short-Time Fourier Transform

İ. *Plotting Short-Time Real Cepstrum*

Figure 20 shows the plotting of Short-Time Real Cepstrum.

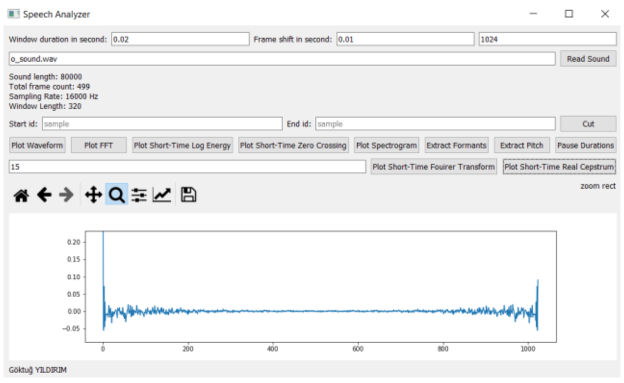
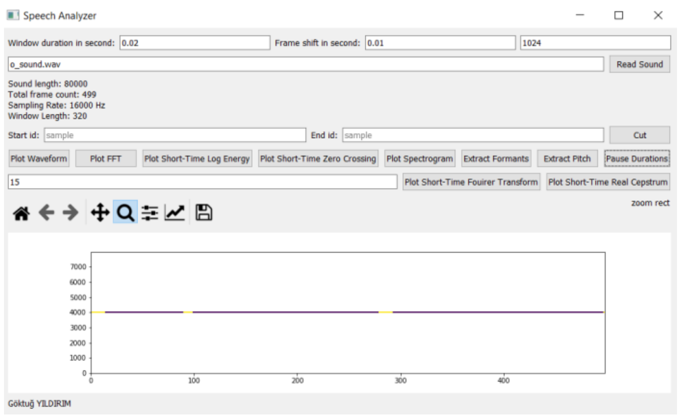

Window duration in second: 0.02 Frame shift in second: 0.01 1024
o_sound.wav
Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320
Start id: sample End id: sample
15
Göktuğ YILDIRIM

Fig. 20 Short-Time Real Cepstrum

J. *Plotting Pause Duration*

Figure 21 shows the plotting of pause durations. Yellow color refers to there is no speech in the frame.



Window duration in second: 0.02
Frame shift in second: 0.01
1024
o_sound.wav

Sound length: 80000
Total frame count: 499
Sampling Rate: 16000 Hz
Window Length: 320

Start id: sample
End id: sample

15

Göktuğ YILDIRIM

Fig. 21 Pause Duration

IV. Conclusion

I accomplished the Udemy course, "Python Arayüz Programlama PyQt5 Dersleri" [3], in order to develop this GUI and I took a certificate. Thanks to this course, I learned how to build a GUI using Python programming language, and I learned fundamentals of event-based programming. Therefore, I have been able to develop this GUI.

As a result of this project, Novice MATLAB users especially, electrical and electronic engineer students who are at the beginning of their undergraduate life, can analyze the speech signals easily with the basic signal analysis techniques by using this GUI. Therefore, they will be able to understand the visual structure of a speech signal because of this GUI that has a user-friendly interface.

V. References

[1] Think-engineer.com. 2020. Window Functions And How We Use Them In DSP-Think Engineer. [online] Available at: <https://think-engineer.com/blog/dsp/window-functions-and-how-we-use-them-in-dsp> [Accessed 26 May 2020].

[2] Che, X., Yang, H., & Meinel, C. (2018). Automatic Online Lecture Highlighting Based on Multimedia Analysis. IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, 11.

[3] Mekatronik, M., 2020. Python Arayüz Programlama Pyqt5 Dersleri. [online] Udemy. Available at: <https://www.udemy.com/course/python-arayuz-programlama-pyqt5-dersleri/> [Accessed 26 May 2020].