Antonio Aodong Chen Gu, Xuan Lin

Dr. Xiaoli Ma

ECE 4271

May 2nd, 2024

Adaptive JPEG Compression

I. Introduction

JPEG is a commonly used lossy image compression algorithm used to store images. However, in classic JPEG algorithm, a constant quality factor is chosen for all regions of the image which could be inefficient. Thus, a novel adaptive JPEG compression algorithm was proposed that could automatically choose quality factors for different parts of an image based on K-means clustering and delentropy. The algorithm shows a higher compression rate in complex images and comparable performance in simple image cases, while keeping the image visually the same.

II. Background Information

Traditional JPEG compression composes of three steps. For the first step, it will divide the image into multiple 8x8 blocks. If the length or width of the image is not multiple of 8, it will zero pad them to be multiple of 8. For each 8x8 2D-DCT will be applied to them. There are two major reasons people are using DCT here. The first reason is that DCT always produces real value while DFT would result in complex value which is difficult to store and calculate. The second reason is that DCT compacts more energy to its lower end which makes it ideal for data compression.

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \quad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

Figure 1. Formula of 2D-DCT where p,q represents the DCT value index, M and N represents the length and width of the block, and A_{mn} represents the input value at index m and n. [2]

For the second step, a quantization table is divided elementwise from the DCT value matrix. Since DCT tends to pact energy to the lower end and human eyes are insensitive to

high frequency details, such quantization table would aim to suppress these high-frequency components to zero. Therefore, when the image is recovered through inverse DCT, the suppressed information will not be noticeable.

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure 2. Standard JPEG Quantization Table.

The quantization table shown in Figure 2 is the most used and people denote it as quality factor of 50. Increasing the quality factor would result in lower values in quantization table and resulting in less compression rate, and vice versa. In classic JEPG, the quality factor is constant across all 8x8 blocks of the image. To update the quantization table based on the quality factor, one can use the following formula:

$$f(qf) = exp\left(6 \cdot \left(\frac{50 - qf}{50}\right) \cdot ln(2)\right)$$

$$Q'[l,k] = round(f(qf) \cdot Q[l,k])$$

The qf represents the quality factor and the Q represents the quantization table.

Even though most data compression happened at the second step, people also use different encoding methods to further compress the data. There are three major methods that JPEG uses: zigzag encoding, run length encoding and Huffman table. The zigzag flattened the 2D matrix into a 1D array and puts the low frequency part at front. Then, with run length encoding, the huge number of zeros created by the quantization can be compressed into a pair of numbers with the first number denoting that it's 0 and the second number denoting how many occurrences of it.

III. Proposed Structure

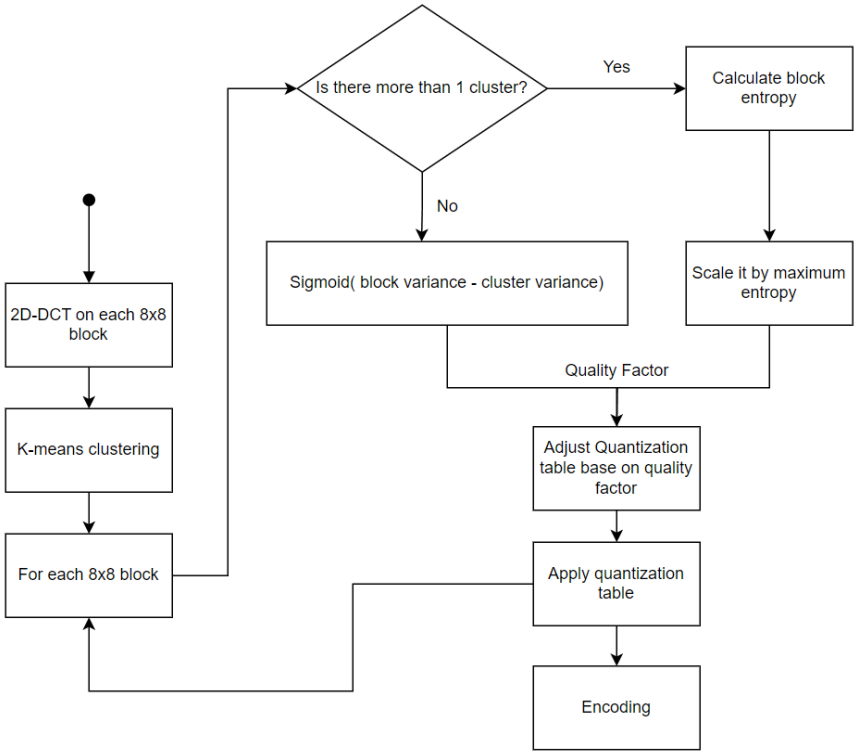

Figure 3. Block Diagram of Adaptive JPEG Compression.

From the classic JPEG compression algorithm, it's obvious that choosing good quality factor is really important for efficient image compression. Using constant quality factor across the whole image is not optimal since most of the images will have parts that contain more important information than other parts to humans.

With the proposed adaptive JPEG compression, different quality factors are assigned to different blocks depending on the amount of information the block contains and the variance in color this block contains compared to a larger block. The algorithm first performs a normal 2D-DCT as classic JPEG compression did. Then, it will perform a k-means clustering to segment image into k parts. To assign different quality factors to each 8x8 block, if all pixels in a block are from one cluster, then the quality factor was determined using sigmoid function with input value of the difference between the block variance and the cluster variance. If a block contains more than 1 group, it will divide the block delentropy by the theoretical maximum value. Noted that both methods output a value from 0 to 1, therefore, to get a quality factor matches the definition, a scaling mechanism is needed. In the current algorithm, the upper limit of the quality factor is set to be 60 and the lower limit is 25 to ensure the minimum quality of the image. A simple linear

scaling is then performed to get the actual quality factor. With quality factor determined, a different quantization table will be applied to each block to achieve optimal compression result.

IV. Mythology

i. K-means Clustering

The initial version of the project compares the block variance with the total variance of the image. However, since the total variance of the image contains too much information, it's not a meaningful comparison.

Figure 4. Original Image vs. Compressed Image using total image variance as criteria.

From this example, it clearly shows that the performance using the total color variance of the whole image is bad. It eliminates the light gray strips on man's sleeve while it's very noticeable to human. The reason of it is because the whole image contains too much information, and its variance would naturally be very big. Thus, the light gray lines would just be unimportant in the comparison of block versus whole image.

One possible solution to it is to segment the image to multiple parts. With each part containing less information regarding the image, the comparison between block variance and the segment variance will be more meaningful. In the proposed algorithm, k-means clustering was used to soft segment the image into k parts hoping that each part would be a stand-out region of the image to human eye. The k-means clustering algorithm first

assigns randomly k data points to be centroids and groups the rest of the data point to these k points base on the distance between these two data points. After that, the centroid would be updated to the center of the group and the algorithm keeps updating till it converges or reaches iteration limits which is currently set to be 100.

Figure 5. Segmented Image using k-means clustering algorithm setting k = 8 and different gray level representing different group.

To perform the k-means on image, the input data were set to be a vector of [color, 0.22*x, 0.22*y] to keep the information of position of pixel. As shown in figure 5, the segmentation is pretty well, with each group having relative the same color and continuous. Within each group, meaningful comparison could be made.

ii. Delentropy

However, at the edge of the groups, it is hard to choose which group variance to use to compare with the block variance. Therefore, delentropy, which is the information entropy of the gradient of the pixel values, was used to determine the quality factors of the blocks at the edge of the groups. The benefit of using delentropy instead of information entropy of pixel values is that delentropy could include spatial correlation of the image into the calculation which is important in image processing. [3]

Firstly, the image along x coordinates and y coordinates are filtered by the first difference filter $g_x = [1, -1]$ to approximate the gradient along x and y coordinates as following,

$$f_x(m,n) = g_x * f = \nabla f(m,n) = f(m,n) - f(m-1,n)$$

$$f_y(m,n) = g_x * f = \nabla f(m,n) = f(m,n) - f(m,n-1)$$

where $f(m,n)$ represents the pixel values at location [m, n].

Since the pixel value ranges from 0 to 255, the image's gradient will range from –255 to 255. Therefore, the histogram of the gradient in the range of [-255, 255] was used to calculate the 2D probability distribution as following:

$$p_{i,j} = \frac{1}{4MN} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \delta_{i,f_x(m,n)}\, \delta_{j,f_y(m,n)}$$

where $i,j \in [-255,255]$ represents the histogram index and $\delta_{x,y}$ represents the Kronecker delta which equals 1 only if x = y and 0 otherwise.

Then, the delentropy is calculated by the formula:

$$H(\nabla f) = -\sum_{i=-255}^{255} \sum_{j=-255}^{255} p_{i,j} \log_2 p_{i,j}$$

After that, the quality factor of edge block is calculated as the linear scaled version of the ratio of the delentropy of the block to the maximum possible block entropy H_{max} such that

$$qf = \frac{H(\nabla f)}{H_{max}} * (upper - lower) + lower$$

where upper is the upper limit of the quality factor and lower is the lower limit of the quality facotr.

The H_{max} is achieved when the distribution is a uniform distribution and for 8 by 8 blocks such that

$$H_{max} = -\frac{1}{64} \sum_{i=1}^{64} \log_2 1/64 = 6$$

V. Results

The images used for testing are baseline JPEG testing images shown in Figure 6. There are three complex images and two quite simple images. The criterias used to measure the performance of each method are the number of non-zeros values and Frobenius norm difference calculated by the formula below:

$$\text{diff} = \log_{10}\left(||I - Iq||_F^2 \right)$$

where I is the original image and Iq is the reconstructed image.

Figure 6. Original Testing Images. [4]

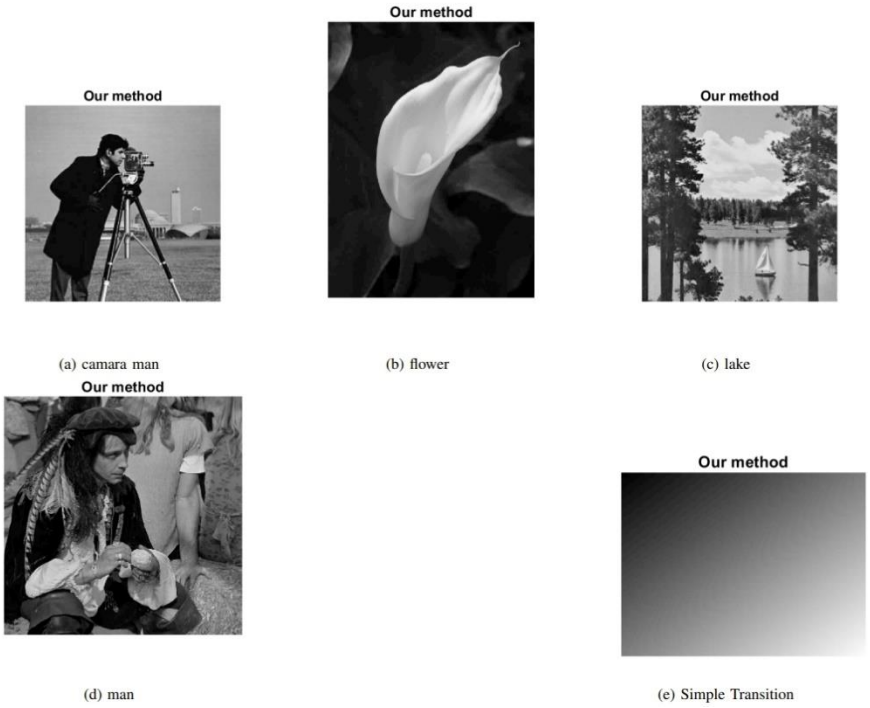

Figure 7. Reconstructed Images using Adaptive JPEG.

Figure 8. Reconstructed Images using Default JPEG

Table 1. The Performances Of Default JPEG And Adaptive JPEG Algorithm

Image Name	Non-Zeros		diff	
	Default	Adaptive	Default	Adaptive
flower	21101	22192	6.3422	6.6178
Camara man	26080	17829	6.343	6.7092
lake	41649	26753	6.9147	7.1637
man	149435	107592	7.4247	7.6323
Transition	7499	7803	5.5863	6.0096

Figures 7 and 8 show the reconstructed images using adaptive JPEG and default JPEG. Visually, the adaptive JPEG and default JPEG have similar quality. In addition, in Figure 10, the two methods also have very close difference values, meaning that the qualities of two methods after reconstruction are similar overall. However, based on Figure 9, for complex images like camara man, lake, and man, the adaptive JPEG has much smaller number of non-zero values, meaning that the adaptive JPEG will have a better compression rate. In addition, for simple images like transition and flower, the adaptive JPEG is slightly worse than the default JEPG.

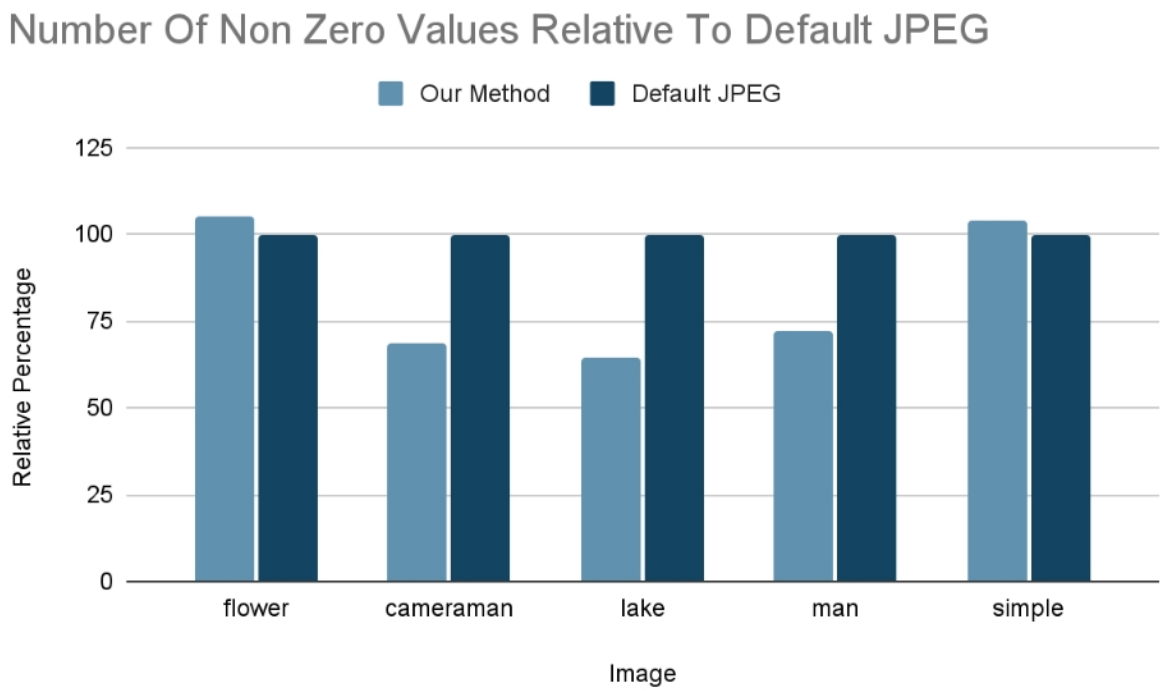

Figure 9. The histogram of number of non zeros using Default and Adaptive JPEG.

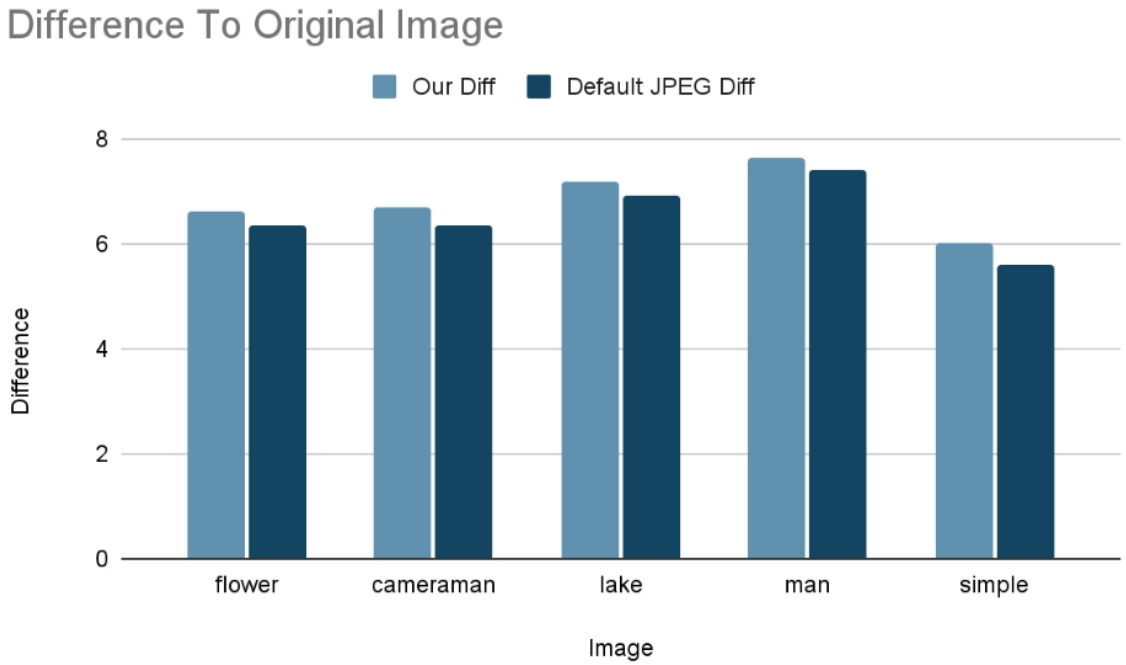

Figure 10. The histogram of differences using Default and Adaptive JPEG.

VI. Conclusion

The adaptive JPEG algorithm updates the quantization table using the quality factor for each 8 by 8 block based on k-means clustering and delentropy to achieve the goal that increasing the compression rate while maintaining the subjective and objective quality. Based on the result, the algorithm successfully achieves this goal for complex images. However, for simple images, the adaptive JPEG algorithm is not working very well because the algorithm tends to have a smaller quality factor due to the low variance of the images.

In addition, because of the k-means clustering, the processing time varies depending on the complexity of the image. Some possible improvements in the future could be more advanced image segmentation algorithm based on edge and object detection. Instead of variance, other criteria to determine the quality factor such as statistical distribution could be used.

Reference:

[1] X. Ma, “Lec-JPEG”

[2] Discrete cosine transform - MATLAB & simulink, https://www.mathworks.com/help/images/discrete-cosine-transform.html (accessed May 1, 2024).

[3] K. G. Larkin, “Reflections on shannon information: In search of a natural information-entropy for images,” 2016.

[4] Image databases, https://www.imageprocessingplace.com/root_files_V3/image_databases.htm (accessed May 2, 2024).