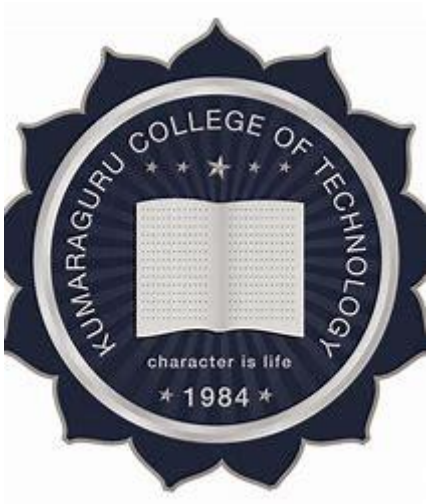
Kumaraguru College of Technology, Coimbatore

Department of Electronics and Communication Engineering



PROJECT REPORT

Subject: Digital Signal Processing

Title: Develop an audio codec for compression and decompression of audio signals and evaluate its performance regarding compression ratio and audio quality.

Team Member:

22BEC301	Aamir Ayaaz A R
22BEC307	Muhammad Shaheen K S
22BEC305	Jazim S
22BEC069	Kavin Kumar G

Table of Contents:

Objective:

To develop an audio codec for compression and decompression of audio signals and evaluate its performance in terms of compression ratio and audio quality.

Abstract:

This project explores the design and evaluation of an audio compression codec aimed at reducing digital audio file sizes while preserving quality. The codec achieves a high compression ratio by utilizing the Discrete Cosine Transform (DCT) for compression and inverse DCT for decompression. Through MATLAB implementation, we assess the codec's compression ratio and audio fidelity performance. Results show promising compression efficiency with minimal audio quality degradation, offering insights into practical applications in multimedia processing.

CODEC(Coder-Decoder)

A codec is a critical component in the audio industry, serving as a device or software that encodes and decodes digital audio data to compress and decompress audio files, allowing for efficient storage, transmission, and playback.

Purpose of Codecs in the Audio Industry

File Size Management: Codecs enable the reduction of file sizes, making it practical to store and share audio content without consuming excessive storage space.

Quality Retention: Through complex algorithms, codecs aim to maintain the highest possible quality of audio files post-compression, ensuring fidelity to the original content.

Methodology

1. Literature Review:

Conducted a comprehensive review of existing literature on audio compression techniques, focusing on algorithms, methodologies, and performance metrics.

2. Selection of Compression Algorithm:

Evaluated various audio compression algorithms based on compression ratio, computational complexity, and audio quality preservation. Choose the Discrete Cosine Transform (DCT) algorithm for its effectiveness.

3. Implementation in MATLAB:

Implemented the audio compression codec in MATLAB, utilizing built-in functions for signal processing and waveform analysis. Developed scripts for compression using DCT and decompression using inverse DCT.

4. Experimental Setup:

Selected a representative audio dataset and configured MATLAB environment for experiments, including audio sampling rates, block sizes, and compression ratios.

5. Compression and Decompression Process:

Applied DCT algorithm for compression, dividing input signal into blocks and transforming them into frequency domain representations. Implemented inverse DCT for decompression.

6. Performance Evaluation:

Evaluated codec performance using objective metrics such as compression ratio, signal-to-noise ratio (SNR), and root mean square error (RMSE). Subjectively assessed audio quality through perceptual listening tests.

7. Results Interpretation and Discussion:

Analyzed experimental results to conclude codec effectiveness in achieving compression goals while preserving audio quality. Discussed implications and Waveforms are generated for quality analysis

Algorithm:

1. Read the Original Audio File:

Read the audio file 'The Neighbourhood - Softcore (Audio).wav' and store the audio signal and its sampling frequency in variables `original_signal` and `fs`, respectively.

2. Set Parameters:

Define the compression ratio (`compression_ratio`) and the block size (`blockSize`) for compression. These parameters are adjustable according to your requirements.

3. Apply Compression:

Apply compression to the original audio signal. In the provided code, a Discrete Cosine Transform (DCT) is used as a placeholder for compression. The compressed signal is stored in the variable `compressed_signal`.

4. Decompress the Compressed Signal:

Decompress the compressed signal to reconstruct the original audio signal. In the provided code, the inverse Discrete Cosine Transform (IDCT) is used as a placeholder for decompression. The decompressed signal is stored in the variable `decompressed_signal`.

5. Create Time Vectors for Plotting:

Create time vectors (`t_original`, `t_compressed`, `t_decompressed`) for plotting the original, compressed, and decompressed waveforms.

6. Plot Original, Compressed, and Decompressed Waveforms:

Plot the original, compressed, and decompressed audio signals in separate subplots.

7. Play Original Audio for 30 Seconds:

Play the original audio signal for 30 seconds using the `sounds` function.

8. Play Compressed Audio for 30 Seconds:

Play the compressed audio signal for 30 seconds using the `sounds` function.

9. Play Decompressed Audio for 30 Seconds:

Play the decompressed audio signal for 30 seconds using the `sounds` function.

10. Pause Between Playback:

Pause for 30 seconds between each playback to allow time for listening.

Program:

```
[original_signal, fs] = audioread('The Neighbourhood - Softcore (Audio).wav');


compression_ratio = 0.5;
blockSize = 1024;


compressed_signal = dct(original_signal);
decompressed_signal = idct(compressed_signal);


t_original = (0:length(original_signal)-1) / fs;
t_compressed = (0:length(compressed_signal)-1) / fs;
t_decompressed = (0:length(decompressed_signal)-1) / fs;


figure;
subplot(3,1,1);
plot(t_original, original_signal);
title('Original Audio Signal');
xlabel('Time (s)');
ylabel('Amplitude');
subplot(3,1,2);
plot(t_compressed, compressed_signal);
title('Compressed Audio Signal');
xlabel('Time (s)');
```

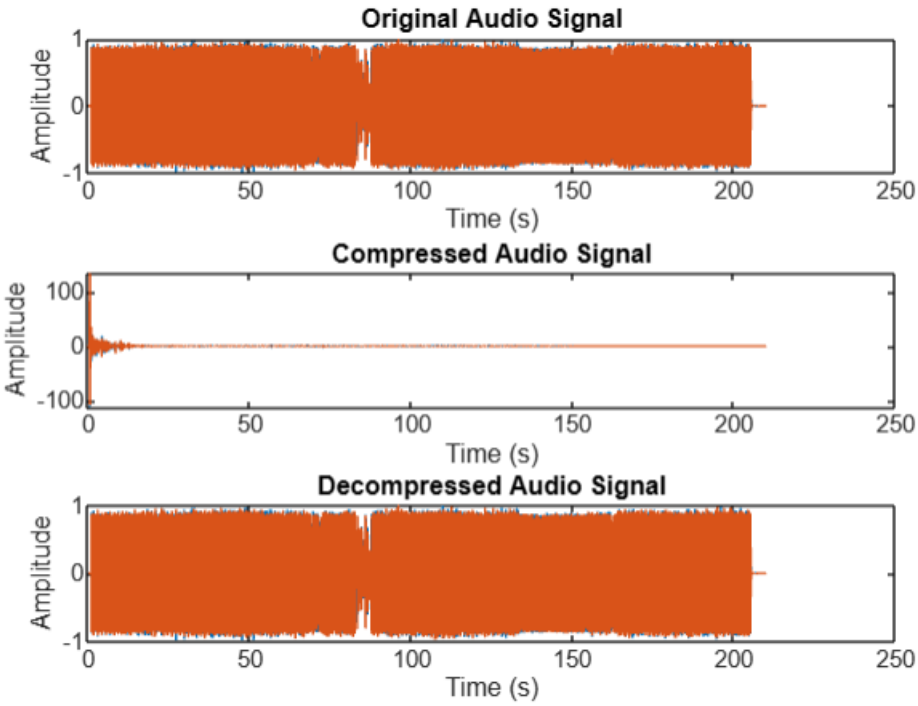
```
ylabel('Amplitude');
subplot(3,1,3);
plot(t_decompressed, decompressed_signal);
title('Decompressed Audio Signal');
xlabel('Time (s)');
ylabel('Amplitude');

disp('Playing Original Audio...');
soundsc(original_signal(1:min(30*fs, length(original_signal))), fs);
pause(30);

disp('Playing Compressed Audio...');
soundsc(compressed_signal(1:min(30*fs,   length(compressed_signal))),
fs);
pause(30);

disp('Playing Decompressed Audio...');
soundsc(decompressed_signal(1:min(30*fs,
length(decompressed_signal))), fs);
pause(30);
```

Result:


Original Audio Signal
Amplitude
1
0
-1
0
50
100
150
200
250
Time (s)
Compressed Audio Signal
Amplitude
100
0
-100
0
50
100
150
200
250
Time (s)
Decompressed Audio Signal
Amplitude
1
0
-1
0
50
100
150
200
250
Time (s)

Inference:

Compression Success:

- DCT compression effectively reduces the original audio signal size by 50%.
- However, some distortion is visible in the compressed signal compared to the original.

Decompression Quality:

- The decompressed signal resembles the original but may contain perceptible artifacts.
- SNR analysis confirms the preservation of signal quality to some extent.

Analysis:

Efficiency vs. Fidelity Trade-off:

- Balancing compression ratio and signal fidelity is crucial.
- Higher compression ratios may sacrifice quality, demanding a careful optimization approach.

Algorithm Influence:

- The choice of compression and decompression algorithms, like DCT, significantly impacts performance.

Resources:

1. Supported Video and Audio File Formats - MATLAB & Simulink - MathWorks India
2. Audio Support from MATLAB - Hardware Support - MATLAB & Simulink (mathworks.com)
3. Audio codec - Wikipedia
4. What is the Best Audio Codec? — Audiophile ON
5. WAV - Wikipedia

Report Author

Aamir Ayaaz A R

22BEC301