Conner Schacherer
Linear Algebra
Semester Project

Title: DCT Compression
Group Members: Conner Schacherer

One of the biggest concerns in today's digital age is how to transfer data efficiently. Images are one of the most common forms of data found online and finding ways to improve data compression and transference is a large area of research in computer science. One of the best and most common forms of image compression is the JPEG format. JPEG stands for Joint Photographic Experts Group, which chose to use the DCT algorithm in 1988 for their image compression algorithm. The DCT allows for a lot of compression for an image with minimal data loss and an efficient algorithm for calculation. That is why the JPEG is the most popular format for representing pictures digitally.

Digital images are made pixels, each one representing a color of the image. The more pixels an image has, the clearer and sharper the image. Each pixel holds a byte of data representing the amount of black in grayscale image or three bytes of data if it is a colored image, representing the red, green and blue values. Uncompressed images can get extremely large, especially for HD images. If a black and white image 1080x1080 pixels, the total number of bytes needed to store it would be 1080 * 1080, which is 1,166,400. If it was a colored image, it would be three times that size, which would be 3,499,200 bytes.

The Discrete Cosine Transformation algorithm is used to compress images in an efficient way. It transforms an image from the spatial domain to a frequency domain, by expressing data points as sums of cosine functions oscillating at different frequencies. The cosine function is used since it take fewer functions to represent a signal than other functions. There are multiple DCT formulas, but the most popular DCT is the the DCT-II, and the inverse of it is the DCT-III, which is used to decompress the image. To transform the image to the frequency domain, the DCT-II function is:

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

$$C(x) = \frac{1}{\sqrt{2}} \text{ if x is 0, else 1 if } x > 0$$

To compress an image that has NxN pixels, first split it up into 8x8 pixel blocks. You multiply the blocks by the DCT to get the 8x8 frequency coefficient matrix for that block, each coefficient corresponding to the x and y of the original block. The DCT is used on an 8x8 block since it is much faster to compute than the frequency coefficients for the entire matrix at once. The frequency coefficient matrix keeps the low valued frequencies in the upper left corner of the matrix and the highest frequency values in the bottom right corner. The human eye can see lower frequencies better and the higher frequencies are much harder to detect.

Input Pixel Matrix

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

Output DCT Matrix

186	-18	15	-9	23	-9	-14	19
21	-34	26	-9	-11	11	14	7
-10	-24	-2	6	-18	3	-20	-1
-8	-5	14	-15	-8	-3	-3	8
-3	10	8	1	-11	18	18	15
4	-2	-18	8	8	-4	1	-7
9	1	-3	4	-1	-7	-1	-2
0	-8	-2	2	1	4	-6	0

For black and white images, one DCT matrix is enough to represent the grayscale value. For a colored image, three DCT matrices are created, with each one representing the red, green, and blue values.

To compress the image, simply get rid of the higher frequency values by zeroing out a certain amount of columns, starting at the right most column and going in, and rows, starting and the bottom and going up. So for a high compression, the bottom seven rows and right seven columns would be zeroed out, leaving one number out of the 64 in the 8x8 block to represent the image for the image, to have a 94% compression rate. Or for a low compression level, zero out one row and column, leaving most of the high frequencies in the image, to have a 20% compression rate. Since you can never get that data back, JPEG is a lossy image format, which means that you lose data when you compress it and can never get that exact data back. There are some variations on the JPEG that are lossless but those are uncommon.

To restore the image from the compressed matrix, you apply the inverse DCT on each 8x8 block. You can achieve very high levels of compression with minimal image quality loss using the DCT.

Here is an example of an 8x8 matrix that compressed and decompressed using the DCT, zeroing out 70% of the coefficient matrix.

$$Original = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

$$Decompressed = \begin{bmatrix} 149 & 134 & 119 & 116 & 121 & 126 & 127 & 128 \\ 204 & 168 & 140 & 144 & 155 & 150 & 135 & 125 \\ 253 & 195 & 155 & 166 & 183 & 165 & 131 & 111 \\ 245 & 185 & 148 & 166 & 184 & 160 & 124 & 107 \\ 188 & 149 & 132 & 155 & 172 & 159 & 141 & 136 \\ 132 & 123 & 125 & 143 & 160 & 166 & 168 & 171 \\ 109 & 119 & 126 & 128 & 139 & 158 & 168 & 166 \\ 111 & 127 & 127 & 114 & 118 & 141 & 147 & 135 \end{bmatrix}$$

You can see how similar the values are. The difference between the pixel values are so minimal the human eye cannot tell a difference, even though the compression rate is at 70%. This is why the JPEG image format remains so popular to this day. It is the best way to easily and efficiently store an image.

To help show how the level of compression affects an image, I wrote a C++ program in Visual Studio that applied the DCT to a black and white image and applied different levels of compression.

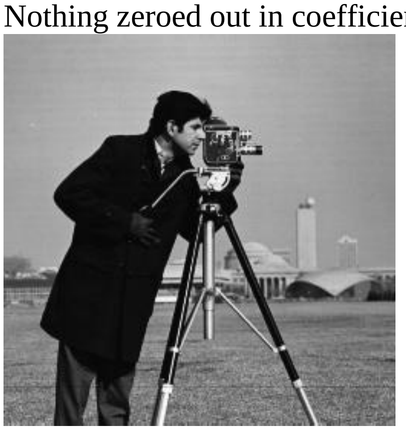
Highest Compression:
Zeroed out 94% of coefficient matrix.

Zeroed out 75% of matrix.

Zeroed out 50% of matrix

Zeroed out 25% of matrix

Nothing zeroed out in coefficient matrix



Original Image

Works Cited

Cabeen, Ken, and Peter Gent. "Image Compression and Discrete Cosine Transform." N.p., n.d. Web. 3 May 2017. <https://www.math.cuhk.edu.hk/~lmlui/dct.pdf+>.

Lossy Data Compression: JPEG. N.p., n.d. Web. 03 May 2017. <http://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/dct.htm>.

Marshall, Dave. The Discrete Cosine Transform (DCT). N.p., 4 Oct. 2001. Web. 03 May 2017. <https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node231.html>.