AUDIO SIGNAL PROCESSING AND ENHANCEMENT

VINITHA R
21ECR224

INTRODUCTION

The "Audio Signal Processing and Enhancement" project is a comprehensive MATLAB script that empowers users to manipulate and enhance WAV audio files. This versatile tool offers three fundamental audio processing functionalities: **amplification, speed alteration, and equalization.**The script generates visual representations of the processed audio signals and facilitates the saving of enhanced audio to separate output files. This project is a valuable resource for audio enthusiasts, researchers, and professionals seeking to meet their specific audio enhancement requirements and creative goals in fields such as music production and digital signal processing.

PROCEDURE

1.**Open MATLAB** and **Load Code.Give Input Audio File.**

e.g., "C:\Users\YourUsername\Documents\audiofile.wav."

2.**Amplification and speed:**Enter the value for amplification factor and Audio Playback Speed.

3. **Equalization**: The code uses a LowPass IIR filter to enhance audio quality.

4. **View Results**: You'll see plots of the original, amplified, speed-adjusted, and equalized audio.The code plays the processed audio for you to hear the effects.

5. **Save Processed Audio**: Processed audio files are automatically saved as "original_audio.wav," "amplified_audio.wav," "speed_increased_audio.wav," and "equalized_audio.wav."

6. **Review Output Files**: Locate and review the saved audio files in the directory where you ran the code.

PROGRAM:

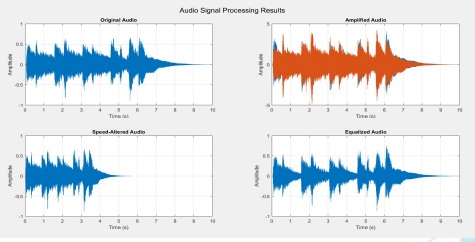
```
% Prompt the user to enter the path to the input WAV audio file
audioFilePath = input('Enter the path to the input WAV audio file: ', 's');

% Read the audio file
[audio, sampleRate]
=audioread('C:\Users\VinithaR\OneDrive\Documents\MATLAB\DSP\Prague.wav'
);
% Prompt the user to specify an amplification factor
amplificationFactor = input('Enter the amplification factor: ');
% Amplify the audio
amplifiedAudio = audio * amplificationFactor;

% Prompt the user to specify a speed increase factor (e.g., 0.5 for increasing
speed)
speedAlteredFactor = input('Enter the speed-Alteration factor: ');

% Calculate the new sample rate for speed increase
newSampleRate = sampleRate * speedAlteredFactor;

% Perform time scaling to achieve speed increase by resampling
speedAlteredAudio = resample(audio, newSampleRate, sampleRate);

% Create a time vector for plotting
t = (0:length(audio) - 1) / sampleRate;
```

```
% Normalize the audio for plotting
audio1 = audio / max(abs(audio));

amplifiedAudioNormalized = amplifiedAudio /
max(abs(amplifiedAudio));

speedAlyteredAudio = speedAlteredAudio /
max(abs(speedAlteredAudio));

% Design a custom IIR equalizer filter for audio equalization

filterType = 'low

cutoffFrequency = 1000

% Design the IIR filter using the Butterworth design
[b, a] = butter(4, cutoffFrequency / (sampleRate / 2), filterType);

% Apply the equalizer to the audio

equalizedAudio = filter(b, a, amplifiedAudio);

% Create a figure with subplots
figure;
sgtitle('Audio Signal Processing Results');
```

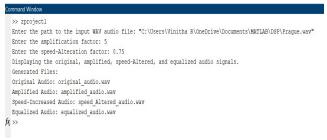
```
% Original Audio Plot
subplot(2, 2, 1);
plot(t, audio1);
title('Original Audio');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
xlim([0, 10]);

% Amplified Audio Plot
subplot(2, 2, 2);
plot(t, amplifiedAudio);
title('Amplified Audio');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
xlim([0, 10]);
%ylim([-5, 5]);

% Speed-Increased Audio Plot
subplot(2, 2, 3);
t_speedAltered = (0:length(speedAlteredAudio) - 1) / (sampleRate /
speedAlteredFactor);
plot(t_speedIncreased, speedIncreasedAudio);
title('Speed-Altered Audio');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
xlim([0, 10]);
% Equalized Audio Plot
subplot(2, 2, 4);
plot(t, equalizedAudio);
title('Equalized Audio');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
xlim([0, 10]);
```

```
% Display the four plots
disp('Displaying the original, amplified, speed-increased, and
equalized audio signals.');

% Play the audio samples
sound(audio1, sampleRate); % Play the original audio
pause(length(audio1) / sampleRate); % Pause for the duration
of the audio
sound(amplifiedAudio, sampleRate); % Play the amplified audio
pause(length(amplifiedAudio) / sampleRate); % Pause for the
duration
sound(speedIncreasedAudio, sampleRate /
speedIncreaseFactor); % Play the speed-increased audio
pause(length(speedIncreasedAudio) / (sampleRate /
speedIncreaseFactor)); % Pause for the duration
sound(equalizedAudio, sampleRate); % Play the equalized
audio
% Save the audio signals to separate output files
audiowrite('original_audio.wav', audio1,sampleRate);
audiowrite('amplified_audio.wav', amplifiedAudioNormalized,
sampleRate);
audiowrite('speed_Altered_audio.wav', speedAlteredAudio,
sampleRate);
audiowrite('equalized_audio.wav', equalizedAudio,
sampleRate);

% Display the names of the output files
disp('Generated Files: ');
disp('Original Audio: original_audio.wav');
disp('Amplified Audio: amplified_audio.wav');
disp('Speed-Altered Audio: speed_increased_audio.wav');
disp('Equalized Audio: equalized_audio.wav');
```

OUTPUT WAVEFORM:



INPUT FROM USER:



GENERATED OUTPUT AUDIO FILES:

CONCLUSION:

The "Audio Signal Processing and Enhancement" project in MATLAB is a user-friendly tool for enhancing audio files. It allows users to effortlessly control audio volume, modify playback speed for creative effects, and fine-tune frequency content using a LowPass IIR filter. The visual representations of the processed audio signals aid in quality assessment and analysis.

The project not only processes audio but also allows users to listen to the effects in real-time. Furthermore, it automatically saves the enhanced audio in separate output files, making it a practical choice for various audio-related tasks, including music production, post-processing, and digital signal processing research.

THANK YOU!