**Chapter 1. Introduction to Still Image Compression.**

**1.1 Introduction**

Before images can be compressed effectively it is important to understand their properties. In this introduction the types of process that are applied to still images will be explained and examined. There are a number of processes that are always applied to an image before still image compression occurs. There are other processes that are used as part of the image compression.

The existing classes of image coder will then be discussed, highlighting the areas which have been explored in detail in this work. The methods applied in this work are generally useful for still image compression but some could be used in video systems, although this is not explored.

**1.2 Image Capture**

Generally in the field of image compression the original image is considered to be a digitally stored version of the original. This stored image is used as a lossless baseline for any process that is applied to the image. However the lossless digital image is usually taken from a real scene and so some other process must have occurred to digitise the image from the analogue original. Although this work does not deal with the problems caused by real systems (such as camera jitter, unfocused images, scratched lens etc.) it is important to understand the effects of cameras and digitisers.

In an ideal system a simple pin hole camera would be used to produce an image, since any object in the viewed environment is always perfectly in focus. Unfortunately the intensity of the light available in a real system makes it impossible to use a pin hole camera. Real systems are forced to use a lens to focus the light from objects onto the image plane of the camera. This system is shown in figure 1-1 and it highlights the problems of using a lens, there is only a limited depth of field where the image is sharply focused. The depth of field can be improved by using a stronger and smaller lens but this tends towards the problem of the pin hole camera, limited image intensity.



Figure 1-1 Diagram of a simple camera lens.

The constraint caused by a limited depth of field is quite useful to image compression techniques. When objects are slightly out of focus the sharp lines defining shapes are degraded and therefore the image is easier to compress. The effect of the depth of field is to produce an active region inside the image, which is important to the user.

There are other effects caused by using lenses in cameras such as spherical and chromatic aberration but these do not affect the performance of image compressors. The most important feature, from the point of view of image compression, is the sampling of the image.

The image produced in the camera is usually converted to an electrical signal by the use of Charge Coupled Device (CCD) cells. These are MOS capacitors that accumulate charge proportional to the number of photons that are incident on them. They have a spectral bandwidth well in excess of the human visual response, continuing into the near infra red, and are available in rectangular arrays allowing a pixelisation of the image formed by the lens. There are several possible faults with the CCD system that could cause problems for the image compressor:

1. Dark Current. Random thermal effects cause a small amount of signal noise, and this becomes worse in particularly dark images. If this occurs then the image coder will be concentrating on areas of the image where only noise activity is apparent.
2. Fixed Patterns. Impurities cause shot noise in the image and although this problem is reduced in video systems, it can be a problem in still image compression.
3. Reset Noise. The CCD is not always completely drained during reset and there can be a small amount of ghosting from the previous image.
4. Thermal Noise at the Amplifier. This very similar to dark current but occurs at the amplifier.

The CCD array samples the image both spatially and temporally. The CCD array uses what is termed as an ‘electronic shutter’ which allows a very crisp representation of the scene to be formed. The usual temporal sampling rate for a CCD camera is 25 or 30 frames/second, although this is only due to standardisation and CCD arrays can work over a larger range of temporal sampling rates, if necessary. The spatial sampling is fixed when the CCD is fabricated, this is typically 752x582 pixels in size, but surprisingly the pixels are not exactly square (8.6x8.2 microns). Although this is a television standard, it is not necessarily the size of pixel array, which will be finally encountered by the image compressor. The picture is usually cropped, sub-sampled or both, before it reaches the compressor, and therefore compressors should be able to handle this general size range. It is possible that the image comes from another source such as a scanner, but they are generally in the same size range.

The final stage of the sampling is to convert the analogue charge in each CCD cell to a digital value. This is done by using a ‘flash’ analogue to digital converter (ADC) and the intensity of the signal is usually split into 28 levels (sometimes 212 for high precision work). The intensity of the sample defines the processes used in the image compressors. Nearly all image compression methods assume that the image has 8 bits of intensity. These methods would not be very effective at compressing images with different intensity ranges, for example binary images.

Although all these stages are necessary to form the final digital image they are nearly always ignored, and the digital images produced are assumed to be lossless. This is the case in this work, but the effects of the camera and digitiser should always be appreciated so that there are no major problems caused, when applying the algorithms to real systems.

**1.3 Colour**

In image compression colour is usually dealt with in two forms:

* Red, Green and Blue (RGB) colour images. These use 8 bits to hold each primary colour and then these colours are combined to produce ‘true’ colours that can be displayed on a television. A normal RGB colour image is usually considered to be stored in 24 bits/pixel.
* Luminance, Chrominance and Saturation (YUV). This is an abstract form of the RGB system that is more useful for image compression. There are equations that linearly transform RGB images to YUV images.

Each red, green and blue image contains information about the detail in the image, which can be represented as a single grey-scale or luminance image. If an image compressor was used on the separate RGB parts of an image it would triple the information needed to compress the colour image. In the YUV format all the details are in the Y image and so most of the compression can be concentrated there while the U and V images are compressed later at a higher compression ratio.

In this work only grey-scale (Y) images will be considered, as colour complicates the situation unnecessarily and is of secondary importance. All the compression techniques described in this work could be applied to colour images, since the U and V parts of the image have similar properties to the Y part of the image and therefore they can be compressed in a similar way.

**1.4 Image Compression**

The problem faced by image compression is very easy to define, as demonstrated in figure 1-2. The original digital image is usually transformed into another domain, where it is highly de-correlated by the transform. This de-correlation concentrates the important image information into a more compact form. The compressor then removes the redundancy in the transformed image and stores it into a compressed file or datastream. The decompression reverses this process to produce the recovered image. The recovered image may have lost some information, due to the compression, and may have an error or distortion compared to the original image.



Figure 1-2 Diagram of a general image compression system. Some compression methods do not have a transform stage but in these cases the transform can be considered to have no effect.

The aim of an image compressor is to produce the maximum compression for the minimum distortion. Although this is a relatively simple statement it is not easy deciding how to approach the problem. Neither the transform, the method of compression, or the distortion/error have been defined. Until some basic constraints are applied to the system it is difficult to proceed.

**1.4.1 Distortion Measure**

The distortion or error caused in the recovered image by the image compression process can be measured in several ways. The standard distortions measures are Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR) and Signal to Noise Ratio (SNR). These distortions are calculated as shown in equations 1.1 to 1.3 respectively. They work by comparing the squared error (power) between the original and the recovered digital images:

 (1.1)

 (1.2)

 (1.3)

where N is the number of pixels in the image

is the original pixel intensity at i,j

and is the compressed pixel intensity at i,j .

A common mistake when calculating PSNR is to replace  with 255. This is the maximum allowed range under the 8 bit intensity system, but the purpose of PSNR is to scale the error in images with less dynamic ranges of pixel intensity. This should make the PSNRs from different images comparable, which is not true for MSE.

None of the main methods for measuring image distortion takes into account, how good the recovered image looks to the human visual system. This is an area called psyco-visual image analysis, and it is an area of research which has a large scope. Unfortunately little progress has been made into an automated method for calculating a psyco-visual distortion measure.

**1.4.2 Transform**

The transform is the defining part of an image compressor and image compressors are usually placed into broad groups based on which transform they use. The image transform should de-correlate the image, so that the image data is in a more compact form, in the new transform domain.

Transforms generally come in pairs of forward and inverse transforms. If both the forward and inverse transforms are applied without compression, then the transform is either perfectly reconstructing (lossless), or the image information is quantised and lost after the transform stage (lossy). A lossless transform does not further complicate an image compressor since it makes no decisions about which parts of the image data are useful (so all these decisions can be made by the compressor algorithm). However a lossy transform can often produce more compression or allow the transform algorithm to run faster, both of which may be beneficial.

The transform can either be orthogonal, orthonormal or non-orthogonal (these are described in more detail in chapter 2). It is common to use orthogonal/orthonormal transforms in image compression, because they are efficient and the transform coefficients are highly decorrelated. The ‘Discrete Cosine Transform (DCT)’ and the ‘Wavelet Transform’ are examples of orthonormal transforms that are used in image compression and these will be looked at in section 1.5.

The transform is rarely applied to the whole image, it usually deals with small regions or image blocks independently. This has the advantage of exploiting local similarities within the image but also leads to a ‘blocking artefact’ effect found in this type of method. The blocks do not have to be a fixed size or shape, but they are usually non-overlapping. Applying an overlapping blocked system duplicates data already contained in other image blocks, and hence can be wasteful. Overlapping blocks have not been fully explored, but work on a lapped orthogonal transform is continuing [7].

**1.4.3 Compression**

Once the image has been transformed, it is necessary to compress the de-correlated data. Compression is achieved by a combination of two methods:

1. Quantisation. The accuracy of the transformed coefficients is reduced, sometimes completely truncating the coefficients. This is often used before entropy coding to improve the compression. Quantisation makes the coder inherently lossy.
2. Lossless Compression or entropy coding. The uncompressed data is converted into an efficient data set, that takes up the minimum size possible. This is achieved using variable length coders such as the Huffman [8] or Arithmetic [9] coders.

These processes will be discussed in more detail in chapter 3.

The compression of images is usually measured in two ways:

1. Compression ratio. The size of the original image is compared to the size of the compressed image.
2. Bits Per Pixel(BPP). This is the number of bits necessary to describe 1 pixel of the image. This is generally an average over the whole image.

For grey-scale images the compression ratio and BPP are related as shown in equation 1.4. In general BPP will be used to measure compression in this work.

 (1.4)

Image compression techniques sometimes have ambiguous compression ratios, when considering colour. This is because the original RGB image is rarely sampled in 8 bits per pixel, but it is sometimes assumed that it is. For this reason all images in this work will be greyscale, when considering MSE and compression, unless otherwise stated. This does not mean that colour is ignored, but it is a secondary consideration.

**1.5 Types of Image Coder**

There are five distinct classes of image compressors. Although it is possible to generally define their operations, it is difficult to be specific. Research is continuing all the time on image compression but JPEG [10] (which was 10 years ago) is the only still image compression standard to have been produced from this work. Proposals for the JPEG 2000 have begun but no standard has yet been fixed.

Each class of image coder will be discussed in general and then specific applications will be discussed in further chapters. The aim of this section is to give an overview of image compression, before concentrating on improvements and implementations.

**1.5.1 Hierarchical Image Partition**

This class of image coder works by dividing the image into different sized and shaped regions. The regions’ surfaces are approximated by very simple functions/transforms. If the transform was applied to a fixed block in the image, it would not be very effective, but the regions are shaped to maximise the effectiveness of the simple transform used. In general this means that small regions are used to approximate areas of high detail (in figure 1-3, the face), while larger regions are used for flatter (less detailed) areas (in figure 1-3, the plain background).

Since the functions which describe a surface are usually defined for a square block, the simplest form of hierarchical image partition is a quad-tree structure [11,57,58,59]. A quad-tree structure works by splitting square blocks into 4 equally sized sub blocks, the blocks that are split and the depth of split is governed by the method used, not the quad-tree structure. This produces an effective method of dividing the image, as shown in figure 1-3, while requiring a small amount of data to describe the quad tree (approximately 1 - 5% of the total compression).



Figure 1-3. Diagram of a quad-tree structure applied to an image.

The partition of the image can be more complex [12], such as using N sided shapes to describe the image, but as the complexity of the partition increases, so does the overhead of storing the segmentation structure. The more complex the partition is, the better the approximation fits to the image, but as a result there are less bits available to approximate the image. This is a common comprise reached in image compression and leads to a large variety of image segmentation based coders.

The sort of approximating function used in segmentation techniques is shown in equation 1.5 and generally they have very few transform coefficients:

 (1.5)

Where is the pixel intensity at i,j

is the transform coefficient at k,l

and is the approximating basis function for coefficients k,l .

The approximating basis function  is not necessarily orthogonal, but there are benefits to this, when using simple segmentation methods. If varyingly shaped blocks are used, it may be difficult to form a basis function that is orthogonal for all of the shaped blocks.

In general segmentation coders can not reach the image quality produced by other image compressors at similar compressions, however the research in these areas has not been fully explored. In chapter four a method is described for efficient image partition based on sorted lists of image block error. In that chapter there are also results for simple image partition algorithms using polynomial, DCT and fractal approximating functions.

**1.5.2 Fractal Transform**

The fractal transform works by exploiting self-similarity within the image. There are usually certain features within the image that are repeated at different resolutions. The fractal transform copies these features from a higher resolution onto features at a lower resolution, enhancing the image. Before this can be achieved, there needs to be some level of image approximation and this is produced by using a simple function to describe image blocks (as with the segmentation methods). In can be seen from figure 1-4 that a larger parent block, with relative coarse features, can be shrunk onto the smaller child block improving the initial approximation.



Figure 1-4. Diagram demonstrating fractal transform (without rotation considered).

The modified version of equation 1.5 for a fractal transform is shown in equation 1.6 :

 (1.6)

where  is the fractal coefficient

and  is the parent shrunk block.

This is the standard form of the fractal transform, but there are clearly a lot of variables. It is not clear, where the parent comes from in the image, how accurate the approximation should be or how much the parent should be shrunk. There has been a lot of research in the area of fractals, since the method was proposed by Barnsley and Jacquin [13] and there has been substantial improvement by Monro [14 to 17], Jacquin [18] and Øien [19].

The fractal transform can be used with image segmentation methods to improve the underlying approximating function, and this has the advantage of being able to use the fractal at different resolutions where necessary.

Although fractal methods have had a reasonable commercial success in the form of Iterated System’s products, there is no comparison of how effective they are compared to plain image segmentation. In chapter four this comparison will be investigated. In chapter three the fractal transform will be examined and the ‘state-of-the-art’ fractal methods will be explained, along with some other improvements related to vector quantisation.

1. **Vector Quantisation**

Linear quantisation is often used in image compression to reduce the size of the compressed transform coefficients. However linear quantisation does not account for spatial features inside the data. Vector quantisation [20] groups the data into sets and quantises these sets in an attempt to increase the overall compression. Since the data sets contain more than one piece of information, they are referred to in vector notation, hence the quantisation shown in equation 1.7 is represented as vector quantisation in equation 1.8.

Quantisation:  (1.7)

Vector Quantisation (VQ):  (1.8)

Where q is the quantisation operator,

 is a coefficient.

 is a quantised coefficient,

A simple example of vector quantisation for a binary image is shown in figure 1-5.



Figure 1-5. Demonstration of simple vector quantisation.

The image is divided into 2x2 blocks and since the image is binary there are 16 possible vectors. Some of these vectors are either not used or only used a few times and so to compress the image some of the redundant vectors are discarded. The compressed image has a slightly reduced quality but the general features of the image are maintained by fitting the best blocks from the quantised set to the original image.

In figure 1-5 it was possible to manually quantise the vectors because of the simplistic situation, but with real images the number of vectors is huge (1x107). It is clearly not easy to find suitable vectors for a specific image, but this problem can be solved by Lloyd [21] and Max [22] quantisation which will be explained in more detail in chapter two. Lloyd-Max quantisation allows an optimal set of vectors to be found for any set of test images. This gives a library of blocks that are easy to fit to the desired image. The method for finding the best library and how to apply this library has been developed in several works [20, 23 - 26].

Vector quantisation can be applied directly to an image, as above, but more favourable results have been found by applying it to a transformed image and wavelets have been shown to be effective [27]. Although vector quantisation could be confused as part of a wavelet transform, it stands alone as an image compression method.

There are drawbacks with vector quantisation. It is quite time consuming to find the optimal library of vectors and once found it is only optimal for its training set of images. The idea of constructing an optimal library for an image and sending it as part of the compressed stream has be discounted due to the time consumed and the size of the library. In chapter three it will be shown that it is possible to do this effectively providing the library is sufficiently simple.

**1.5.4 Discrete Cosine Transform**

The Discrete Cosine Transform (DCT) [28] was one of the first transforms to be used in image compression. It is very similar to the Fourier transform but the frequency data is more tightly packed. The DCT is effectively the real part of the Fourier transform, offset and sampled at twice the rate so that it is calculated over the centre of the pixels. The general form of the DCT (for two dimensional image surfaces) is shown in equations 1.9 and 1.10:

** (1.9)

 (1.10)

Where N, M are block side lengths,

is the image pixel intensity at i,j,

are the transform coefficients at k,l,

and ,

.

The DCT is an orthogonal transform and the coefficients of the DCT are generated by forming the inner product of the image/surface. The DCT basis functions are shown in equation 1.11 :

 (1.11)

where  is the DCT basis functions.



Figure 1-6. Diagram of the first 5 DCT basis functions or primitives (The greyscale primitive has not been shown, and is simply flat) .

The DCT basis functions, shown in figure 1-6, are very similar to the basis functions found by Principal Component Analysis (PCA) [60,61]. PCA finds the ideal orthogonal basis function for a set of image blocks (training sets). The difference is that the DCT does not assume anything about the image statistics, whereas PCA generates basis functions that are ideal for the training sets. This means that the principal components may not be ideal for all image blocks.

The DCT is perfectly reconstructing, when all the coefficients are calculated and stored to their full resolution. This means that when the coefficients are compressed it is possible to obtain a full range of compressions and image qualities.

The DCT has been used as the standard for still image compression in the form of JPEG [10] for about 10 years. The standard is now somewhat dated and it has caused the DCT to be seen in a bad light. The JPEG [10] standard is not close to what can be achieved with the DCT, if better techniques are applied (such as sensible quantisation and appropriate transform block sizes). The JPEG standard was designed to work with 1980’s hardware and as a result, there are some unnecessary features as well as some useful ideas. The JPEG standard is presently being reviewed.

The compression of the DCT can be approached in a number of ways and these ideas will be explored in chapter three and in chapter six. There are three basic methods for using the DCT.

1. Compression of all the coefficients by some method, often linear quantisation.
2. Truncation of the DCT coefficients to some degree and the application of a variable sized tiling of the image.
3. A combination of the above ideas.

**1.5.4 Wavelet Transform**

Wavelet transforms are based on sub-sampling high and low pass filters (Quadature Mirror Filters (QMF) [29]). These filters are matched in such a way that they split the data into high and low pass bands without losing any information. Wavelet filters have been designed for a wide range of applications and many different sets of filters have been proposed for different applications. In the case of image compression a common choice is constituted by the family of Daubechies orthogonal filters [30, 31] or the biorthogonal filters [32, 33], which have the advantage of linear phase.

Wavelet filters are explained in more depth in chapter seven, but they basically have two possible constraints:

* Regularity: The sum of a regular wavelet’s filter coefficients must be , as shown in equation 1.12 .

 (1.12)

where is the filter coefficient *i*,

* Orthonormality (or perfect reconstruction):The inner product of 2 wavelet filters is 1 or 0, depending on the filter as shown in equation 1.13 :

 (1.13)

where is the filter coefficient *i*,

and 

Neither of these constrains have to be satisfied for a wavelet to exist, but they usually required for image compression.

In image compression the wavelet filters are first applied to the image in the horizontal direction and then in the vertical direction to produce images as shown in figure 1-7.



Figure 1-7. Diagram showing the application of wavelet filters.

The four bands that are formed are referred to as the low-low (LL), low-high (LH), high-low (HL) and High-High (HH). The LL band still has image-like information and so it is possible to apply the set of wavelet filters, in the same way as applied to the original image. This process of dividing the image into sub-bands can be continued as far as desired (to the resolution of the image), but for image compression it is usually only continued to 4 or 5 levels. A typical final image is shown in figure 1-8.

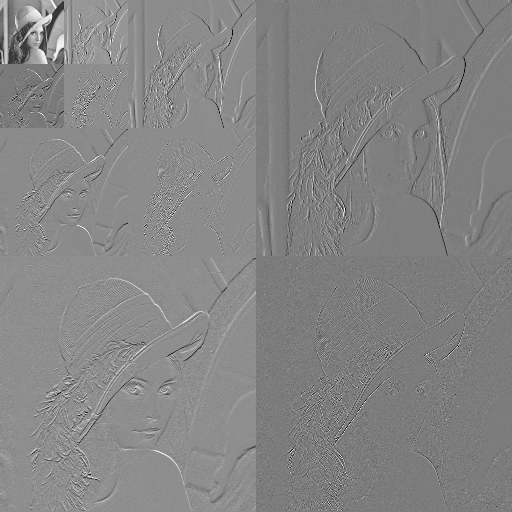


Figure 1-8. Diagram showing the final stage of wavelet filtering.

The sub-bands in figures 1-7 and 1-8 have been offset with 128 (out of 256) grey levels to make them visible and scaled according to the sub-band they are in. The actual information content of the sub-bands can be visualised by showing their average signal power, as shown in figure 1-9. It can be seen that most of the energy is very highly packed into the smallest Low-Low sub-band. This means that the wavelet transform is ideal for image compression, giving a high de-correlation, while also being orthogonal.



Figure 1-9. Diagram of the power distribution produced by a wavelet transform.

When the transform has been applied, it is a matter of compressing the transform coefficients. There are a number of ways to achieve this, and the most effective methods have only recently been introduced. Originally the wavelet sub-bands were linearly quantised in some way to produce compression [ 34, 35] and this led to Vector Quantisation being used to quantise the sub-bands [27]. The VQ methods are constantly being improved but the other main contender is Shapiro’s zero tree method [36]. This exploits the pyramid like nature of the wavelet filtered image and produces exceptional results. An improvement to this method was produced by Said and Pearlman [37] which improves the zero tree method by using significance sorting. The advantage of these two methods is that the compressed image data is not only progressive, it is also embedded. This means that the required compression ratio can be reached exactly and any compression ratio up to the required compression can be decoded from the compressed data. Unfortunately this does not allow the image to be progressively reconstructed in real time since the inverse wavelet transform would be required after each update of the wavelet image.

Wavelet transforms are generally accepted as the next generation of image compression and although there has been little research done on them in this work they have been used for comparison. Chapter seven explains Shapiro’s method [36] in detail and highlights the changes that have been made.

1. **Summary**

The aim of this work is to improve image compressors. This chapter explained the basics of still image compression, in chapters which follow the ideas touched on here will be extended. Large improvements to still image compression technology have been made in this work and these are explained in chapters two through to seven. In chapter eight the methods described will be compared and conclusions drawn as to which image compressors are best to use.