Speech Signal Enhancement: A Comprehensive Analysis

Muhammad Sibtain Haider
2021459
Department of Computer Engineering
Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Topi, Pakistan
u2021459@giki.edu.pk

***Abstract*—This paper presents a systematic approach to enhancing the quality of a speech signal through the application of a custom bandpass filter to remove unwanted noise. The methodology includes loading an audio file, analyzing its frequency content, designing and applying a custom bandpass filter, and finally saving the enhanced signal as a new WAV file. The implementation is discussed with relevant code snippets and visualizations, and the effectiveness of the enhancement process is evaluated. The paper concludes with insights into potential enhancements and the adaptability of the approach to various scenarios.**

I. Introduction

The goal of this project is to enhance the quality of a speech signal by employing a custom bandpass filter to filter out unwanted noise. The implemented code loads an audio file, conducts a comprehensive analysis of its frequency content, designs and applies a custom bandpass filter, and finally saves the enhanced signal as a new WAV file.

II. Original Signal Analysis

The original audio signal is initially loaded and visually represented in both time and frequency domains. The time-domain plot illustrates the amplitude of the speech signal over time, while the frequency-domain plot showcases the spectrum of the signal. These visualizations aid in understanding the characteristics of the original speech signal and identifying the frequency range of interest.

A. Code Implementation

The following code segment demonstrates the initial analysis of the original audio signal:

Listing 1. Original Signal Analysis Code

```
[f, fs] = audioread('voice.wav');
pOrig = audioplayer(f, fs);
N = size(f, 1);
figure;
subplot(4, 1, 1);
plot(1:N, f);
title('Original␣Audio␣Signal');
xlabel('Sample');
ylabel('Amplitude');
```

III. Custom Bandpass Filter Design

To enhance the speech signal by removing unwanted noise, a custom bandpass filter is designed. The filter design parameters, including the filter order and frequency range, are chosen based on the characteristics of the signal and the identified noise frequencies.

A. Code Implementation

The following code segment demonstrates the design of the custom bandpass filter:

Listing 2. Custom Bandpass Filter Design Code

```
% Filter design parameters
n = 21; % Filter order
beginFreq = 800; % Start frequency of the
    red↪  bandpass filter
endFreq = 12000; % End frequency of the
    red↪ bandpass filter
% Design a custom bandpass filter
[b, a] = my_bandpass_filter(n, beginFreq,
    red↪  endFreq, fs);
```

B. Custom Bandpass Filter Function

The custom bandpass filter is designed using a windowed sinc function. The window function, in this case, a rectangular window, is applied to shape the sinc function, and the resulting filter is normalized to have unity gain at DC.

1) Code Implementation: The following code segment demonstrates the custom bandpass filter function:

Listing 3. Custom Bandpass Filter Function Code

```
% Function for custom bandpass filter
function [b, a] = my_bandpass_filter(
    red↪ order, beginFreq, endFreq, fs)
% Design a bandpass filter using a
    red↪ windowed sinc function
f1 = beginFreq / fs;
f2 = endFreq / fs;
midFreq = (f1 + f2) / 2;
% Create a time vector for the filter
    red↪ coefficients
t = (-order/2 : order/2) / fs;
```

```
% Rectangular window (you can replace
    red↪ this with another window
    red↪ function)
window_func = ones(1, order + 1);
% Windowed sinc function
sinc_func = sin(2 * pi * midFreq * t) ./
    red↪ (pi * t);
% Apply the window to the sinc function
bandpass_filter = sinc_func .*
    red↪ window_func;
% Normalize the filter to have unity gain
    red↪  at DC
bandpass_filter = bandpass_filter / sum(
    red↪ bandpass_filter);
b = bandpass_filter;
a = 1;
end
```

IV. Applying the Bandpass Filter

The designed bandpass filter is then applied to the original speech signal to attenuate frequencies outside the specified range. The filtered signal is subsequently visualized in both time and frequency domains.

A. Code Implementation

The following code segment demonstrates the application of the bandpass filter:

Listing 4. Applying the Bandpass Filter Code

```
% Apply the bandpass filter to the
    red↪ original signal
fOut = filter(b, 1, f);
% Create an audioplayer object for the
    red↪ filtered signal
pFiltered = audioplayer(fOut, fs);
% Plot the filtered audio signal
subplot(4, 1, 3);
plot(1:N, fOut);
title('Filtered␣Audio␣Signal');
xlabel('Sample');
ylabel('Amplitude');
```

V. Filtered Signal Analysis

The frequency spectrum of the filtered signal is then analyzed to assess the effectiveness of the bandpass filter in attenuating noise. Additionally, the filtered signal is saved as a new WAV file for further evaluation and comparison.

A. Code Implementation

The following code segment demonstrates the analysis of the filtered signal:

Listing 5. Filtered Signal Analysis Code

```
% Compute the frequency spectrum of the
    red↪ filtered signal
yFiltered = fft(fOut, N) / N;
yFiltered2 = fftshift(yFiltered);
% Plot the frequency spectrum of the
    red↪ filtered signal
subplot(4, 1, 4);
plot(w, abs(yFiltered2));
title('Filtered␣Audio␣Spectrum');
xlabel('Frequency␣(Hz)');
ylabel('Magnitude');
% Save the filtered output as a WAV file
    red↪ (replace if exists)
outputFilename = 'filtered_output.wav';
audiowrite(outputFilename, fOut, fs);
```

VI. Techniques Used

A. Fourier Series and Time-Domain Analysis

In the time-domain analysis, the code utilizes the Fast Fourier Transform (FFT) to compute the frequency spectrum of the original and filtered signals.

1) Code Implementation: The following code segment demonstrates the computation of the frequency spectrum:

Listing 6. Fourier Series and Time-Domain Analysis Code

```
% Compute the frequency spectrum of the
    red↪ original signal
df = fs / N;
w = (-(N/2):(N/2)-1) * df;
y = fft(f, N) / N; % For normalizing, but
    red↪  not needed for our analysis
y2 = fftshift(y);
% Plot the frequency spectrum of the
    red↪ original signal
subplot(4, 1, 2);
plot(w, abs(y2));
title('Original␣Audio␣Spectrum');
xlabel('Frequency␣(Hz)');
ylabel('Magnitude');
% Compute the frequency spectrum of the
    red↪ filtered signal
yFiltered = fft(fOut, N) / N;
yFiltered2 = fftshift(yFiltered);
% Plot the frequency spectrum of the
    red↪ filtered signal
subplot(4, 1, 4);
plot(w, abs(yFiltered2));
title('Filtered␣Audio␣Spectrum');
xlabel('Frequency␣(Hz)');
ylabel('Magnitude');
```

B. Amplitude Modifications

The concept of amplitude modification is implicitly involved in the filtering process. The bandpass filter is designed to selectively modify the amplitudes of frequencies within the specified range, attenuating unwanted frequencies while preserving those of interest.

1) Code Implementation: The following code segment demonstrates the amplitude modification process:

Listing 7. Amplitude Modifications Code

```
% Design a custom bandpass filter
[b, a] = my_bandpass_filter(n, beginFreq,
    red↪  endFreq, fs);
% Apply the bandpass filter to the
    red↪ original signal
fOut = filter(b, 1, f);
```

C. Frequency Domain Analysis

Frequency domain analysis involves visualizing the amplitude spectra of the original and filtered signals. Fourier series concepts come into play when analyzing how different frequencies contribute to the overall signal.

1) Code Implementation: The following code segment demonstrates the computation and visualization of frequency spectra:

Listing 8. Frequency Domain Analysis Code

```
% Compute the frequency spectrum of the
    red↪ original signal
df = fs / N;
w = (-(N/2):(N/2)-1) * df;
y = fft(f, N) / N; % For normalizing, but
    red↪  not needed for our analysis
y2 = fftshift(y);
% Plot the frequency spectrum of the
    red↪ original signal
subplot(4, 1, 2);
plot(w, abs(y2));
title('Original␣Audio␣Spectrum');
xlabel('Frequency␣(Hz)');
ylabel('Magnitude');
% Compute the frequency spectrum of the
    red↪ filtered signal
yFiltered = fft(fOut, N) / N;
yFiltered2 = fftshift(yFiltered);
% Plot the frequency spectrum of the
    red↪ filtered signal
subplot(4, 1, 4);
plot(w, abs(yFiltered2));
title('Filtered␣Audio␣Spectrum');
xlabel('Frequency␣(Hz)');
ylabel('Magnitude');
```

In summary, Fourier series concepts, including frequency domain analysis and amplitude modifications, are integral to understanding and implementing the filtering process for speech signal enhancement. The provided code effectively employs these principles to selectively modify the frequency content of the original signal, resulting in an enhanced and filtered output.

VII. Conclusion

In conclusion, the provided code demonstrates a systematic approach to enhancing speech signal quality through custom bandpass filtering. The design and application of the bandpass filter are based on a careful analysis of the original speech signal's characteristics and the identification of noise frequencies. The visualizations in both time and frequency domains provide valuable insights into the signal's structure and guide the selection of filter parameters.

The effectiveness of the enhancement is assessed through the comparison of the original and filtered signal spectra. The bandpass filter successfully attenuates unwanted frequencies, preserving the essential components of the speech signal. The filtered output is saved as a new WAV file, allowing for further examination and comparison.

While this specific implementation demonstrates a manual approach to filter design, further enhancements could involve automated techniques for parameter selection based on signal analysis. This could include adaptive algorithms that adjust filter parameters in real-time to accommodate varying noise conditions.

In summary, the code showcases a practical method for enhancing speech signal quality through thoughtful filter design. The approach can be adapted and extended to suit different scenarios and may serve as a foundation for more sophisticated signal processing applications. The evaluation of the filtered signal provides valuable insights into the success of the enhancement process, contributing to the overall understanding and improvement of speech signal quality.