

# Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

## **IMAGE & VIDEO PROCESSING LABORATORY (EC69211)**



## Mini Project: JPEG Compression

Submitted By:

Shreya Sainath Desai - 21EC39022

Shreeya Singhal - 21EC3903

## Introduction:

- Image compression is the process of reducing the file size of an image without highly compromising its quality. This is done by removing redundant or unnecessary data from the image. The two primary types of image compression are lossy and lossless.
- Lossy compression discards some of the image data to reduce size which leads to a loss in quality but results in high compression ratios.
- Lossless compression preserves all the original image data and ensures that the image can be restored to its exact original form but the compression ratios are lower.
- JPEG (Joint Photographic Experts Group) compression is one of the most widely used methods in this domain. It balances compression efficiency and image quality which makes it an ideal choice for many practical applications.
- The JPEG standard uses a combination of lossy and lossless compression techniques to get high compression ratios as well as preserve enough data for restoring a reasonably good image.
- It exploits the limitations of human visual perception which is more sensitive to brightness than colors and hence this technique compresses color data more aggressively than brightness information.

## Objective:

- The goal of this experiment is to understand the core principles of the JPEG compression algorithm and to implement it in Python.
- We will dissect each stage involved in the JPEG process which includes color space conversion, chroma subsampling, Discrete Cosine Transform (DCT), quantization, and Huffman coding and understand the role these stages play in the compression process as well as look into the effectiveness of the overall process.

## Algorithm:

- **Color Space Conversion:**

We first convert the image from RGB to YCbCr. This is done because we want to more aggressively compress the color information as compared to the brightness information as the human eye is more sensitive to brightness than it is to color. Hence, by converting RGB to YCbCr, we separate the luminance (Y) from the chrominance (Cb and Cr) channels and hence we can compress the chrominance channel more aggressively. A function is implemented in the code which performs this conversion using the formula given below.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- **Chroma subsampling:**

A function is implemented for chroma subsampling in which the 4:2:0 scheme is used for the subsampling. This method reduces the resolution of the chrominance channels by averaging the color information in 2x2 pixel blocks. This down sampling reflects the reduced sensitivity of the human eye to color details, contributing significantly to data reduction.

- **DCT:**

A function is implemented that 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.

This is done so that the high-frequency elements can be compressed more aggressively as compared to the low-frequency components as our eyes are more sensitive to low-frequency components in the images.

- **Quantization:**

The block after DCT, goes through Quantization where the block is divided by the quantization block. The Quantization block for chrominance has higher values as compared to luminance for more aggressive compression of the chrominance channel and higher loss also occurs for chrominance. Similarly, for the reason of more aggressive compression of high-frequency elements, the quantization values for higher components of frequency are much higher as compared to the lower ones.

- **Run-Length Encoding:**



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

- **Huffman Coding:**

A function takes the run-length encoded data and implements Huffman coding on it which is a type 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 input data.


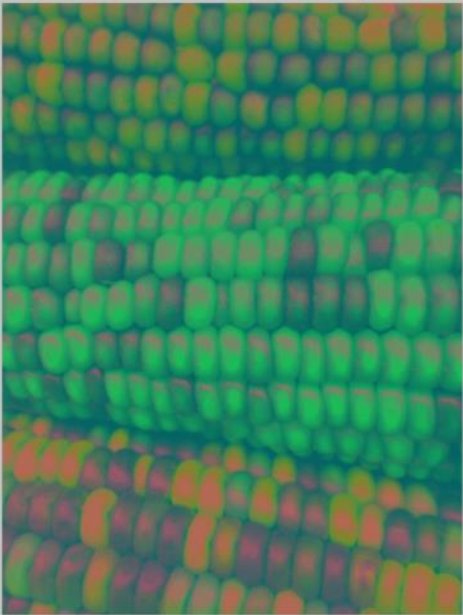
## Results:

Enter the name of the input file: cameraman.jpg

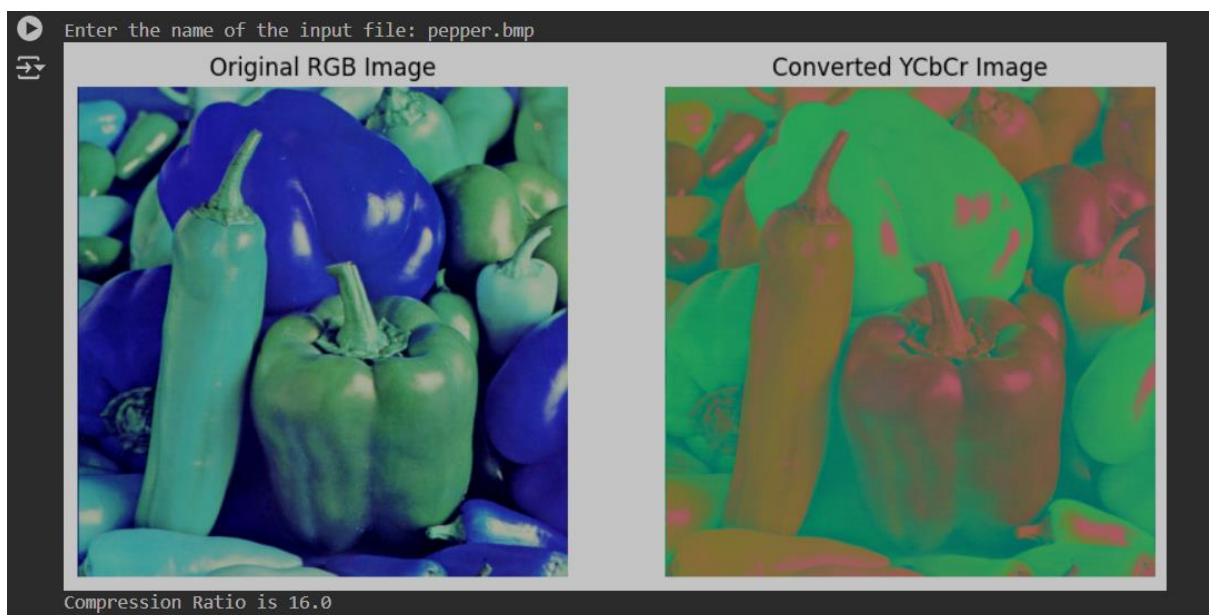
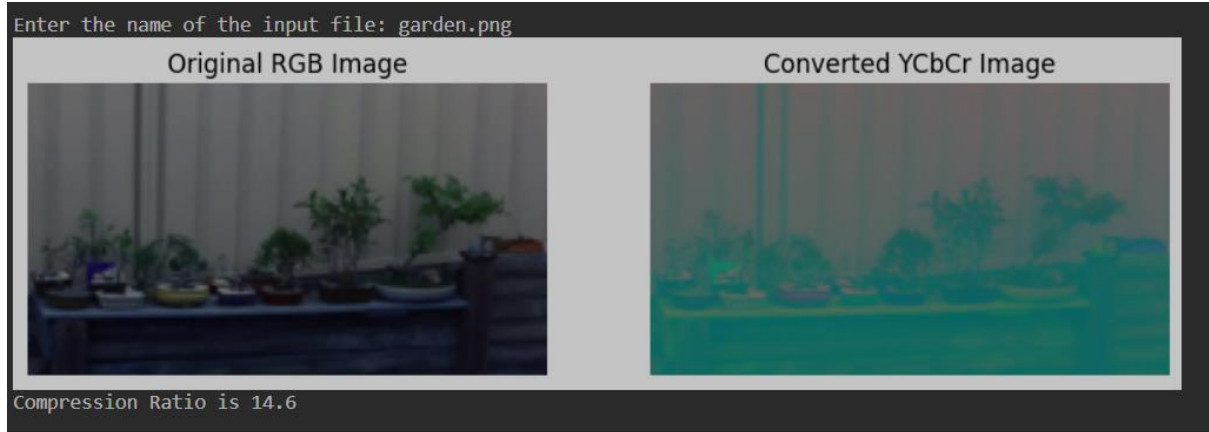
Original RGB Image	Converted YCbCr Image
	

Compression Ratio is 16.0

Enter the name of the input file: corn.bmp

Original RGB Image	Converted YCbCr Image
	

Compression Ratio is 16.0



- Compression Ratio: Original Bits / Compressed Bits
- Original Bits: This represents the total number of bits required to represent the original image before compression  

$$\text{Original Bits} = (\text{len}(y) \times \text{len}(y[0]) \times 8) + (\text{len}(cr) \times \text{len}(cr[0]) \times 8) + \text{component}(\text{len}(cb) \times \text{len}(cb[0]) \times 8)$$

luminance (Y), chrominance (Cr) , chrominance (Cb)

- Compressed Bits: This represents the total number of bits required to represent the compressed image after performing various compression operations.

$$\text{Compressed Bits} = (\text{len}(y\_compressed) \times \text{len}(y\_compressed[0]) \times 8) + (\text{len}(cr\_compressed) \times \text{len}(cr\_compressed[0]) \times 8) + \text{component} (\text{len}(cb\_compressed) \times \text{len}(cb\_compressed[0]) \times 8)$$

## **Discussion and Conclusion:**

- We implemented JPEG compression algorithm and performed 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.
- This technique leverages the fact that the human eye is more sensitive to brightness than color and low-frequency components than high-frequency components. Hence, the later ones are more aggressively compressed.
- Converting RGB to YCbCr and applying 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 and luminance channel, acknowledging the perceptual significance to the human eye.
- Serialization is conducted in a zigzag pattern followed by run-length encoding that targets trailing zeros, common after serialization due to the absence of specific frequencies.
- In the end, Huffman coding is implemented which eliminates repetitive data redundancies, ultimately resulting in the successful compression of vast amounts of image data.
- 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.