Enriching lives, opening minds.

ESCOLA SUPERIOR DE TECNOLOGIA IPCA

SIGNAL PROCESSING REPORT

Prepared by

Seidy KANTE

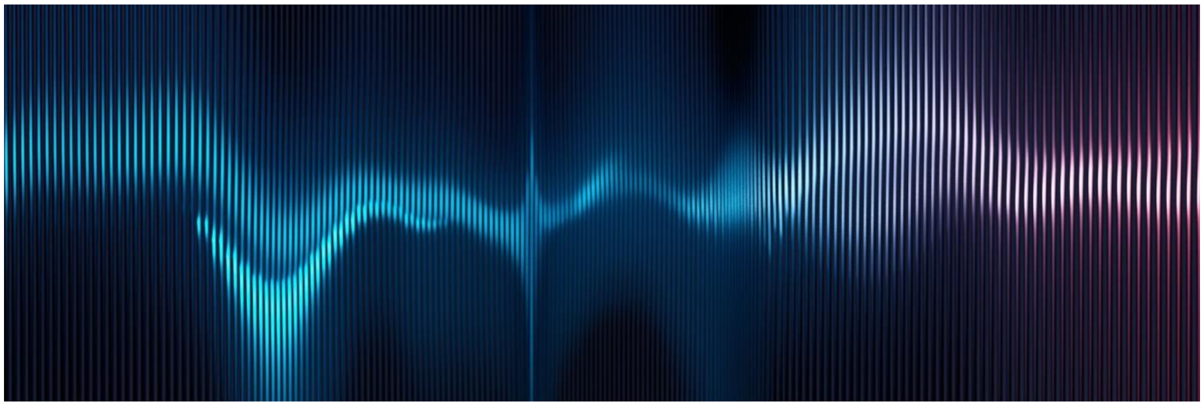
a50896@alunos.ipca.pt

JAN 18, 2023

INTRODUCTION

The goal of this project is to develop a MATLAB script for the acquisition, analysis, and processing of voice audio signals. The project is divided into two main topics: Topic 1 focuses on basic functions such as loading, playing, plotting audio signals, and gender recognition, while Topic 2 addresses the filtering of corrupted audio signals.

TOPIC 1: Audio Signal Functions



1. Load Function

The project involves implementing a load function capable of importing audio files in .mp3 or .wav formats.

```matlab
function [signal, fs] = loadAudio(filePath)

    % Loads audio signal from a .wav or .mp3 file
    %
    % Input:
        % filePath: Path to the audio file
    %
    % Output:
        % signal: Audio signal data
        % fs: Sampling frequency of the audio signal

    [signal, fs] = audioread(filePath);
end
```

The script successfully loads audio signals from the specified file paths.

2. Play Function

A playAudio function has been implemented to play the imported audio signal.

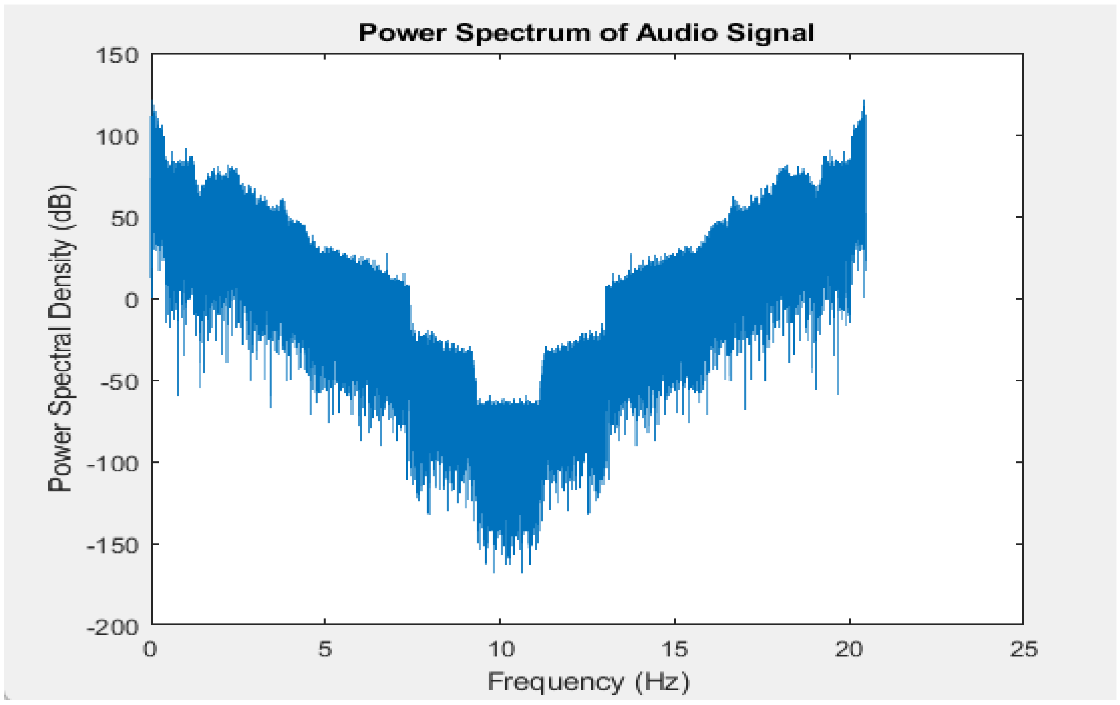
```
function [player] =playAudio(signal, fs)

    % Plays the audio signal using MATLAB's audio player
    %
    % Input:
        % signal: Audio signal data
        % fs: Sampling frequency of the audio signal

    [player] = audioplayer(signal, fs);

    player.play;

end
```

This functionality utilizes MATLAB's audioplayer, providing a convenient way to audibly assess the loaded audio.

3. Plot Function

The plotAudioSignals function allows for the visualization of the imported audio signal in the time domain, as well as the display of its power spectrum.

```
function plotAudioSignals(signal, fs)
    % Plots the imported audio signal in time domain and its power spectrum
    %
    % Input:
        % signal: Audio signal data
        % fs: Sampling frequency of the audio signal

    % Time domain plot
    t = 1:length(signal);
    time = t/fs;
    plot(time, signal);
    title('Audio Signal in Time Domain');
    xlabel('Time (s)');
    ylabel('Amplitude');

    % Power spectrum plot
    f = (0:length(signal)-1)/fs;
    Pxx = abs(fft(signal)).^2;
    Pxx_db = 20*log10(Pxx);
    plot(f, Pxx_db);
    title('Power Spectrum of Audio Signal');
    xlabel('Frequency (Hz)');
    ylabel('Power Spectral Density (dB)');

end
```



This feature aids in understanding the characteristics of the audio signal over time and in the frequency domain.

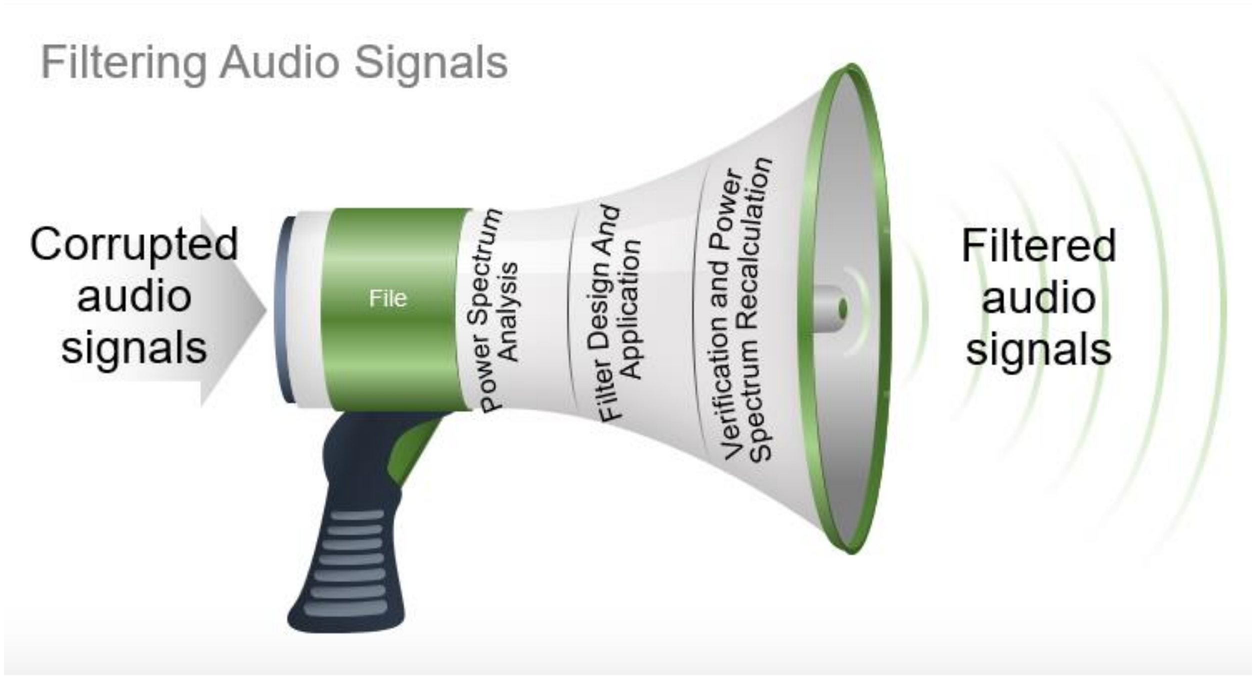
4. Gender Recognition Function

The project includes a gender recognition function (recognize_binary_gender) that analyzes the power spectrum of the audio signal to determine the likely gender of the speaker.

```
function gender = recognizeBinaryGender(audio, fs)

    nfft = 2^nextpow2(length(audio));
    audio_fft = fft(audio, nfft);
    power_spectrum = abs(audio_fft).^2 / (fs * length(audio));
    freq = (0:nfft-1) * fs / nfft;

    % Find maximum frequency
    [~, max_idx] = max(power_spectrum); % ~:max_power
    max_freq = freq(max_idx);

    % Calculate distances
    male_distance = abs(125 - max_freq);
    female_distance = abs(200 - max_freq);

    % Determine gender based on distances
    if male_distance <= female_distance
        gender = 'Male';
    else
        gender = 'Female';
    end
end
```

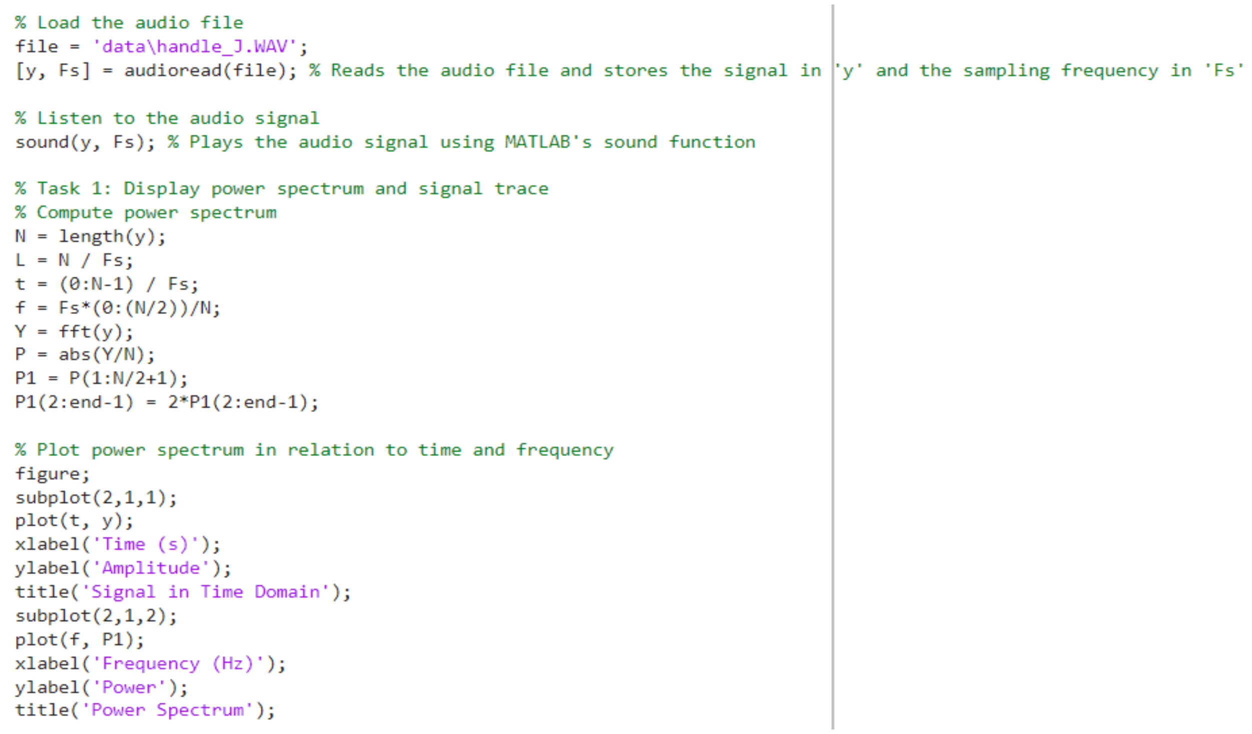
The implementation considers peak distances at specified frequency ranges to categorize the voice as male or female.

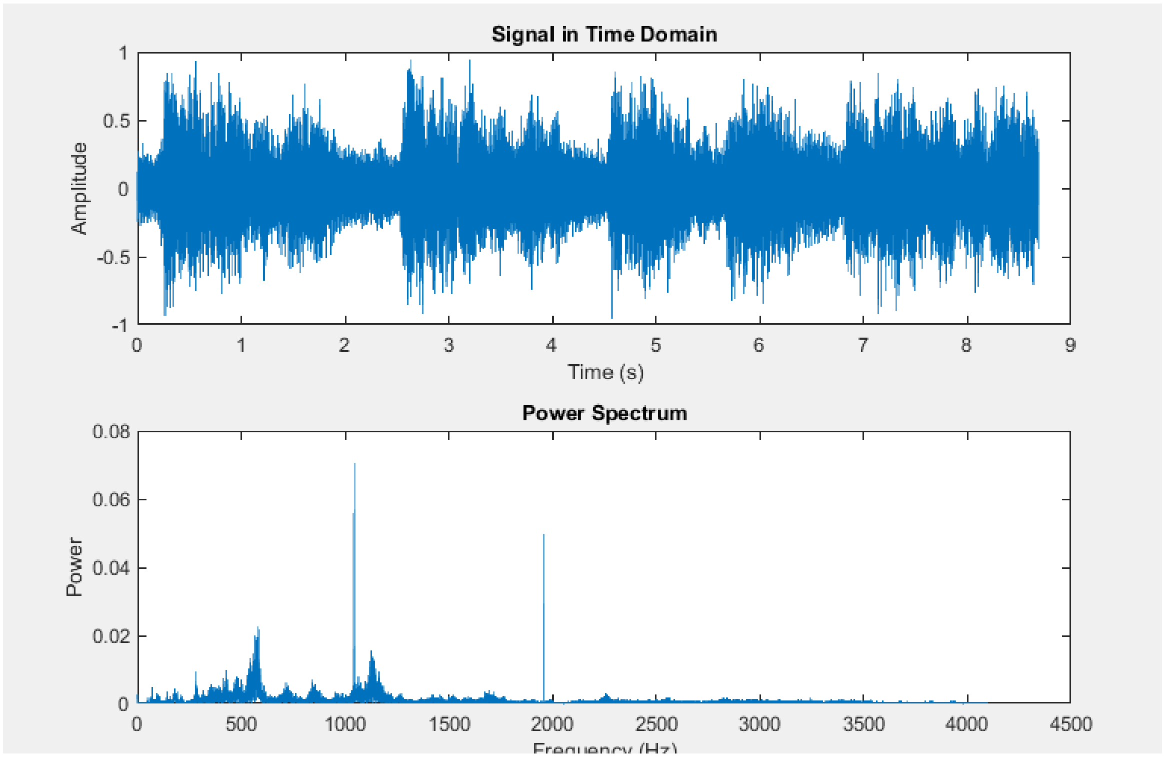
TOPIC 1: Filtering Audio Signals



1. Power Spectrum Analysis

The script loads a corrupted audio file, 'handle_J.WAV,' and displays its power spectrum in relation to time and frequency.


```
% Load the audio file
file = 'data\handle_J.WAV';
[y, Fs] = audioread(file); % Reads the audio file and stores the signal in 'y' and the sampling frequency in 'Fs'

% Listen to the audio signal
sound(y, Fs); % Plays the audio signal using MATLAB's sound function

% Task 1: Display power spectrum and signal trace
% Compute power spectrum
N = length(y);
L = N / Fs;
t = (0:N-1) / Fs;
f = Fs*(0:(N/2))/N;
Y = fft(y);
P = abs(Y/N);
P1 = P(1:N/2+1);
P1(2:end-1) = 2*P1(2:end-1);

% Plot power spectrum in relation to time and frequency
figure;
subplot(2,1,1);
plot(t, y);
xlabel('Time (s)');
ylabel('Amplitude');
title('Signal in Time Domain');
subplot(2,1,2);
plot(f, P1);
xlabel('Frequency (Hz)');
ylabel('Power');
title('Power Spectrum');
```



This initial analysis helps identify noise frequencies and plan the subsequent cleaning process.

2. Filter Design and Application

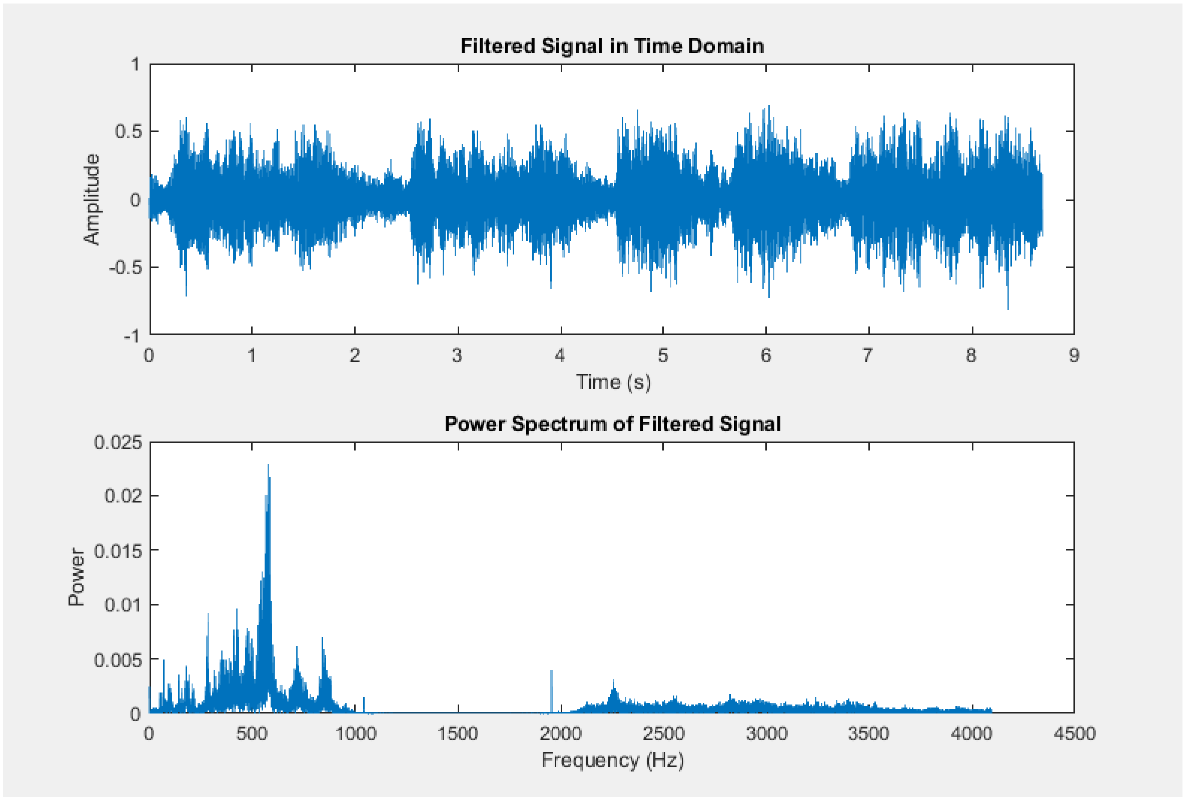
Noise frequencies are identified, and a band-stop filter is designed to remove the noise. The design process involves specifying frequencies to filter out and determining the bandwidth. The filter is then applied to the audio signal, resulting in a cleaner version.

```
% Task 2: Identify noise frequencies and design filter(s)
% Analyze the spectrum to identify noise frequencies
% Design and apply a filter to remove noise

frequencies_to_remove = [1000 2000]; % Define frequencies to filter out
bandwidth = 200; % Bandwidth for the filter

% Design band-stop filter
d = designfilt('bandstopiir','FilterOrder',20, ...
               'HalfPowerFrequency1',frequencies_to_remove(1)-bandwidth/2, ...
               'HalfPowerFrequency2',frequencies_to_remove(2)+bandwidth/2, ...
               'DesignMethod','butter','SampleRate',Fs);

% Plot the frequency response of the filter
% fvtool(d, 'Fs', Fs);

% Task 3: Filter the audio signal
y_filtered = filter(d, y); % Apply the designed filter to the audio signal
```

3. Verification and Power Spectrum Recalculation

The script allows the user to listen to the filtered audio and verifies the removal of noise. Additionally, it recalculates and plots the power spectrum of the filtered signal, providing a comprehensive assessment of the cleaning process.

```
% Task 4: Listen to the filtered audio signal and calculate power spectrum
sound(y_filtered, Fs); % Play the filtered audio signal using sound function

% Calculate power spectrum of the filtered signal
Y_filtered = fft(y_filtered);
P_filtered = abs(Y_filtered/N);
P1_filtered = P_filtered(1:N/2+1);
P1_filtered(2:end-1) = 2*P1_filtered(2:end-1);

% Plot power spectrum of the filtered signal
figure;
subplot(2,1,1);
plot(t, y_filtered);
xlabel('Time (s)');
ylabel('Amplitude');
title('Filtered Signal in Time Domain');
subplot(2,1,2);
plot(f, P1_filtered);
xlabel('Frequency (Hz)');
ylabel('Power');
title('Power Spectrum of Filtered Signal');
```



Conclusion

In conclusion, the project successfully addresses the fundamental aspects of audio signal processing, including loading, playing, and analyzing signals. The implementation of gender recognition and the filtering of corrupted audio signals demonstrates a practical application of signal processing techniques.