***A PROJECT REPORT***

***ON***

***SIMPLE JPEG Codec IMPLEMENTATION***

**Project Team**:  Sharath Chandra Nandikula

Lalitha Priya Janampet

Bhavani Siva Jyothi Pinniboina

**Abstract**

This project targets the development of a JPEG codec with a simpler encoding process as a means of presenting the fundamentals of image compression. The encoding process consists of converting RGB images to the YCbCr color context, which means the two color hues component and the brightness component are precisely separated, thus, resulting in inefficient compression. Using an 8 x 8 rectangular block-based approach, the DCT is employed to each block to shift the spatial domain to the frequency domain, thereby focusing on energy compactness. After that, quantization is used to infringe on frequencies that are less visible, thus, it helps to create lossy compression. In the next step, the quantized data scan information is further entropy coded through a zigzag scan and Huffman coding respectively, in order to minimize storage requirements. The decoding process uses these methods in reverse, namely, inverse quantization, inverse data type change, and the color space is rebuilt in order to return to the original image, with some quality loss due to compression.

The project uses MATLAB for computer algorithm exploration and Python for functional implementation, making use of libraries to make light of computational procedures. The main problems come in keeping your stability in DCT operations and being able to control the artifacts from quantization. Performance is evaluated through the measures of Peak Signal-to-Noise Ratio (PSNR) and visual inspection of the recovered pictures. The expected outcomes of this project are a workable codec that shows the balance between image quality and compression efficiency, and hence this image can be used to learn the benefits of the JPEG standard almost in the actual world. This is a type of project that is the theoretical part of coding as the practical part becomes the starting point for the exploration of the intricate subject of image possesses.

**1. Introduction**

Image compression is a fundamental aspect of modern digital imaging, reducing the storage and transmission requirements while maintaining acceptable quality. The JPEG standard is one of the most widely used compression formats, employing lossy techniques to achieve significant size reduction. This project explores the development of a simplified JPEG codec that demonstrates key compression steps such as color space conversion, Discrete Cosine Transform (DCT), quantization, and Huffman coding. The project provides a hands-on approach to understanding the trade-offs between image quality and compression efficiency, bridging theoretical and practical aspects of image processing.

**1.1 About Project**

This project is about designing a JPEG codec that is easier to use for both encoding and decoding images. The compression process that JPEG uses is the first step followed by the Huffman coding technique. Therefore, the project clearly describes how JPEG does the compression process. The two codes give flexibility to utilize them for code development and testing. Furthermore, deeper insight into algorithmic efficiency is reached.

**1.2 Existing System**

Modern image compression systems involve state-of-the-art and efficient JPEG algorithms that give high compression ratios with visual quality remaining unaltered. These systems are strong and can accommodate a distinctive resolution of images and other types. However, black-box models may be handy and trendy, albeit that they are not as useful as they should be for the conceptual fundamentals of compression.

**1.3 Proposed System**

The proposed system presents a basic implementation of a JPEG codec, which simplifies the process of encoding and decoding for a better understanding. The system makes use of an 8x8 block-based DCT, a quantization matrix for compression, and Huffman encoding for this purpose. The decoder reverses these steps to reconstruct the image, showcasing the effects of lossy compression.

**1.4 Problem Definition**

Even though JPEG is widely used, many implementations are so convoluted and mysterious that they pose a barrier for students and researchers. This project seeks to develop a simple and transparent JPEG implementation that users can easily use to learn about basic components, drawbacks, and trade-offs in compression.

**1.5 Objectives**

• Simply put, this is a JPEG encoder and decoder which is DCT-based and it works using quantization, and Huffman coding.

• To judge the codec performance with image quality metrics like PSNR.

• To deliver learning experiences of the relationship between compression ratio and image quality.

• To contribute to the development of practical knowledge in image compression that helps in debugging and coding the algorithms.

• To be the initial stepping stone for the exploration of advanced image processing techniques.

**2. Literature Review**

**2.1 JPEG Standards and Their Evolution**

The JPEG (Joint Photographic Experts Group) standard was approved in 1992 and it has dramatically changed the way of digital image storage and transmission. Developed for the purpose of compressing efficiently photographic images, JPEG uses lossy techniques which responsibly reduce the file sizes. Over the years, modifications such as JPEG 2000 have come up with new wavelet-based compressions that promise better quality at higher compression ratios. Regardless of upgrades, the one you should have is the original one as its convenience, its flexibility, and its equipment compatibility all over the world are the key features. This standard is basic, and apart from consumer photography, it has been used in web graphics.

**2.2 Importance of JPEG Processes**

1. Color Space Conversion

JPEG images are generally saved in the YCbCr color space instead of RGB. This two-channel separation of chrominance from the luminance of the object serves to enhance the image of the area containing color while thickness is compressed due to a similar type of techno.

2. Discrete Cosine Transform (DCT)

DCT spatial image data is transformed into frequency domain, where the energy thus generated is stored in a few of low-frequency. Performing such extreme transformations helps in decreasing mutual pixel correlations, but still containing the most developed.

3. Quantization

Quantization reduces the accuracy of the DCT coefficients, which have high frequency components. This means that the less important high-frequency data is simply thrown away, which results in a remarkable decrease of the size.

4. Zigzag Scanning

As a result of this technique, the two-dimensional array of the quantized coefficients is transformed into a one-dimensional sequence, the coefficients being ordered in such a way that the low-frequency components are put first. Step two is to encode the data entropy.

5. Huffman Coding

Along with other entropy coders, Huffman coding uses short-length codes for frequently occurring patterns, thus further decreasing the file size. This part guarantees the storage and transmission of the compressed data in a very efficient way.

**2.3 Existing Tools and Frameworks**

The widely-used image compression tools and frameworks have been implemented in the optimized JPEG version. The popular libraries such as libjpeg and Pillow in Python respectively provide, image; compression; and de-compression functions in a very robust way. These instruments abstract the greatest part of the complexity which makes the JPEG operation as seamless as possible without you even glancing at the internal mechanisms. The most famous new formats (JPEG 2000 and WebP) which offer better compression, i.e., a low compressed file, are already around, but we will pay the price of having higher computational complexity of the compression.

**2.4 Relevance to This Project**

The simplified JPEG codec implemented in this project seeks to clear the shrouded procedure of compression. These manual methods as DCT, quantization, and Huffman coding of the main elements have been mastered and the project is of course, a very detailed example of how JPEG is able helped quite a lot understandable to deal with the thinning of the JPEG towards the balance of compression as well as quality retention. This exercise is the foundation for this point in the deepen the study of the existing standards and methodologies in image compression.

**3. Methodology**

This part identifies the step-by-step process used to code a JPEG codec with reduced complexity, including both encoding and decoding. MATLAB builds the prototype and then Python is utilized to move to the implementation stage, as outlined below.

**3.1 General Workflow**

The JPEG compression and decompression process is a choreographed workflow.

1. Encoding Phase: The input image is changed into a compressed format by means of color space transformation, DCT, quantization, zigzag scanning, and Huffman coding.

2. Decoding Phase: The compressed data makesit possible to approximate the original image by conducting an inverse operation of the encoding process.

A flowchart outlining the workflow is attached, which shows the different stages the procedure goes through.

**3.2 Encoding Process**

1. Color Space Conversion

* Images are transformed from RGB to YCbCr format, where the luminance (Y) is isolated from the chrominance (Cb, Cr).
* The foreground is that the human vision system is more sensitive to luminance than colours. This ensures the priorities involved in the compression process.

2. Block-Wise Discrete Cosine Transform (DCT)

* The image is sub-divided into 8x8 blocks.
* DCT (the inverse technique) transforms these blocks from the spatial domain to the frequency domain, concentrating image energy in fewer coefficients.
* Low frequency items are considered more essential image features, while the higher frequencies hold the details.

3. Quantization

* Quantization Matrix is applied to DCT coefficients, which results in lower precision for high-frequency components.
* This lossy stage dramatically size down data by sacrificing non-necessary detail.

4. Zigzag Scanning

* Quantized coefficients are reordered into a 1D sequence by executing a zigzag sampling.
* Low-frequency coefficients are preferred in the synthesized order, which results in the better entropy coding.

5. Huffman Coding

* The 1D sequence is coded via Huffman coding, thus shorter codes are given to more frequent values.
* In this manner, compression is efficient because there is less redundancy.

**3.3 Decoding Process**

1. Huffman Decoding

* The encoded data is decompressed to restore the zigzag-ordered coefficients. It is quantized and zigzag ordered coefficients are decompressed to the original data.

2. Inverse Zigzag Scanning

* The 1D sequence is transformed back into 2D blocks by cutting the zigzag pattern.

3. Dequantization

* A set of Quantized coefficients are obtained by the division of the quantization matrix with the DCT coefficients, thus only the ABCd of details will be restored.

4. Inverse DCT (IDCT)

* The depth of the pixel each block is acquired using the inverse of the DCT (IDCT).

5. Color Space Reconstruction

* The compressed image is being converted back into RGB, which was the original format of the image before the compression.

**3.4 Implementation**

1. Programming Language

* MATLAB was employed in the prototype stage thanks to its ability to deal with matrices and plotting tools. The final implementation is based on Python for its flexibility and fast access to image processing libraries such as NumPy and Pillow.

2. Structure and Algorithms

* The MATLAB code is actually modular in the sense that the functions such as DCT, quantization, and zigzag scanning are separated into different files. Aside from that, Python aligns with the modular structure of MATLAB, and also comes with added real-time performance optimization.
* Both the PSNR metric in MATLAB or Python is used to estimate the quality of digital images.

**MATLAB Implementation Details**

1. Image Reading and Preprocessing

* A monochrome image is used for the reason of busyness, that is, compression of the lightness information.

2. Block-Wise DCT

* A tool called a DCT is applied to 8x8 blocks of the image, which then transform the data into a frequency domain representation.

3. Quantization and Zigzag Scanning

* First, a quantization matrix is applied followed by taking zigzag scanning which arranges the coefficients into one compact order.

4. Reconstruction

* Using the inverse of zigzag and dequantization, the transformed DCT coefficients are obtained. The IDCT step does the reconstruction of the spatial image that is the nearby equivalent.

5. PSNR Calculation

* The degree of image superiority or PSNR serves as the comparison measure between the original and the reconstructed images.

**4. Implementation Details**

**4.1 Data Preparation**

The process is commenced by incorporating the input image, which is either in RGB or grayscale. In the absence of color, it is transferred to the YCbCr color space in order for luminescence, which is the image brightness, and chrominance, the color intensity, to be presented separately. Grayscale images directly go through the process. The image is divided into sections of 8x8 in the process of duplication according to the size that is supplemented if proper dimensions are not provided for further block-basis compression.

**4.2 Core Algorithms**

1. DCT Implementation:

* Each 8x8 block is subjected to the Discrete Cosine Transform, changing the pixel values into frequency components. The energy now mainly is in the top-left corner of the block, which is the lower frequency region of the block.

2. Quantization Matrix Design:

* The quantization matrix that is already set is used for the DCT output. As the higher frequencies are not that important for the human visual system and can be used as a real quantity but much smaller, the verified best-case quantization is used.

3. Zigzag Scan:

* The quantized coefficients are steapped in zigzag patterns in such a manner that low-frequency coefficients are the first to be filled, hence, the significant data is recorded for the entropy encoder.

4. Huffman Encoding and Decoding:

* The entropy encoder uses Huffman coding to assign shorter codes to the most frequently occurring values of the zig-zagged coefficients after this operation they can be called zig-zagged. In the process of decoding, these codes are broken down to quantized coefficients.

**4.3 Testing Framework**

The implementation is tested by encoding multiple pictures, reconstructing them through decoding, and matching the results with the original pictures. Examples like Peak Signal-to-Noise Ratio (PSNR) which are used to determine the quality of the photos brought back, but the level of compression ratios to the one that evaluates the degree of size reduction.

**4.4 Challenges**

One of the numbers of issues was losing copious precision during DCT and IDCT processes, which caused minor artifacts in reconstructed images. Huffman coding required uniquely implementing algorithms to skillfully handle the encoding of variable-length messages and was therefore quite efficient. On the other hand, optimizing for larger images and computing overheads during each block-wise operation turned out to be difficult problems.

**5. Results and Discussion**

**Mat lab Results:**

**A tree with the sun shining through it

Description automatically generated**

**A tree with a bright light shining through it

Description automatically generated**

**PSNR: 36.314 dB**

**Python results:**

A collage of a lake with trees and mountains

Description automatically generated

**5.1 Visual Results**

The comparison of the original and reconstructed images shows the codec's ability to compress and keep the important visuals. In areas with fast-changing content, there are minor artifacts, however, the basic structure and color accuracy are mostly preserved. The samples show that the codec can make ratios good enough without damage to the quality, and this way it can be used in such applications where a reasonable tradeoff is required. Despite that, this fact only draws the public's attention to the parts of the image where the compressing may introduce noticeable distortions in the areas with fine textures or complex color gradients.

**5.2 Quantitative Analysis**

Metrics like PSNR and compression ratio allow a clear assessment of the codec's performance in numbers. For example, a PSNR of 36.314 dB indicates that the reconstructed images are relatively high and consist of only the minimum possible noise and distortions compared to the original. The compression ratio is an indicator of the amount of data saved without being visibly compromised in terms of quality. But on the other hand, it may take place along with some of the losses in detail and slight blurring, which refers to the trade-off between compression ability and visual quality. The check-up shows even though the codec is a good performer at typical photographic images, it may loose detail in images with a high color palette or those having a varying color distribution.

**5.3 Limitations**

The very fact that compression is lossy means that fewer details in the image will be lost, especially in high-frequency regions such as edges and fine textures. The distortions that appear in the reconstructed images are the proof of this, too. Specifically, these distortions are the most obvious at points where two areas contrast each other the most. The codec may experience performance problems in usage or devices with a heavy amount of compression or extremely high-quality requirements, thus, during medical imaging and professional photography. Even though the compression ratio and image quality are contradictory, artifacts like ringing and blocking are more localized in areas with large contrast.

**Comparison of MATLAB and Python Implementations**

|  |  |  |
| --- | --- | --- |
| **Feature** | **MATLAB** | **Python** |
| Ease of Use | Intuitive high-level functions | Extensive libraries, requires setup |
| Performance | Optimized for matrix computations | Dependent on library efficiency |
| Flexibility | The best for prototyping and analysis | Suitable for system integration |
| PSNR Value | 36.314 dB | Similar, based on testing |

**6. Conclusion and Future Work**

**6.1 Conclusion**

This project was meant to help explain the basic concepts of image compression, for example, DCT (Discrete Cosine Transform), Quantization, Zigzag Scanning, and Huffman Coding. Along with the use of DCT to differentiate images, it also, through quantization, using a zigzag technique, scanning, and selecting the shortest coding, aided in the identification of particular redundancies being cut down and all this being the coding process that specifically does compression while the decoding process is the reconstruction. The most important were expressed in the results, shedding light on the compression results due to the very high degree of file size reduction while the image is still of lower but bearable quality which is referred to by metrics such as PSNR. It is impossible not to mention the implementation of the brick-and-mortar of compression coding that the theory provided, through which useful lessons were learned about balancing between intrusion and visual integrity.

**6.2 Future Work**

Future work can focus on trying to make the codec more sophisticated way such as progressive JPEG, which will civilize the lack of atrocious experience of images loading incrementally. Driving the way from Huffman, clustering often unsolvable problems and manufacturing adaptive quantization matrices for effective dynamic compression-based image content shall later guarantee the successful weightage of road accidents. Moreover, facilitating the roll-out of the method to support chroma subsampling or associating other entropy coding methods (such as arithmetic coding) might reduce compression rations. Making the programming functionality multi-threaded or introducing GPU with its speed capabilities can eliminate the code bottlenecks and switch to working with bigger and larger images.

**7. References**

* MATLAB Code: Refer to the uploaded PDF for detailed implementation.
* Python Notebook: Detailed implementation is shown in the uploaded notebook.
* Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing. Pearson.