JPEG COMPRESSION USING THE DISCRETE COSINE TRANSFORM

CHARLES PARK AND JAMIE MORROW

1. Abstract

In this report, we will discuss the Discrete Cosine Transform (DCT) and its application to image compression, specifically in the JPEG format. We begin with the motivation for a lossy image compression that still retains a significant amount of energy, and argue that for this, the DCT is a superior algorithm to a Discrete Fourier Transform. We then explain the one-dimensional DCT, extend it to the two-dimensional case, and use it to create a compression algorithm. We finish with examples of two different compression qualities, a discussion of the JPEG format drawbacks, and what improvements can be made.

2. Motivation

Digital images are everywhere in today's world. With the advent of the Internet as a medium for social interaction, these images have infiltrated nearly every aspect of our live, from our cell phones to browsing the Internet, to the silly picture of your cat that your mom sends you. Raw images, especially at the size that modern cameras shoot at, require an absurd amount of storage space, so there is a massive need to compress this huge number of photos. If we allow our compression to be lossy, that is to say we can remove data to make a smaller file, we can look at the relationship between pixels and their neighbors. The Discrete Cosine Transform (DCT) is perfect for this, for a number of reasons. Most notably, however, is that although it has roughly the same complexity as the Fast Fourier Transform (FFT), the DCT retains much more energy, encoding more information in the same or smaller number of bytes.

3. Theory

3.1. **The One-Dimensional DCT.** Generally, the DCT is an invertible, linear transformation that transforms some real signal into a vector of coefficients of cosine functions. Similar to the Fourier transform, it uses periodic basis elements, but rather than complex exponentials of increasing frequency, it uses cosines of increasing frequency of the form:

$$\sqrt{\frac{2}{N}} \cos\left(\left(\frac{2\pi n + N}{2N}\right) k\right)$$

where N is the number of discrete values to be transformed, $n = 0, 1, \ldots, N-1$ and $k = 0, 1, \ldots, N-1$.

Date: March 27, 2016.

The transform itself involves taking the inner product of these basis elements with a signal $x[n]$, such that the transformed signal, $x(k)$ will be:

$$\sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x[n] \cos\left(\left(\frac{2\pi n + N}{2N}\right) k\right)$$

3.2. **One-Dimensional Example.** To illustrate, we take a vector of 8 random integers between 1 and 10, $\{3, 1, 6, 8, 7, 7, 8, 7, 1, 2\}$ and calculate its DCT to be:

$$\{22.36, -5.74, -5.61, 0.95, -0.54, 2.41, -2.39, -1.24, -1.94, 0.066\}$$

where each entry, beginning at an index of zero corresponds to the cosine at frequency k.

3.3. **Extension into Two Dimensions.** We can extend this to a two dimensional signal by introducing another indexing variable, j. Then instead of a one-dimensional vector, we get a two-dimensional matrix T, where each entry is calculated as:

$$T_{i,j} = \frac{1}{\sqrt{2N}} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \cos\left(\frac{(2n_1+1)i\pi}{2N}\right) \cos\left(\frac{(2n_2+1)j\pi}{2N}\right)$$

Where N is once again the number of discrete values in the signal, and $i, j \in \{0, 1, 2, \ldots, N-1\}$

4. The JPEG Compression Algorithm

The actual math behind the compression is relatively easy with a basic knowledge of linear algebra. We take a square image that and represent it as a matrix M_0 of $n \times n$, where n is the number of pixels along an edge. Before anything else, we shift and scale the entries so that instead of its entries ranging in intensity from 0 to 1, they range from -128 to 127. That is to say, for every entry i, j, $M_{i,j} = 255M_{0(i,j)} - 128$. As well we take T to be a normalized $n \times n$ matrix of DCT values from $i, j \in \{0, 1, \ldots, n-1\}$, which we generate using the following piecewise equation:

$$T_{i,j} = \begin{cases} \frac{1}{\sqrt{n}} & : i = 0 \\ \sqrt{\frac{2}{n}} \cos\left(\frac{(2j+1)i\pi}{2n}\right) & : i > 0 \end{cases}$$

We then transform both the rows and the columns of M by multiplying it by both T and the transpose of T.

$$D = TMT^{\intercal}$$

Continuing, we break D into smaller 8×8 submatricies and multiply every entry by the corresponding quantization matrix Q_x of some quality x, and round so that any negligable terms go to zero.

$$C_{i,j} = round\left(\frac{D_{i,j}}{Q_{i,j}}\right)$$

To decompress, we then multiply the elements of the rounded C and the quantization Q matrix to make a matrix R:

$$R_{i,j} = Q_{i,j} \times C_{i,j}$$

And represent R as an image N by taking its inverse DCT:

$$N = T^{\intercal} R T$$

To illustrate numericaly, we take M_0 to be the 8×8 image matrix:

$$M_0 = \begin{pmatrix} 0.173 & 0.690 & 0.750 & 0.032 & 0.053 & 0.112 & 0.743 & 0.715 \\ 0.816 & 0.310 & 0.701 & 0.596 & 0.217 & 0.198 & 0.463 & 0.968 \\ 0.616 & 0.531 & 0.099 & 0.915 & 0.419 & 0.818 & 0.512 & 0.981 \\ 0.506 & 0.852 & 0.865 & 0.495 & 0.970 & 0.179 & 0.402 & 0.728 \\ 0.730 & 0.271 & 0.744 & 0.796 & 0.849 & 0.383 & 0.544 & 0.428 \\ 0.886 & 0.326 & 0.373 & 0.632 & 0.060 & 0.748 & 0.259 & 0.666 \\ 0.268 & 0.113 & 0.107 & 0.395 & 0.330 & 0.548 & 0.839 & 0.755 \\ 0.281 & 0.027 & 0.920 & 0.720 & 0.244 & 0.841 & 0.149 & 0.115 \end{pmatrix}$$

Which shifting the intensity values to be on the interval $[-128, 127]$, we get a matrix M to be:

$$M = \begin{pmatrix} -83.885 & 47.950 & 63.250 & -119.840 & -114.485 & -99.440 & 61.465 & 54.325 \\ 80.080 & -48.950 & 50.755 & 23.980 & -72.665 & -77.510 & -9.935 & 118.840 \\ 29.080 & 7.405 & -102.755 & 103.325 & -21.155 & 80.590 & 2.560 & 122.155 \\ 1.030 & 89.260 & 92.575 & -1.775 & 119.350 & -82.355 & -25.490 & 57.640 \\ 58.150 & -58.895 & 61.730 & 74.980 & 88.495 & -30.335 & 10.720 & -18.860 \\ 97.930 & -44.870 & -32.885 & 33.160 & -112.700 & 62.740 & -61.955 & 41.830 \\ -59.660 & -99.185 & -100.715 & -27.275 & -43.850 & 11.740 & 85.945 & 64.525 \\ -56.345 & -121.115 & 106.600 & 55.600 & -65.780 & 86.455 & -90.005 & -98.675 \end{pmatrix}$$

And a T:

$$T = \begin{pmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .277 & .0975 & -0.975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{pmatrix}$$

and the product matrix D:

$$D = TMT^{\intercal} = \begin{pmatrix} 20.607 & -46.409 & 56.128 & -101.92 & 61.773 & 68.296 & 89.362 & -25.872 \\ 58.758 & 20.519 & 150.024 & -115.820 & 3.245 & -74.640 & -91.904 & 52.030 \\ -153.305 & -61.778 & 40.015 & -56.019 & -139.287 & 58.998 & 59.373 & -17.557 \\ -57.039 & 4.937 & 108.347 & -5.227 & -42.441 & 29.708 & -19.699 & 50.644 \\ -5.036 & 107.456 & -134.863 & -72.622 & -102.382 & -57.745 & -13.071 & 159.254 \\ -23.415 & -102.314 & 64.744 & -8.845 & -66.753 & -104.772 & -123.321 & 19.308 \\ -4.121 & 27.292 & -8.074 & 46.460 & -36.079 & -1.692 & 57.776 & -146.170 \\ 0.982 & -109.561 & -33.359 & 58.358 & 10.380 & 46.900 & -61.711 & -25.397 \end{pmatrix}$$

With a quantization matrix of quality 50:

$$Q_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

We compress D to be:

$$D_{compressed} = \begin{pmatrix} -5 & 4 & 6 & -7 & -5 & -2 & 1 & 1 \\ 7 & -4 & 4 & 1 & -3 & -1 & 0 & 2 \\ 2 & 1 & -6 & 4 & -1 & 1 & 0 & 2 \\ 0 & 5 & 4 & 0 & 2 & -1 & 0 & 1 \\ 3 & -3 & 2 & 1 & 1 & 0 & 0 & 0 \\ 4 & -1 & -1 & 1 & -1 & 1 & -1 & 0 \\ -1 & -2 & -1 & 0 & 0 & 0 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \end{pmatrix}$$

To decompress the image, we multiplying the rounded C by the quantization matrix Q_{50} to get:

$$\begin{pmatrix} -80 & 44 & 60 & -112 & -120 & 80 & 51 & 61 \\ 84 & -48 & 56 & 19 & -78 & -58 & 0 & 110 \\ 28 & 13 & -96 & 96 & -40 & 57 & 0 & 112 \\ 0 & 85 & 88 & 0 & 102 & -87 & 0 & 62 \\ 54 & -66 & 74 & 56 & 68 & 0 & 0 & 0 \\ 96 & -35 & -55 & 64 & -81 & 104 & -113 & 0 \\ -49 & -128 & -78 & 0 & 0 & 0 & 120 & 101 \\ -72 & -92 & 95 & 98 & -112 & 100 & -103 & -99 \end{pmatrix}$$

And apply the Inverse DCT to get :

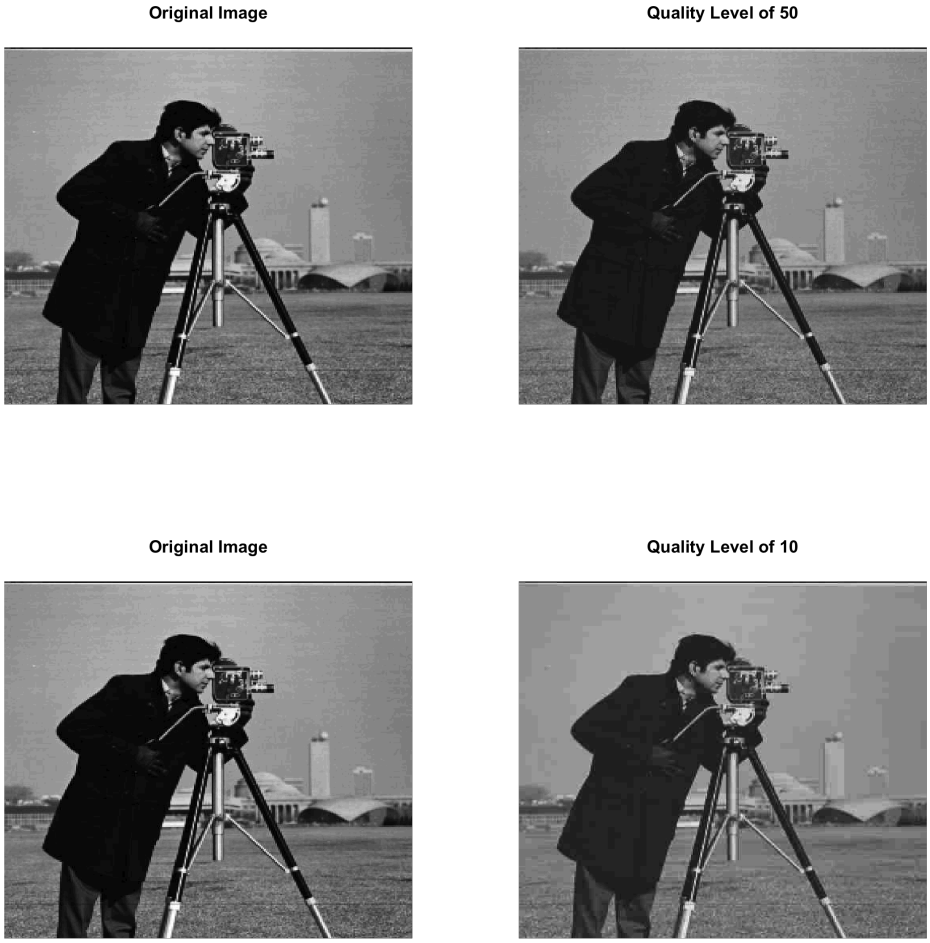
$$\begin{pmatrix} 34.659 & 0.675 & 59.552 & -136.719 & 86.532 & 8.177 & 103.190 & 0.069 \\ -4.981 & 89.804 & 120.679 & -83.001 & -1.151 & -64.185 & -98.037 & 8.235 \\ -152.475 & -40.339 & 37.422 & -78.543 & -96.236 & 66.310 & 95.509 & -37.001 \\ -12.850 & 70.457 & 63.506 & 22.429 & 0.920 & -11.477 & -1.908 & 68.782 \\ 17.711 & 68.308 & -100.902 & -87.162 & -147.374 & -60.023 & -64.288 & 101.739 \\ -34.776 & -39.174 & 122.304 & -63.268 & 33.840 & -96.398 & -192.458 & 48.934 \\ -9.611 & 98.969 & -31.747 & 50.992 & -93.940 & -26.174 & 65.513 & -131.014 \\ -48.375 & -97.190 & 42.923 & -9.262 & -10.044 & 18.358 & -47.359 & -37.057 \end{pmatrix}$$

Giving us a scaled intensity image matrix of:

$$\begin{pmatrix} 0.635 & 0.502 & 0.732 & -0.034 & 0.838 & 0.531 & 0.903 & 0.500 \\ 0.480 & 0.850 & 0.971 & 0.175 & 0.495 & 0.249 & 0.117 & 0.532 \\ -0.095 & 0.342 & 0.646 & 0.193 & 0.124 & 0.759 & 0.873 & 0.355 \\ 0.449 & 0.775 & 0.748 & 0.587 & 0.503 & 0.455 & 0.492 & 0.768 \\ 0.569 & 0.766 & 0.105 & 0.159 & -0.075 & 0.265 & 0.248 & 0.897 \\ 0.364 & 0.346 & 0.977 & 0.252 & 0.632 & 0.123 & -0.251 & 0.691 \\ 0.462 & 0.886 & 0.375 & 0.699 & 0.133 & 0.397 & 0.755 & -0.011 \\ 0.311 & 0.120 & 0.667 & 0.463 & 0.460 & 0.571 & 0.315 & 0.355 \end{pmatrix}$$

5. Example: The Cameraman

We used this algorithm to compress the perennially processed "Cameraman" at two different qualities, 50 and 10, shown on the following page. Although the quality of the image was sacrificed, we successfully compressed the image, nearly to half of its original size, with original image being 199 KB and the compressed at quality 10 being 140 KB.



6. The Drawbacks of JPEG Compression

Although JPEG compression preserves images relatively well, there are a number of drawbacks to using the format. First and foremost of which are JPEG artifacts, the lack of detail and weird grids that appear in large areas of a similar tones or color. These artifacts are usually easiest to spot in gradients, where the JPEG compression will change it into discrete steps of tone instead of a continuous spectrum. In addition, JPEGs are not good for images that are predominantly one value with small lines of a radically different tone. In these images, the compression will diffuse the sharp lines, as well as disrupt the background around the lines. These problems can be rectified using a wavelet transform that allows perfect reconstruction, but the JPEG format that supports wavelets has failed to usurp JPEG as the predominant image compression format. Despite these drawbacks however, for applications where the image quality is not of the utmost priority, JPEG

compression allows images to be a reasonable size, while still having the quality to show off your latest vacation to your friends.

7. Bibliography

Broughton, S.A. and K. Bryan. "The Discrete Cosine Transform." *Discrete Fourier Analysis and Wavelets*, 105-237. 2009.

Cabeen, K. and P. Gent. "Image Compression and the Discrete Cosine Transform" Online.

Strang, G. "The Discrete Cosine Transform" *SIAM Review*, Vol. 41, No. 1, 135-147. 1999.