Parallelization of the JPEG Compression Algorithm

Kyle Barrows
Undergrad Student, Dept. Computer Science
University of Central Florida
Orlando, Fl, USA
barrowsk@knights.ucf.edu

Caleb Chase
Undergrad Student, Dept. Computer Science
University of Central Florida
Orlando, Fl, USA
calebchase2001@knights.ucf.edu

Mari Peele
Undergrad Student, Dept. Computer Science
University of Central Florida
Orlando, Fl, USA
mpmorrell@knights.ucf.edu

***Abstract*— JPEG image compression is widely used to reduce storage size of photographs for internet applications, digital cameras, medical imaging, surveillance systems, and wide variety of other uses. This lossy compression allows the user to adjust their output image to the ratio of image quality and file size appropriate to their use case. The compression method uses discrete cosine transformation (DCT) combined with a statistical encoder. In this project we investigated the possible benefits of parallelization on each the DCT and encoder algorithms. The experimental results were analyzed in two ways. The first by comparing runtime and correctness of parallelized vs non-parallelized implementation of the algorithm we created. The second method was to compare the runtime of our parallelized algorithm to well-known JPEG compression tools. The analysis involved statistical methods and visualization of data collected during the analysis describing execution time and image quality. (HOLD SPACE FOR RESULTS AND CONCLUSION)**

Index Terms—JPEG, concurrent, c++, lossy, DCT

I. Introduction

The JPEG format was first released as a standard in 1992 as a joint project from the Joint Photographic Experts Group committee, a joint working group of the International Standardization Organization (ISO), the International Telecommunication Union (ITU), and the International Electrotechnical Commission (IEC). Since that time, the standard has been updated eight times to include features like compliance testing, extensions, and a file interchange format.

JPEG image compression takes advantage of the human eye's increased sensitivity to luminance over chrominance. In other words, human vision is more sensitive to light than to color tone. This difference allows a method of compression that discards high-frequency information through discrete cosine transformation (DCT) and creates an image that appears almost identical but does not contain much of the original information. This is a type of lossy compression that greatly reduces the image storage size.

A compression ratio 10:1 has little effect on the image quality while still significantly reducing image size. The user can choose the level compression or image quality that fits their specific needs. This methodology works best with images like photographs that lack sharp transitions and variations in tone. Images like sketches or photographs with sharp textures are more likely to create artifacts that distort the output image.

Additional caution should be taken with repeated edits, resizing, or cropping of the original JPEG output. Each alternation of the file decompresses and recompresses the file and loses additional information through lossy compression. Some tools have been created to provide lossless JPEG editing, but they only work for specific alternations and images of a minimum quality.

In comparison, PNG is an alternative image format that uses lossless compression to decrease image size while maintaining all original image data. PNG uses a palette of colors stores within the image data and lossless compression to create images that are capable of transparency and editing without losing quality or information. They are capable of handling images with sharp transitions, such as crisp logos. This capability comes at a cost of increased file storage size compared to JPEG images.

II. Methodology

A. Algorithms

For our experiment we will first implement two different versions of the JPEG compression algorithm. There are numerous variations on the JPEG compression algorithm and the exact specification used has not been decided yet.

The first algorithm we will implement will be a standard JPEG compression algorithm that does not utilize any parallelization. The purpose of this implementation is to provide a baseline in terms of performance.

The second JPEG compression algorithm will utilize parallelization. During the compression process, the algorithm runs in 8x8 blocks on the image. Parallelization will be used to run these 8x8 block operations on separate threads using dynamic load balancing.

In addition, we plan to include a third implementation from the ImageMagick software suite. Using this third algorithm will give insight into how our algorithm compares to industry ones. ImageMagick was chosen due to its maturity and widespread use.

B. Testing and Analysis

To test these algorithms, the following process will be applied. As a note, this process is subject to change.

1. Image input to the algorithms will be prepared. Ten images at a pixel size of 2560 x 1440 will be selected in lossless format. These images will be then down sampled to a pixel size of 1280 x 720, and 640 x 480.
2. Next these images will be fed into the non-parallel, parallel and ImageMagick JPEG compression algorithms. The time to compress each image for each algorithm will be recorded. We will repeat this process n-number of times. (n has not been decided on yet)
3. To analyze our results we will provide statistical analysis that aims to answer the follow questions:
    a. What is the relationship between image size and runtime?
    b. Is there significant variation between image contents for a given size and runtime?
    c. What were the performance effects on the parallelized algorithm compared to the non-parallelized algorithm.
    d. How do our implementations of the algorithm compare to the ImageMagick algorithm?
4. The results of the statistical analysis will be provided in tables and data visualizations. In addition, written analysis will be included to accompany the visualizations.

Once the testing process has been completed, we will hypothesize explanations for the results of the analysis and provide our thoughts on the results

III. Background

A. YCbCr Color Space

YCbCr is a type of color space often used in digital media and photography. Like the RGB color space (red, green, blue), each letter in YCbCr represents its color format. Y represents the luminance component. Cb and Cr are the chroma components that can generally be described as the blue chrominance and red chrominance values. To utilize YCbCr, an image's RGB pixel values are converted. From Microsoft's specifications [8], a method is provided to convert RBG to YCbCr using matrix multiplication:

$$[Y\ Cb\ Cr] = [RGB] \begin{bmatrix} 0.299 & -0.168935 & 0.499813 \\ 0.587 & -0.331665 & -0.418531 \\ 0.114 & 0.50059 & -0.081282 \end{bmatrix}$$

Fig. N. RGB to YCbCr Matrix [8]

The reason that YCbCr is used over other color spaces is because YCbCr corresponds with the sensitivity of the human eye to light. Humans tend to be more sensitive to change in luminance than chroma in images [7]. Therefore, YCbCr allows for compression of Cb and Cr components without having a minimal impact on the visible quality of the image.

B. Chroma Subsampling

In the JPEG compression algorithm, YCbCr properties are taken advantage of in a process called chroma subsampling. For a 2D array of pixels, the subsampling of the image is controlled by a parameter J:a:b. "J" represents a filter of J x 2 pixels. "a" represents the number of chrominance samples in the first row and "b" represents the number of chrominance samples in the second row. For example, 4:4:4 would provide no subsampling whereas 4:2:2 provides subsampling by a factor of 2.

A. *Discrete Cosine Transform*

The Discrete Cosine Transform (DCT) is the main algorithm behind JPEG compression. DCT is a block compression algorithm meaning it compresses the data into blocks (typically 8x8 pixels for standard DCT) and attempts to represent each block as a sum of cosine functions with varying frequencies.

The input into the DCT algorithm is an array of integers, where represents the intensity of the pixel. Each pixel has a value ranging from 0-255. Using a set of precomputed basis functions, the DCT method will take each image block and output an array of integers containing DCT coefficients ranging from -1024 to 1023 [4]. The DCT coefficient represents the weight, or how much each one of the basis functions influences the final image.

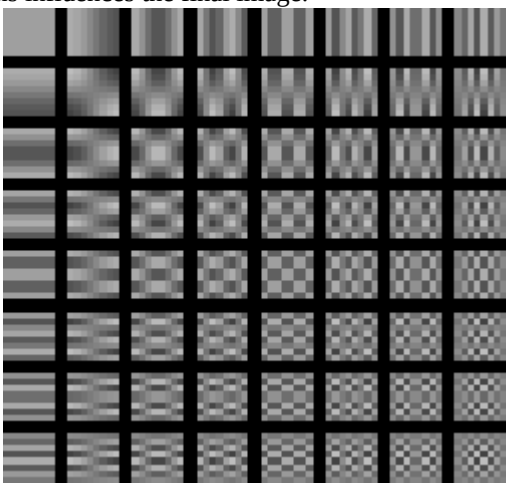

Fig. N. DCT: basis images [9]

This image shows the 64 basis functions that the original image can be summed to recreate the original image. Each cell in the grid represents a cosine wave with a different frequency. As it moves right, the horizontal frequency increases and as it moves down vertical frequency increases. As higher frequency ranges are not as perceivable to the human eye, the DCT coefficients for high frequency cells are often smaller.

Each 8x8 block of an image is given a matching 8x8 matrix, each cell represents a DCT coefficient for how much the matching basis image influences the final image. A higher coefficient means it has more influence on the final appearance. The top left cell is called the direct coefficient, or DC coefficients and represents an image with no frequency variance (i.e. no color differences). The remaining cells are alternating coefficients, or AC coefficients, and represent cells with different degrees of frequency variance. Generally, the lowest frequency cells have the highest coefficients and are thus the most important which lends itself to the next step.

B. *Quantization*

Once the DCTII values are calculated, JPEG compressions uses coefficient quantization to reduce the number of values

stored for the image. It does this by reducing some of the values to zero so they can be compressed using runtime-length encoding. By using a precomputed quantization matrix that varies based on the compressor and the quality of the compression selected, the quantized value for each position in the DCT matrix is obtained with [5]:

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

Fig. N. Quantization matrix example [10]

The larger the value in the quantization matrix, the more information will get discarded for each frequency range. This results in many values for less important, higher-frequency areas getting set to zero.

C. *Encoding*

To store the values in the table, each value gets collected in a zig-zag manner collecting data from lowest frequency to highest-frequency.

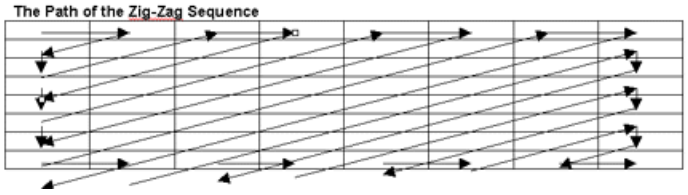

Fig. N. Order cells are collected [6]

This means the most quantized parts of the image are grouped together. This can result in long strings of zeros since many values are usually set to zero during quantization. Using run-length encoding (RLE), a lossless compression algorithm, sequences of repeated data are represented by a single integer representing how many times it appears in the sequence [6].

REFERENCES

[1] "JPEG vs. PNG: Which one should you use? | adobe." [Online]. Available: https://www.adobe.com/creativecloud/file-types/image/comparison/jpeg-vs-png.html. [Accessed: 11-Mar-2023].

[2] K. Iqbal, "JPEG - image file format," *JPEG - Image File Format*, 08-Aug-2020. [Online]. Available: https://docs.fileformat.com/image/jpeg/. [Accessed: 10-Mar-2023].

[3] "Overview of JPEG 1," *JPEG*. [Online]. Available: https://jpeg.org/jpeg/index.html. [Accessed: 10-Mar-2023].

[4] The discrete cosine transform (DCT). https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node231.html.

[5] *Lossy data compression: JPEG*. https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/coeff.htm.

[6] *Lossy data compression: JPEG*. https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/lossless.htm.

[7] S. Winkler, M. Kunt, and C. J. van den Branden Lambrecht, "Vision and video: Models and Applications," *Vision Models and Applications to Image and Video Processing*, pp. 201–229, 2001.

[8] "[MS-RDPRFX]: Color conversion (RGB to ycbcr)," *[MS-RDPRFX]: Color Conversion (RGB to YCbCr) | Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rdprfx/b550d1b5-f7d9-4a0c-9141-b3dca9d7f525. [Accessed: 10-Mar-2023].

[9] IMAGE PROCESSING TOOLBOX. *Image Processing Toolbox Documentation*. https://www.mathworks.com/help/images/.

[10] JPEG STANDARD QUANTIZATION TABLE | DOWNLOAD SCIENTIFIC DIAGRAM. https://www.researchgate.net/figure/JPEG-standard-quantization-table_fig1_331969197.