Dept. of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur

IMAGE & VIDEO PROCESSING LABORATORY (EC69211)

Mini Project: JPEG Compression

Submitted By:
Rahul Singh – 20EC39026
Jaya Kisnani – 20EC39015
[Group - 8]

Introduction

Background

In the digital age, the efficient storage and transmission of visual information have become crucial. Images, constituting a significant portion of this data, often require large amounts of storage space and bandwidth for transmission. This necessity has propelled the development of effective image compression techniques, which aim to reduce the size of image files while preserving as much of the original quality as possible. Image compression finds its application in various fields, from web browsing and digital photography to medical imaging and satellite imagery.

JPEG (Joint Photographic Experts Group) compression is one of the most widely used methods in this domain. It strikes a balance between compression efficiency and image quality, making it an ideal choice for many practical applications. The JPEG standard utilizes a combination of lossy and lossless compression techniques to achieve high compression ratios, thus reducing file size significantly without a proportional loss in image quality.

Objective

The primary goal of JPEG compression is to minimize the storage space and bandwidth required for images. The objective of this report is to delve into the JPEG compression process, focusing on a Python implementation that illustrates the algorithm's core principles. We aim to explore how JPEG compression reduces file size, the trade-offs it makes, and the implications of these trade-offs on image quality. By examining the specific implementation details, this report seeks to provide a comprehensive understanding of the JPEG algorithm and its practical applications.

Scope

This report will specifically focus on the Python code provided for JPEG compression. We will dissect the stages involved in the JPEG process, including color space conversion, chroma subsampling, Discrete Cosine Transform (DCT), quantization, and Huffman coding. Each of these stages plays a critical role in the compression process and contributes to the overall effectiveness of the JPEG standard.

We will analyze the provided code to understand how it implements these stages, and evaluate the effectiveness of the compression it achieves. The report will not only highlight the technical aspects of JPEG compression but also discuss its practical implications, limitations, and potential areas for improvement.

Algorithm

JPEG Overview

JPEG (Joint Photographic Experts Group) compression is a widely used method for reducing the size of images with minimal loss of quality. The JPEG algorithm involves several key stages:

1. Color Space Conversion: Transforming the image from the RGB color model to YCbCr.
2. Chroma Subsampling: Reducing the resolution of chrominance information (color) while retaining the full resolution of luminance (brightness).
3. Discrete Cosine Transform (DCT): Converting the image data from the spatial domain to the frequency domain.
4. Quantization: Reducing the precision of the high-frequency components, which are less perceptible to the human eye.
5. Entropy Coding: Applying lossless compression techniques like Run-Length Encoding (RLE) and Huffman coding.

Code Analysis

1. Color Space Conversion

The rgb2ycc function in the code is responsible for converting the RGB color space of an image to the YCbCr color space. This conversion is crucial because the human eye is more sensitive to changes in luminance (brightness) than in chrominance (color). The YCbCr model separates the luminance (Y) from the chrominance (Cb and Cr), allowing more aggressive compression of the color information.

2. Chroma Subsampling

In the chromasub function, chroma subsampling is implemented, specifically the 4:2:0 scheme. This method reduces the resolution of the chrominance channels by averaging the color information in 2x2 pixel blocks. This downsampling reflects the reduced sensitivity of the human eye to color details, contributing significantly to data reduction.

3. DCT and Quantization

The DCT function performs the Discrete Cosine Transform. This transformation converts the image data from the spatial domain into the frequency domain. The DCT helps to separate the image into parts of differing importance with respect to the image's visual quality.

The lumtrans and chrtrans functions apply quantization to the DCT coefficients. Quantization reduces the number of distinct values for these coefficients, particularly for higher frequencies, which are less critical to the image's overall appearance. The arrays LUM and CHR represent the quantization matrices for the luminance and chrominance channels, respectively.

4. Run-Length Encoding and Huffman Coding

After quantization, the serialize function rearranges the DCT coefficients in a zigzag order, ensuring that low-frequency components are processed first. The run_length function then applies Run-Length Encoding to these coefficients, which is particularly effective in encoding sequences of zeros that are common in quantized blocks.

The huff function implements Huffman coding, a form of entropy coding used for lossless data compression. It assigns shorter codes to more frequent elements and longer codes to less frequent elements, based on the frequency distribution obtained from the image data.

5. Integration and Flow

The provided code integrates these stages to compress an image. It begins with color space conversion, followed by chroma subsampling for the color components. The DCT is then applied to 8x8 blocks of the image, and the resulting coefficients are quantized. The serialized and quantized data undergo run-length encoding, followed by Huffman coding to achieve further compression.

The algorithm processes the luminance (Y) and chrominance (Cb, Cr) components separately, reflecting their different roles in human visual perception. The Y component retains more detail due to its higher importance, while the Cb and Cr components are more heavily compressed.

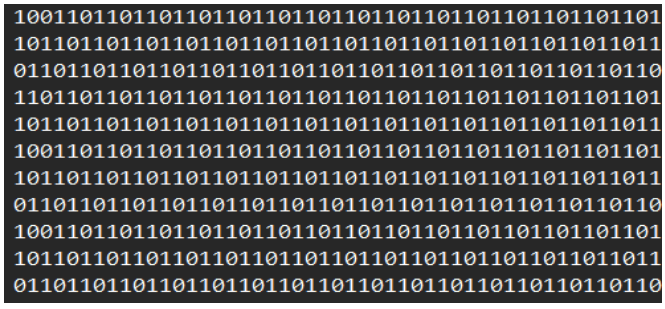
In conclusion, this section of the code encapsulates the core of JPEG image compression, balancing the need for reduced file size with the preservation of image quality. The efficiency of JPEG lies in its use of the characteristics of human vision, exploiting the fact that certain kinds of detail are less important for the perceived image quality.

Output Results

Input Image

Output .bin file Screenshot

Compression Ratio is 16.0



Compression Ratio is 14.6

$$\text{Compression Ratio}=\frac{Compressed\ Bits}{Original\ Bits}$$

- Original Bits: This represents the total number of bits required to represent the original image before compression. It is calculated as the sum of the bits required for the luminance (Y) component (len(y)×len(y[0])×8), chrominance (Cr) component (len(cr)×len(cr[0])×8), and chrominance (Cb) component (len(cb)×len(cb[0])×8).

- Compressed Bits: This represents the total number of bits required to represent the compressed image after performing various compression operations. It is calculated as the sum of the lengths of the compressed representations of the luminance (Y), chrominance (Cr), and chrominance (Cb) components.

Discussion and Conclusion

Data Compression Techniques in Image Processing

In the realm of image processing, leveraging the inherent redundancies within images has proven to be a powerful strategy for efficient data compression. One prominent approach involves manipulating images to exploit redundancies, primarily within the chrominance streams. The process begins with Discrete Cosine Transform (DCT), a mathematical transformation applied to emphasize low-frequency content while downplaying high-frequency elements. High-frequency content, often associated with image noise introduced during capture, is deemed less critical to the human eye. Subsequent quantization further refines the compression, reducing less perceptually sensitive data. The serialization step employs a zigzag pattern, strategically increasing the number of zeros at the array's end, enhancing redundancy. To address trailing zeros, run-length encoding is implemented, taking advantage of the absence of certain frequencies in the serialized array. Finally, Huffman coding efficiently removes repetitive data redundancies, culminating in the successful compression of substantial amounts of image data.

Optimizing Compression: A Multifaceted Approach

The optimization of image data compression involves a multifaceted strategy, combining various techniques to achieve maximum efficiency. At its core, the process revolves around identifying and capitalizing on redundancies within images. Leveraging the chrominance streams and employing Discrete Cosine Transform (DCT) initiates the prioritization of essential visual information. The subsequent quantization step refines the data by assigning greater emphasis to low-frequency content, acknowledging the perceptual significance to the human eye. Serialization, conducted in a zigzag pattern, strategically introduces zeros, augmenting redundancy for subsequent stages. Run-length encoding targets trailing zeros, common after serialization due to the absence of specific frequencies. The final touch, Huffman coding, excels at eliminating repetitive data redundancies, ultimately resulting in the successful compression of vast amounts of image data. This comprehensive approach underscores the nuanced and intricate methods employed in the pursuit of efficient image compression.

Conclusion

In conclusion, the provided Python implementation of JPEG compression serves as a robust foundation for understanding and experimenting with image compression techniques. It effectively demonstrates the core principles of JPEG compression, achieving a balance between reducing file size and maintaining image quality. However, there are opportunities for improvement, particularly in adapting the algorithm to specific image characteristics and

optimizing for performance. Future work could explore adaptive quantization, more sophisticated entropy coding techniques, and optimization for speed and memory usage, making the algorithm more versatile and suitable for a wider range of applications.

References

1. https://ieeexplore.ieee.org/document/125072
2. https://en.wikipedia.org/wiki/Color_space
3. https://en.wikipedia.org/wiki/Chroma_subsampling
4. https://en.wikipedia.org/wiki/Discrete_cosine_transform
5. https://en.wikipedia.org/wiki/Run-length_encoding
6. https://narainsreehith.medium.com/huffman-coding-on-image-d6092bed5821