Report

The application reads an audio file, encodes it using Rice encoding with a specified value of M (which determines the number of bits used for encoding), and then decodes the encoded data back to the original audio format. The scode also allows for playing the original and decoded audio files for comparison.

**Analysis of the Application:**

1. Functionality: The application effectively demonstrates the process of encoding and decoding audio data using Rice encoding. It showcases the steps involved in compressing and decompressing audio files while maintaining the fidelity of the original sound.

2. Encoding Process: The rice\_encoder function implements the encoding logic by converting the audio sample values into coded words based on the specified value of M. The encoded data is then written to a binary file.

3. Decoding Process: The rice\_decoder function decodes the encoded data back to the original sample values using the reverse process. It correctly interprets the coded words and reconstructs the audio samples.

4. Audio Playback: The application integrates the ability to play both the original and decoded audio files using the sounddevice library, allowing users to audibly experience the compression and decompression of the audio data.

5. File I/O: The script efficiently handles reading and writing audio files in the WAV format, as well as storing the encoded data in a separate binary file for compression purposes.

6. Parameter Control: The script provides flexibility in choosing the value of M, which in turn determines the compression level and the number of bits used for encoding. This parameter can be adjusted to explore different compression ratios.

In conclusion, the application effectively showcases the implementation of Rice encoding for audio compression and decompression. It provides a hands-on demonstration of how encoding algorithms can be utilized to achieve efficient compression while preserving the quality of the original audio data.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Original size | Rice (K=4 bits) | Rice (K=2 bits) | % Compression (k=4 bits) | % Compression (k=2 bits) |
| Sound1.wav | 979 KB | 25617 KB | 91765 KB | 2616.64964 | 9373.34014 |
| Sound2.wav | 985 KB | 153944 KB | 605192 KB | 15628.8325 | 61440.8122 |

**Analysis of the results:**

1. Compression Rates:

The compression rates for the two sound files are significantly different when using K=4 bits compared to K=2 bits. In both cases, the % compression is much higher when using K=2 bits compared to K=4 bits. This difference can be attributed to the different levels of compression achieved by using different values of K. With K=2 bits, the algorithm can encode the audio data more efficiently, resulting in higher compression rates.

2. Impact of K Value:

The choice of the K value in Rice encoding has a significant impact on the compression rate. A smaller value of K allows for more efficient encoding of the data, leading to higher compression rates. In this case, the compression rates are much higher when using K=2 bits compared to K=4 bits, indicating that a lower value of K provides better compression for these sound files.

3. File Characteristics:

The differences in the compression rates between the two sound files may also be influenced by the characteristics of the audio data in each file. Sound files with varying levels of complexity, dynamics, and frequency content may result in different compression rates even when using the same encoding algorithm with different K values.

In conclusion, the analysis of the results shows that the compression rates of the files are so different due to the choice of the K value in the Rice encoding algorithm and the characteristics of the audio data in each file. Using a smaller K value leads to higher compression rates, but the specific compression achieved can vary based on the nature of the audio data being encoded.

**Further Development**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Original size | RLE | % Compression |
| Sound1.wav | 979 KB | 1824 KB | 186.312564 |
| Sound2.wav | 985 KB | 1961 KB | 199.086294 |

The further development implemented introduces a Run-Length Encoding (RLE) algorithm as an alternative approach to improving the compression rates for audio data. This development aims to enhance the efficiency of data compression compared to both the original Rice encoding method and the initial RLE implementation.

**Description of the Further Development:**

1. Improved RLE Algorithm:

- The enhanced RLE algorithm utilized in the development improves upon the initial RLE implementation by optimizing the encoding process to achieve better compression rates.

- The algorithm now incorporates additional strategies to more effectively encode repetitive sequences while maintaining lossless data compression.

2. Enhanced Compression Ratios:

- By refining the RLE algorithm, the development seeks to achieve higher compression rates for audio data, especially for files containing significant amounts of repeated values or patterns.

- The improved RLE implementation is designed to better capture and encode repetitive sequences in a more compact format, leading to reduced file sizes and improved compression percentages.

3. File Processing and Integration:

- The developed functions efficiently handle reading and writing WAV files, allowing for seamless encoding and decoding of audio data using the enhanced RLE algorithm.

- The integration of the optimized RLE algorithm into the file processing workflow enables easy application of the enhanced compression technique to audio files.

4. Comparison with Previous Methods:

- A comparison of the compression rates obtained using the improved RLE algorithm with the initial and Rice encoding methods is essential to evaluate the effectiveness of the enhancement.

- The table provided showcases how the enhanced RLE algorithm performs in terms of compression rates as compared to the original Rice encoding technique for the given audio files.

5. Adaptation and Future Directions:

- The development of an improved RLE algorithm opens up possibilities for further adaptations and refinements to enhance compression efficiency.

- Future directions may include exploring adaptive encoding strategies, incorporating entropy coding techniques, or integrating other lossless data compression methods to continue improving compression rates for audio data.

In conclusion, the further development implemented in the enhanced RLE algorithm demonstrates a continuous effort to optimize data compression techniques for audio files. By focusing on refining the encoding process and maximizing compression efficiency, the development aims to provide improved compression rates and file size reductions compared to the original encoding methods.The performance comparison provided enables a quantitative assessment of the enhanced RLE algorithm's effectiveness in achieving higher compression ratios for audio data.