Voice Spectrum Analyzer

Farah Sabry, Hana Shanan, Mohamed Ashraf
Nile University

Abstract
This project offers a flexible audio recording and analysis tool, built with MATLAB, that gives users the ability to customize various recording parameters. Users can specify settings such as maximum recording duration, frequency, and graphing details to tailor the recording experience. The tool allows for user-initiated recordings, providing the option to stop the process at any point or continue until the predetermined time limit is met. After the recording, the code generates graphical representations, including time-domain and frequency-domain plots, along with an energy graph, providing detailed insights into the recorded audio signal.

Contents

1 Introduction

In audio signal processing, having a simple yet effective tool for recording and analysis is crucial. This MATLAB code provides a flexible platform that allows users to set specific parameters to customize their audio recording. The interface asks for details like maximum recording time, frequency settings, and time-domain graphing preferences, making the tool adaptable to different needs. During recording, users can stop the process at any time or let it continue until the set time limit is reached. After the recording, the code generates visual representations, including time-domain and frequency-domain plots, as well as an energy graph. These graphics give users a clear understanding of the recorded audio signal, making the MATLAB code a useful resource for analyzing both the time and frequency aspects of audio data.

2 Methodology

We designed the voice spectrum analyzer using MATLAB, creating a user-friendly graphical interface with built-in UI controls. The main function, voiceCapture, includes:

2.1 Main Figure

Created using the figure function to set up the GUI.

2.2 UI Components

Six labels and entry fields, and three buttons (Start, Stop, and Play), all defined using uicontrol.

2.3 Public Variables

Eight variables, initialized to zero, to avoid callback errors, including the recorder, voice signal, and frequency settings.

2.4 StartCapture()

Triggered by the Start button, it captures audio for a user-defined duration (Tmax) and frequency (Fs). It checks for stop button presses and stores the audio in voiceSignal, displaying the time-domain graph.

2.5 StopCapture()

Activated by the Stop button, it stops the recording and displays the output.

2.6 PlaySignal()

Plays the recorded audio using the sound function.

2.7 DisplayTimeDomain()

Retrieves user-defined min/max values for time and frequency and plots the time-domain and frequency-domain graphs. It uses subplot() for multiple graphs and fft() for the frequency analysis. Energy is calculated by squaring the magnitude of each coefficient.

---

3 Results

Small description

Upon running the MATLAB code, a user-friendly audio recording and analysis tool presents a range of interactive features. The user is prompted to enter several parameters, such as:

- Frequency (in Hz)
- Initial time for the graph
- Final time for the graph
- Initial frequency
- Final frequency

When the user clicks start, the recording begins, with the option to press the stop button to halt the process immediately or wait until the set maximum time is reached. After the recording—whether stopped by the user or completed after the time limit—the MATLAB code generates insightful graphical outputs. The recorded audio is displayed in the time domain, visualizing its temporal behavior. Additionally, the Fourier Transform of the signal is shown in the frequency domain, highlighting its frequency components.

To further enhance the analysis, an energy graph is generated, offering a detailed view of energy distribution across frequencies. This combination of graphical representations allows users to thoroughly examine and interpret the recorded audio data. Additionally, users can playback the recorded audio by pressing the "play" button. In conclusion, this MATLAB code not only allows users to customize the audio recording process but also provides rich visual insights into the recorded audio signal. The time and frequency domain graphs, along with the energy distribution graph, make it a powerful tool for detailed analysis of audio signals in MATLAB.

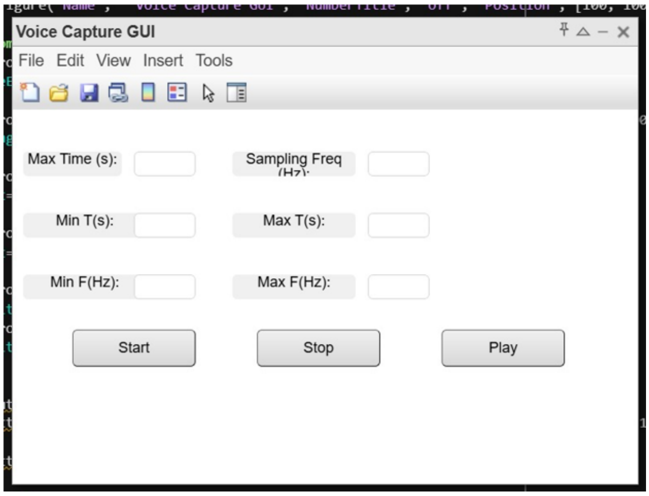

Figure 1: The GUI Appearing After Running The file

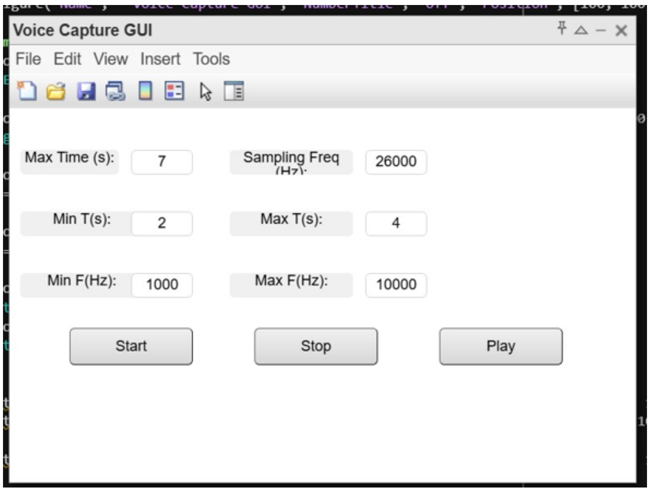

Figure 2: After Filling The Required Info.

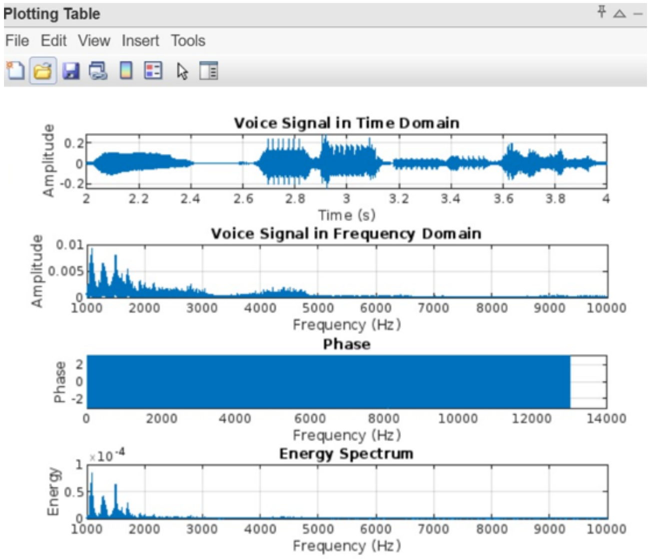

Figure 3: The Graphs Resulted From The Recorded Audio File

4 Conclusion

The MATLAB code provides an intuitive audio recording and analysis tool. It enables users to customize their experience by setting parameters like maximum recording time, frequency, and graphing details. The recording process begins at the user's command, with the option to stop it immediately or allow it to continue until the specified time limit is reached. The generated graphs, which represent the signal in both the time and frequency domains, along with the energy distribution, offer a thorough visual analysis of the recorded audio. This tool is a valuable asset for users looking to analyze and understand audio signals in MATLAB, providing insights into their temporal and frequency characteristics.

5 Appendix

```
function voiceCapture
    % Create GUI
    fig = figure('Name', 'Voice␣Capture␣GUI', 'NumberTitle', 'off', 'Position', [100, 100, 500, 300]);

    % UI components
    uicontrol('Style', 'text', 'String', 'Max␣Time␣(s):', 'Position', [10, 250, 80, 20]);
    maxTimeEdit = uicontrol('Style', 'edit', 'Position', [100, 250, 50, 20]);

    uicontrol('Style', 'text', 'String', 'Sampling␣Freq␣(Hz):', 'Position', [180, 250, 100, 20]);
```

```
    samplingFreqEdit = uicontrol('Style', 'edit', 'Position', [290, 250, 50, 20]);

    uicontrol('Style', 'text', 'String', 'Min T(s):', 'Position', [10, 200, 100, 20]);
    minEdit=uicontrol('Style', 'edit', 'Position', [100, 200, 50, 20]);

    uicontrol('Style', 'text', 'String', 'Max T(s):', 'Position', [180, 200, 100, 20]);
    maxEdit=uicontrol('Style', 'edit', 'Position', [290, 200, 50, 20]);

    uicontrol('Style', 'text', 'String', 'Min F(Hz):', 'Position', [10, 150, 100, 20]);
    minnEdit=uicontrol('Style', 'edit', 'Position', [100, 150, 50, 20])
    uicontrol('Style', 'text', 'String', 'Max F(Hz):', 'Position', [180, 150, 100, 20]);
    maxxEdit=uicontrol('Style', 'edit', 'Position', [290, 150, 50, 20]);


    startButton = uicontrol('Style', 'pushbutton', 'String', 'Start', 'Position', [50,95, 100, 30], 'Callback', @startCapture);
    stopButton = uicontrol('Style', 'pushbutton', 'String', 'Stop', 'Position', [200,95, 100, 30], 'Callback', @stopCapture);

    playButton = uicontrol('Style', 'pushbutton', 'String', 'Play', 'Position', [350, 95, 100, 30], 'Callback', @playSignal);

    % Variables
    recorder = [];
    voiceSignal = [];
    Fs=0;
    min=0;
    max=0;
    minn=0;
    maxx=0;
    stopButtonPressed = false;

    % Callback functions


    function startCapture(~, ~)
        % Retrieve inputs

        Tmax = str2double(get(maxTimeEdit, 'String'));
        Fs = str2double(get(samplingFreqEdit, 'String'));


```

```
        % Create audiorecorder object
        recorder = audiorecorder(Fs, 16, 1);

        stopButtonPressed = false;

        % Record voice for Tmax seconds
        record(recorder, Tmax);

        % Wait for recording to complete

        for t = 1:Tmax
            if stopButtonPressed
                % Stop the recorder and exit the loop
                stop(recorder);
                break;
            end
            % Pause for 1 second
            pause(1);
        end

        % Get recorded voice signal

        voiceSignal = getaudiodata(recorder);

        % Display voice signal in time domain
        displayTimeDomain(voiceSignal, Fs);

    end

    function stopCapture(~, ~)
        % Stop the recorder
        stopButtonPressed=true;
        stop(recorder);

        % Get recorded voice signal
        voiceSignal = getaudiodata(recorder);

        % Display voice signal in time domain
        displayTimeDomain(voiceSignal, Fs);


    end

    function playSignal(~, ~)
        % Play the recorded signal
        sound(voiceSignal, Fs);
    end

    function displayTimeDomain(signal, Fs)
        % Display voice signal in time domain
        min = str2double(get(minEdit, 'String'));
```

```
        max = str2double(get(maxEdit, 'String'));
        minn = str2double(get(minnEdit, 'String'));
        maxx =str2double(get(maxxEdit, 'String'));
        time = (0:length(signal)-1) / Fs;
        figure('Name', 'Plotting Table', 'NumberTitle', 'off');
        subplot(411);
        plot(time, signal);
        xlim([min max]);
        xlabel('Time (s)');
        ylabel('Amplitude');
        title('Voice Signal in Time Domain');
        grid on;

        % Display voice signal in frequency domain nd plots the amplitude
        % of the signal
        N = length(signal);
        Y = fft(signal);
        f = Fs * (0:(N/2))/N;
        Z=((1/Fs)*Y);
        P = abs((1/Fs)*Y);
        subplot(412);
        plot(f, P(1:N/2+1));
        xlim([minn maxx]);
        title('Voice Signal in Frequency Domain');
        xlabel('Frequency (Hz)');
        ylabel('Amplitude');
        grid on;

        %plots the phase of the signal
        subplot(413);
        plot(f, angle(Z(1:N/2+1)));
        title('Phase');
        xlabel('Frequency (Hz)');
        ylabel('Phase');
        grid on;

        %plots energy
        subplot(414);
        plot(f, (P(1:N/2+1)).^2);
        xlim([minn maxx]);
        title('Energy Spectrum');
        xlabel('Frequency (Hz)');
        ylabel('Energy');
        grid on;

    end
end
```

6 Refrences

Donald E. Knuth. The TEX book. Addison-Wesley, 1984. Pabon, P., Trenstorm, S. (2018, September 27). Feature maps of the acoustic spectrum of the voice.

6.1 LaTex document code link

https://www.overleaf.com/read/ngbryznjnwxmd14a23