Introduction
JPEG Compression
Example Images

An Introduction to JPEG Compression

Tim & Jesse Trutna

12/14/01

Abstract

The amount of information required to store pictures on modern computers is quite large in relation to the amount of bandwidth commonly available to transmit them over the Internet and applications like video where many thousands of pictures are required would be prohibitively intensive for use on most systems if there wasn't a way to reduce the storage requirements of these pictures. This article covers lossy JPEG compression, one of the most common methods of compressing continuous-tone greyscale or color still images (JPEG compression can be applied to movies as well, it is simply done one frame at a time).

1. Introduction

When the eye perceives an image on a computer monitor, it is in actually perceiving a large collection of finite color elements, or pixels as shown in Figure 1, 5164 pixels a square inch, to be precise. Each of these pixels is in and of itself composed of three dots of light; a green dot, a blue dot, and a red dot. The color the eye perceives at each pixel is a result of varying intensities of green, red, and blue light emanating from that location. A computer image can thus be represented as 3 matrixes of values, each corresponding to the brightness of a particular color in each pixel. If one were to examine one of these intensity matrices with the relative intensity being represented as a color between black and white, it would appear to be a greyscale image. Indeed, this is the basis for the RGB color scheme. Three greyscale intensity images, one for each color, that when superimposed give a resulting "full" color image as demonstrated in Figure 2.

The intensity map of black to white occurs in incremental steps according to the "bit-depth" of the image. That is, the colors from black to white are arranged along a number line, The greater the number

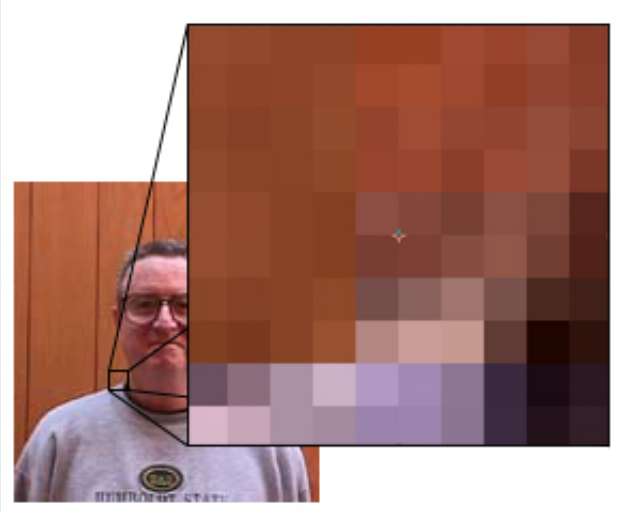

Figure 1: Images are created out of pixels

Introduction
JPEG Compression
Example Images

Figure 2: Greyscale to Color

Introduction
JPEG Compression
Example Images

of steps, the larger the bit-depth. If each intensity value is represented as an 8-bit number, then there are 256 variations. If the intensity values are represented as 16-bit numbers, there are 32,000 variations between absolute black and pure white. Figure 3 demonstrates a black to white gradient in 4-bits of intensity.

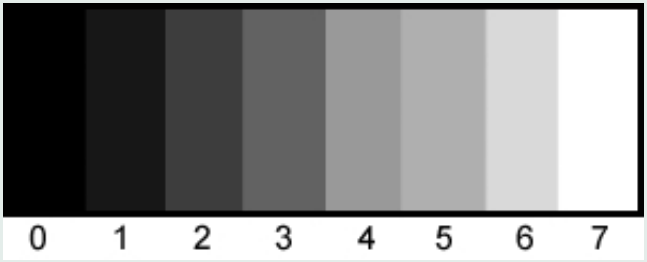

Figure 3: 4-bit Black to White Gradient

Representing images in this fashion, however, takes a great deal of space. Consider a 4-inch square image on a computer screen. A 4-inch square image, if it were represented as a 4*72=288 element square matrix, would require 82,994 pixels, at 8-bits a pixel thats 82,994 bytes for each greyscale image, or 82,994*3=248,832 bytes for the color image. That's a quarter of a megabyte for a 4 square inch image. If the resolution of that image were increased to 16-bit, the file size would double and 24-bit and 32-bit images aren't uncommon either. The file sizes involved make it difficult to transmit pictures over limited bandwidth systems like the Internet. Clearly some way is needed to "compress" the image, or reduce the amount of space it takes up.

The method of image compression that will be analyzed in this paper is the lossy JPEG image compression technique. The most popular compression technique for continuous-tone greyscale and color images.

JPEG compression is able to greatly reduce file size with minimal image degradation by throwing away the least "important" information. Just how it decides what is "important" will be covered later in this paper. Because it actually eliminates information, it is considered a "lossy" compression technique. That is, the final image resulting from the compression technique is not exactly the original image.

Introduction
JPEG Compression
Example Images

2. JPEG Compression

The JPEG compression process is broken into three primary parts as shown in Figure 4. To prepare for processing, the matrix representing the image is broken up into 8x8 squares (the size was determined when the JPEG standard was created as a balance between image quality and the processing power of the time) and passed through the encoding process in chunks. To reverse the compression and display a close approximation to the original image (how close depends on the quantization matrix supplied by the user which will be covered later) the compressed data is fed into the reverse process as shown in Figure 5. Color images are separated into the different channels (each equivalent to a greyscale channel) and treated individually.

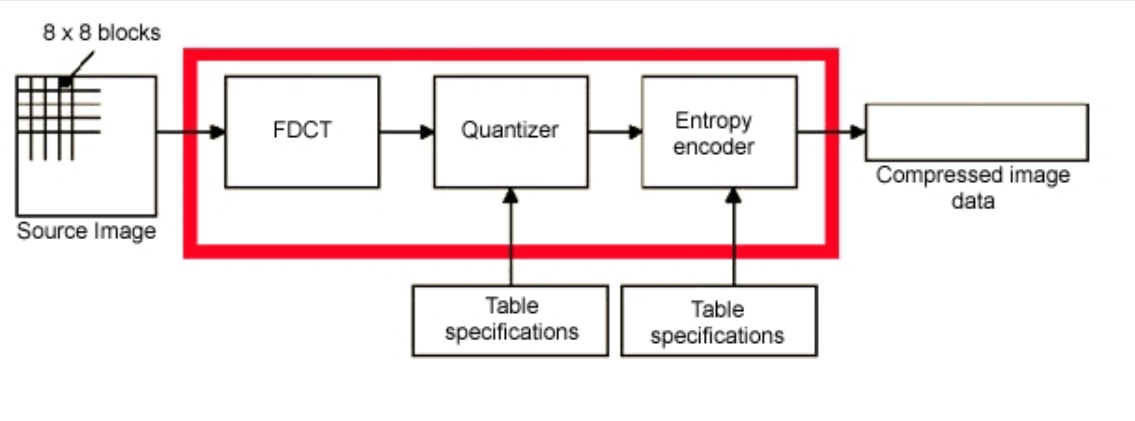

Figure 4: Encoding Schematic

Additionally, before being fed into the *DCT* every value in the matrix is shifted from an unsigned integer with range $[0, 2^{p-1} - 1]$ to a signed integer with range $[-(2^{p-1}), 2^{p-1} - 1]$ by subtracting 2^{p-1} from the value where p is the number of bits per channel (or bits per value). In the case of the standard 8-bit channel the numbers are shifted from $[0, 255]$ to $[-128.127]$ by subtracting 128. This centers the activity around 0 for easier handling with *cosine* functions.

Introduction
JPEG Compression
Example Images

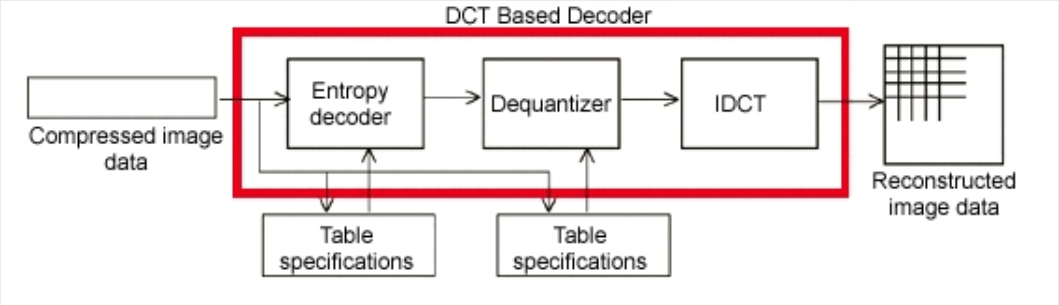

Figure 5: Decoding Schematic

2.1. Discrete Cosine Transform

$$FDCT(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7}p(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

$$\begin{aligned} IDCT(x,y) &= \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)DCT(u,v)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right] \\ &\approx p(x,y) \end{aligned}$$

where

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

and p(x,y) is the matrix of values from the original image.

The discrete cosine transform (DCT) is closely related to the Discrete Fourier Transform (DFT). Both take a set of points from the spatial domain and transform them into an equivalent representation in the frequency domain. The difference is that while the DFT takes a discrete signal in one spatial dimension and transforms it into a set of points in one frequency dimension the Discrete Cosine Transform (for an

8x8 block of values) takes a 64-point discrete signal, which can be thought of as a function of two spatial dimensions x and y, and turns them into 64 basis-signal amplitudes (also called DCT coefficients) which are in terms of the 64 unique orthogonal two-dimensional "spatial frequencies" or "spectrum" shown in Figure 6. The DCT coefficient values are the relative amounts of the 64 spatial frequencies present in the original 64-point input. The element in the upper most left corresponding to zero frequency in both directions is the "DC coefficient" and the rest are called "AC coefficients."

Calculating the coefficient matrix this way is rather inefficient. The two summations will require N^2 calculations where N is the length (or width) of the matrix. A more efficient way to calculate the matrix is with matrix operations. Set C equal to

$$C_{i,j} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N}\right] & \text{if } i > 0 \end{cases}$$

Since we always deal with $8x8$ matrices when dealing with JPEG compression this can be calculated once and referenced to for all calculations as computed in Equation 2. Using C the $FDCT$ can be found by

$$FDCT(u,v) = C * P * Transpose[C] \tag{1}$$

where P is the matrix of values from the image being compressed.

$$C = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix} \tag{2}$$

Let us take an example matrix of values say

Introduction
JPEG Compression
Example Images

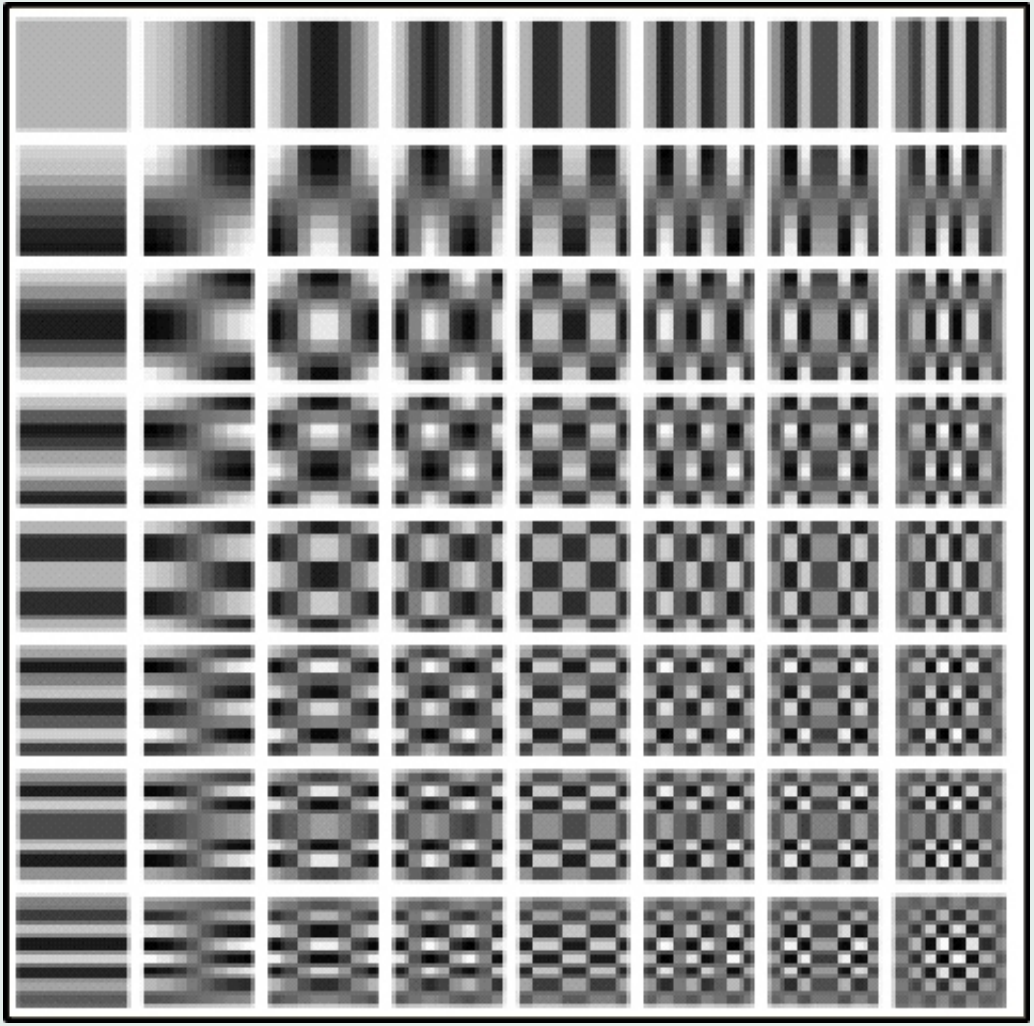

Figure 6: 64 two-dimensional spatial frequencies

$$P = \begin{bmatrix} 124 & 85 & 161 & 160 & 135 & 76 & 138 & 113 \\ 147 & 165 & 103 & 126 & 122 & 136 & 184 & 155 \\ 263 & 162 & 199 & 118 & 192 & 150 & 128 & 124 \\ 280 & 176 & 177 & 192 & 211 & 213 & 153 & 85 \\ 177 & 112 & 94 & 191 & 179 & 114 & 159 & 176 \\ 176 & 100 & 91 & 115 & 204 & 211 & 234 & 170 \\ 139 & 61 & 118 & 109 & 98 & 91 & 137 & 154 \\ 110 & 180 & 162 & 85 & 147 & 137 & 145 & 161 \end{bmatrix}$$

We then subtract 128 from every element

$$P = \begin{bmatrix} -4 & -43 & 33 & 32 & 7 & -52 & 10 & -15 \\ 19 & 37 & -25 & -2 & -6 & 8 & 56 & 27 \\ 135 & 34 & 71 & -10 & 64 & 22 & 0 & -4 \\ 152 & 48 & 49 & 64 & 83 & 85 & 25 & -43 \\ 49 & -16 & -34 & 63 & 51 & -14 & 31 & 48 \\ 48 & -28 & -37 & -13 & 76 & 83 & 106 & 42 \\ 11 & -67 & -10 & -19 & -30 & -37 & 9 & 26 \\ -18 & 52 & 34 & -43 & 19 & 9 & 17 & 33 \end{bmatrix}$$

And feed those values into Equation 1 to get

$$FDCT = \begin{bmatrix} 162.29 & 4.49 & 33.58 & 71.88 & 48.26 & 39.39 & -2.55 & 62.78 \\ 23.81 & 75.72 & -23.93 & -2.07 & 10.34 & 29.95 & -6.29 & -0.09 \\ -133.14 & -40.95 & 16.36 & -108.35 & -58.02 & -23.96 & -8.77 & 12.34 \\ -61.87 & -115.48 & -16.13 & -12.27 & 47.01 & 55.68 & -34.09 & -7.66 \\ 15.75 & 44.84 & -55.86 & -33.21 & 12.75 & -3.39 & -12.04 & -12.64 \\ -10.22 & 66.91 & -32.01 & 13.41 & 14.01 & 45.41 & 57.56 & 27.71 \\ 58.13 & 7.34 & -15.27 & 37.19 & -17.91 & -26.81 & 24.14 & 47.34 \\ -71.62 & 18.14 & 11.32 & -74.14 & 57.60 & 7.68 & 12.55 & 19.63 \end{bmatrix}$$

At this point we haven't really accomplished any compression. In fact the matrix $FDCT$ takes up more space than P because it has the same number of elements but each element is a floating point number with a range of $[-1023, 1023]$ instead of an integer with range $[0, 254]$. The important thing is the data has been organized in terms of importance. The human eye has more difficulty discriminating

Introduction
JPEG Compression
Example Images

between higher frequencies than low and most computer data is relatively low frequency. Low frequency data carries more important information than the higher frequencies. The data in the $FDCT$ matrix is organized from lowest frequency in the upper left to highest frequency in the lower right. This prepares the data for the next step, quantization.

2.2. Quantization

Quantization is the step where we actually throw away data. The DCT is a lossless procedure. The data can be precisely recovered through the $IDCT$ (this isn't entirely true, in reality the $FDCT$ and $IDCT$ contain transcendental functions which no physical implementation can compute with perfect accuracy). During Quantization every element in the $8x8$ $FDCT$ matrix is divided by a corresponding element in a quantization matrix Q to yield a matrix Q_{FDCT} according to the formula:

$$Q_{FDCT} = round\left(\frac{FDCT(u,v)}{Q(u,v)}\right)$$

The goal of quantization is to reduce most of the less important high frequency coefficients to zero, the more zeros we can generate the better the image will compress. The matrix Q generally has lower numbers in the upper left that increase in magnitude as they get closer to the lower right. While Q could be any matrix the JPEG committee has recommended certain ones that seem to work well, such as the example matrix Q_{50} below.

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (3)$$

Another convenience of this method is that it allows the user to customize the level of compression at runtime to fine tune the quality/compression ratio. If the user wants better quality at the price of compression he can lower the values in the Q matrix. If he wants higher compression with less image quality he can raise the values in the matrix.

If we take our example matrix $FDCT$ and apply Equation 3 to it (at a Q_{50} quality level) we arrive at:

$$Q_{FDCT} = \begin{bmatrix} 10 & 0 & 3 & 4 & 2 & 1 & 0 & 1 \\ 2 & 6 & -2 & 0 & 0 & 1 & 0 & 0 \\ -10 & -3 & 1 & -5 & -1 & 0 & 0 & 0 \\ -4 & -7 & -1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 2 & -2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Quite a few more zeros than before. But if we raise the values in the Q matrix to say some quality level Q_{30} according to the formula

$$Q_x = \begin{cases} \frac{100-x}{50} * Q_{50} & \text{if } Q_x > Q_{50} \\ \frac{50}{x} * Q_{50} & \text{if } Q_x < Q_{50} \end{cases}$$

we arrive at the quality matrix

$$Q_{30} = \begin{bmatrix} 27 & 18 & 17 & 27 & 40 & 67 & 85 & 102 \\ 20 & 20 & 23 & 32 & 43 & 97 & 100 & 92 \\ 23 & 22 & 27 & 40 & 67 & 95 & 115 & 93 \\ 23 & 28 & 37 & 48 & 85 & 145 & 133 & 103 \\ 30 & 37 & 62 & 93 & 113 & 182 & 172 & 128 \\ 40 & 58 & 92 & 107 & 135 & 173 & 188 & 153 \\ 82 & 107 & 130 & 145 & 172 & 202 & 200 & 168 \end{bmatrix}$$

Introduction
JPEG Compression
Example Images

yielding a matrix Q_{FDCT} with even more zeros

$$Q_{FDCT} = \begin{bmatrix} 6 & 0 & 2 & 3 & 1 & 1 & 0 & 1 \\ 1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 \\ -6 & -2 & 1 & -3 & -1 & 0 & 0 & 0 \\ -3 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

This is the point at which the compression has become "lossy." When we reverse the process by the following formula

$$FDCT(u, v) = Q_{FDCT} * Q(u, v)$$

we will only loose up to one percent of the value of $Q(u, v)$ from each element of $FDCT(u, v)$ that didn't get reduced to zero. But those values that were reduced to zero are gone forever and cannot be reconstructed. The coefficients we have lost however were the higher frequency, less important ones. We have succeeded in turning most of the higher frequency coefficients into zeros which is good because the more zeros there are the more compact we can make the data. Actually, it's how many zeros we can get in a row that determines how much we can compress the data. That is were Entropy Encoding comes in.

2.3. Entropy Encoding

Instead of reading across the rows, row after row, JPEG compression reads along the diagonals. This tends to groups the lower coefficients in the beginning of the string and distribute the zeros in longer continuous strings. The pattern the data is read off is demonstrated in Figure Figure 7.

Reading in that pattern turns our matrix Q_{FDCT} from Equation 4 into a string of values like so

$$EE_{Q_{FDCT}} = \begin{bmatrix} 6 & 0 & 1 & -6 & . & . & 0 & 0 & 0 & 0 \end{bmatrix}$$

Instead of delivering that to the receiving decoder, the string of zeros is replaced with a code that describes how many zeros there are in a line. For example a string of fifteen zeros 000000000000000000 might be described as 01111 (1111 is binary for the number fifteen). This condensed string of numbers

Introduction
JPEG Compression
Example Images

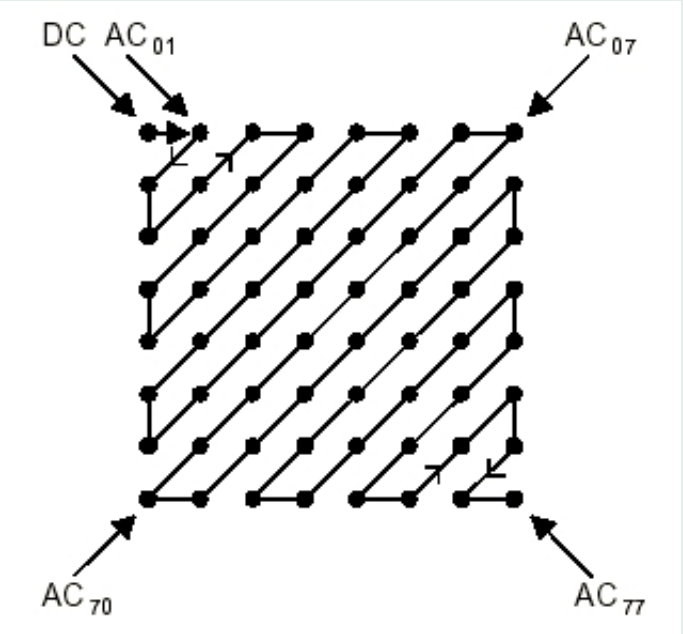

Figure 7: The zig-zag pattern of Entropy Encoding

Introduction
JPEG Compression
Example Images

is then passed on to a Huffman or arithmetic encoder. These further condense the string by looking for recurring sequences of numbers that are placed into a dictionary and referenced. The data is then sent to a receiving computer that performs the reverse operation on the data and reconstructs an image that may be visually indistinguishable from the original image while requiring substantially less storage space.

3. Example Images

We can best illustrate the power of JPEG compression by comparing an image and varying levels of compression. We begin with an initial matrix P and accompanying graph Figure Figure 8.

$$P = \begin{bmatrix} 124 & 85 & 161 & 160 & 135 & 76 & 138 & 113 \\ 147 & 165 & 103 & 126 & 122 & 136 & 184 & 155 \\ 263 & 162 & 199 & 118 & 192 & 150 & 128 & 124 \\ 280 & 176 & 177 & 192 & 211 & 213 & 153 & 85 \\ 177 & 112 & 94 & 191 & 179 & 114 & 159 & 176 \\ 176 & 100 & 91 & 115 & 204 & 211 & 234 & 170 \\ 139 & 61 & 118 & 109 & 98 & 91 & 137 & 154 \\ 110 & 180 & 162 & 85 & 147 & 137 & 145 & 161 \end{bmatrix}$$

And then examine the the varying levels of compression, the difference between the reconstructed matrix and the original, and the graphical output that is the result.

$$QDCT(75) = \begin{bmatrix} 20 & 1 & 7 & 9 & 4 & 2 & 0 & 2 \\ 4 & 13 & -3 & 0 & 1 & 1 & 0 & 0 \\ -19 & -6 & 2 & -9 & -3 & -1 & 0 & 0 \\ -9 & -14 & -1 & -1 & 2 & 1 & -1 & 0 \\ 2 & 4 & -3 & -1 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ -2 & 0 & 0 & -2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Introduction
JPEG Compression
Example Images

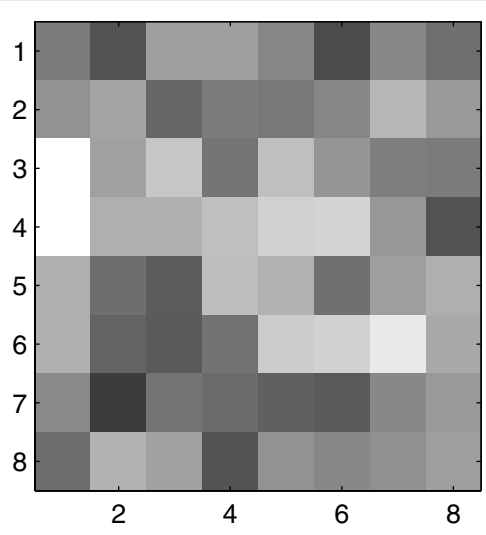

Figure 8: P Original

Introduction
JPEG Compression
Example Images

$$\text{diff}(75) = \begin{bmatrix} -1 & 2 & -4 & 8 & 4 & 2 & -2 & -3 \\ -7 & -5 & 0 & 0 & -3 & -18 & 22 & 0 \\ 5 & 4 & 15 & -18 & 10 & 12 & -10 & -3 \\ -14 & 7 & -21 & 18 & -1 & 0 & -6 & 13 \\ 16 & -4 & 2 & -4 & -6 & -5 & 3 & -8 \\ -16 & 5 & 18 & 3 & -4 & 18 & -9 & 2 \\ 20 & -12 & -13 & -1 & 2 & -2 & 8 & -5 \\ -6 & 5 & 4 & -3 & 6 & -9 & 0 & 6 \end{bmatrix}$$

At quality level 75 the graph (Figure 10) is very similar to the original and has a decent quantity of zero elements.

$$QDCT(50) = \begin{bmatrix} 10 & 0 & 3 & 4 & 2 & 1 & 0 & 1 \\ 2 & 6 & -2 & 0 & 0 & 1 & 0 & 0 \\ -10 & -3 & 1 & -5 & -1 & 0 & 0 & 0 \\ -4 & -7 & -1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 2 & -2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{diff}(50) = \begin{bmatrix} 5 & 32 & -24 & -30 & 24 & -13 & 11 & -7 \\ -22 & 10 & 5 & 23 & -6 & 7 & -35 & 30 \\ 19 & -17 & 34 & -31 & 12 & 20 & 8 & -27 \\ 18 & -23 & -38 & 37 & -8 & -32 & 11 & 14 \\ 2 & 15 & 4 & -4 & -23 & 18 & -14 & -3 \\ -13 & 2 & 7 & -1 & 23 & -3 & -14 & 4 \\ 3 & 0 & -7 & 10 & -25 & -26 & 42 & -15 \\ 10 & -9 & 10 & -12 & 9 & 31 & -5 & -6 \end{bmatrix}$$

At quality level 50 image quality begins to degrade (Figure 11) but we have a lot of zero elements.

Introduction
JPEG Compression
Example Images

$$QDCT(25) = \begin{bmatrix} 5 & 0 & 2 & 2 & 1 & 0 & 0 & 1 \\ 1 & 3 & -1 & 0 & 0 & 0 & 0 & 0 \\ -5 & -2 & 1 & -2 & -1 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{diff}(25) = \begin{bmatrix} 11 & -39 & -6 & 49 & -5 & -31 & -3 & -5 \\ 6 & 52 & -28 & 38 & -15 & 26 & 29 & 7 \\ -2 & -31 & 12 & -41 & -18 & -1 & -38 & -29 \\ 21 & 5 & 6 & 18 & -22 & 70 & 32 & -8 \\ -9 & 7 & -28 & 45 & -51 & -51 & -4 & 30 \\ 3 & -8 & -26 & 2 & 10 & 44 & 26 & -41 \\ 4 & -41 & 5 & 44 & -18 & -7 & -16 & 0 \\ -11 & 52 & -1 & -19 & 9 & 20 & -20 & 10 \end{bmatrix}$$

At image quality 25 the image (Figure 12) is really starting to differ from the original, although it would have an excellent compression ratio.

$$QDCT(10) = \begin{bmatrix} 2 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Introduction
JPEG Compression
Example Images

$$\text{diff}(10) = \begin{bmatrix} 30 & -21 & 46 & 49 & 32 & -32 & 7 & -39 \\ -15 & 13 & -38 & -7 & -10 & 2 & 48 & 18 \\ 29 & -35 & 38 & -36 & 23 & -22 & -21 & 2 \\ 30 & -22 & 27 & 40 & 24 & 14 & -12 & -37 \\ -30 & -47 & -26 & 58 & 0 & -88 & -16 & 39 \\ 24 & -25 & -13 & -3 & 48 & 31 & 61 & 15 \\ 15 & -59 & 1 & -13 & -36 & -57 & -22 & -10 \\ -10 & 49 & 23 & -47 & 25 & 12 & -1 & -6 \end{bmatrix}$$

At image quality 10 the reconstructed matrix has little resemblance to the original (Figure 13). The image may compress excellently but it does us little good if we can't see what the image is supposed to be.

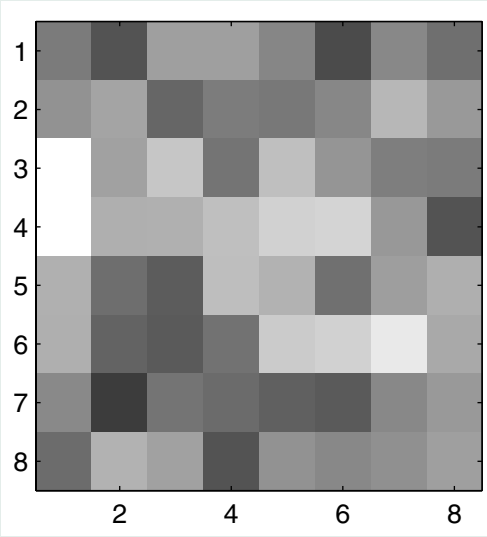

Figure 9: P Original

Introduction
JPEG Compression
Example Images

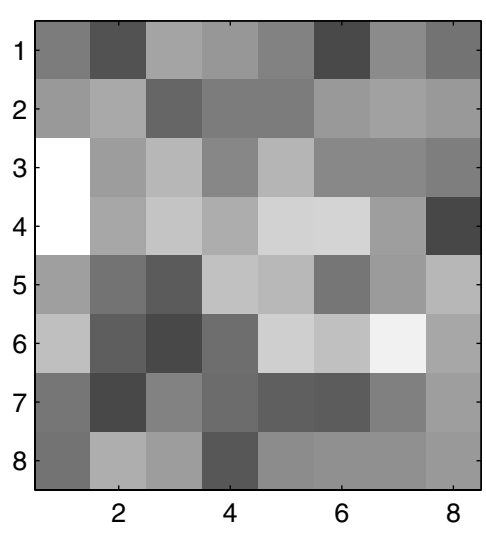

Figure 10: P(75)

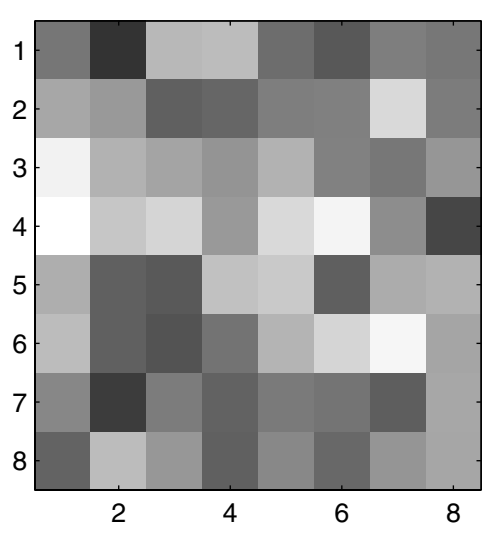

Figure 11: P(50)

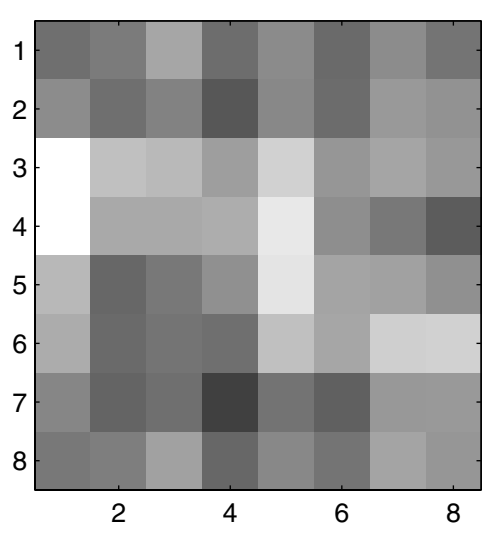

Figure 12: P(25)

Introduction
JPEG Compression
Example Images

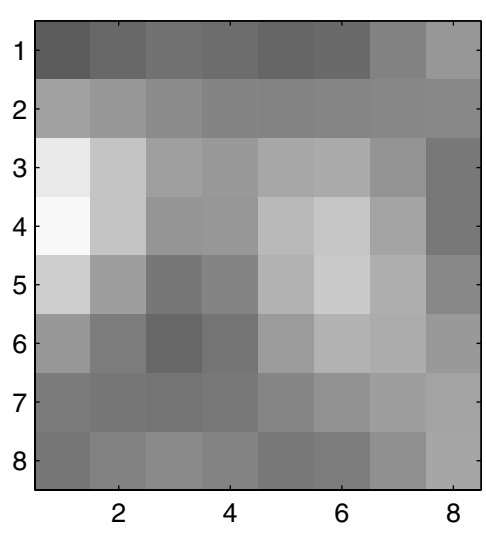

Figure 13: P(10)

4. Conclusion

The JPEG lossy compression scheme is one of the most popular and versatile compression schemes in widespread use. It's ability to attain considerable size reductions with minimal visual impact with relative light computational requirements and the ability to fine tune the compression level to suit the image at hand has made it the standard for continuous-tone still images. It has also been extended to work on moving pictures in the MPEG (motion jpeg) standards that are beginning to play a vital role in the online distribution of film. More research is also being done to incorporate wavelet technology into the standard as well. The JPEG standard has proven a versatile and effective tool in the compression of data.

References

[1] Dave Arnold *Fall 2001 Linear Algebra Class*

[2] The JPEG Picture Compression Standard *IEEE Standards*

[3] JPEG - JFIF *http://www.jpeg.org*

[4] Lossy Graphics Compression *Image Compression Methods*

Introduction

JPEG Compression

Example Images