An introduction to Image Compression

A JPEG Tutorial

Simon Lewis 2004

School of Engineering and Applied Science
Aston University
Aston Triangle
Birmingham
B4 7ET

Contents

1 Introduction

With the advent of peer to peer distribution systems, the need to compress multimedia data is obvious. By compressing data, the transmission time is reduced. This represents a saving in both time and money.

This tutorial will look into the fundamental principles of JPEG image compression. JPEG is of significance because it contains many different compression techniques. JPEG is related to a whole family of multimedia compression schemes, as is shares the same mathematical transform.

As you work through the exercises you will gain an understanding of the different tricks that are used. The accompanying software will provide visual confirmation of the techniques described.

After completing this tutorial, you will have gained the skills necessary, to enable you to discuss and progress onto more complex forms of compression.

2 Digital Image Representation

2.1 Raster and Vector Image formats

In a computer system an image may be stored in many ways. The two most common ways are as vector or raster files. A vector file format is more suitable for 'drawings' whereas Raster (bitmaps) are more suitable for storing photographic images.

A vector file contains a sequence of drawing operations, for example consider the following:

```
Rectangle((0,0),20,40,black)  //A black rectangle
                              // at point (0,0)
                              // width 20, height 40
```

A complex image would require many instructions like this, to construct an image.

A bitmap file consists of a 2 dimensional array of pixels. A pixel is the smallest atomic unit within a raster image.

$$\begin{array}{c|cccc|} & 0 & n_2 & \ldots & N_2-1 \\ \hline 0 & \ldots & \ldots & \ldots & \ldots \\ n_1 & \ldots & x[n_1, n_2] & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ N_1-1 & \ldots & \ldots & \ldots & \ldots \end{array} \tag{1}$$

Depending on whether or not the image is colour, each pixel will be made up of several components. This will be considered further when colour space is considered.

Depending on the image, each pixel will require a certain number of bits to encode. For a true colour image each pixel will be encoded with 24 bits per pixel (8 bits per component).

2.2 Colour Space

An image is made up of a number of components. These may be considered one or more, 2 dimensional matrices placed one behind the other. A black and white image must have one component and a colour image must have at least 3. Each pixel is made up of parts from each component. It is possible to have components of different sizes to create an image. JPEG uses this technique to reduce the size of an encoded image[1].

A colour space describes the way an images colour is made up. It defines what data each component specifies. Some common colour spaces are defined below:

Gray-scale This is what we perceive as 'black and white'. There is only one component. This component specifies the luminance of the image.

RGB This colour space has 3 components. The R component specifies the 'redness' of a pixel. The G component specifies the 'green-ness' of a pixel. The B component specifies the 'blueness' of a pixel. This colour space is commonly used in computer displays. The value of the components, defines how much energy is supplied to the 3 electron guns in the display. Each of the components contributes equally to the information in the image

YCrCb This colour space is used by television sets. The Y component specifies the Luminance of the image (grayness).The Cr component specifies the red chrominance of the image (red colour). The Cb component specifies the blue chrominance of the image (blue colour).

CYMK The abbreviations stand for Cyan, Magenta, Yellow and Black. This format is typically used in printers.

2.3 Image Format Quality

With vector images the quality of a image is implementation specific. As the vector image is described in terms of drawing instructions. The detail of the image is related to the number of instructions.

With raster images the quality of an image is related to the pixel density. If not enough pixels are used the image will seem 'blocky'.

When images are compressed, they may use one of two types of schemes, lossy or lossless. In a lossless scheme the image that is reproduced from a file is exactly the same as the original image [2]. In a lossy scheme, information that is perceived

[1]The process of reducing a components size is called sub-sampling.
[2]Allowing for truncation errors.

as redundant is discarded. This discarded information introduces a distortion in the reproduced image [5].

JPEG is an example of a primarily Lossy scheme [3]. It was designed for the compression of digital photographs. With a lossy scheme compression ratio's of 40:1 may be achieved. This amount of compression is not possible with a Lossless scheme. As you will see in later sections, there is a tradeoff between image quality and file size. To make the file smaller you must discard data. If too much data is discarded then the image distortions becomes unbearable and the image is no longer useful.

From this point on we will only be considering Raster images. Raster images are required for the image processing applications that we will be considering.

2.4 Measuring Distortion

The distortion of an image may be described quantitatively or qualitatively. When the original raster image pixel data is available, it is possible to calculate the difference between the encoded and original data. The following formulae may be used when describing the numerical quality of images:

$$\mathrm{MSE} \triangleq \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} (x[n_1, n_2] - \hat{x}[n_1, n_2])^2 \tag{2}$$

In this definition $\hat{x}$ is the value of the encoded pixel at $[n_1, n_2]$. The number of pixels is $N_1 N_2$.

Using this definition, the Peak Signal to Noise Ratio (PSNR) may be defined as:

$$\mathrm{PSNR} \triangleq 10 \log_{10} \frac{(2^B - 1)^2}{\mathrm{MSE}} \tag{3}$$

Analysis of this formula shows that the quality of an image can be related to the square of the number of quantisation levels.

Taubman [5] says that good reconstructed images have PSNR of 30dB or more. Its is widely known that calculated errors, show little resemblance to the perceived image quality. Because of this a subjective scale of image quality will be used

Kevin Jones [2] describes a subjective quality scale:

[3] As JPEG has a number of modes, including the new lossless JPEG-LS

Excellent An image of extremely high quality, as good as could be desired.

Fine An image of high quality, errors are not objectionable.

Passable An image of acceptable quality, errors are not objectionable.

Marginal An image of poor quality, errors are somewhat objectionable.

Inferior A very poor quality image, subject still recognisable.

Unusable Image cannot be recognised.

This scale will be used, when rating the quality of the images encoded with the accompanying software.

2.5 Questions

I) If a pixel is encoded with 24 bit precision (3 components), how many different colours may be represented in the image?

II) Why is the CYMK colour space used in printers?

III) Compare and Contrast the issues that are involved in increasing the image size of both Vector and Raster file formats.

IV) An image consists of four pixels. The original data and the encoded data are shown below. The original picture has pixel values in the range 0..100. These have been encoded with 2 bits per pixel. **Calculate the MSE of the image.**

$$\begin{aligned}\textbf{Original} &= \begin{bmatrix} 13 & 67 \\ 1 & 99 \end{bmatrix} \\ \textbf{Encoded} &= \begin{bmatrix} 00 & 10 \\ 00 & 11 \end{bmatrix}\end{aligned} \tag{4}$$

What are the quality implications of your answer? How could the error be reduced?

3 Compression Techniques

3.1 Overview

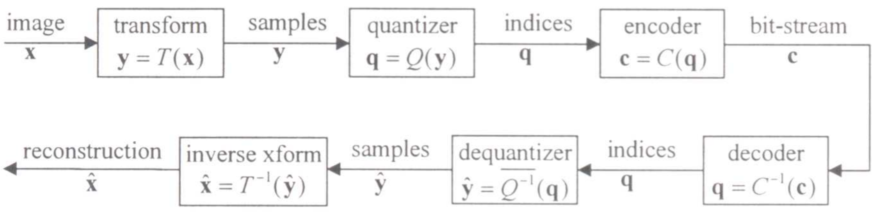


Figure 1: Structured Compression Overview
[5, *from Taubman,JPEG 2000,pg14*]

As you can see in Figure 1 the compression–decompression cycle should fit into six operations. Three operations with their corresponding inverses.

$\hat{x}$ corresponds to the reproduced image. It is distinguished from the input image x, because in a lossy scheme the two will be different.

For and image to be readily compressed the image must exhibit patterns and redundancy. Patterns in the data allow an image to be simplified by exploiting repetition. Redundancy allows an image to be reduced by discarding information considered irrelevant.

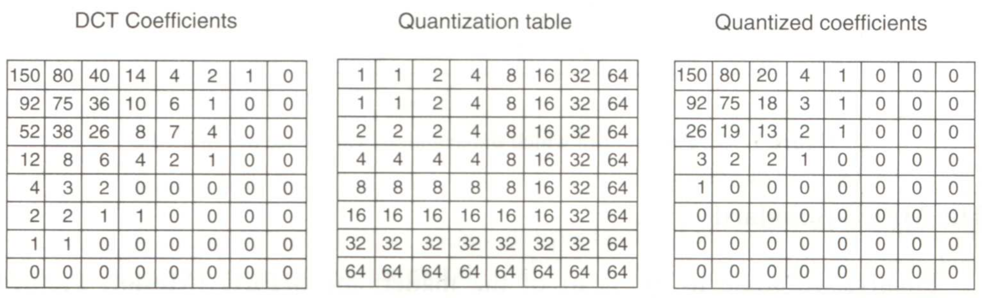
The transform step is used to convert one set of numbers to another set of numbers. For this reason it is also sometimes called a *Mapper*. The transform operation, may also transfer the data from one domain to another (hence transform). In JPEG the pixels are linearly mapped and then transformed to the frequency domain. This may seem a little complicated, but thats because it is. The transform reduces the amount of disorder in the image and makes it more efficient to encode.

The quantiser is the part of the process that introduces loss into the encoding system[4].

Figure 2 shows the output of an image transform[5]. Each of the numbers from the transform is divided by its corresponding coefficient from the *Quantisation Table*, this is then rounded to the nearest integer. This leaves a matrix of the same size

[4]Assuming infinite precision

[5]In this example the Discrete Cosine Transform (DCT), this is the transform used in the JPEG standard (more on this later!)



DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
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

Figure 2: Quantisation Overview
[6, *from Tanenbaum,C.N.,pg699*]

containing *Quantised Coefficients*. As you can see in Figure 2 the quantised coefficients show a reduced amount of variation. For efficient encoding their should be large runs of similar numbers (the reason for this will become clear).

3.2 Run Length Encoding

This technique is used to compress data with large runs of the same value.

Consider a sequence of bytes that needs encoding:

$$S = 22_{16}\ 22_{16}\ 22_{16}\ 22_{16}\ 44_{16}\ 44_{16}\ 77_{16}$$
$$||S|| = 7 \text{ bytes}$$

In run length encoding, the number of equal consecutive values, followed by the value is given. For example:

$$08_{16}00_{16}$$

expands to:

$$00_{16}00_{16}00_{16}00_{16}00_{16}00_{16}00_{16}00_{16}$$

Using this terminology S is encoded into the string:

$$\begin{aligned} RLE_s &= 04_{16}22_{16}02_{16}44_{16}01_{16}77_{16} \\ ||RLE_s|| &= 6 \text{ bytes} \end{aligned}$$

In this case there is only a saving of 1 byte. For larger runs there would be considerable saving. Notice that if no runs exist, the encoded size will be twice as big as the original sequence.

3.3 Huffman Encoding

Huffman encoding is a form of variable length coding. It exploits the statistical patterns in a data set to produce an *optimum* code. Any fixed size natural alphabet, will have a biased frequency usage. Consider the English language, E is the most common letter. In the traditional approach each letter would be encoded with a fixed length code. An example of this type of encoding is the ASCII character set. In the ASCII set each symbol is assigned an 8 bit code[6].

If the symbols with higher frequency usage, are encoded with the shortest code words, an optimum coding scheme is produced. In Huffman, no codeword is allowed to prefix a longer code. This requirement will be better understood with an example.

Consider a fixed length alphabet with the following symbols and usage:

Table 1: An example alphabet

Symbol	Frequency
A	$\frac{2}{16}$
B	$\frac{2}{16}$
C	$\frac{3}{16}$
D	$\frac{9}{16}$

The Huffman algorithm may be summarised by repeating the following steps. The pool is the collection of symbols with their associated probabilities:

[6]The letters are encoded using 7 bits.

1. Locate the two symbols or tree nodes with the lowest frequencies and remove them from the pool.If more than one item has the lowest frequency, an arbitrary choice can be made.
2. Create a new tree node and make the items in the previous step its two branches.
3. Make the frequency of the new tree node the sum of the frequencies of the children.
4. Add the new node to the pool.

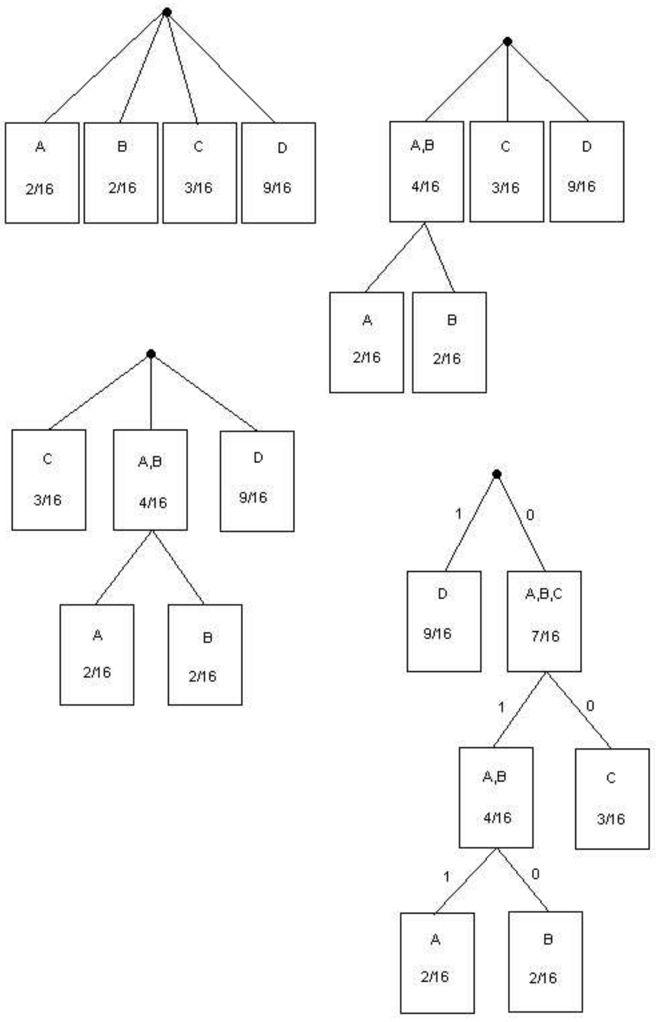


Figure 3: Huffman Example

The variable length code word for each symbol, is obtained by tracing down the binary tree, noting the value on the branches. Branches in one direction are labelled with a '1', the other direction is labelled with a '0'.

D = 1

C = 00

B = 010

A = 011

As you can see from the above definition no code prefixes any other code. Consider the decoding of the example bitstream.

$$\begin{aligned}&\ldots 0100111011010010011\\&= \text{B A D A B B A}\end{aligned}$$

3.4 Questions

I) As Huffman encoding relies on predetermined statistics, what are the performance issues when considering a computer based system?

II) What happens if the statistics of an image change, without the statistics being recalculated?

4 Image Transforms

4.1 The 2 Dimensional DCT

The purpose of an image transform is to convert the data so that is more readily compressible. In JPEG the transform that is used is the Discrete Cosine Transform (DCT). As you will see this transform has a lot of similarities to the Discrete Fourier Transform. Before we consider the mathematics let us first look at the effect of the transform.

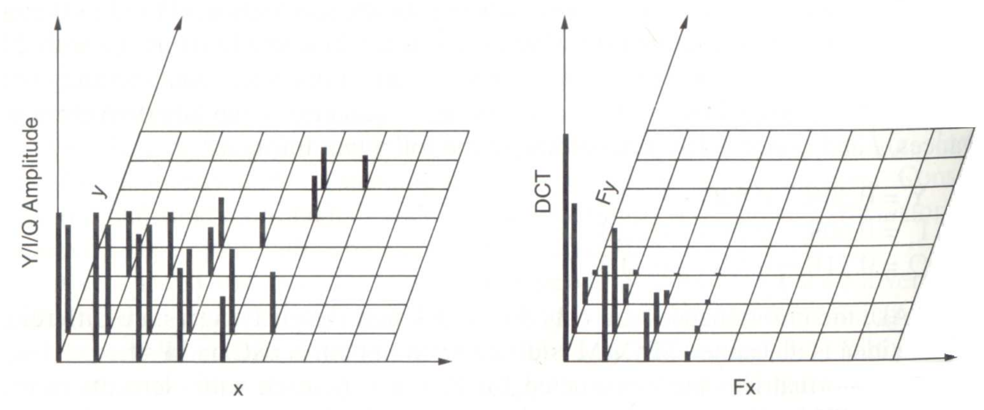


Figure 4: Domain Transform
[6, *from Tanenbaum,C.N.,pg698*]

Consider the left most picture in Figure 4. It represent the intensities of different component pixel values. The amplitude axis is labelled with the YCbCr colour space notation[7].The data represented by the left hand picture is in terms of intensity levels. The DCT has the effect of converting from the spacial domain (intensity levels), to the frequency domain. What the right hand picture in Figure 4 shows, is the coefficients of increasing frequency cosine waves[8].

What is achieve by this transform is that the data is concentrated. The image data is exactly the same, it is just represented in a different way. In the new representation the data has less entropy.

This transform is a *Block Transform*. It works on blocks of pixels. In JPEG each

[7]It is important to note that the JPEG specification is colourblind, it make no reference to a particular colour model. The JFIF file format insists you use the YCbCr colour space, hence the notation.

[8]Don't Panic, Don't Panic.

component is broken up into 8 by 8 blocks of pixels. As you will have guessed 8 was chosen because of its numerical computability.

Here comes the science bit:

As shown in Figure 1 the transform must be reversible. The DCT is no different. The forward transform is referred to as the FDCT and the inverse the IDCT. The 2-dimensional definitions are as follows [4]:

$$\text{FDCT} \triangleq T[i,j] = c(i,j) \sum_{x=o}^{N-1} \sum_{y=o}^{N-1} V[y,x] \cos\left(\frac{(2y+1)i\pi}{2N}\right) \cos\left(\frac{(2x+1)j\pi}{2N}\right) \quad (5)$$

Where

$$c(i,j) = \frac{2}{N}, \; i \text{ and } j \neq 0 \quad (6a)$$

$$c(i,j) = \frac{1}{N}, \; i \text{ and } j = 0 \quad (6b)$$

$$\text{IDCT} \triangleq V[y,x] = \sum_{i=o}^{N-1} \sum_{j=o}^{N-1} c(i,j) T[i,j] \cos\left(\frac{(2y+1)i\pi}{2N}\right) \cos\left(\frac{(2x+1)j\pi}{2N}\right) \quad (7)$$

The reason for the $c(i,j)$ term, is to ensure that the transform is orthonormal. This means that the magnitude of the transform is always 1 ($\|T_{i,j}\| = 1$)[5]. Orthonormal also means that the transform is orthogonal.

In JPEG the value of 128 is subtracted from all pixel values to bring them into the range -128 to 127 [9]. This ensures that the numbers are signed. When the DCT transform is applied it ensures that the cosine coefficients are of similar size. This ensure that the DC coefficient is as small as possible.

From Equation 5, you can see that what is produced are coefficients of increasing frequency. The first coefficient resulting from this transform is special, it is related directly to the average value of the block; because of this it is called the *DC Coefficient*. All the other coefficients are termed *AC coefficients*. The AC coefficients add or subtract pixels values at different locations in the image. The effect of the AC coefficients is to add detail to the image.

[9]This is assuming that 8 bits are used to encode each component of a pixel.

As mentioned, when using quantisation, many of the AC coefficients are quantised to zero. This quantisation has the effect of increasing compression but reducing the fidelity of the image (as the detail *cannot* be added).

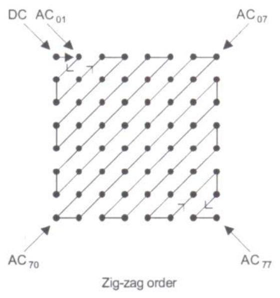


Figure 5: ZigZag Encoding Order
[1, *from T.81, pg 16*]

Once the transform on an 8 by 8 block is completed, you are left with an 8 by 8 block of DCT coefficients. Figure 5 Shows the position of the DC coefficient and the 63 AC coefficients. Because the higher frequency coefficients are towards the bottom right of the coefficient block, the coefficients are encoded in a zig–zag order. When the DCT matrix has been quantised, this ordering makes compression by run length encoding very efficient.

4.2 A 1 Dimensional DCT example

To consolidate your understanding of the DCT process, this section gives numerical examples of the 1D-DCT.

The 1 Dimensional FDCT and IDCT are given by the following formulae:

$$\text{FDCT} \triangleq T[i] = c(i) \sum_{n=o}^{N-1} V[n] \cos\left(\frac{(2n+1)i\pi}{2N}\right) \tag{8}$$

Where

$$c(i) = \sqrt{\frac{2}{N}},\ i \neq 0 \tag{9a}$$

$$c(0) = \sqrt{\frac{1}{N}} \tag{9b}$$

$$\text{IDCT} \triangleq V[i] = \sum_{n=o}^{N-1} \underbrace{c(n)T[n]}_{Amplitude} \cos\left(\frac{(2i+1)n\pi}{2N}\right) \tag{10}$$

It is clear from the formula that the IDCT is simply summing up sinusoids of different amplitudes.

4.2.1 Example 1

Consider a 1D image, A row vector of pixels. Each pixel is encoded with 8 bits per sample. The maximum value of each sample is 255.

$$X[n] = \begin{bmatrix} 190 & 184 & 186 & 182 & 167 & 123 & 63 & 38 \end{bmatrix}$$

These pixel values are then mapped to the range -128..127 by subtracting 128 from the original values:

$$X'[n] = \begin{bmatrix} 62 & 56 & 58 & 54 & 39 & -5 & -65 & -90 \end{bmatrix}$$

We will now calculate the DC coefficient by using Equation 8.

$$\begin{aligned} N &= 8 \\ i &= 0 \end{aligned}$$

$\therefore$ Equation 8 becomes:

$$\begin{aligned} T[0] &= \sqrt{\frac{1}{8}} \sum_{n=o}^{7} V[n] \cos(0) \\ &= \sqrt{\frac{1}{8}} \sum_{n=o}^{7} V[n] \\ T[0] &= \sqrt{\frac{1}{8}} (62 + 56 + 58 + 54 + 39 + (-5) + (-65) + (-90)) \\ &= 38.5 \end{aligned}$$

The DC coefficient is simple enough to evaluate. To calculate the other AC coefficients is a little tedious. To calculate the values the following C++ code may be used [4]:

```
double c = 1.0 / sqrt(n);    // To hold the scaling factor
unsigned int N;              // To hold the sample size (8 in JPEG)
double input[];              // To hold the pixel input values
double output[];             // To hold the Output coefficients

int main()
{
    for(unsigned int i = 0; i < N; i++)
    {
        output[i] = 0;
        for(unsigned int j = 0; j < N; j++)
        {
            output[i] += c *(input[j])
                           *cos ((2*j+1)*i*PI/2.0/N);
        }
        c = sqrt(2.0/N);
    }
    return 0;
}
```

4.2.2 Example 2

Consider an image with the following properties:

$$\begin{array}{c} \text{n} \\ \text{Input} \\ \text{DCT} \end{array} \left[\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 128 & \mathbf{88} & 40 & 0 & 0 & 40 & 88 & 128 \\ 181.0 & 0.0 & 136.6 & 0.0 & 0.0 & 0.0 & 4.6 & 0.0 \end{array}\right]$$

Using Equation 10 we will now calculate the pixel value of $X[1]$ using the IDCT.

$$
\begin{aligned}
= 181.0\sqrt{\frac{1}{8}}\ \cos(0) + 0.0\sqrt{\frac{2}{8}}\ \cos\left(\frac{3\pi}{16}\right) + 136.6\sqrt{\frac{2}{8}}\ \cos\left(\frac{6\pi}{16}\right) \\
+ 0.0\sqrt{\frac{2}{8}}\ \cos\left(\frac{9\pi}{16}\right) + 0.0\sqrt{\frac{2}{8}}\ \cos\left(\frac{12\pi}{16}\right) \\
+ 0.0\sqrt{\frac{2}{8}}\ \cos\left(\frac{15\pi}{16}\right) + 4.6\sqrt{\frac{2}{8}}\ \cos\left(\frac{18\pi}{16}\right) + 0.0\sqrt{\frac{2}{8}}\ \cos\left(\frac{21\pi}{16}\right)
\end{aligned}
$$

$$
\begin{aligned}
&= 64.0 + 0.0 + 26.1 + 0.0 + 0.0 - 2.1 + 0.0 \\
&= 88.0 \text{ (As Required)}
\end{aligned}
$$

As the cosine terms are not dependent on any variable, these values may be gathered from a 'lookup table'.

4.3 Questions

I) Given:

$$X[n] = \begin{bmatrix} 10 & 52 & 69 & 0 & 100 & 85 & 99 & 56 \end{bmatrix}$$

Calculate the DC coefficient value of the image.

II) After the transform has been applied, what will the effect of quantising the DC coefficient to zero be?[10]

III) What would happen if the transform was completed with a block size of 10 rather than 8? What are the reasons to avoid using a value of N that is not a power of 2?

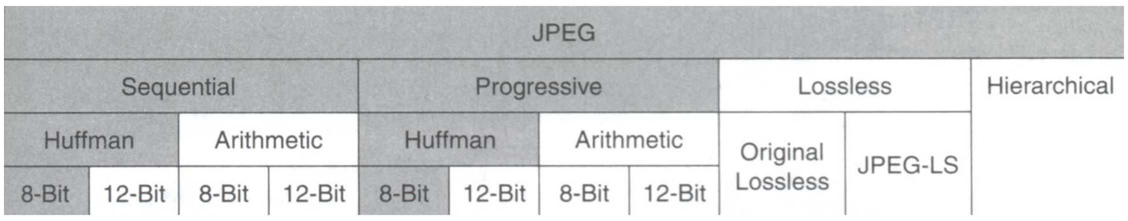
[10]This can be done using the software provided

5 JPEG Compression

5.1 Standards and Definitions

The JPEG standard was defined in T.81 [1]. It stands for "Joint Photographic Experts Group". The 'Joint' comes from the fact that it was developed by two standards authorities; the ISO/IEC working group

There are four types of JPEG compression they are summarised below.



JPEG										
Sequential				Progressive				Lossless		Hierarchical
Huffman		Arithmetic		Huffman		Arithmetic		Original Lossless	JPEG-LS	
8-Bit	12-Bit	8-Bit	12-Bit	8-Bit	12-Bit	8-Bit	12-Bit			

Figure 6: JPEG Types
[4, *from Miano,C.I.F.F.,pg37*]

As you move to the right in Figure 6 the modes of JPEG get more and more complicated. In this tutorial we will only consider the most basic type: JPEG sequential mode with 8 bit samples, using Huffman encoding.

5.2 Overview

So far have considered the prerequisites to JPEG image compression. In this section we shall look at the specifics of the JPEG compression technique. JPEG uses the following 'dirty tricks' to reduce the file size:

Component Sub-sampling The components of the image are sampled at different amounts. This has the effect of reducing the size of some components. This can save half the file space straight away.

Domain Transformation The image is transformed into the frequency domain using the DCT. This enables data to be interpreted as 'redundant' increasing the opportunity for compression.

Quantisation The DCT blocks are divided by a quantisation table. This has the effect of deleting the higher order coefficients of the DCT. This is the lossy step in JPEG. It reduces the quality enabling higher compression.

Huffman Encoding The remaining DCT coefficients are encoded with a special form of Huffman encoding. The data is then written to the file. We will assume that the file format in use is the JFIF file format[11].

This may be summarised by the following:

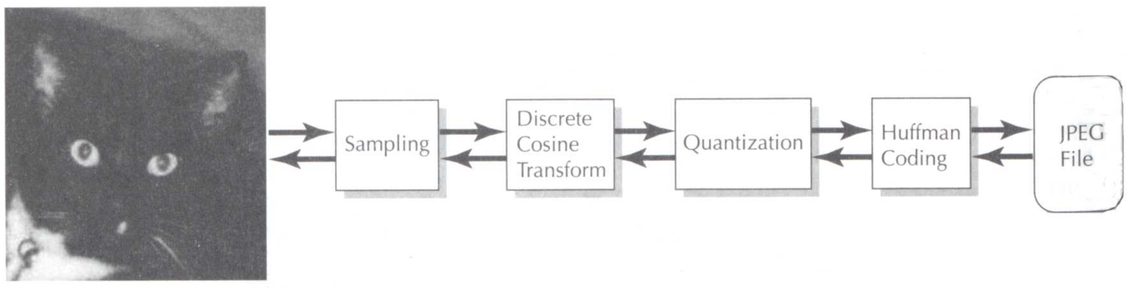


Figure 7: The steps of creating a JPEG file
[4, *from Miano,C.I.F.F.,pg44*]

The overview of what happens to each block is described beautifully in Figure 8.

As you can see the AC and DC components are encoded separately. The DC components under go differential encoding, where as the AC coefficients are encoded using run length encoding.

5.3 Sub-sampling and Scans

When all components are the same size, and the image is a multiple of the block size (8×8) life is simple. When this is not the case things become a little more difficult.

5.3.1 Sub-Sampling

In the JFIF[12] format the image is required to be in the YCbCr format. The Human Visual System (HVS) is not as sensitive to colour as it is to black and white. Because of this you can sub-sample the chrominance components. This has the effect of reducing the quality, of the colour in the image. The most common way to reduce the

[11]This in most cases will be true. There was an official file format released after JFIF but this is not as popular.
[12]JFIF = JPEG File Interchange Format

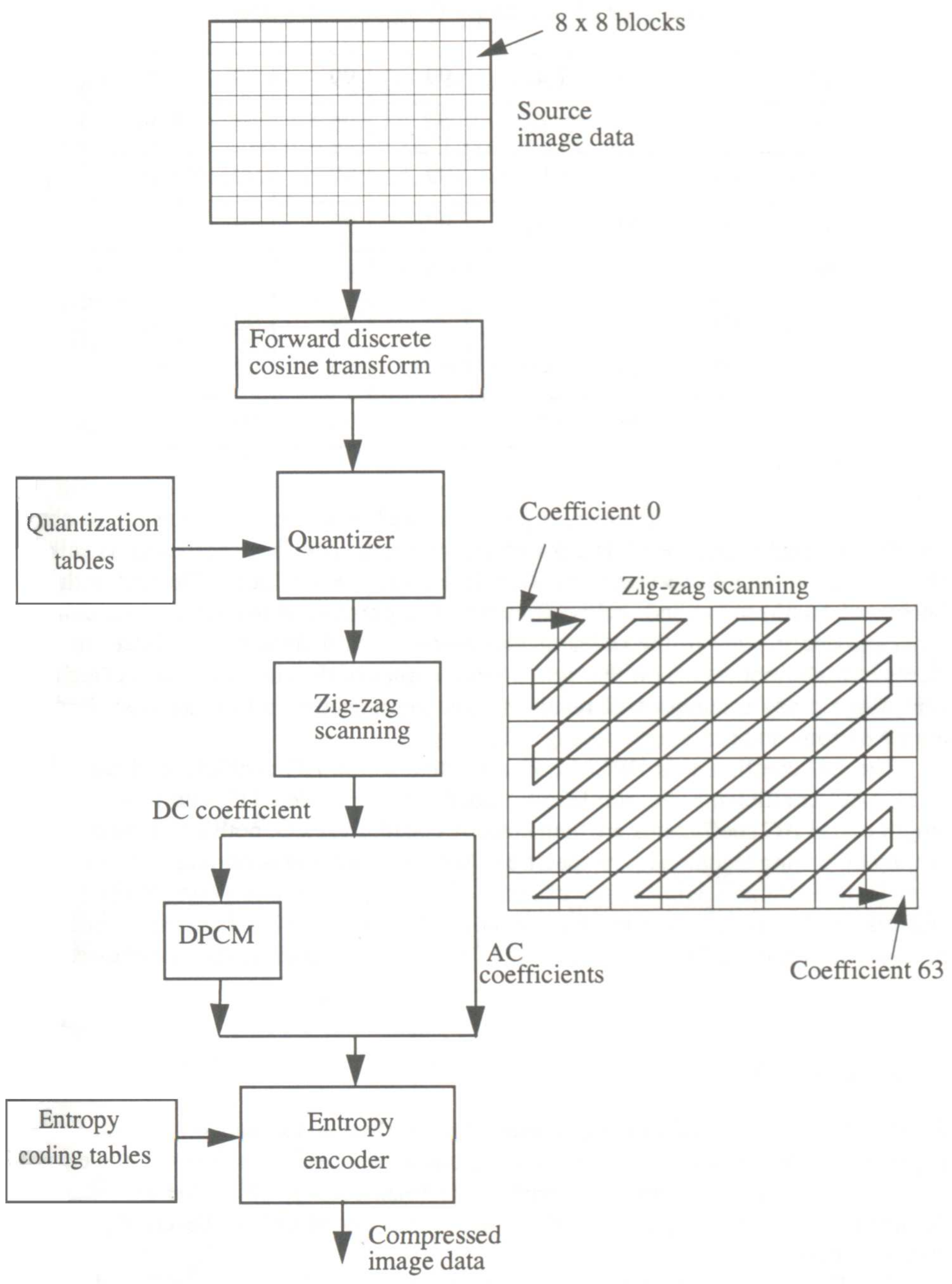


Figure 8: Block processing Overview
[3, *from Guojun Lu, C.C.D.M.S.,pg87*]

size of the Cb and Cr components is to half them in both directions. Obviously sub-sampling causes a loss of information. When the image is decoded from the file at the other end of the transmission system, the image must be Up-sampled, before it can be displayed. This involves increasing the resolution of the sub-sampled components. Up-sampling causes blocking effects, the intermediate values are interpolated. In order to reduce blocking effects filtering techniques may be used.

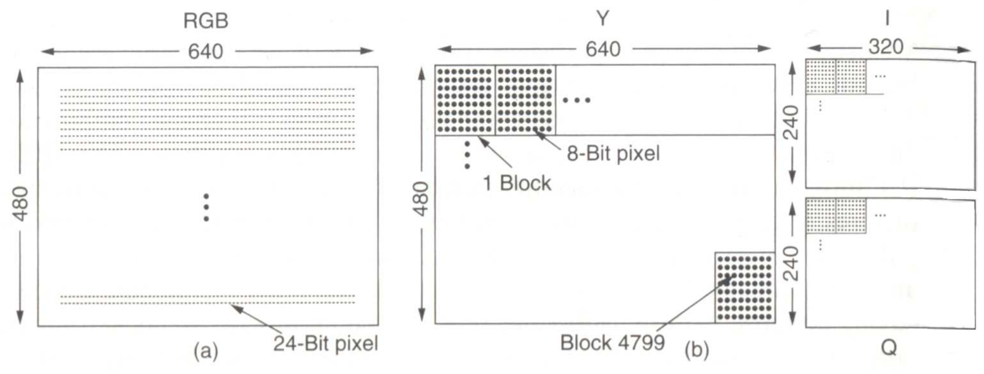


Figure 9: Sub-sampling Example
[6, *from Tanenbaum,pg698*]

Figure 9 shows the effect of sub-sampling the image. The chrominance components have been sub-sampled by a factor of two in both directions. The overall effect of this technique is to reduce the overall size of the image to half the original.

If an image is not a multiple of the block size then padding is added to the image to make up it's size. The image should be padded with the same pixel values as the outer pixels of the image.

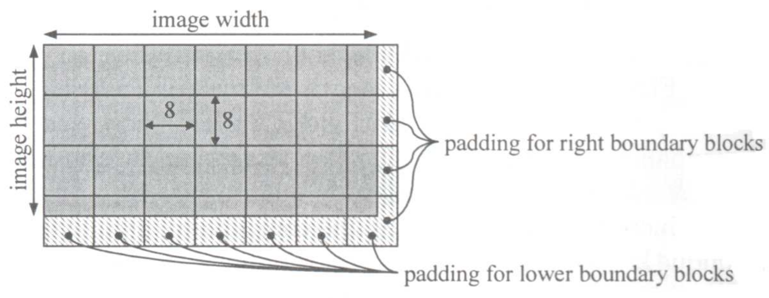


image width
image height
8
8
padding for right boundary blocks
padding for lower boundary blocks

Figure 10: Padding an Image
[5, *from Taubman,pg722*]

The JPEG standard defines a number of formulae to describe the sampling ratios.

$$x_i = \left\lceil X \times \frac{H_i}{H_{max}} \right\rceil \text{ and } y_i = \left\lceil Y \times \frac{V_i}{V_{max}} \right\rceil$$

Where

x_i is the number of columns

y_i is the number of rows

H_i and V_i are the componet sampling factors

H_{max} and V_{max} Are the maximum sampling factors of the image

5.3.2 Sampling Example

This example comes directly from the standard. An image has the following sampling factors:

Component	Horizontal Frequency	Vertical Frequency
Y	4	1
Cb	2	2
Cr	1	1

An image with a maximum resolution of 512×512 pixels, will have components of resolution:

Component	Horizontal Resolution	Vertical Resolution
Y	512	256
Cb	256	512
Cr	128	256

5.3.3 Image Scans

When an image has sub-sampled components the process of encoding the data is somewhat complicated.

In the usual case each component is encoded in a single scan. In this form each component is read from top to bottom and stored in a ‘scan’ in the file. A file would then contain the data of the 3 components sequentially in the file. In this mode a file would contain 3 scans, one for each of the components.

In an interleaved scan, data from each component is interleaved. When all the components are the same size, one block is taken from the first component, one from the second, and then one from the third. This continues until all the data units have been read.

When some of the components have been sub-sampled, the ordering of components is a little complicated. A Minimum Coded Unit (MCU) is the number of blocks, from different components that are encoded together. The JPEG standard has set the maximum MCU size to 10 (8 × 8) blocks.

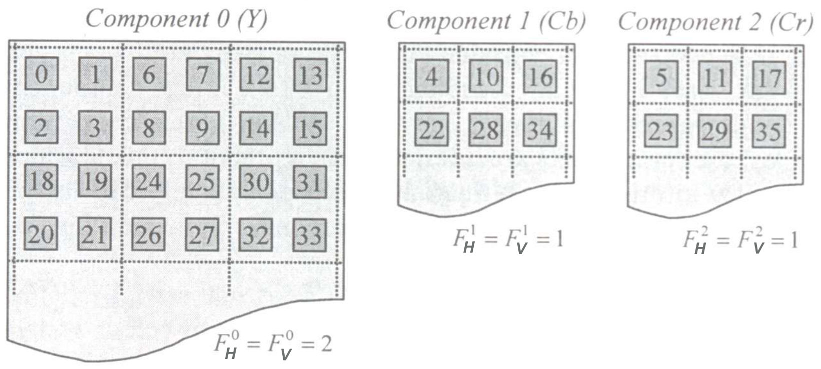


Figure 11: MCU's
[5, *from Taubman,pg730*]

Figure 11 shows a typical YCbCr image. The chrominance components have been sub sampled by a factor of two in both directions. The MCU size is 6 blocks. The encoding order is shown in the figure. The sampling ratios, that were mentioned in the previous section are labelled.

5.4 Quantisation Exercise

This section makes good use of the software written by the author. You should be familiar with the concept of quantisation from the previous sections. Load up the software and open a bitmap image.

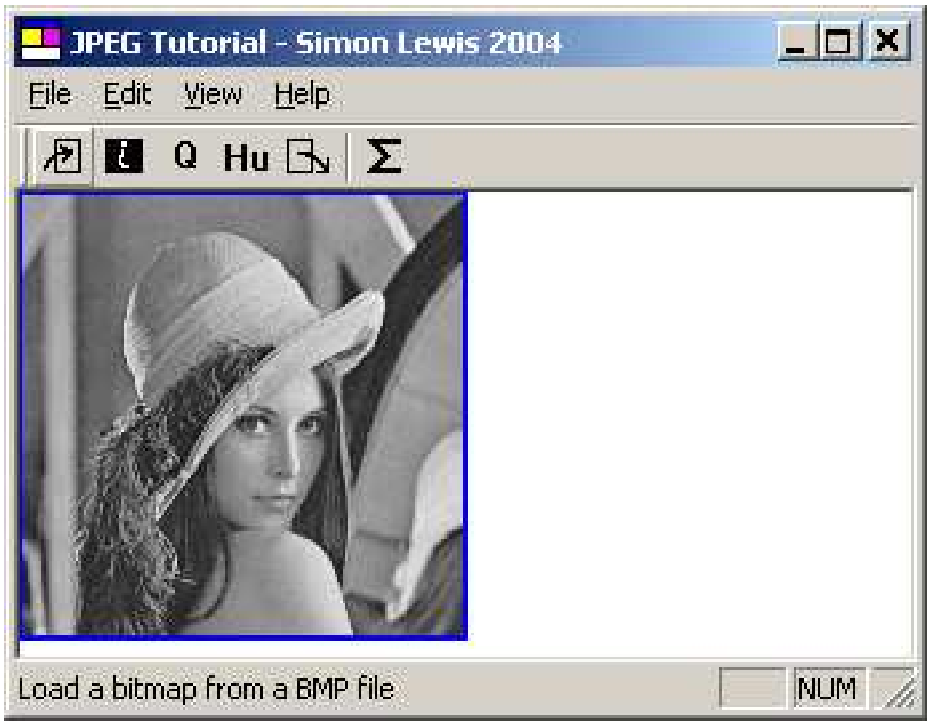

JPEG Tutorial - Simon Lewis 2004
File
Edit
View
Help
Q
Hu
Σ
Load a bitmap from a BMP file
NUM

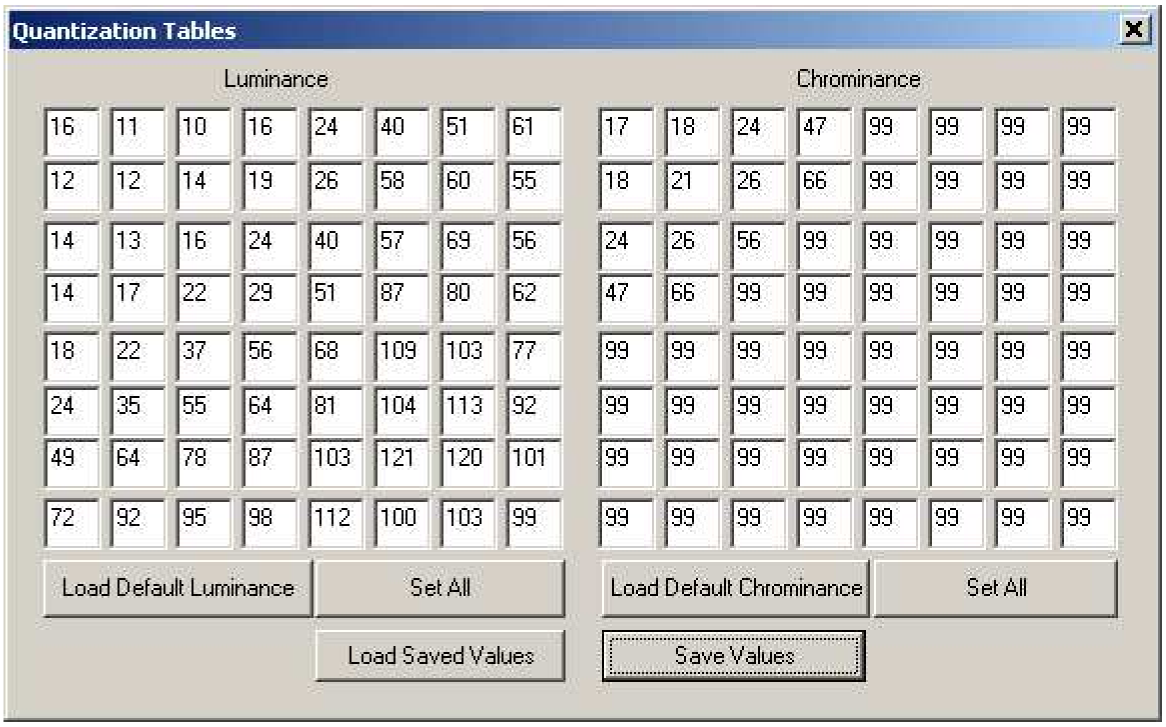
Figure 12: Load in a bitmap

Set the quantisation coefficients. This is achieved by clicking the ‘Q’ button on the toolbar.

By clicking the default button you can use the quantisation coefficients that are specified as ‘good’ in the JPEG specification. Try experimenting with different quantisation values. By clicking save, and then clicking the ‘Write to File‘ button in the toolbar; the encoded and original can be observed side by side.

The JPEG picture that is output, is in the same directory as the original. Locate the original and estimate the amount of compression.

Try changing the quantisation values, you can change the way the image looks. Try investigating different values. Comment on the results. The JPEG files content can be inspected by using the program JPEGdump [4] on the output file.



Luminance

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

Chrominance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Figure 13: Set the Quantisation Coefficients

5.4.1 Questions

I) What are the effects of quantisation when the quantisation values are large?

II) Name some circumstances when picture quality loss would not be allowed.

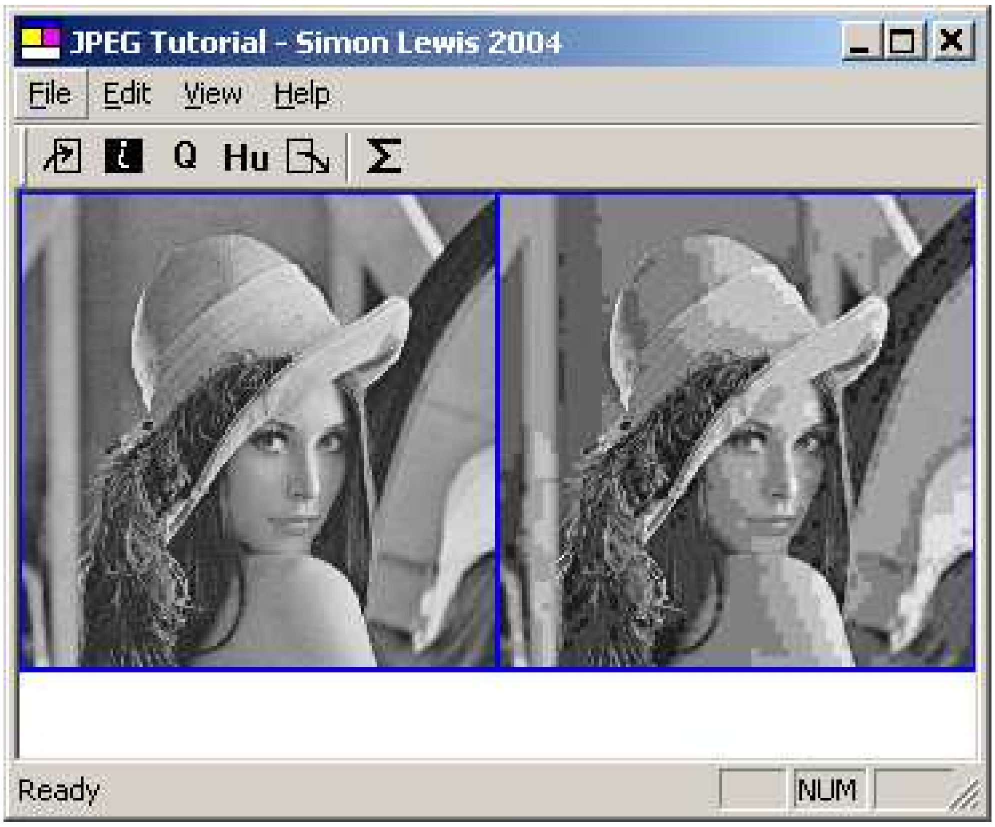


Figure 14: An example of distorted output

5.5 Huffman Encoding

The Huffman encoding in JPEG sufferers from a large amount of indirection. As shown in Figure 8 the AC and DC components are processed differently. The assumptions made about the type of image affect the way it is encoded. In JPEG it is assumed that the image is continuous, this would make the DC difference between pixels very small (or zero). After the AC coefficients have been quantised, there will be large runs of 0's introduced in the zig-zag path.

With the DC coefficients the difference between the last DC component and the new DC coefficient is stored. This difference is encoded into a 'byte code'. The byte code tells the decoder how many raw bits to read from the file stream. The bits read in tell us the DC difference. The byte codes are subject to Huffman encoding. The byte code's for the DC coefficients are stored in the DC Huffman table. This will become clear when the reader consults the example at the end of section.

The AC coefficients are stored a little differently. The values are still encoded into 'byte codes'. The first 4 bits of the code say how many leading zeros their are before the next value. The last 4 bits of the byte code tells the decoder how many raw bits to read from the bit stream. There are two special reserved 'byte code's when decoding the AC coefficients. 00_{16} tells us that all the rest of the coefficients in the block are zero. $F0_{16}$ tells us that there are 16 zero AC coefficients next.

5.5.1 Example

Consider the block:

$$\begin{bmatrix} 10 & -6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Table 2: DC Coefficient Encoding

Byte Code	Raw Bits	Value Of Bits	Last DC	New DC
04_{16}	1010_2	10	0	10

Table 3: AC Coefficient Encoding

Byte Code	Zero Run	Raw Bits	Coefficient Value
03_{16}	0	010_2	-6
04_{16}	0	1001_2	9
$E3_{16}$	14	100_2	8
$F0_{16}$	16	NONE	NONE
$F0_{16}$	16	NONE	NONE
$C1_{16}$	12	1_2	1
01_{16}	0	0_2	-1

For the AC coefficients, when the raw bits are read, the first bit tells the decoder the sign. If the first bit is zero it means that the coefficient is negative. It is still counted as a logical one, in terms of its value.

$$010_2 \equiv -|110_2|$$

5.5.2 Questions

I) Why is the Huffman code length in JPEG limited to 16?

II) Encode the following block. Assume the DC coefficient was 3 In the previous block.

$$\begin{bmatrix} 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

5.6 JPEG Questions

I) As JPEG is such a complicated standard, why is it in common use today?

II) Why is there a JFIF file format in existence?

III) The quantisation tables used are usually the standard values. What problems might you have if the image you are compressing does not conform to the JPEG assumptions?

6 Conclusions

You should now have a greater understanding of compression techniques, in particular the main features of the JPEG technique.

Although not comprehensive, this tutorial has tried to fully explain the majority of aspects in the JPEG standard.

The interested reader should consult the many good resources mentioned in the reference section for more information.

Simon Lewis
24/04/04

References

[1] CCITT. Information technology - digital compression of continuous tone still images. *T.81*, 1993.

[2] Kevin Jones. Image compression demonstration. Master's thesis, Aston University, 1994.

[3] Guojun Lu. *Communication and Computing for Distributed Multimedia Systems.* Artech House Publishers, 1st edition, 1996.

[4] John Miano. *Compressed Image File Formats.* Addison Wesley, 1st edition, 1999.

[5] David S. Taubman and Michael W. Marcellin. *JPEG2000 - Image Compression Fundamentals, Standards and Practices.* Kluwer Academic Publishers, 1st edition, 2002.

[6] Andrew S. Tanenbaum. *Computer Networks.* Prentice Hall, 4th edition, 2003.