INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

DIGITAL IMAGE PROCESSING LABORATORY

Mini Project-JPEG Compression

ELECTRONICS AND ELECTRICAL COMMUNICATIONS ENGINEERING

VISION AND INTELLIGENT SYSTEMS

Name	Roll Number
Harsh Sukheja	19EC39013
Sreyan Biswas	19EC39036

❖ Aim of the experiment:

- Design the entire JPEG pipeline.
- Code should be in Python/ Cpp.
- Code should work for Colored Images for all sizes i.e. variable height and width.

❖ Introduction:

Background of Image Compression

- The objective of an image compression technique is to represent an image with smaller number of bits without introducing appreciable degradation of visual quality of decompressed image. These two goals are mutually conflict in nature. In a digital true color image, each color component that is R, G, B components, each contains 8 bits data. Also color image usually contains a lot of data redundancy and requires a large amount of storage space.
- In order to lower the transmission and storage cost, image compression is desired. Most color images are recorded in RGB model, which is the most well known color model. However, RGB model is not suited for image processing purpose.
- For compression, a luminance-chrominance representation is considered superior to the RGB representation. Therefore, RGB images are transformed to one of the luminance-chrominance models, performing the compression process, and then transform back to RGB model because displays are most often provided output image with direct RGB model.
- The luminance component represents the intensity of the image and look likes a gray scale version. The chrominance components represent the color information in the image.

Why use JPEG Compression?

- JPEG stands for **Joint Photographic Experts Group**. JPEG is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.
- We perform such type of compression to reduce the size of the file without damaging its quality to a huge extent.
- By reducing the size we can store it in a huge amount which was not possible earlier. Reducing the size of images will also improve the **efficiency** of the system as it will give less load on it.
- The compression ratio of lossless methods (e.g., Huffman, Arithmetic, LZW) is not high enough for image and video compression. JPEG uses transform coding, because:
  A large majority of useful image contents change relatively slowly across images, i.e., it is unusual for intensity values to alter up and down several times in a small area, for example, within an 8 x 8 image block

How does the compression work ?

- JPEG uses a lossy form of compression based on the discrete cosine transform (**DCT**). This mathematical operation converts each frame/field of the video source from the **spatial (2D) domain** into the **frequency domain** (a.k.a. transform domain).
- A perceptual model based loosely on the human psychovisual system discards **high-frequency** information, i.e. **sharp transitions** in **intensity**, and **color hue**. In the transform domain, the process of reducing information is called **quantization**.
- **Quantization** is a method for optimally reducing a large number scale (with different occurrences of each number) into a smaller one, and the transform-domain is a convenient representation of the image because the high-frequency coefficients, which contribute less to the overall picture than other coefficients, are characteristically small-values with high compressibility. The quantized coefficients are then sequenced and losslessly packed into the output bitstream.
- The compression method is usually **lossy**, meaning that some original image information is lost and cannot be restored, possibly affecting image quality.
- There is an optional **lossless** mode defined in the JPEG standard. However, this mode is not widely supported in products.
- The **degree of compression** can be adjusted, allowing a selectable **tradeoff** between **storage size** and **image quality**. JPEG typically achieves 10:1 compression with little perceptible loss in image quality.

DCPM Encoding

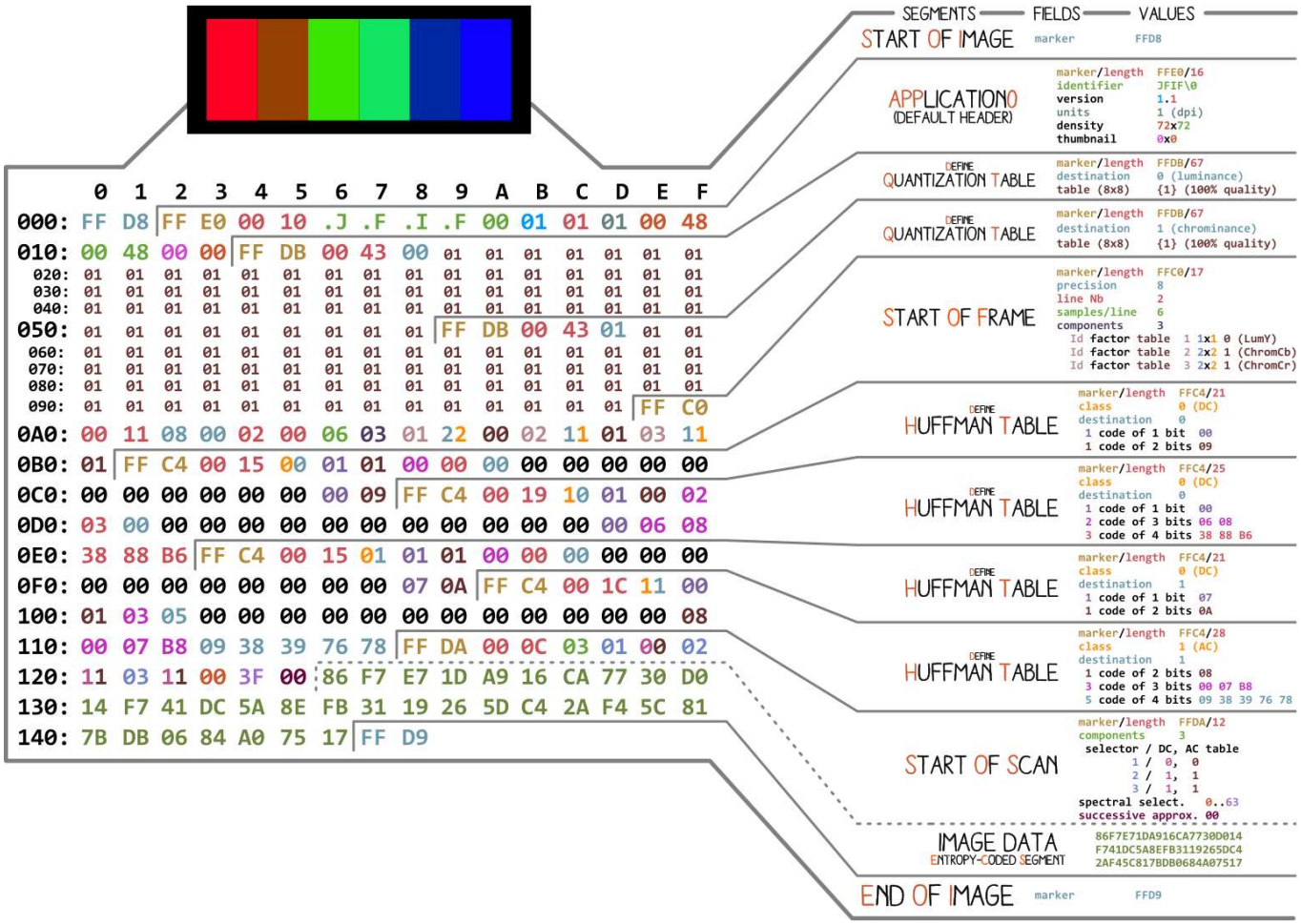
- Differential pulse-code modulation (DPCM) is a signal encoder that uses the baseline of pulse-code modulation (PCM) but adds some functionalities based on the prediction of the samples of the signal. The input can be an analog signal or a digital signal.
- If the input is a continuous-time analog signal, it needs to be sampled first so that a discrete-time signal is the input to the DPCM encoder.
  - Option 1: take the values of two consecutive samples; if they are analog samples, quantize them; calculate the difference between the first one and the next; the output is the difference.
  - Option 2: instead of taking a difference relative to a previous input sample, take the difference relative to the output of a local model of the decoder process; in this option, the difference can be quantized, which allows a good way to incorporate a controlled loss in the encoding.
- Applying one of these two processes, short-term redundancy (positive correlation of nearby values) of the signal is eliminated; compression ratios on the order of 2 to 4 can be achieved if differences are subsequently entropy coded because the entropy of the difference signal is much smaller than that of the original discrete signal treated as independent samples.

DCT

- Run-length encoding (RLE) is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.
- This is most efficient on data that contains many such runs, for example, simple graphic images such as icons, line drawings, Conway's Game of Life, and animations. For files that do not have many runs, RLE could increase the file size.

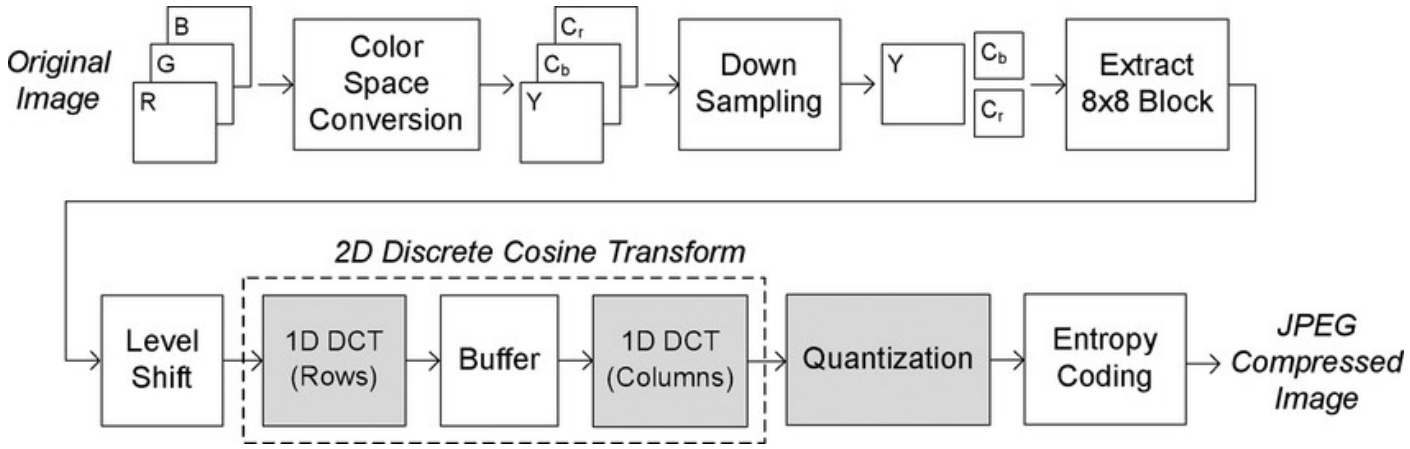
JPEG Header

- JPEG files (compressed images) start with an image marker which always contains the marker code hex values **FF D8 FF**. It does not have a length of the file embedded, thus we need to find JPEG trailer, which is **FF D9**.



Overview of the process:

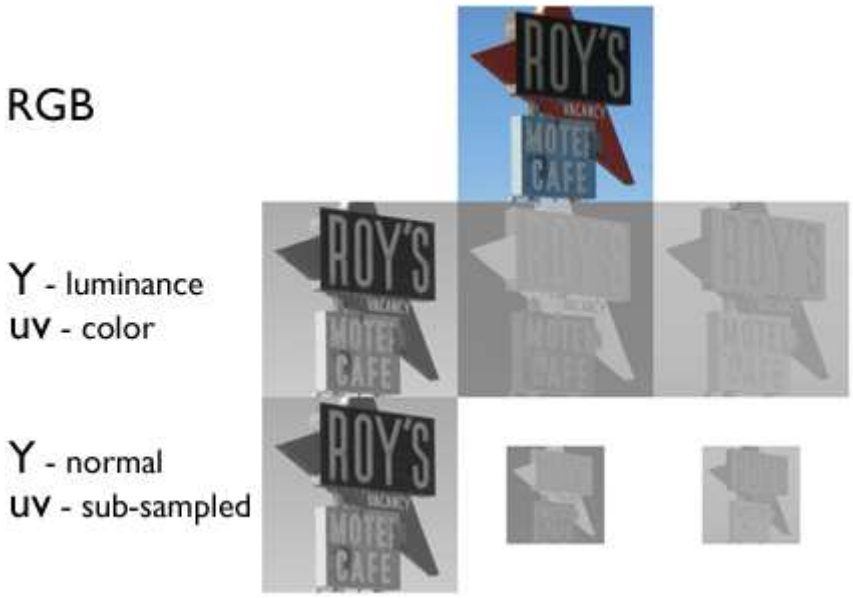
- Firstly, we convert the R, G, B color format to Y, Cb, Cr format. Some colors are more sensitive to human eyes and thus are high-frequency colors. Some colors of chromium compounds like Cb and Cr are less sensitive to human eyes thus can be ignored.
- Then we reduce the size of pixels in **downsampling**. We divide our image into 8*8 pixels and perform forward DCT(Direct Cosine Transformation).
- Then we perform **quantization** using quantum tables and we compress our data using various encoding methods like run-length encoding and Huffman encoding.



❖ Algorithm:

Color Conversion and Subsampling

- Starting with the RGB data, this data is divided into the luminance and the color components. The data is converted to Y, Cb,Cr.
- This step *does not reduce the amount of data* as it just changes the representation of the same information. But as the human observer is much more sensitive to the intensity information than to the color information, the color information can be **sub-sampled** without a significant loss of visible image information. But of course the amount of data is reduced significantly.



Block-Processing and DCT

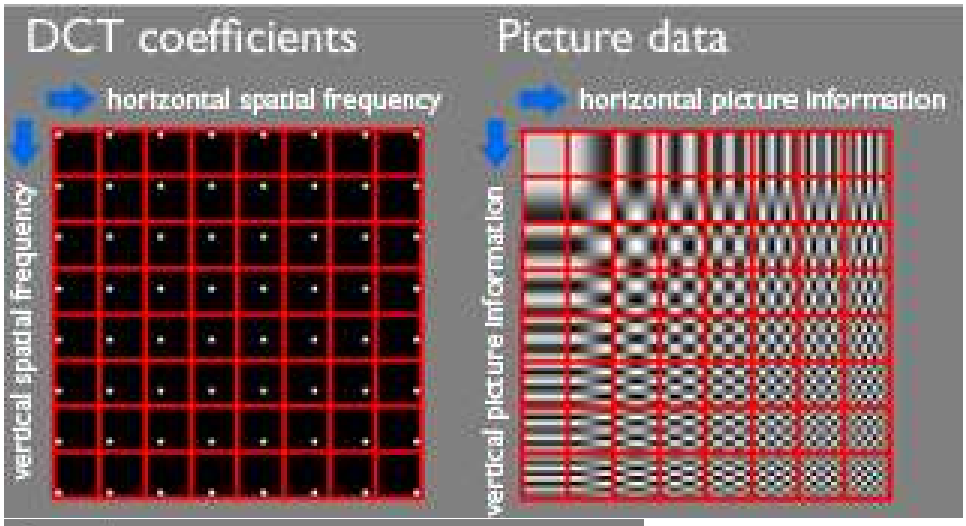
- The Yuv image data with the subsampled color components is then divided into 8x8 pixel blocks. The complete algorithm is performed from now on these pixel blocks. Each block is transformed using the discrete Cosines Transformation (DCT).
- In the spatial domain (so before we have transformed) the data is described via digital value for each pixel. So we represent the image content by a list of pixel number and pixel value.
- After the transformation, the image content is described by the coefficient of the spatial frequencies for vertical and horizontal orientation.

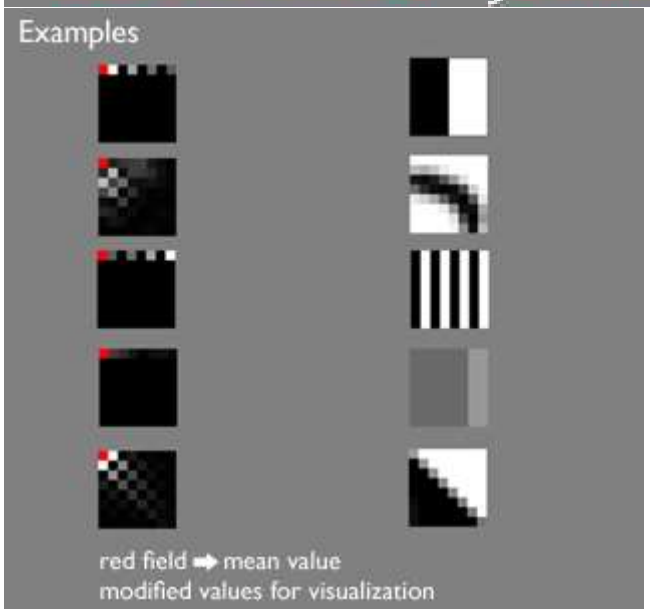
- So in the spatial domain, we need to store 64 pixel values. In the frequency domain, we have to store 64 frequency coefficients. No data reduction so far.
- The matrix after DCT conversion can only preserve values at the lowest frequency that to in certain point. Quantization is used to reduce the number of bits per sample.
  $F(w) = a$

The DCT equation (Eq. 1) computes the i,jth entry of the DCT of an image.

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases}$$

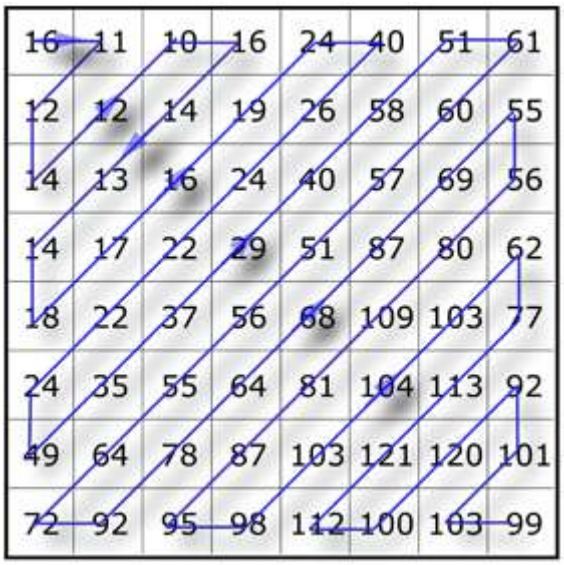




Quantization

- To reduce the needed amount of data to store the 64 coefficients, these are quantized using the quantization table.
- Depending on the size of the quantization steps, more or less information is lost in this step.
- Most of the times, the user can define the strength of the JPEG compression.
- The quantization is the step where this user information has influence on the result (remaining image quality and file size).
- There are two types of Quantization: Uniform Quantization , Non-Uniform Quantization

- The **zigzag scan** is used to map the 8x8 matrix to a 1x64 vector. Zigzag scanning is used to group low-frequency coefficients to the top level of the vector and the high coefficient to the bottom. To remove the large number of zero in the quantized matrix, the zigzag matrix is used



DPCM and Variable Length Encoding

- Basic concept of **Differential pulse-code modulation** (DPCM) - coding a difference, is based on the fact that most source signals show significant correlation between successive samples so encoding uses redundancy in sample values which implies lower bit rate.
- Realization of basic concept is based on a technique in which we have to predict current sample value based upon previous samples (or sample) and we have to encode the difference between actual value of sample and predicted value (the difference between samples can be interpreted as prediction error). Because it's necessary to predict sample value DPCM is form of predictive coding.
- Implementing DPCM in the current JPEG Encoder: Encode the difference between the current and previous 8x8 block. The smaller the number the fewer bits is required. The DC component value in each 8x8 block is large and varies across blocks, but is often close to that in the previous block

- **Run-length encoding (RLE)** is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.
- Depending on the quantization, more and more coefficients are reduced to zero. And the probability is very high, that these coefficients with value 0 are found in the higher frequencies rather than in the lower frequencies.
- So the data is reordered that way, that the values are sorted by the spatial frequency, first the low frequencies, high frequencies come last. After reordering, it is very likely, that we have some values at the beginning and then a lot of 0. Instead of storing “0 0 0 0 0 0 0 0 0 0”, we can easily store “10x 0”. That way, the amount of data is also reduced significantly.
- For **encoding**, we take our **quantized output** and then serialize into our file in a **zig-zag fashion** and we use **Huffman encoding** to shrink them down.

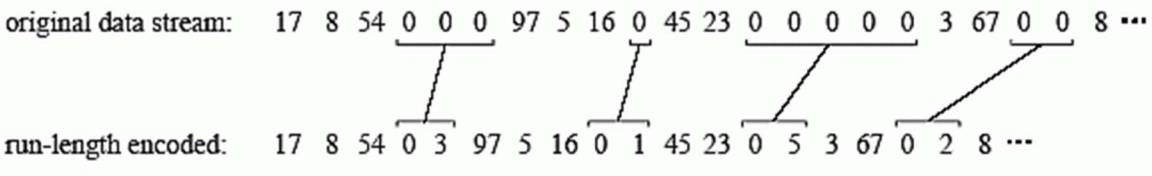


FIGURE 27-1
Example of run-length encoding. Each run of zeros is replaced by two characters in the compressed file: a zero to indicate that compression is occurring, followed by the number of zeros in the run.

Decompression

- In the second stage, we decompress our data, It involves decoding where we decode our data, and we again de-quantize our data by referring to the quantization table.
- We decode the Huffman tables, the Huffman encoding, we unquantized by multiplying by all our values in the quantization table.
- Then we apply the inverse DCT, to obtain our block back. And we do this for every little 8 by 8 image in our picture. If our image isn't a multiple of 8, then we have to add some padding bytes at the end to make it work.
- Then we perform Upsampling to convert it into original pixels and finally, color transformation takes place to convert the image into its original color format.

❖ Results:

1. Input Image**:**

2. Output Image after encoding and then decoding:

❖ Discussion:

- The input image is passed through the JPEG Encoder and the output is produced in Y, Cb, Cr format as:
  i. **Y.txt** : Brightness (luma) component intensities in Binary code format
  ii. **Cb.txt** : Blue minus luma (B-Y) component intensities in Binary code format
  iii. **Cr.txt** : Red minus luma (R-Y) component intensities in Binary code format

- The Decoder uses the above generated text files to reproduce back the image file and produce the output as the file **out_image.jpg**
- The JPEG compression is a block based compression. The data reduction is done by the subsampling of the color information, the quantization of the DCT-coefficients and the Huffman-Coding (reorder and coding).
- The user can control the amount of image quality loss due to the data reduction by setting (or chose presets).
- For a high quality compression, the subsampling can be skipped and the quantization matrix can be selected that way, that the information loss is low. For high compression settings, the subsampling is turned on and the quantization matrix is selected to force most coefficients to 0. In that case, the image get clearly visible artifacts after decompression.