Applications of Fractals to Image Data Compression

ANDREW PHILIP WILTON

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

March 1996

The University of Aston in Birmingham

Applications of Fractals to Image Data Compression

ANDREW PHILIP WILTON

Doctor of Philosophy

1996

Summary of Thesis

Digital image processing is exploited in many diverse applications but the size of digital images places excessive demands on current storage and transmission technology. Image data compression is required to permit further use of digital image processing. Conventional image compression techniques based on statistical analysis have reached a saturation level so it is necessary to explore more radical methods. This thesis is concerned with novel methods, based on the use of fractals, for achieving significant compression of image data within reasonable processing time without introducing excessive distortion.

Images are modelled as fractal data and this model is exploited directly by compression schemes. The validity of this is demonstrated by showing that the fractal complexity measure of fractal dimension is an excellent predictor of image compressibility. A method of fractal waveform coding is developed which has low computational demands and performs better than conventional waveform coding methods such as PCM and DPCM.

Fractal techniques based on the use of space-filling curves are developed as a mechanism for hierarchical application of conventional techniques. Two particular applications are highlighted: the re-ordering of data during image scanning and the mapping of multi-dimensional data to one dimension. It is shown that there are many possible space-filling curves which may be used to scan images and that selection of an optimum curve leads to significantly improved data compression. The multi-dimensional mapping property of space-filling curves is used to speed up substantially the lookup process in vector quantisation.

Iterated function systems are compared with vector quantisers and the computational complexity of iterated function system encoding is also reduced by using the efficient matching algorithms identified for vector quantisers.

Keywords: image compression, fractals, space-filling curves, vector quantisation, iterated function systems.

Acknowledgement

I would like to thank the following:

My research supervisor, Dr. Geof Carpenter, for his help over the duration of this work, and for his advice during the preparation of this thesis.

The Electronic Engineering and Applied Physics department of Aston University for allowing me time off from my duties to complete this research.

My colleagues from the EEAP department for their understanding when I have been pre-occupied with my studies.

My wife, Helen, for her constant encouragement to continue with my studies.

Applications of Fractals to Image Data Compression

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction to Thesis

1.1. Objectives

In this thesis an attempt is made to link two areas of study, image data compression and fractals. Each of these topics has proved fruitful for mathematicians and engineers but their union has been less well-received. Indeed, a healthy scepticism has become apparent with regard to the applications of fractals.

> "... this is an interesting case, which seems to happen more and more often, of how an explanation in search of a phenomenon may lay claim to far more territory than it can handle." [Madd 86]

There has perhaps been an over-enthusiasm for seeing fractals as a panacea for many demanding problems in Science and Engineering. Clearly not all objects and processes are fractal. Many physical processes exhibit a fractal behaviour but only over a limited range of scales. This makes them inadmissible as fractals in the Platonic, mathematical, sense but does not necessarily invalidate the use of fractals to generate useful results. There may be no true fractals in nature but there are no true circles or straight lines either.

It is the aim of this thesis to show that ideas from the mathematics of fractals can be applied to image processing and to image data compression in particular.

The initial motivation for this work was the intuitively attractive idea that the space-filling curves and fractals in general are so interesting and have such unique properties that they ought to have practical applications. This has been confirmed. This issue now is one of quantification.

1.2 Why apply fractals to image compression?

The demand for better image compression and storage has led to the development of numerous image compression methods some of which have been exalted to the status of proprietary or open standards. The development of open standards is a major undertaking involving significant technical collaborative work between major manufacturers. This usually means that such standards lag behind the most recent advances in the field. This leads to an opportunity for a proprietary standard to gain a foothold. Since the main benefit of open standards - the ability for different manufacturers' equipment to work together - is so clear, it becomes increasingly difficult for a new proprietary standard to be accepted unless its performance gain is dramatic (perhaps an order of magnitude or more). There is unlikely to be a new

standard (other than enhancement of existing standards) for still image coding in the near future unless a new technique is devised which offers such a gain.

There are two possible answers to the preceding argument.

1. If improvements to existing algorithms can be identified which are simple to implement, requiring little overhaul of existing systems, then they may well be adopted.

2. Fractals offer such a novel way of interpreting image data that many authors feel they have the potential to provide the dramatic performance increase necessary to supplant current standards.

1.3. Structure of thesis

The thesis is organised as follows:

Chapter 2 presents an introduction to image data compression. It includes an overview and a suitable taxonomy of image compression techniques, providing a framework into which fractal methods may be placed. It also includes a comparative study of the 'state-of-the-art' in image compression. No new methods are presented in this chapter but benchmark performance figures are derived by applying existing techniques to the same data sets which are used for the novel approaches in later chapters.

Chapter 3 provides a grounding in fractals and their properties. There is no particular attempt at mathematical rigour; the purpose is to describe those properties of fractals which contribute to their usefulness in image processing. Methods of measuring the fractal dimension of image data are examined. It is shown that the fractal dimension provides a natural and effective measure of the complexity of an image which in turn can be used to predict how effectively the image data can be compressed. Space filling curves are introduced as examples of fractals.

Chapter 4 presents original work on space-filling curves. The curves are analysed in detail and methods of generating the curves in two dimensions are compared. Hierarchical extensions to the curves and variations for non-square grids are developed. A comparison is made of the different curves and techniques for creating new curves. The curves are shown to have interesting and useful properties with general applicability which is exploited in each of the later chapters. In Chapter 4 their property of re-ordering data in a two-dimensional array is exploited in a novel way to generate many different scans of images. It is shown that scanning images in this way improves the effectiveness of predictive methods of image data compression.

Chapter 5 presents original work on one-dimensional waveform coding using fractal methods. A method of lossy image coding called fractal waveform coding (FWC) is presented which is based on the fractal measurement of coastlines. This is shown to be both easy to implement and computationally efficient, yet it yields excellent results with an easily-controlled trade-off between compression ratio and image quality. This method of image compression has previously been proposed but without attention to practical difficulties. In Chapter 5 a thorough analysis is performed with extensive simulation from which solutions to the practical problems are derived.

Chapter 6 is something of a departure from the main direction of the thesis. It covers vector quantisation (VQ) of images. Conventionally this multi-dimensional approach has been used to better exploit the redundancy in image data but has suffered from severe computational difficulties. In this chapter original methods of greatly increasing the performance of VQ systems are identified including a novel technique using another property of space-filling curves, the ability to map between one and many-dimensional spaces, for the vector lookup problem.

In Chapter 7 fractal image coding based on iterated function systems (IFS) is discussed. This is a new and controversial technique, containing aspects of many of image coding methods identified in previous chapters. Original implementations of IFS image compressors based on several proposals have been created and comparative results are presented. IFS coding shares similar computational difficulties with VQ and the original improvements to the VQ lookup problem identified in Chapter 6 have also been applied here with some success.

It will become apparent that there are connections between all of the chapters. In Chapter 8 a discussion of these connections is presented. As with any work which is new, every answer leads to further questions but time does not permit exploration of all these avenues. Topics deemed worthy of further attention are discussed in Chapter 8.

It has not been possible to consider all applications of fractals to image processing. In particular the quite well-developed topic of synthetic image creation has not been discussed at all. It is felt that this is sufficiently distinct from the focus of this thesis which is the application of fractals to coding real images of natural scenes and, in particular, ways in which the fractally coded representation of the images allows compression of the image data.

Chapter 2. Image Data Compression

2.1 Introduction

This chapter contains an introduction to the image processing terminology used in this thesis. A taxonomy of the principal components of an image compression system is presented together with illustrations from original and cited experimental data. Many of the concepts explored in this chapter have been used by the Joint Photographic Experts Group (JPEG) in defining the current standard for still image compression. This standard is also described in detail later in the chapter when the key ideas have been outlined.

Digital image processing has found applications in many subject domains such as videoconferencing, satellite remote sensing, document and medical imaging, radar, sonar, facsimile (FAX) and remotely-piloted vehicles for military and space use. The typical image compression sequence of activities is shown in Fig. 2.1. Typically an analogue representation of a real-world scene is acquired by a sensor and converted to digital form by sampling and then quantising each sample. The digital signal has advantages for subsequent computer processing (e.g. to enhance the image, detect hidden or distorted features), but the analogue-to-digital conversion increases the signal bandwidth considerably. The processed image data may then be passed directly to an output device such as a computer screen or printer but, more typically, it has to be stored or transmitted to a remote system. In either case, at this stage the performance of the system becomes limited by the volume of image data.

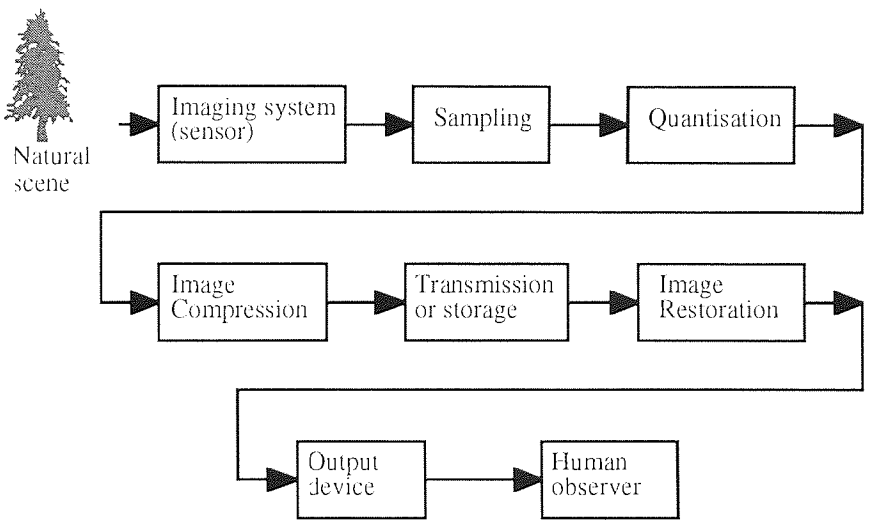


Fig. 2.1. Digital image compression sequence

A digital image is a two-dimensional array of pixels each of which is a number, corresponding to some property of the image scene such as intensity, represented by a finite number of bits. Since a single digital image may have as many as 16 million pixels, these images comprise large amounts of information. Even though the capacities of modern storage media and the bandwidths of transmission systems are high, and still increasing, they are necessarily limited. For over forty years now there has been development of methods of reducing the storage and transmission requirements of images. This is the subject of image data compression.

More recently, digital image compression has been crucial to the growth of multimedia computing and may be seen as an 'enabling technology' [Gonz 92]. i.e. Its importance in its own right is less significant than the advances it yields in other fields. In addition, image data compression is required to handle the increased bandwidth of modern imaging sensors and evolving broadcast television standards. In the same way that the use of digital audio storage with compact discs has revolutionised the recording industry, digital video introduces a whole new range of applications such as electronic cameras, video-conferencing systems, video programme delivery by cable and satellite and high-definition TV (HDTV). The relatively low audio bandwidth of approximately 20 kHz, which leads to a digital data rate of only 1.4 Mb/s, has permitted the development of digital audio without requiring data compression but the far higher bandwidth of image and video data makes data compression essential.

Some international standards for image compression have recently been established such as the JBIG bi-level standard and the JPEG colour image standard (both of which are discussed in Section 2.8). However, images are rapidly becoming so large that they cannot adequately be compressed for transmission or archival with current techniques so considerable research activity is still devoted to this problem. Some examples of the storage and data raterequirements for uncompressed digital images for various applications are given in Table 2.1.

Still images		Moving images (image sequences)	
Image type	**Storage**	**Image type**	**Data rate**
512 by 512 grey-scale (256 levels)	0.25 MB	Digital Broadcast TV	166 MBit/s
1024 by 1024 full-colour (24 bit)	3 MB	HDTV	1766 MBit/s
FAX	0.01 MB	Low quality video phone	0.4 MBit/s
Photo disc image	6 MB	Video Conferencing (CIF)	48 MBit/s

Table 2.1. Image storage and bandwidth requirements

Practical systems for each of these applications have differing requirements in compression ratio, image distortion, processing time etc. Often, techniques can be found which perform extremely well within a single narrow domain but successful innovations which evolve into standards must have wider applicability.

2.1.1 Is image compression necessary?

There are critics of the necessity to compress images for transmission. While the broadcast spectrum is inherently limited, it can been argued that increasing use of optical fibre for transmission provides ample bandwidth for the foreseeable future. With current technology, optical fibre links are not always economically viable - in the same way that building motorways between every town is not a solution to traffic congestion. Furthermore, their greater subjective quality is leading to an ever-increasing demand for higher resolution images. The size of uncompressed image data grows as the square of the image resolution so that, without compression, the seemingly vast optical bandwidth could be exhausted more quickly than might be expected. Conversely, the greater the image resolution, the easier the data is to compress, making it even more reasonable to apply compression.

The case for compression of stored images is stronger. Even with the recent rapid decline in the cost of storage media, the capacities shown in Table 2.1 are hard to achieve for significant sets of images.

It is also worth considering whether the subject of image compression is now saturated with techniques. Surprisingly complex algorithms are now being employed to achieve relatively modest compression gains. Advances in hardware now allow digital signal processing in real-time at low cost, opening up new applications in areas such as communications, consumer electronics, medicine, robotics and defence which previously were not feasible. However, hardware developments do not remove the incentive for searching for more efficient algorithms. It has been pointed out by Aho and Ullman that:

> "As computation becomes cheaper and machines become faster our desire to perform larger and more complex problems will also grow. Thus the discovery of more efficient algorithms (i.e. those whose growth rates are low) becomes *more* rather than less important." [Aho 95]

2.2 Classification of images

Fig. 2.1 shows the first stage of an image processing system to be the acquisition of an image by a sensor system. This could equally be replaced by the acquisition of a pre-digitised image from some other source. In general a digital image may pass through

many image processing systems before it is viewed by a human observer. The issue considered in this thesis is how the compression and subsequent decompression cycle may be optimised and how it is influenced by the other processes such as sampling (Section 2.3) and quantisation (Section 2.4). It will not be considered how the original image was captured - only how to maintain the fidelity of the image compared with its original digital form.

There are several categories of digital image which might need to be compressed. The essential factors which determine appropriate compression techniques are: whether the image is natural or synthetic; the image resolution; whether the image is monochrome or colour and whether the image is single or part of a set such as a moving sequence.

2.2.1 Natural and synthetic images

The most fundamental division in image classes is between natural images (i.e. those acquired by some sort of camera or scanning device) and synthetic images (i.e. those generated by computer for animation etc.). Simple synthetic images such as cartoons and geometric drawings are usually not good candidates for conventional image compression. In general, they are far better represented by an object-based description than a pixel based description. For example, Fig. 2.2 shows a simple bi-level synthetic image containing three overlapping geometric shapes with different shading.

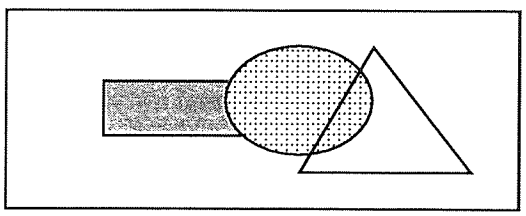


Fig. 2.2. Simple synthetic image

When displayed even at a low resolution of 72 pixels per inch this image requires approximately 12,000 pixels to represent it, yet the computer drawing program only requires a few bytes to store the geometry coordinates and to specify the fill patterns.

Images of a natural scenes will, in general, contain far more information since they represent the complexity of the real world and thus are much harder to compress. It is appealing to consider ways in which object-based descriptions could be applied to natural images. This is the principle of model coding which is discussed briefly in Section 2.6.5. Unfortunately this is fraught with difficulties since it is hard for a computer system to identify the objects in an image and even harder to then find concise descriptions for the objects.

Synthetic images can now be produced with techniques such as ray-tracing and fractal forgery (the process of creating synthetic computer images which look like natural scenes [Barn 88b]) which appear to be natural images. For these images both object-based and pixel based coding are applicable, with the trade-off depending on the computational complexity of recreating the image from the object description

2.2.2 Image resolution

The term resolution has varied meanings. For the human visual system this is most appropriately considered to be the minimum feature resolution which is the smallest angular difference over which two intensities may be discerned. This angular resolution is approximately $\frac{1}{60}$ of a degree [Scha 89]. However, the perceived image resolution depends on the overall imaging system, including any lenses, the separation of the observer and the display screen, the viewing angle and the pixel spacing of the screen. It is therefore convenient and common to describe the image resolution as referring to the number of pixels on each axis of the image.

A typical modern computer display has a viewing area approximately 25 cm square which, when viewed from a distance of 50 cm, subtends an angle of approximately 22 degrees. To provide an angular resolution comparable with the angular resolution of the human visual system such a display would need a resolution of approximately 1300 by 1300 pixels. Depending on the particular application, digital images in common use have resolutions between 128 by 128 pixels (low quality video phones) and 4096 by 4096 pixels (high quality satellite imagery).

2.2.3 Monochrome and colour images

Monochrome images may be bi-level or have a grey-scale. Colour images may be paletted, where the image contains only a relatively small number of colours, typically 256 (but these may be chosen from a large palette), or full-colour, where the image may contain an essentially unlimited number of colours since there are usually more available colours than pixels.

It is well known that the human perception of colour can be modelled by tristimulus colour theory [Wyze 67] in which colour is represented by the intensities of any three independent *primary* colours. Any set of physically realisable primary colours will require some negative coefficients to match some colours but certain shades of red, green and blue give the broadest range of colours for positive coefficients only. This restriction is necessary in systems such as CRT-based displays since they cannot produce negative amounts of primary colours. There is an internationally defined

standard, CIE 1931 (with some recent modifications), which uses three primaries which are not physically realisable, but which allows other colours to be represented with positive coefficients. Real displays represent a subset of all the colours defined by this standard.

The human visual system can distinguish approximately 2^6 shades of red, 2^7 shades of green and 2^5 shades of blue. Thus a pixel may be represented accurately by approximately 18 bits of data. With current technology using byte-oriented hardware it is usual to allocate one byte to each colour giving 24 bits or 3 bytes per pixel for a full-colour system.

Colour images can be considered as three independent 'grey-scale' images, where the 'grey-scales' are the axes of the tristimulus colour coordinate system. Often, the most conceptually simple coordinate system based on red, green and blue (RGB) axes is used but there are numerous transformations into other colour spaces [Jain 89], such as those used for television systems, which have practical advantages since they can exploit the different sensitivities of the human visual system to luminance and chrominance.

2.2.4 Still images and image sequences

Processing of a sequence of images could be performed simply by processing each image as a still image (intra-field coding) but this would neglect much of the temporal redundancy. Practical schemes usually identify differences in successive frames (inter-field differences) and code these using the same techniques as are used for still images. More advanced predictive coding is also used to provide motion compensation where part of the new image may be represented by a simple linear translation from a previous frame. Currently the leading standards for image sequence coding are H.261 (also known as P x 64) [Liou 91] developed for videoconferencing and MPEG [LeGa 91] for entertainment-quality video in cable-TV and satellite broadcasting.

2.2.5 Image types used in this thesis

In this thesis the main theme is the illustration of the novel applications of fractal mathematics to image compression. Of the image types identified in Sections 2.2.1 to 2.2.4 it is required only that source images should contain sufficient complexity to demonstrate the ideas.

- Only pixel-based images of natural scenes are considered. These are far more representative of typical images to be compressed. Any synthetic image could

easily be converted to pixel form and processed in the same way but this would rarely be appropriate.

- The images have a resolution of 512 by 512 pixels. Images of 128 by 128 pixels or smaller are rarely acceptable, as indicated by the poor take-up of current video-phones. Larger images are expensive to work with since specialised equipment is required to view images larger than 1000 by 1000 pixels. The intermediate resolution chosen is typical of images which are commonly used and is convenient because it allows the original and processed images to be compared side-by-side on the same computer screen to give an accurate subjective measure of quality. It is also high enough to show detailed information in the images such as edges. The choice of square images whose resolution is a power of two is convenient for the preponderance of image processing algorithms which require block-based or recursive decomposition of the image.

- In general, only grey-scale images are considered. The exception is made for bi-level images in Chapter 4 where dithering is discussed. Elsewhere in this thesis, all of the techniques presented are applicable to colour images if they are treated as having three independent colour axes. This would not be optimal since the colour coordinates, in any coordinate system, are unlikely to be completely independent, but the processing - as far as data compression is concerned - would not be fundamentally different. The number of grey levels is set to 256, which provides a useful level of quantisation since it is slightly greater than the number of grey levels distinguishable by the human eye. It is also convenient to work with images where each pixel is represented by a single byte.

- Only still images are considered. This decision is taken for practical reasons. Image sequence coding is a qualitatively different problem. Temporal (i.e. inter-frame) data redundancy is usually significantly greater than spatial (i.e. intra-frame) redundancy so most developments in image sequence coding concentrate on exploiting this. A good technique for still image coding is required to process the first frame of any sequence and whenever there is an abrupt change of scene and the work presented here could be employed in these cases. In the MPEG standard, for example, so that the decoder can reliably resynchronise in the presence of gross transmission errors, it is specified that every fifteenth frame is encoded without reference to previous frames.

To summarise, experimental work reported in this thesis is restricted to 256-level, grey-scale, still images of natural scenes with a spatial resolution of 512 by 512 pixels.

2.3 Image sampling

In Fig. 2.1 it was seen that the first stage of converting an image to digital form is to sample the image on a discrete grid. Older scanning devices such as vidicon camera tubes operate by scanning the image row by row and then sampling each row. More modern devices such as charge-coupled-devices (CCD) cameras provide a true two-dimensional scanning ability (i.e. the image can be scanned element by element in any order [Flor 85]). The scanning sequence is almost invariably a simple raster and, in all current television standards, interlacing (where each frame of the image is split into two fields containing alternate lines) is used to reduce the signal bandwidth. The possibility of using more complex non-raster scans for subsequent image processing including compression is explored in Chapter 4.

2.3.1 Shape of sampling grid

Two sampling patterns are in common use, rectangular and interlaced (or quincunx). If the horizontal and vertical spacings are equal then the interlaced pattern is the same as a rectangular pattern rotated through 45°. With a different scaling factor, the interlaced pattern can be described as hexagonal. A two-dimensional sampling lattice may be described by a 2 by 2 sampling matrix λ whose columns are the vectors specifying the two nearest samples.

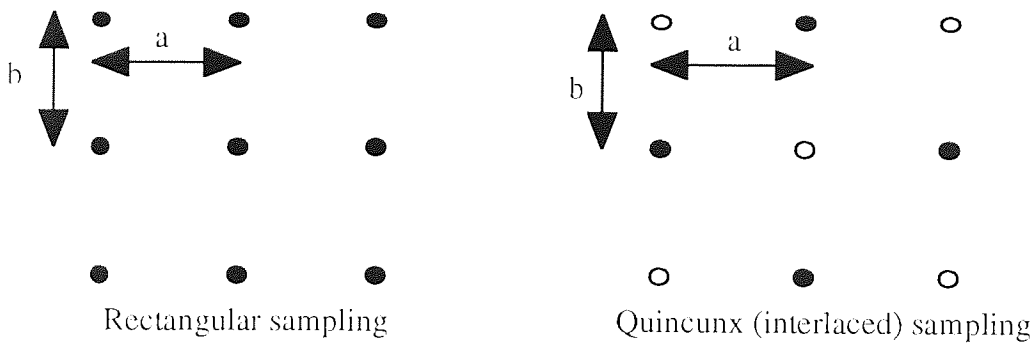


Fig. 2.3. Rectangular and interlaced sampling

The sampling matrices λ_r and λ_q for the rectangular and quincunx sampling grids shown in Fig. 2.3 are:

$$\lambda_r = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad \lambda_q = \begin{bmatrix} 2a & a \\ 0 & b \end{bmatrix} \qquad \text{.....(2.1)}$$

It can then be shown [Jain 89] that, if an image $f(x,y)$ is sampled on a grid λ then its Fourier transform $F(\zeta_x,\zeta_y)$ is replicated in frequency space on the reciprocal lattice Λ where:

$$\lambda^T\Lambda = \mathbf{I} \qquad \text{.....(2.2)}$$

Hence:

$$\Lambda_r = \begin{bmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{b} \end{bmatrix} \quad \Lambda_q = \begin{bmatrix} \frac{1}{2a} & 0 \\ \frac{1}{2b} & \frac{1}{b} \end{bmatrix} \qquad \text{.....(2.3)}$$

To avoid aliasing an image should be bandlimited with a low-pass filter before it is sampled. The bandwidth of the sampled image is the spatial frequency ζ_0 where the Fourier transform $F(\zeta_x,\zeta_y)$ is zero outside a bounded region in the frequency plane.

$$F(\zeta_x,\zeta_y) = 0, \text{ for } |\zeta_x| > \zeta_0 \text{ and } |\zeta_y| > \zeta_0 \qquad \text{.....(2.4)}$$

The ideal rectangular image sampling function is a two-dimensional infinite grid of Dirac delta functions on a rectangular grid with spacing Δx, Δy. This is the comb function where:

$$\text{comb}(x,y;\Delta x,\Delta y) \equiv \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\delta(x-m\Delta x, y-n\Delta y) \qquad \text{.....(2.5)}$$

Then the sampled image $f_s(x,y)$ is given by:

$$\begin{aligned} f_s(x,y) &= f(x,y)\text{comb}(x,y;\Delta x,\Delta y) \\ &= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(m\Delta x,n\Delta y)\delta(x-m\Delta x, y-n\Delta y) \end{aligned} \qquad \text{.....(2.6)}$$

It follows from Shannon's sampling theorem that a bandlimited image sampled uniformly on a rectangular grid with spacing Δx, Δy can be recovered without error from the sample values $f(m\Delta x,n\Delta y)$ provided that the sampling rate is greater than the Nyquist rate, i.e. $\frac{1}{\Delta x} = \zeta_{xs} > 2\zeta_{x0}$, $\frac{1}{\Delta y} = \zeta_{ys} > 2\zeta_{y0}$ and that the reconstructed image is given by :

$$f(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(m\Delta x,n\Delta y)\,\text{sinc}(x\zeta_{xs}-m)\,\text{sinc}(y\zeta_{ys}-n) \qquad \text{.....(2.7)}$$

The important restriction here is that the input signal is bandlimited. In a digital image processing environment, the preliminary low-pass filtering is, in effect, performed by

the lens and scanning aperture combination during acquisition of the image. The concern during image compression and restoration is only that this bandlimited image can be recovered.

2.4 Image quantisation

In Fig. 2.1 quantisation was shown as being an essential component of any digital image processing system. Although most images start as real-world scenes, with a continuous scalar value for the intensity at each point in the scene, it is necessary in a practical system to allocate only a finite number of bits to each pixel value. This process is termed amplitude quantisation, or simply quantisation. This initial quantisation is usually combined with the sampling stage during image acquisition. In addition, most of the image compression schemes discussed in this thesis include a further quantiser somewhere in the system (e.g. for coding transformed coefficients, edge sizes etc.). The terminology and notation for quantisation introduced here is required in Chapters 4 to 7.

2.4.1 Definition of quantisation

A quantiser Q maps a continuous scalar variable f into a discrete variable $\hat{f}$ having a finite set of L reconstruction levels using a set of L transition or decision levels. This can be expressed by:

$$\hat{f} = Q(f) = r_i, \text{where } d_{i-1} < f \le d_i \quad \ldots\ldots (2.8)$$

where r_i for $1 \le i \le L$ and d_i for $0 \le i \le L$ denote the reconstruction levels and the decision boundaries respectively.

The mapping is generally a staircase function (see Fig. 2.4) with a simple mapping rule: the input is mapped to the first reconstruction level which includes the input. The quantiser design has to determine the optimum transition and reconstruction levels, given the probabilities and an optimisation condition. This involves a trade-off between simplicity and performance.

Clearly there will, in general, be some error in the reconstructed values. This is termed quantisation noise e_Q where:

$$e_Q = \hat{f} - f \quad \ldots\ldots (2.9)$$

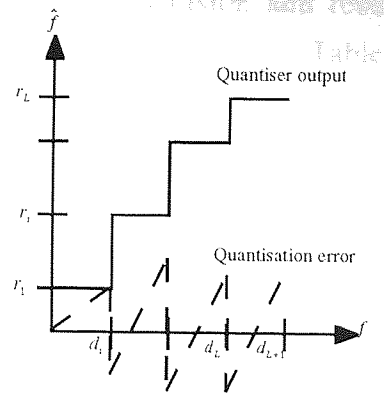


Fig. 2.4. Quantisation

The design of a quantiser is based around ways of minimising some error criterion based on the average distortion D . If the minimum mean square error (MMSE) criterion is used then D is given by:

$$D = E\left[e_Q^2\right] = E\left[\left(\hat{f} - f\right)^2\right] = \int_{f_0=-\infty}^{\infty} p_f\left(\hat{f} - f_0\right)^2 df_0 \qquad \text{.....(2.10)}$$

2.4.2 Comparison of quantisers.

The simplest quantiser is the uniform or linear quantiser where the reconstruction and decision levels are uniformly spaced. Some improvement is gained by using a logarithmic quantiser which uses closer decision levels for small signal levels and increases the decision level spacing for large signals where the quantisation noise is effectively masked. The quantisation noise is signal dependent but, for certain signal types such as those with Gaussian or Laplacian amplitude distributions, quantisers known as Lloyd-Max quantisers [Max 60] have been designed which are optimal (i.e. the mean-square quantisation noise is minimised for a fixed number of quantisation levels).

Quantiser	Decision levels	Reconstruction levels
Linear	$d_i - d_{i-1} = \Delta, \quad 1 \le i \le L$	$r_i = \frac{d_i + d_{i-1}}{2}, \quad 1 \le i \le L$
Logarithmic	$d_i - d_{i-1} = \Delta, \quad 1 \le i \le L$	$r_i = \left(\frac{L}{e-1}\right)\left[\exp\left(\frac{i}{L}\right) - 1\right]$
Lloyd-Max	$d_i = \frac{r_i + r_{i+1}}{2}, \quad 1 \le i \le L-1$	$r_i = \frac{\int_{f_0=d_{i-1}}^{d_i} f_0 p_f(f_0) df_0}{\int_{f_0=d_{i-1}}^{d_i} p_f(f_0) df_0}, \quad 1 \le i \le L$

Table 2.2. Derivation of quantiser levels

Formulae for the derivation of the decision and reconstruction levels of linear, logarithmic and Lloyd-Max quantisers are given in Table 2.2. Fig. 2.5 shows three 4-bit quantisers over the same range 0 - 15. The Lloyd-Max quantiser is for a Gaussian source

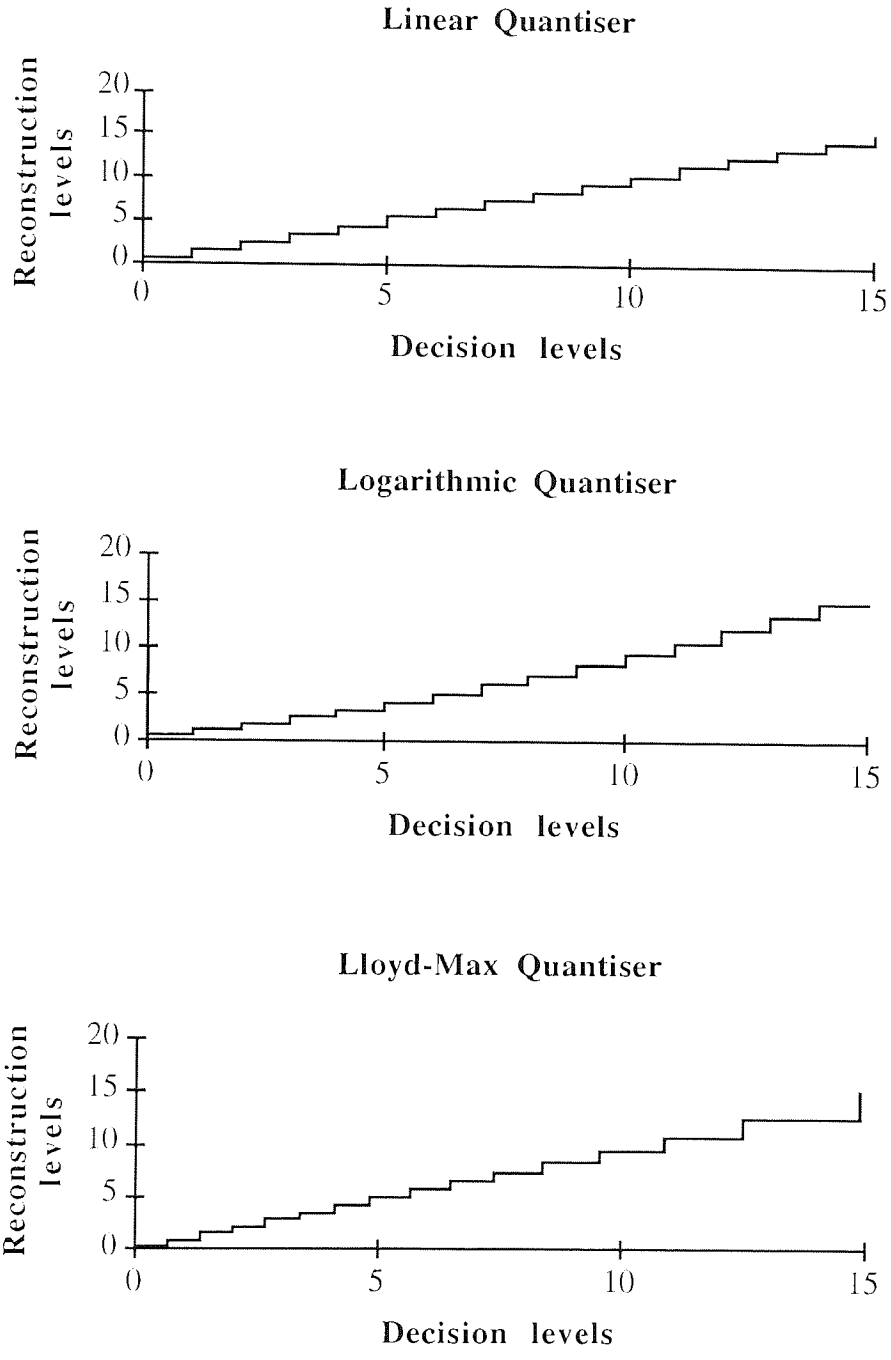


Fig. 2.5. Linear, logarithmic and Lloyd-Max quantisers

The Lloyd-Max quantiser is not considered suitable for practical systems since it requires *a priori* knowledge of the frequency distribution of the signal to be quantised. However, when a practical quantiser is embedded in a larger system, then it can be

replaced during simulation by an optimum Lloyd-Max quantiser (designed to match the known image frequency distribution) to enable analysis to be made of the contribution made to the total error by the quantiser.

If each scalar is quantised separately the process is termed scalar quantisation. If two or more variables are jointly quantised the process is termed vector quantisation, a concept which is examined in much more detail in Chapter 6.

2.4.3 Quantisation taking into account visual perception.

The choice of quantiser in a digital image processing application depends on which aspect of the image is being coded. If the variable being coded is the pixel intensity, then, since there is no reason why a particular image should have a particular intensity distribution, no advantage is gained by using a non-linear quantiser and linear quantisers are generally preferred [Deni 91]. Linear quantisers are universally used for video quantisation since, averaged over many frames, a uniform distribution of pixel intensities is found.

It is notable that the usual image display system, the cathode ray tube, has a very non-linear response to intensity but this is, fortunately, counteracted by the response of the human eye. Hence, a linear quantiser can be used here without significant perceived distortion.

A significant property of the human visual system is its sensitivity to intensity contours in local structure. While the luminance (intensity) of an image feature is independent of the luminances of the surrounding features, the perceived brightness of an object depends on the luminance of the surrounding features. This is summarised in Weber's law (Eqn. 2.11) which states that, if the luminance of an object f_O is just noticeably different from the luminance f_S of its surroundings, then their ratio is constant. i.e.

$$\frac{|f_s - f_o|}{f_o} = \text{constant} \qquad \ldots\ldots(2.11)$$

The value of this constant is found to be approximately 0.02 which implies that only about 50 contrast levels are required for an image. This is, however, a rather simplistic conclusion since an image may contain contrasting features at different scales.

Excessively coarse luminance quantisation gives rise to perceptible contouring. There are two main methods of suppressing this: contrast quantisation and pseudo-random noise quantisation (dithering).

In contrast quantisation [Kret 75] the contrast is represented by some non-linear transformation of the luminance to give approximately 50 levels of contrast. Hence, if uniformly quantised, each pixel which would otherwise require 8 bits is represented by 5 to 6 bits. If an optimum mean square quantiser is used this can be reduced to 4 to 5 bits.

In pseudo-random noise quantisation which is known as dithering [Robe 62] a small amount of uniformly distributed noise is added to the luminance signal. In order to display the image, the same, or another, pseudo-random sequence is subtracted from the quantiser output. The mean value of the quantised pixels is the same with or without noise but any contours are smoothed. The amount of dither must be small enough to maintain spatial resolution but large enough to allow the luminance values to vary randomly about the quantiser decision levels. Dithering schemes utilising fractal, space-filling curves are examined in Chapter 4.

2.4.4 Summary of quantisation

Quantisers are required during digital image acquisition, processing and display. Linear quantisers of contrast rather than luminance are appropriate for image capture and display. At other stages in an image processing system the choice of quantiser depends on which variable is being quantised. Optimal quantisers are known for variables with well-defined statistics but many variables (e.g. edge magnitudes, transform coefficients) require a more empirical approach.

2.5 Data compression

General purpose data compression schemes are those which are independent of the nature of the data source. It is appropriate to discuss general data compression before considering image data compression because many of the principles are applied in image data compression.

Data compression consists of taking an input stream of symbols and transforming them into an output set of codes. The data is judged to be compressed if the size of the codes is smaller than the original symbols. Strictly, there are two distinct components, data modelling and data coding. The decision to output a particular code for a given symbol (or set of symbols) is based on a *model* of the data which defines the symbol probabilities. Frequently, the theoretical distinction between modelling and coding is blurred and it is common to discuss image coding schemes which are really modelling schemes but perhaps include an implicit form of coding.

2.5.1 Data modelling

General purpose data compression generally uses one of two types of models, statistical and dictionary based. Statistical models encode each symbol based on its probability whereas dictionary systems replace entire strings of symbols by a single code.

2.5.1.1 Statistical modelling

The simplest statistical model is a *static* table of symbol probabilities. Representative blocks of data can be used to build such a table. In the case of image data no universal table of pixel intensity probabilities exists since, averaged over many images, the probabilities of all levels are equal. However, statistics can be gathered for each image and used to form a static probability table for that image. Since these image-specific statistics need to be transmitted to the decoder, this limits statistical models of this kind to order 0 or order 1 (i.e. blocks of only one or two pixels) since an order n model of a data stream with symbols having L values requires $\left(2^L\right)^n$ probabilities.

The solution to this difficulty is to employ *adaptive* models. Instead of requiring an initial pass through the data to generate statistics, the statistics are continually generated during compression as new symbols are encountered. The decoder encounters symbols in the same sequence as the coder and can therefore update its copy of the statistics in the same way. Such systems are inevitably poor for small data sets (since most of the message will have been transmitted before a representative symbol probability table can be built) but are appropriate for large data sets such as images. Various enhancements can be made such as assigning different weightings to old and recent symbols. Adaptive modelling is usually carried out with a finite context, using only a certain number, or order n, of previous symbols. The order is generally small since memory requirements grow exponentially with n.

2.5.1.2 Dictionary modelling

Dictionary based modelling schemes scan input data searching for groups of symbols which occur in a dictionary. When a match is found, only the dictionary index needs to be transmitted. Static dictionaries have similar disadvantages to static probability models, an overhead in forming the dictionary and the need to transmit the dictionary along with the compressed data. The solution is also similar - adaptive creation of the dictionary. Nelson [Nels 92] cites the use of acronyms as a simple analogy to this process. For example, the first time in this thesis that the Joint Photographic Expert Group (JPEG) is mentioned, both its dictionary definition and substitution string are defined. At any later point in the thesis only the substitution string is required.

2.5.2 Data coding

Data coding is the method for representing the chosen codes with as few bits as necessary to convey the information content. A coder is said to be optimal if every generated code has exactly the number of bits required by the information content of the coded symbol.

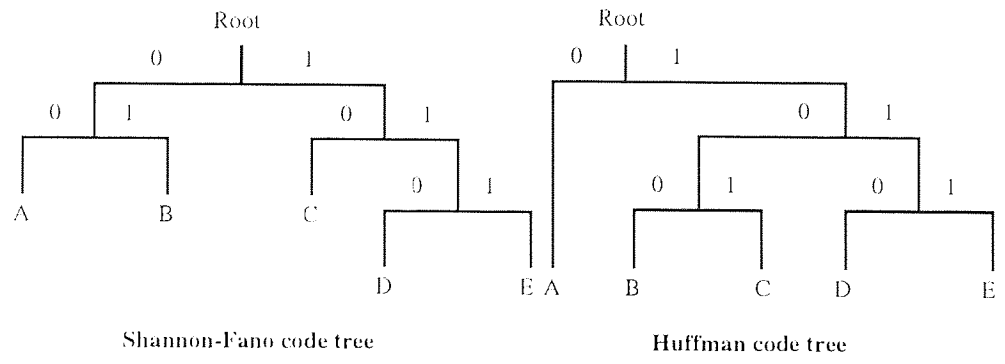
Efficient coding techniques based on the idea of data entropy have been known for over 40 years. Compression is achieved by taking into account statistical properties of the data. There are only three fundamentally different approaches, minimal redundancy or entropy coding (Shannon-Fano coding, Huffman coding), arithmetic coding and substitutional coding, but these have been modified to produce families of related schemes.

2.5.2.1 Shannon-Fano coding

Shannon-Fano coding [Shan 48] was the first well-known method for effectively coding symbols. It requires knowledge of the probabilities of the symbols in the message and assigns codes for each symbol with low-probability symbols having more bits and high-probability symbols having fewer bits. Although the symbol codes have different lengths they can be arranged as a binary tree giving unique prefixes so that they can be uniquely decoded. The codes generated are frequently but not always optimal. Fig. 2.6 shows the Shannon-Fano code tree for the short set of symbols A-E derived from a set of probabilities given in Table 2.3

			Shannon-Fano code		Huffman code	
Symbol	Count	Probability	Codeword	Bits	Codeword	Bits
A	15	0.3846	00	2	0	1
B	7	0.1795	01	2	100	3
C	6	0.1538	10	2	101	3
D	6	0.1538	110	3	110	3
E	5	0.1282	111	3	111	3

Table 2.3. Shannon-Fano and Huffman codes



Shannon-Fano code tree

Huffman code tree

Fig. 2.6 Shannon-Fano and Huffman code trees

2.5.2.2 Huffman coding

Huffman coding [Huff 52] shares all of the desirable properties of Shannon-Fano coding but the code tree is built bottom-up instead of top-down leading to a different set of codes. The Huffman code is optimal only when all symbol probabilities are integral powers of ½ because it uses an integral number of bits for each code. It is efficient when the symbols to be coded have widely varying probabilities and when there is a large data set to be coded. For a given frequency distribution, there are many possible Huffman codes, each giving the same total compressed length. A pure Huffman coder has to pass a complete set of symbol statistics to the decoder. As the coder collects more data and tries to achieve better compression the size of the statistics increases. However, in practice the symbol probabilities converge after a relatively small fraction of the total data. Huffman proved that his coding system could not be bettered by any integral bit-width coding scheme including Shannon-Fano coding. Since the computational complexity is similar, Huffman coding is therefore almost always used instead of Shannon-Fano coding.

Adaptive Huffman coding is also possible. An existing Huffman code tree can be updated to take account of a new symbol without having to rebuild the entire tree because Huffman trees exhibit the sibling property (a binary tree where the nodes can be listed in order of increasing weight with every node appearing adjacent to its sibling in this list). There are practical problems with Huffman trees for large data sets due to numeric overflow of the integers which are typically used to represent the codes but these can be solved by periodic but infrequent rescaling of the tree [Nels 92]. Huffman coding is extremely widely used both as a compression scheme in its own right and as a back-end entropy coder to more sophisticated compression systems such as JPEG.

2.5.2.3 Arithmetic coding

Huffman and Shannon-Fano codes must have integral bit widths so they are optimal only in the rare cases when the symbol probabilities are integral powers of ½. This problem is most apparent when some symbols have very high probabilities. For example, a symbol with probability 0.9 would have an optimum code size of 0.11 bits but a Huffman coder would have to assign a code of 1 bit. Arithmetic coding avoids this restriction by using a fractional number of bits for each coded symbol with the entire message having a single code which is represented by an interval of real numbers between 0 and 1. As the message becomes longer, the interval needed to represent it becomes smaller and smaller, and the number of bits needed to specify that interval increases. Successive symbols in the message reduce this interval in accordance with

the probability of that symbol. Arithmetic coding is not efficient for short messages but the efficiency approaches 100% for long messages.

Arithmetic coding requires a technique for building a model of the data which can be used by the encoder. The model can be static (e.g. a table of standard letter frequencies for English text) or adaptive, taking into account intersymbol probabilities as well as the symbol probabilities.

Arithmetic coding methods are computationally intensive compared with Huffman coding. On data sets containing some symbols of very high probability arithmetic coders can achieve significantly better compression than Huffman coders but, in general, the improvement is small [Witt 87]. For example, Pennebaker's Q-coder [Penn 88], which is acknowledged to be excellent, only achieves 5-10% better compression and, since it has been patented, is usually replaced with a Huffman coder to avoid licence fees.

2.5.2.4 Substitutional coding

Substitutional (or dictionary-based) compressors replace an occurrence of a particular symbol or group of symbols in a data stream with a reference to a previous occurrence of that symbol or symbols. Substitutional compressors can have static or adaptive dictionaries. A static dictionary is only appropriate when the data is known to be restricted to a narrow range of values and all well-known schemes use adaptive dictionaries. There are two main classes of schemes, both proposed by Ziv and Lempel, LZ77 [Ziv 77] and LZ78 [Ziv 78].

LZ77-based schemes maintain a lookahead buffer and move a fixed-size *sliding-window* over the data trying to find a match between the buffer contents and the window. Using LZ77 it is relatively simple to implement compressors which are fast [Nels 92] which has led to the development of many popular computer archiving programs (e.g. `lha`, `zip`, `zoo`). It is also used for data compression in quarter-inch tape drives which use the QIC-122 standard.

LZ78-based schemes build the dictionary from *all* previously encountered symbols instead of just a subset of them. In addition, dictionary entries can be formed by concatenation of new symbols with existing dictionary entries. A practical version of this scheme which can discard and rebuild the entire dictionary during compression is Welch's LZW scheme [Welc 84]. This is used in the UNIX `compress` program, the Graphics Interchange Format (GIF) standard for paletted colour images and the V.42bis standard which has been adopted by modem manufacturers to replace the previous MNP-5 Huffman-based standard.

2.6 Classification of image compression systems

The purpose of *image compression* is to design a model and coder which take advantage of the properties which are unique to images (i.e. they are two-dimensional data sets with features which have to be recognised by the human visual system). Given this additional knowledge, image compression systems should be able to perform significantly better than general purpose compressors.

2.6.1 Mathematical models of images

In his pioneering work Shannon [Shan 48] viewed communication signals as a series of symbols generated according to some stochastic process. This idea has been widely used in image processing. For example, images are often modelled as first order Markov processes, which are stochastic processes in which the future is determined by the present and is independent of the past. However, this is only valid for parts of images such as areas of fairly uniform texture (grass, clouds etc.). The parts of images which are badly modelled by such processes are the feature edges. These may well represent a small set of pixels compared to the total image size but they are most important for human perception [Marr 82].

Another image model based on metric spaces is also used in this thesis. This model is described in Chapter 7.

It is the difficulty in finding an accurate but mathematically tractable model for images which leads to the requirement for experimental work based on simulation to determine the effectiveness of image compression algorithms.

2.6.2 Lossless and lossy compression

Whereas general purpose data compression techniques are invariably lossless (i.e. they produce exact duplicates of the input data after compression and restoration), image compression may be lossy. The level of redundancy in images is so much greater than in other forms of data that it is acceptable to discard large amounts of data if is not of high importance. Data such as digitised images and digitised speech are imperfect representations of analogue phenomena (due to sampling and quantisation) to start with, so the idea of losing a certain amount of data during subsequent compression is not unreasonable. In particular, this becomes even more acceptable if the level of loss is known (or at least predictable) and can be traded against compression ratio in a controlled manner. Lossy compression does not permit perfect reconstruction of the original but it can provide very good quality at a fraction of the bit-rate. A well-

designed lossy system can minimise information loss (image distortion) for a given storage space or communication rate (i.e. when bits are scarce the lossy system can devote the bits to the data of most importance).

Image coding schemes may be broadly divided into three categories depending on which aspect of the image is used. Different transformations may be applied to the image data to move it into a more suitable domain for quantisation and codeword assignment. The categories are waveform coders, transform coders and model coders and are discussed in Sections 2.6.3 to 2.6.5. In these sections, common practice has been followed in referring to the systems as coders although the key differences are really in the way in which the data is modelled.

2.6.3 Waveform coders (spatial domain)

In waveform coders the image intensity or some simple variation, such as difference in intensities between pixels, is coded. Waveform coding has a major advantage in its simplicity, both conceptually and computationally. Since these techniques do not exploit features specific to any class of signals, they may be used for speech as well as images. For example, techniques developed for audio data such as pulse code modulation (PCM) and delta modulation (DM) can be used for image coding but are found to be less effective because images do not have the same repetitive nature as sounds. Detailed description of these techniques is deferred to Chapter 5 where they provide clearer contrast with the fractal waveform coding method described there.

Waveform coders also exhibit the highly desirable property not shown by other coding methods of graceful degradation of image quality at high compression ratios in contrast to the blockiness' apparent in the reconstructed images from transform coding.

In principle, any of the methods outlined in Sections 2.4 and 2.5.2 can be used for quantisation and codeword assignment but, for simplicity, scalar quantisation and uniform codeword assignment are usually employed.

Adaptive waveform coding of images requires a method of predicting later pixel values from information about previous pixels. The prediction is usually carried out based on only a few surrounding pixels. For example, Fig. 2.7 shows a target pixel X at the current position (x, y) and four previous predictor pixels *A*-*D*

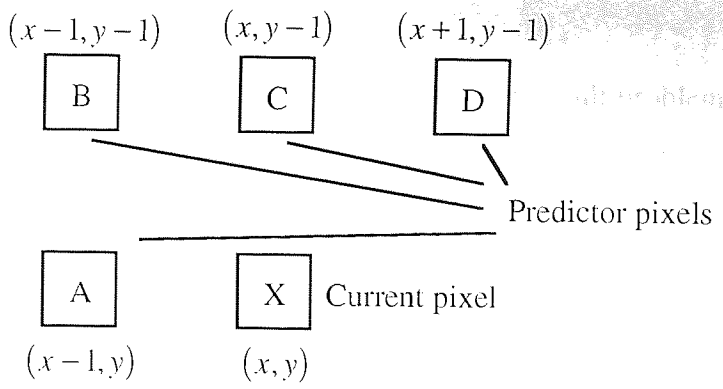


Fig. 2.7. Pixels used for waveform coding predictors

Even using only four pixels to form a prediction there are many possible predictors with some of the more common shown in Table 2.4. The lossless mode of the JPEG standard uses only pixels *A*-*C* as predictors with eight possible prediction equations which are also shown in Table 2.4.

4-pixel prediction equations	JPEG 3-pixel prediction equations
$X = A$	No prediction
$X = B$	$X = A$
$X = C$	$X = B$
$X = (A + C)/2$	$X = C$
$X = (A + D)/2$	$X = A + C - B$
$X = (A + (C + D)/2)/2$	$X = A + (C - B)/2$
$X = A + (C - B)$	$X = C + (A - B)/2$
$X = A + ((D - B)/2)$	$X = (A + C)/2$

Table 2.4. Possible predictor equations

The justification for using waveform coders depends on the redundancy present in the sequence of pixels. In waveform coding errors are causally related to the scanning mechanism. The scanning mechanism is analysed in Chapter 4 and fractal-based waveform coding is examined in Chapter 5.

Thus there are three issues which have to be addressed in a waveform coder:

- The scan order in which the data is processed.
- The way in which data is reduced (e.g. how samples are selected, how redundant data is discarded).
- Compression of the resultant reduced data (bit reduction etc.)

2.6.4 Transform coders

Transform analysis is a general technique for solving difficult problems by moving or *transforming* the problem into another domain where the problem is *easier* to solve. For example, Brigham [Brig 74] cited the example of solving a relatively complicated problem, long division, by taking logarithms and using the simpler operation of subtraction. In image transform coders the image is transformed into another domain (e.g. Fourier or Cosine) which is significantly different from the intensity domain and the transform coefficients are coded. The transformation itself does not compress at all but a transform may be selected such that many coefficients are either zero so they need not be transmitted or sufficiently close to zero that they may be quantised with fewer bits. Clearly any picture containing useful information is correlated (i.e. there is a fall-off in energy at high frequencies) so that, when transformed, the set of samples in each output block has values which are no longer correlated and most of the 'energy' (information) is contained in a small proportion of the block. 'Energy compaction' can then be achieved by rank-ordering the transform coefficients in order of decreasing significance.

Transform coding is generally handled in an open-loop process where, in principle, each block can be handled separately unlike the closed-loop sequential nature of DPCM. This has two advantages: the coder can be made adaptive to local image characteristics and the overall storage and computational requirements are reduced. Although smaller sub-image block sizes increase the efficiency, using blocks which are too small reduces the ability to exploit correlation among neighbouring sub-images. Typical block sizes used are 8x8 and 16x16.

The general structure of a transform coding system is shown in Fig. 2.8.

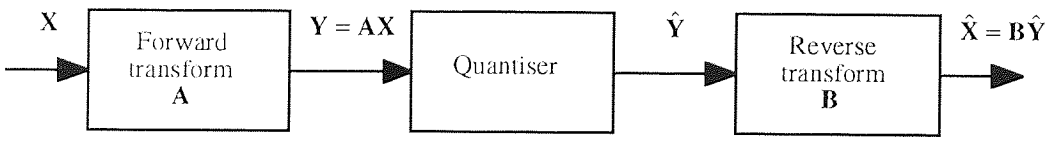


Fig. 2.8. Transform coding system

The input set of N pixels $\mathbf{X}$ may be either a one-dimensional set from the vertical or horizontal directions in the image or a rectangular (typically square) block. The input set is transformed by the N by N matrix $\mathbf{A}$ to give a set of N transform coefficients $\mathbf{Y} = \mathbf{AX}$. These coefficients are quantised to yield an approximation $\hat{\mathbf{Y}}$. At the receiver the quantised coefficients are re-transformed by the matrix $\mathbf{B}$ to yield $\hat{\mathbf{X}} = \mathbf{B}\hat{\mathbf{Y}}$.

For this process to work well it is desirable that the matrix **A** should have the following properties

- It should generate coefficients which are as uncorrelated as possible. This is a corollary to the requirement for energy compaction. The implication is that many coefficients can be discarded without significantly degrading the transformed image.

- It should be linear. i.e. allow a one-to-one mapping between the pixel and transform domains. Thus $\mathbf{B} = \mathbf{A}^{-1}$

- Ideally, it should be orthonormal. The energy in both domains should be the same so that no energy is lost or carried redundantly.

Some of the most important transforms are the discrete Karhunen-Loeve transform (KLT) (also called the Hotelling transform) [Andr 75], the discrete Fourier transform (DFT) [Brig 74], the Walsh-Hadamard transform (WHT) [Prat 69] and the discrete Cosine transform (DCT) [Ahme 74]. There are many variations on these : e.g. Haar, Slant, and singular value decomposition (SVD) but their performances are not sufficiently better for compression to make them widely used. There have been several comprehensive analyses made of image transforms (e.g. [Andr 75]) and their fast implementations (e.g. [Elli 83, Jain 89]). The conclusions are illustrated in Table 2.5.

Transform	Complexity for n by n block	Energy compaction	Notes
DFT	$O(n^2 \log_2 n)$	Good	Good
DCT	$O(n^2 \log_2 n)$	Excellent	Near-optimal for highly-correlated images
WHT	$O(n^2 \log_2 n)$	Good	Some objectionable distortion
Haar	$O(n^2)$	Fair	Poor compression
Slant	$O(n^2 \log_2 n)$	Good	Similar to WHT
KLT	$O(n^4)$ but can be approximated	Optimal	Used for performance evaluation. Best on average
SVD	$O(n^4)$ but can be approximated	Optimal	Used for performance evaluation. Best for a given image

Table 2.5. Comparison of transform coders

The complexity of the direct implementation of any of these transforms is $O(n^4)$ which is excessive for large images. However, if the transform is separable (i.e. it can be performed using two one-dimensional transforms), this reduces to $O(n^3)$. Most of these transforms (including DFT, DCT, HT, Haar and Slant) can be simplified still

further by expressing the transformation matrix as sets of products of several smaller matrices. This leads to fast algorithms for their implementation. In particular if the smaller matrices have just a few non-zero entries then the complexity reduces to $O(n^2 \log_2 n)$.

2.6.4.1 Theoretical transforms

The KLT has theoretical importance as it is the only transform which fulfils the three requirements identified above. The elements of the KLT transformation matrix are obtained by diagonalising the covariance matrix of the image data, giving completely uncorrelated coefficients. Also the variance of the KLT coefficients decreases in value monotonically with increasing order so it is optimal in having the best energy compaction properties. It is not used in practice because of the computational requirements. Even more importantly, the image statistics are different for each image block and require different transformation matrices for each coefficient. If a new matrix is required for each block then the overhead outweighs the advantages. The usual solution is to employ sub-optimal transforms which can use a fixed transformation matrix for all blocks. The SVD is similar to the KLT except it concentrates the most energy in a given number of coefficients whereas the KLT concentrates the average energy in a given number of coefficients.

2.6.4.2 Discrete cosine transform

Each of the practical transforms listed in Section 2.6.4 is useful in some area of image processing such as image enhancement, or motion estimation, but for image compression, the DCT has emerged as the clear leader.

Given an n by n image block $f(x,y)$ then the DCT of the block, denoted $F(u,v)$, where u and v are the x and y direction spatial frequencies, is given by Eqn. 2.12

$$F(u,v) = \frac{1}{\sqrt{2n}} C(u)C(v) \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x,y) \cos\left\{\frac{\pi u(2x+1)}{2n}\right\} \cos\left\{\frac{\pi v(2y+1)}{2n}\right\} \quad \text{.....(2.12)}$$

where $C(i) = \begin{cases} 1/\sqrt{2} & i = 0 \\ 1 & i \neq 0 \end{cases}$.

The DCT has all of the desirable transform properties, including an efficient algorithm requiring only real arithmetic. It may be viewed as giving a harmonic analysis of an image block with the DCT coefficients giving the relative amounts of the two-dimensional spatial frequencies in the input block. The first coefficient corresponds to no x or y direction variation and is thus known as the 'DC' coefficient with the

remaining coefficients being termed 'AC' values. Because images typically vary slowly across the relatively small blocks (typically 8 by 8 pixels) which are considered in transform coding, almost all the signal 'energy' is contained in the first few coefficients.

It is apparent that direct application of Eqn. 2.12 leads to an extremely inefficient method of calculating the DCT with complexity $O(n^4)$. However, many of the terms could be replaced by table lookup leading to a practical method using a pre-calculated cosine transform matrix **C** where:

$$c_{ij} = \begin{cases} \frac{1}{\sqrt{n}} & i = 0 \\ \sqrt{\frac{2}{n}} \cos\left[\frac{i\pi(2j+1)}{2n}\right] & i \neq 0 \end{cases} \quad \text{.....(2.13)}$$

Then, using matrix multiplication:

$$\mathbf{F} = \mathbf{CfC}^T \quad \text{.....(2.14)}$$

2.6.4.3 Zonal and threshold coding

Once the transform coefficients of an image block have been found a strategy is needed for deciding which coefficients to code and how to allocate the number of available bits among the coefficients. In zonal coding all coefficients within a pre-defined region of the coefficient matrix are coded. In threshold coding, the whole of the coefficient matrix is considered but only coefficients exceeding a fixed threshold are coded. This requires extra information about the locations of the selected coefficients which is typically provided by run-length coding the coefficients. If the transform has good energy compaction properties a regular sequence of coefficients may be used such as the zig-zag scan used in the JPEG system (see Section 2.8.4). Once the selected coefficients are chosen the total number of bits allocated to the block is divided among the coefficients proportionally to the expected variances of the coefficients. In the DCT, for example, the expected variance is much larger for low-frequency components.

2.6.5 Image model coding (parametric coding)

The term image model coding is usually only applied when a more sophisticated image model is derived and the model parameters are coded. The reconstructed image is then synthesised from the model parameters. Image model coding is still in its infancy but it is thought that future applications will be in areas where intelligibility of the images is

far more important that faithful reproduction of image intensities. For example, some success has been achieved with 'cartoon-like' images for sign-language in areas such as video phones for the deaf [Pear 90]. The potential compression attainable by such systems is significantly greater than is likely to be achieved by other methods but there is a limited number of applications where intelligibility is the only important factor.

There are specific areas within conventional coding where model coding can be applied. Many images contain areas of constant 'texture' where the image data varies in the region but is statistically very similar. If a model can be found for the statistical variation (repeat frequency, shading etc.) then it is sufficient to encode just the boundary of the region and a description of the texture. For example, it might be sufficient to code just sufficient information to specify that a certain region of an image contains blue sky with a few clouds and let the decoder decide where to place the clouds. This is closely related to the use of fractal 'forgery' in image synthesis where the object is to create realistic-looking synthetic images. This is not considered further in this thesis since the theme is the processing and compression of real image data.

2.7 Comparing image compression schemes

Numerous image compression methods are known. The choice of a particular technique for a given application depends on four main criteria: the choice of images; the required quality of the reconstructed image; the compression ratio; the speed of operation. To clarify evaluation of the techniques introduced in this thesis, Sections 2.7.1 to 2.7.4 describe how these factors are measured.

2.7.1 Choice of test images

Once the type of image has been established (see Section 2.2), the content of the image to be compressed clearly has a considerable influence on effectiveness of the compression techniques. The optimum choice of test image subject from the unlimited possibilities is an intractable problem. In some applications there is a clear choice (e.g. human head and shoulders for development of videoconferencing systems) but the issue is rarely so clear-cut. An analogous situation arises when vendors of computers publish benchmark performance data for their machines. The benchmark programs are usually independently designed to try and represent realistic tasks but commercial pressures have forced some vendors to optimise their compilers (and in some cases the hardware) specifically for the benchmarks [Patt 93]. For image compression, as an extreme example, one could design an algorithm (Fig. 2.9) which provides excellent data compression of one particular image but is otherwise not useful.

```
encoder:    IF image is the lena image THEN
                transmit '1'
            ELSE
                transmit '0'
                transmit uncompressed image
            ENDIF
```

Fig. 2.9. Image-specific compression algorithm

This type of problem has to be avoided by using a set of significantly different test images. There are advantages in using 'standard' images since they are readily available and their use permits direct comparison with other workers' results.

In this thesis two sets of images are used extensively.

- A primary image set of three well-known test images having significantly different properties (see Table 2.6). This set is used to develop algorithms and to give comparisons with other work. The images are shown in Appendix A.

Image Name	Notes
LENA	The 'classic' test image. Mostly easy to compress and reconstruct, with some regions of fine detail. The version used here is derived from the original RGB format.
BOATS	An intermediate difficulty image. The sharp edges of the boat masts need to be picked out from the plain background. Legibility of the boat name is also a good test.
BARBARA	A difficult image to compress. The fine pattern of stripes on the clothes causes trouble for all coders. There are large regions of fine detail with sharp edges.

Table 2.6. Summary of primary test images

- A secondary set of 26 images of widely-varying subjects (fine-art, landscapes, portraits, machinery, sports scenes etc.) collected from generally available sources including several image processing research laboratories and Usenet newsgroups. It is not known in all cases how each image was originally captured but it can be argued that, in a sense, this makes the set more useful since it introduces greater diversity. To prevent the introduction of unnecessary errors due to re-sampling and re-quantisation, original images were chosen which had a resolution of at least 512 by 512 and at least 256 grey levels. Higher resolution images were cropped from the centre rather than rescaled. This larger image set has been used to fine-tune algorithms and to generate more meaningful statistics once algorithms have been optimised on the small primary image set. The secondary image test set is shown in full in Appendix A

2.7.1.1 Image complexity

It is useful to have some objective measure of image complexity in order to judge the effectiveness of any compression. The usual approach is to compute the 'entropy' or information content of the image.

Shannon showed that the entropy H of a set of symbols with L possible values is given by:

$$H = -\sum_{i=1}^{L} P_i \log_2 P_i \quad \text{.....(2.15)}$$

where P_i is the probability that the symbol value is i.

Unlike thermodynamic entropy, there is no absolute number which represents the absolute information content of a message. When information entropy is calculated, the symbol probabilities depend on the model not just on the data so, for example, the probabilities depend on which order entropy is being used.

H can be interpreted as the average amount of information contained in each symbol. The zero'th order entropy of the image is obtained by considering only individual pixel intensities. However, it is not sufficient to consider just single pixel values, since images contain features which extend over many pixels. Better estimates can be obtained by examining the frequency distributions of blocks of adjacent pixels. For example the first order entropy is obtained by considering all consecutive pairs of pixel intensities. As the block size n increases the estimates provide better approximations to the entropy of the image. Table 2.7 shows the zero'th and first order entropies of the primary test set images.

Image Name	Zero'th order Entropy / bits-per-pixel	First order Entropy bits-per-pixel	Compression ratio lossless JPEG	Compression ratio lossy JPEG (Q50)
Lena	7.25	5.95	1.86	14.26
Boats	7.07	5.94	1.74	11.15
Barbara	7.51	6.66	1.60	9.00

Table 2.7. Complexity of primary test images

Determination of image complexity can be achieved by an inverted argument. If a standard image compression algorithm such as JPEG (which is discussed in Section 2.8) is applied, then the compression ratio achieved is a good indication of the image complexity. There are dangers in this approach. For example, Table 2.7 shows that the compression ratio achieved with the lossless JPEG model does not vary greatly between images but the lossy JPEG model achieves quite different compression ratios.

These problems with determining a good measure of image complexity support the case for finding a more appropriate measurements such as the fractal dimension which is discussed in Chapter 3.

2.7.2 Measurement of image quality

One of the key measures of any image compression system is the quality or fidelity of the restored image. The requirements for the quality of the reconstructed images produced by a compression system depend on the application. Some data, such as historical records and space probe data, may be irreplaceable or very expensive to obtain in which case lossless or information-preserving coding is required. In other applications, such as entertainment television, high subjective quality is of paramount importance but much of the original information may be lost so long as the result is acceptable to human viewers. There are also applications, such as remotely piloted vehicles and robot vision where intelligibility is overwhelmingly important but visual quality is not. However, in each case, the lower the acceptable quality and intelligibility the lower the required bit rate.

It is necessary to have some reproducible and objective measure of image quality. Clearly it is desirable for this measure to reflect the perceptual quality or usefulness of the restored image in a particular application. Unfortunately, the complexity of coding schemes required, especially at very low bit-rates, inevitably results in radically different errors and unwanted image artefacts which are not directly comparable between schemes. For example, it is hard to predict whether a group of human observers will prefer a generally sharp image with a few major defects or a rather blurred image with no other defects. There is no easily computable measure which is accepted to accurately represent human preferences for image quality.

An additional desirable property of distortion measures is ease of computation and tractability in further analysis. This explains the popularity of measurements based on the squared error since there is a wealth of theory and associated numerical methods which are based on optimising the mean squared error. Definitions of squared error based measurements including the mean-squared-error (MSE), the signal-to-noise ratio (SNR) and the peak signal-to-noise-ratio (PSNR) for an M by N pixel source image f and restored image f' with source image variance σ^2 are given in Table 2.8.

Distortion measure	Definition
Mean squared error	$\mathrm{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left\|f(i,j) - f'(i,j)\right\|^2$
Mean signal-to-noise ratio	$\mathrm{SNR} = 10\log_{10}\left(\frac{\sigma^2}{\mathrm{MSE}}\right)$,
Peak signal-to-noise ratio	$\mathrm{PSNR} = 10\log_{10}\left(\frac{(\text{peak value of source image data})^2}{\mathrm{MSE}}\right)$

Table 2.8. Squared-error based distortion measures

SNR is commonly used in general signal processing applications but, for image processing, PSNR is more common. It has even been suggested that this is because the figures look better (typically PSNR is 4-6 dB greater than SNR)! There are problems with the squared error. For example, it gives a large numerical distortion for a small spatial or intensity shift in the image even though these errors would give no visual distortion. Equally, a single damaging but highly localised visual defect may only give rise to a small average distortion. MSE does not take into account masking effects of high activity near edges or the subjective significance of coherent errors like block structure, ghosting and contouring.

In general, it is found that MSE measures are better predictors at higher bit rates but poor at low bit rates. A reasonable compromise which can be taken is to design a system to minimise the mean squared error but then use a more complicated distortion measure to evaluate the quality of the system.

For these reasons a wide variety of objective image quality measures have been proposed [Ahum 93] which attempt to incorporate knowledge of the human visual system (sensitivity to edges, insensitivity to textures, masking effects). Such methods include the absolute error and the cube root of the sum of the cubed errors (L_3). None of these measures has had wide acceptance

Use of subjective measures is a potential minefield, frequently leading to inconsistent results. Attempts have been made to design objective measures which predict subjective measures for applications such as entertainment video. Typically [Marm 86] this would involve plotting the objective measure against scores awarded for subjective quality and then trying to fit a curve to the resultant points.

Cosman *et al* [Cosm 94] made a detailed analysis of objective and subjective measure of image quality of compressed medical images. They made the surprising and controversial conclusion that, in many cases, the compressed and restored images are superior to the originals when judged subjectively (by medical experts - not image

processing experts). It is thought that this is mainly due to the inherent noise reduction when clustering methods such as vector quantisation are used. As far as objective measures are concerned they make a strong case for the use of segmental SNR (SSNR) [Jaya 84] which was developed as a measure of coded speech quality, since it averages out small distortions over large areas and large distortions over small areas. SSNR is calculated by dividing the image into blocks of small to medium size (e.g. block sizes from 2 x 2 to 32 x 32 pixels), calculating the SNR for each block and then averaging the values for each block size.

To test the predictive value of each of these quality measures, images in the primary test set (see Section 2.7.1 and Appendix A) were distorted by a variety of effects (random pixel variation, JPEG compression, coarse quantisation) and each objective distortion measure was applied. The parameters were chosen to give a range of quality values representative of compressed images derived by the methods discussed in this thesis. The results averaged over the three test images are summarised in Table 2.9.

Distortion by random pixel variation			
variation	**PSNR / dB**	**SNR / dB**	**SSNR / dB**
0.01	44.84	40.50	28.85
0.02	38.67	34.33	25.26
0.04	32.71	28.37	21.29
0.08	26.72	22.38	17.82
Distortion by coarse quantisation			
grey levels	**PSNR / dB**	**SNR / dB**	**SSNR / dB**
64	46.39	42.05	29.88
32	40.93	36.58	26.44
16	34.91	30.57	22.76
8	28.88	24.54	18.94
Distortion by JPEG compression			
quality factor	**PSNR / dB**	**SNR dB**	**SSNR / dB**
90	38.93	34.39	25.18
50	34.11	28.77	22.23
25	30.67	27.33	20.69
10	28.24	23.90	18.56

Table 2.9. Summary of restored image quality measures

The absolute value of each objective quality measure evaluated in Table 2.9 is less important than achieving an equal ranking between the objective and subjective measures. The distorted versions of each of the images were ranked in order of subjective quality by a group of non-specialist observers using a 'bubble-sort' of paired comparisons. Fig. 2.10 shows the mean ranking obtained from each of the objective quality measures plotted against the mean subjective ranking. It is not necessary also to evaluate SNR as a predictor since, for a given image, it is proportional to the PSNR.

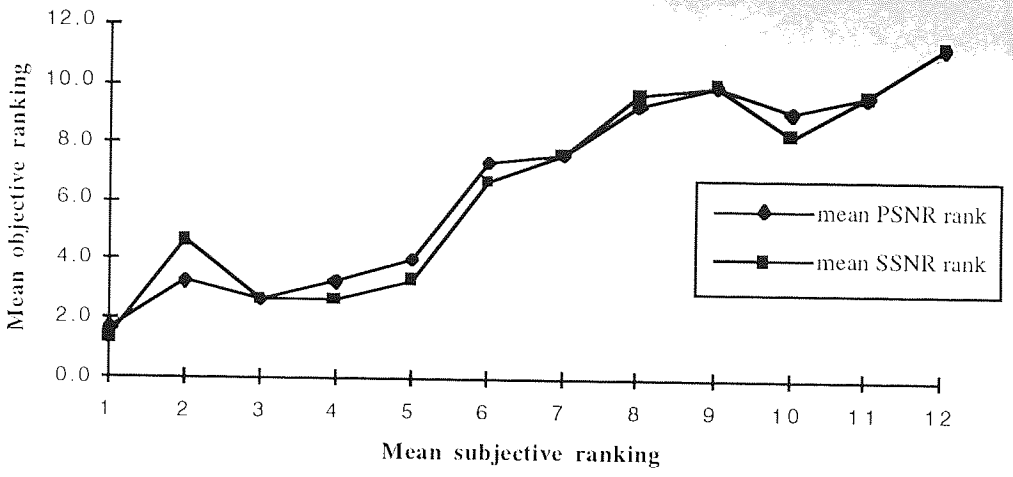


Fig. 2.10. Subjective quality of distorted images

Some interesting results can be seen from close examination of the statistical data. For example, images with 2-4% random pixel variation were nearly always preferred by observers to images with only 1% random variation! The data is considered reliable since control experiments were introduced by the presence of image pairs which ought to be indistinguishable and such image pairs were consistently ranked equal. Overall it can be seen from Fig. 2.10 that neither PSNR nor SSNR achieves completely uniform objective and subjective ranking but that both measures do provide approximate predictions of subjective acceptance. The closest straight line fit is for the SSNR but the PSNR is almost as good so, to facilitate comparison with other work, the simplest and most common measure, PSNR, is used for the experimental results reported here. Where the term SNR is used it should be taken to mean the PSNR unless specifically stated otherwise.

2.7.3 Compression ratio

Data compression is the process of reducing the amount of data required to represent a given amount of information. Data is not synonymous with information. Differing amounts of data may represent the same amount of information. Data which adds no relevant information or duplicates existing information is said to be redundant. If two data sets contain n_1 and n_2 information units then the compression ratio C_R is: $C_R = n_1 / n_2$

Of course, when compressing images, the true information content of each image is unknown so a rather more practical measurement is used. The compression ratio is simply defined as the ratio of the original image size (which is always 262,144 bytes

for experimental work reported here) to the compressed image size as given in Eqn. 2.16.

$$C_R = \frac{\text{Size of original image}}{\text{Size of compressed image}} \quad \text{.....(2.16)}$$

Results quoted do not necessarily correspond to the actual size of the compressed files to which, for simulated systems, it is not necessary to perform optimal packing of compressed data into bytes. Instead the results are quoted as if this packing has been performed. In addition, unless specifically stated, the results do not assume that any further compression such as entropy coding (see Section 2.5.2) has been applied to the compressed data.

It should be noted that there is also some disagreement over whether compression ratio is a useful measure. It has been suggested that the absolute size of the compressed image which can be reconstructed to give a perceptually acceptable image is more useful. The reasoning is that initial images with very high resolution can be compressed greatly, giving high quality reconstructed images but still requiring excessive storage for the compressed data. In contrast, an initial image with low resolution will be difficult to compress significantly. Perhaps unsurprisingly, this argument is promoted by proponents of scale-independent image compression schemes such as iterated function system compression which is discussed in Chapter 7.

2.7.4 Coding and decoding speeds

Clearly, in order to be useful, any image data compression must be achievable in a finite and preferably short processing time. In the past it has been important to ensure that a coding scheme is practical to implement in hardware. This issue is now becoming less significant with developments such as: reconfigurable hardware [Atha 94] which is blurring the distinction between hardware and software implementations; the embedding of high-performance microprocessors into systems to execute complex algorithms.

Where the time taken to perform image compression and decompression is an important factor it necessary to provide some measure of the complexity of the computation. Obviously no measure based on the speed of a particular processor, such as the Sun workstation as used here, is likely to be useful. Neither is it helpful to give results which only apply to a specific size of images. The appropriate measure is the computational complexity of the algorithms which provides an indication of the rate of growth of the computation time as the scale of the problem increases.

In this thesis the computational complexity of the algorithms presented is described with the 'order notation' proposed by Knuth [Knut 76]. If f and g are functions of the integer n which represents the scale of the problem then the order of $f(n)$, denoted $O(f(n))$, is the set of all $g(n)$ such that there exist positive constants c and n_0 such that $|g(n)| \leq cf(n)$ for all $n > n_0$. For example, complexities of $n^2/2$ and $13n^2 + 5n$ are both $O(n^2)$. It is helpful to think of $O(f(n))$ as meaning order *at most* $f(n)$. For example, a complexity of $4n$ is also $O(n^2)$, which may be seen by setting $c = 2$ and $n_0 = 1$. In general, algorithmic complexities greater than $O(n^3)$ rapidly become impractical whereas algorithms with extremely low growth rates such as $O(\log_2 n)$ are extremely useful.

For each algorithm the relative speeds of compression and decompression also need to be considered. Speed of compression usually matters much more if data is to be transmitted rather than stored. The decompression speed is quite important for storage and retrieval and is vital for reception of transmitted data. Some compression methods are approximately symmetric with respect to the compression and decompression speeds. Others, such as IFS coding, examined in Chapter 7 are extremely asymmetric, with compression times currently exceeding decompression times by several orders of magnitude.

2.8 Current image compression methods

Current image compression methods are split into two categories. The first category is methods which have been adopted as standards. These are well-understood and have performances which are generally good but not outstanding. They are, however, robust. Current image compression standards have been developed by the ISO and the CCITT organisations to encompass binary, grey-scale and colour images. The state of the art as far as standards are concerned is the JPEG standard.

The second category consists of areas of current research. Current research work in image compression is focused on wavelets, object-based coding, vector quantisation (VQ) and fractal image compression. The last two topics are closely related to each other and are addressed in this thesis in Chapters 6 and 7.

2.8.1 General purpose data compression of images

Although general purpose compression techniques outlined in Section 2.5 perform well on data such as text they are not very effective for compressing continuous tone images. Statistical methods fail because pixel intensities tend to be quite uniformly distributed and dictionary methods fail because the repeated features, which might

otherwise be good candidates for dictionary entries, tend to be slightly different each time they are encountered so that only short matching strings can be found. Table 2.10 shows the performance of several general-purpose compression systems on the image data. In all cases the compression is poor. The best compression ratio achieved is 1.32:1 by `gzip` on the BOATS images. Clearly, general-purpose compressors are not useful for image data which is why compression schemes have to be devised specifically for images.

Compression name	Basic compression scheme	LENA compressed size/Bytes	BARBARA compressed size/Bytes	BOATS compressed size/Bytes
none	raw image	262144	262144	262144
UNIX `compress`	LZW	219893	259187	217457
lzh, lha	LZ77	218866	235557	203742
gzip	LZ77	212930	232157	198978
zoo	LZ78	240941	282201	225062

Table 2.10. Performance of general-purpose compression systems

2.8.2 Elementary image compression

It is possible to achieve significant image compression simply by sub-sampling either in the spatial domain or in the intensity domain. The effectiveness of this depends on the resolution of the source image and the requirements of the display. For example the three images in the primary test set have been sub-sampled at various resolutions or block sizes and then (linearly) interpolated back to their original resolution. Fig. 2.11 shows the resulting compression and distortion resulting from this simple process.

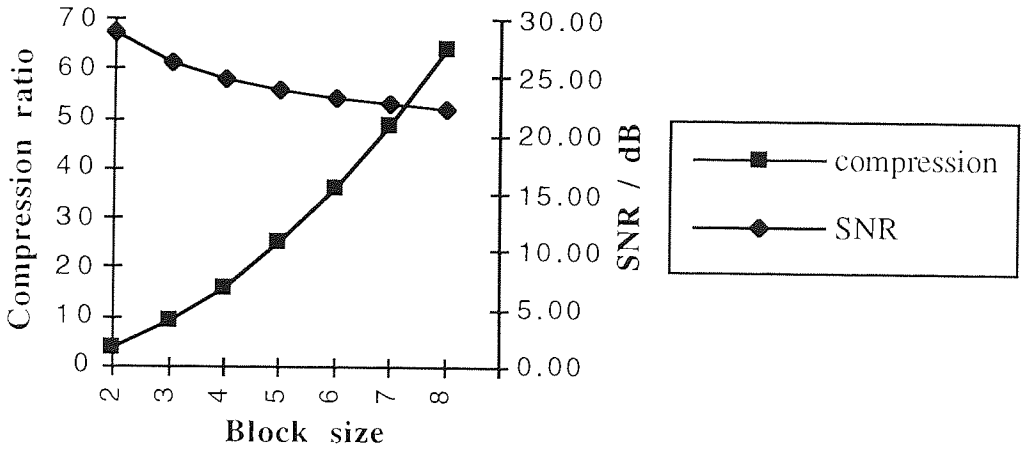


Fig. 2.11. Compression by sub-sampling

The objective measure of SNR makes this seem effective but, subjectively, the restored images have most noticeable 'blockiness' as shown in Fig. 2.12 where the block size is 8 by 8 pixels.

Fig. 2.12. Blockiness due to sub-sampling

Similarly, the three primary test images have been coarsely quantised. Fig. 2.13 shows the resulting compression and distortion resulting from this simple process. Again, in spite of an apparently good objective measure of SNR, subjectively, the restored images have most noticeable contouring as shown in Fig. 2.14 where the image is quantised to only 8 grey levels.

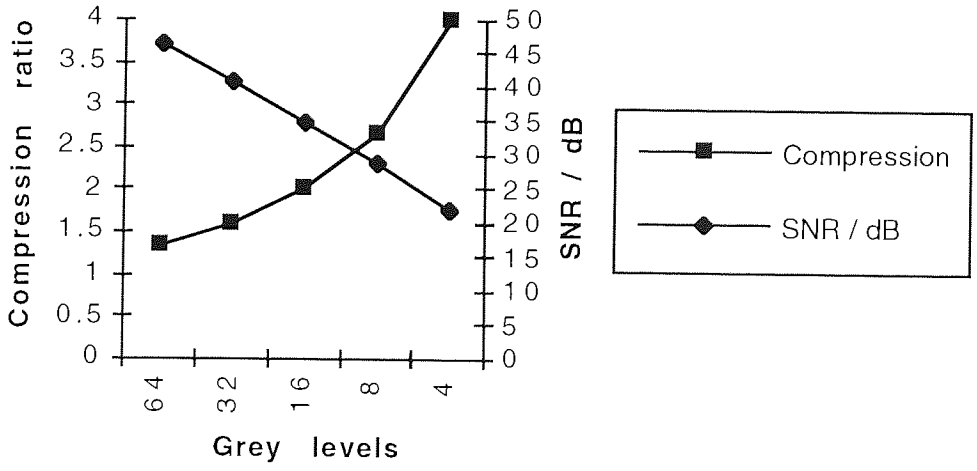


Fig. 2.13. Compression by quantisation

Fig. 2.14. Contouring caused by coarse quantisation

The significance of these results is that considerable compression can be achieved very simply using only sampling and quantisation. If the compression achieved in this way is sufficient for a particular requirement then there is no need to resort to any of the more complex image compression schemes.

2.8.3 Binary image compression standards

2.8.3.1 CCITT Group 3 and Group 4 FAX

The most widely used standards are the CCITT Group 3 [Fax3] and CCITT Group 4 [Fax4] standards. The Group 3 system uses non-adaptive one-dimensional run-length-coding in which some lines are coded two-dimensionally. The Group 4 system uses only two-dimensional coding. In both cases the two-dimensional coding is based on relative address coding (RAC) [Gonz 92]. The CCITT used a set of eight representative documents to test the standards. On these documents the compression ratio achieved is approximately 15:1 [Urba 92].

2.8.3.2 JBIG

A further committee, the Joint Bi-level Image Experts Group (JBIG) has produced an improved standard which performs 10-15% better than CCITT Group 4 on text and line-art and copes better with half-tone images and other documents which are sometimes expanded by the original FAX standards [Arps 94]. JBIG can be used on grey-scale or even colour images by applying the algorithm to one bit-plane at a time but the overhead in processing the separate bit-planes means that beyond about 6 bits per pixel the JPEG standard (Section 3.8.4) works better.

JBIG uses predictive coding from a set of nearby (but previously scanned) pixels called the template. An example template might be the two pixels preceding the current one on the same line, and the five pixels centred above this pixel on the previous line. The pixel difference is then arithmetically coded based on the state so formed. The specified arithmetic coder is the Q-coder [Penn 88] but the standard permits use of a Huffman coder to avoid licence fees.

2.8.4 Continuous-tone image compression standards

There are numerous widely-used proprietary image file formats such as the popular GIF format. These are usually lossless systems, mostly based on run-length encoding or substitutional encoding which are employed more as a matter of convenience and to distinguish rival image processing software than to perform real image compression. For example a GIF-compressed image typically has the same storage requirement as it would if compressed with a general purpose data compression system. There is only one standard, JPEG, which has been universally adopted and which marks the current 'state-of-the-art'.

2.8.4.1 JPEG

The Joint Photographic Experts Group (JPEG) is an experts group of ISO, IEC and CCITT members which has defined a compression standard for image coding [Wall 91, Penn 93]. JPEG is designed for compressing either full-colour or grey-scale digital images of 'natural', real-world scenes. The JPEG standard defines a 'baseline' lossy algorithm and four modes of operation:

- Sequential encoding. Each image is independently compressed.
- Progressive encoding. This is intended to support real-time transmission of images. It allows the DCT coefficients to be sent incrementally in multiple 'scans' of the image. With each scan, the decoder can produce a higher-quality rendition of the image. Thus a low-quality preview can be sent very quickly, then refined as time allows.
- Hierarchical encoding. This allows images to be compressed at multiple resolutions. The higher-resolution images are coded as differences from the next smaller image.
- Lossless encoding. This does not use the DCT, since roundoff errors prevent a DCT calculation from being lossless. The lossless mode uses predictive coding chosen from one of eight specified predictor functions. The sequence of

differences between actual and predicted values is encoded using Huffman coding as in the lossy mode. The lossless mode is particularly appropriate for the final scan in a hierarchical sequence.

It is worth outlining the baseline compression algorithm shown in Fig. 2.15 since it is essentially a sequence of the same transformations, quantisations and encodings which are discussed in Sections 2.4 to 2.6.

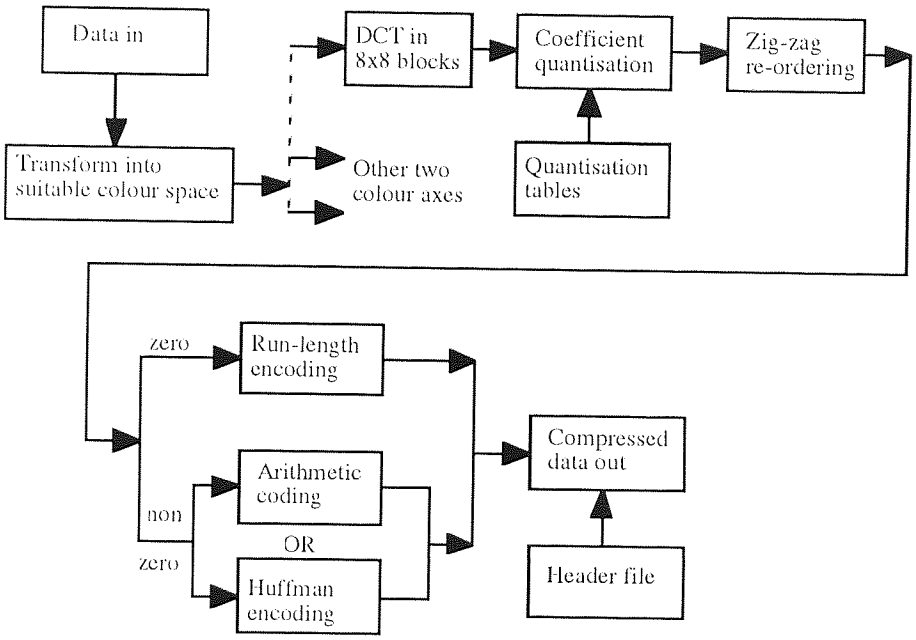


Fig. 2.15. JPEG image compression

1. The image is transformed into a suitable colour space. This is irrelevant when the image is grey-scale but for colour images it is preferable to transform from the red-green-blue (RGB) space into a luminance/chrominance colour space. Some compression can then be applied along the chrominance axes by undersampling as the human eye is not as sensitive to high-frequency colour information as it is to high-frequency luminance. The remainder of the algorithm operates on each colour component independently but if the colour space is not changed compression will be less since each component is coded at luminance quality.

2. In each colour axes the pixel values are grouped into 8x8 blocks. The DCT of each block is calculated.

3. For each block, each of the 64 frequency components is divided by a separate 'quantisation coefficient', and the results are rounded to integers. This is the fundamental compression step. A quantisation coefficient of 1 loses no information; larger coefficients lose successively more information. The higher frequencies are

normally reduced much more than the lower. All 64 quantisation coefficients are parameters to the compression process and can be defined for each image but most existing coders use simple multiples of the example tables given in the JPEG standard.

5. The reduced coefficients are encoded. JPEG coding is so effective that many (typically more than half) of the coefficients are reduced to zero during the quantisation stage. The zero values are coded with a run-length coder and the non-zero values are coded using arithmetic or Huffman coding. The specified arithmetic coder is the patented Q-coder [Penn 88] so it is rarely used in practice. As a further refinement, the coefficients are scanned in a zig-zag order (see Fig. 2.16) thus selecting coefficients in order of probable energy content and maximising the expected run lengths of zeros.

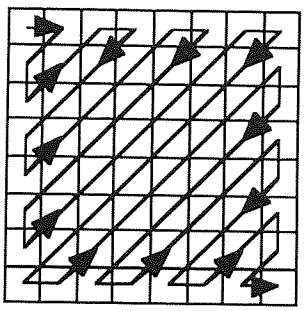


Fig. 2.16. JPEG coefficient block scan order

6. Appropriate headers are added to produce an 'interchange' JPEG file. All of the compression parameters are included in the headers so that the decompressor can reverse the process but for specialised applications, where the decoder knows which parameters were used, this overhead of several hundred bytes may be omitted.

The decompression algorithm reverses this process and typically adds some smoothing steps to reduce pixel-to-pixel discontinuities at the block boundaries.

2.8.4.1.1 Performance of JPEG encoding

The JPEG compression process has numerous parameters which can be used to trade off compressed image size against reconstructed image quality over a very wide range. In practice these parameters are usually derived from a single value called the 'quality' setting which varies in the range 0 to 100. The way in which these parameters are derived varies between JPEG implementations so no direct comparisons can be made between different implementations.

The performance of a typical JPEG coder [Imag 94] has been evaluated here with results as shown in Fig. 2.17. This shows the mean performance over the three

primary test images with Q varying from 10 to 100. It can be seen that the SNR is approximately proportional to, and the compression ratio approximately inversely proportional to, the quality setting except for the two extreme values. The discontinuity at Q=100 is caused by the coder switching to the lossless mode and the smaller discontinuity at Q=10 is caused by excessive rounding during the quantisation stage. In most implementations the useful range of JPEG quality settings is approximately 25 - 90.

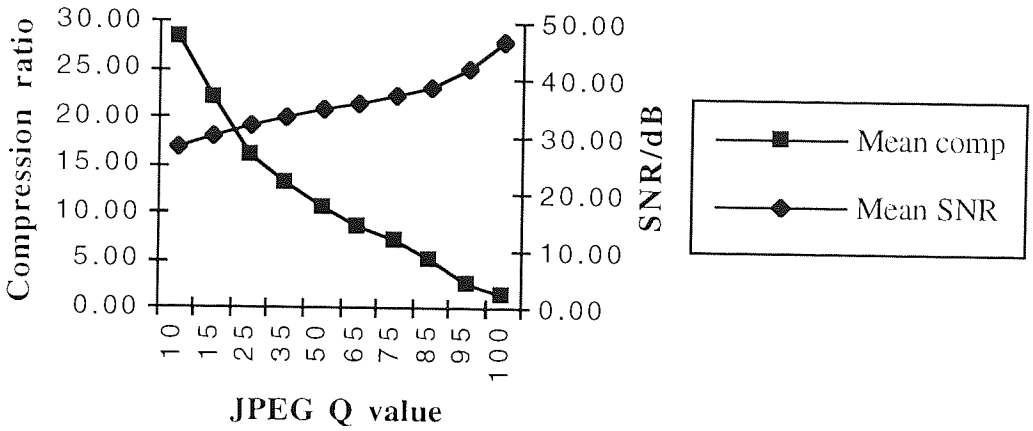
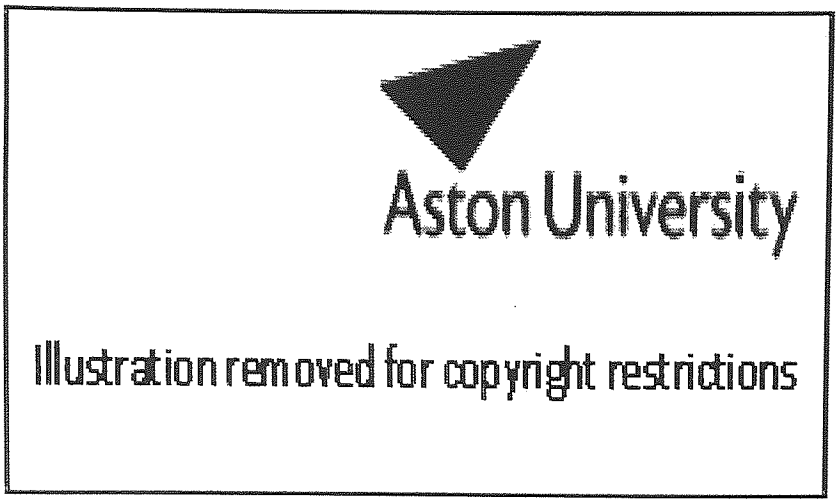


Fig. 2.17. Performance of a typical JPEG compressor

From Fig. 2.17 it is clear that the JPEG compressor achieves very good results (30 dB SNR at a compression ratio of 20:1) but at the higher compression ratios the subjective quality is poor with considerable 'blockiness' in the restored image. For example, Fig. 2.18 shows detail from the Lena test image when compressed at a ratio of 50:1 using JPEG. It can be observed that there is obvious degradation due to each block being represented by only a few DCT coefficients. In addition artefacts with a ripple-like appearance are seen around edges. This effect, the Gibb's phenomenon associated with 'ringing' in the frequency response of the human visual system is inevitable with frequency-domain-based compression.



Detail from original image **Detail from compressed image**

Fig. 2.18. LENA compressed at 50:1 with JPEG encoding

JPEG has been adopted as the image compression system for the Adobe PostScript printing system language, the raster image part of the ISO Document interchange format, the CCITT colour FAX standard and the European videotext standard.

2.9 Summary

Data compression is a well-developed topic and provably optimal compression can be achieved for certain types of data. Compression systems may be broadly divided into lossless and lossy systems. There are applications where only lossless compression can be used and, for such data, techniques such as Huffman coding, arithmetic coding and substitutional coding can provide significant compression ratios. Images do not compress at all well using lossless compression techniques. Fortunately, a digital image, being an approximation of analogue data to start with, is an excellent example of data for which lossy compression is appropriate.

There is a need for scepticism with regard to claims made for the effectiveness of proposed image compression schemes. It is enlightening to compare compression ratio and distortion figures with Fig. 2.11 which shows the objectively (but not subjectively) good performance of an elementary compression method such as sub-sampling.

Although well-established, lossy image compression has become so important that considerable development work is still taking place. The pressure for a working standard has resulted in the JPEG standard which has rapidly become widely adopted. It gives good compression in the range 10:1 to 20:1 with subjectively acceptable distortion and objectively measured distortion in the range 30-35 dB SNR.

JPEG compression is computationally demanding. Hardware advances have alleviated this problem considerably but there is still a requirement for conceptually and computationally simple compression techniques. In Chapter 4, novel scanning strategies are presented which can improve many existing image compression schemes with little increase in complexity. A further, computationally simple, fractal compression technique is presented in Chapter 5.

At higher compression ratios greater than approximately 20:1 artefacts caused by JPEG compression become subjectively unacceptable. Vector quantisation offers hope for significantly higher compression if the computational difficulties can be solved. This is addressed in Chapter 6.

A further drawback with JPEG is its resolution dependence. Even though it provides a hierarchical mode, this is not the same as resolution-independence. Any attempt to display the image at a resolution higher than the highest coded resolution still results in blockiness from the required pixel replication. It will be seen in Chapter 7 that is not the case when iterated function system fractal-based methods are used.

Most research is applied to incremental developments of themes which have been known for many years. The demand for significantly better compression systems requires a more radical approach. Although the idea is controversial, the concept of using fractals for image compression appears to offer a solution.

Chapter 3. Fractals and Images

3.1 Introduction

This thesis is concerned with the 'applications of fractals to image compression'. In Chapter 2 it was shown that the problem of image data compression reduces to two separate problems: (1) modelling the image data so that redundancy may be determined and (2) coding the data according to the model so as to require the minimum storage. In this chapter it is shown that images can be modelled as fractal data and that this interpretation leads to new ways of compressing images.

In addition fractals provide a natural tool for applying conventional image data compression techniques in a systematic and hierarchical way to the complexity of image data which is present at many scales of measurement. Here, some of the ways in which fractal forms arise from image data are identified. The procedures used to determine whether data is fractal are also used to measure the complexity of the data and it is shown how this relates to the conventional complexity measure of entropy.

This chapter gives a brief and non-rigorous description of the properties of fractals together with notation sufficient to explain the image compression methods explored in Chapters 4 to 7.

3.2 What is a fractal?

Unfortunately it is not easy to give definition of a fractal. Many, otherwise rigorous, mathematics texts evade this issue since any particular definition seems either to exclude sets which are usually thought of as fractals or to include sets which are not. The reasoning is well expressed by Falconer who wrote:

> "My personal feeling is that the definition of a fractal should be regarded in the same way as the biologist might regard the definition of life. There is no hard and fast definition, but just a list of properties characteristic of a living thing. ... Most living things have most of the properties - but there are exceptions to all of them." [Falc 90]

For a fractal set F such a set of properties might include:

(1) F has a fine structure. i.e. detail at arbitrarily small scales.

(2) F is too irregular to be described by traditional geometrical language - both locally and globally.

(3) F may have some form of self-similarity, perhaps approximate or statistical.

(4) Usually the fractal dimension $\mathfrak{D}$ of F (however defined) is greater than its topological dimension. Definition and practical measurement of the fractal dimension are considered in Section 3.5.

(5) In most cases F is defined in a simple way, perhaps recursively.

Since fractals are only defined here in such loose terms it is appropriate to give an example of an object which is considered to be fractal. One of the most famous examples is the coastline of Great Britain. Mandelbrot [Mand 67] compared the lengths of coastlines on various maps. He observed that as the maps became more and more detailed the measured length of the coastlines increased in a regular way. Thus there is no single coastline length; it depends how it is measured. However, the rate of increase of the coastline length at different levels of magnification can be measured.

The procedure is to measure the length at various scales and plot the logarithm of the length against the logarithm of the unit measure (i.e. the smallest measurable unit at a given resolution). Any logarithm base may be used, provided it is the same for both axes, since it is the ratio which is important. The slope of the plotted line is negative since, as the scale increases, the length decreases. Informally, the fractal dimension is one (the topological dimension of a line) minus the slope of the plotted line. This is illustrated in Fig. 3.1. The fractal dimension of the coastline of Britain, measured in this way is approximately 1.28. A more practical method of measuring the fractal dimension of image data is described in Section 3.4.

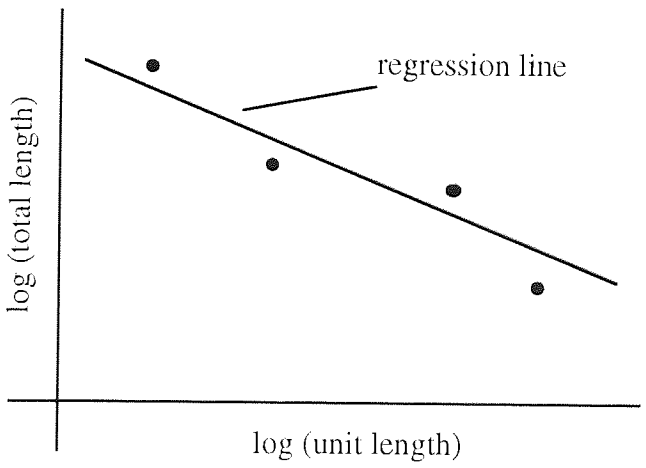


Fig. 3.1. Experimental measurement of fractal dimension

Measurement of coastlines may seem a long way from image data compression but it will be shown in Chapter 5 that exactly the same process may be used to 'measure' image data at various scales. If a simplified version of the data is created at a chosen scale, and detail at smaller scales is ignored, then a compressed version of the image is produced.

3.3 Sources of fractals

Fractals are not merely an elusive mathematical concept. Dirac reputedly remarked that if a theory is mathematically beautiful and elegant, it is inconceivable that nature does not have a use for it. It could certainly be argued that is the case for fractals. Real-world objects and processes often exhibit fractal properties and it is increasingly accepted that there is a deep, probably causal, relationship between the fractal structure and appearance of natural objects and the processes which form them [Gell 94].

Fractal analysis has been found to have extremely diverse applicability. Some of the most fruitful areas have been:

- Quantitative measurement of the irregularity of physical properties. For example, the roughness of ocean floors and the distribution of earthquakes along fault lines [Piet 86]. Measurement of fractal properties has also been used to characterise materials and hence identify the physical processes which formed them [LeMe 91].

- Fractal modelling of physical processes [Cher 91, Peit 92]. For example, fractals have been used to model lightning, clustering of stars, Brownian motion, spread of diseases, population growth etc.

- Computer graphics [Peit 88]. For example, synthetic scene generation for flight simulators and films [Pent 84].

- Analysis of the growth of biological forms such as plants [Meak 86].

These areas are clearly related. For example, the success of fractal modelling of plants and trees for synthetic imagery is due to the image formation techniques corresponding well to the way in which the real objects are formed [Saup 89].

Many examples of scaling laws or power laws (i.e. the quantity under study is inversely proportional to the rank) are known in a wide variety of disciplines (populations of cities, earthquake magnitudes, word distributions). Laws of this form

have long been known [Zipf 49], but have only recently been interpreted as manifestations of underlying fractal phenomena [Mand 82, Gell 94]. In a completely general form Mandelbrot's finding may be summarised with a single rule:

(Measurement of process) $\propto$ (scale at which process is measured)$^{-\mathfrak{D}}$ ..... (3.1)

where $\mathfrak{D}$ which is the fractal dimension of the process.

Mandelbrot was the first to argue that Nature *is* fractal. The premise adopted here is that this can be exploited for image compression by assuming that *images of nature* are also fractal. This is not a proven assumption but some confidence can be gained from a result derived by Kaye that, if a Markov chain[1] of events exists (as is often assumed for consecutive pixel intensities), then there will probably be a fractal pattern embedded in the chain of events [Kaye 89].

3.4 Fractal dimension and complexity

Entropy was introduced in Chapter 2 as a measure of data complexity which has been found acceptable for theoretical analysis. However, it was noted that entropy does not provide a good measure of the complexity of specific data types such as image data.

Experimental work has been performed here to measure the complexity of images assuming that they have a fractal form to determine if such measurements are better predictors of the compressibility of the image data than entropy. In order to achieve this, a practical, automated method of measuring the fractal dimension of an image is required.

The theoretical determination of the fractal dimension of a set requires knowledge of the properties of the set and the metric space in which it lies. Although there are several different definitions of fractal dimension used by mathematicians, the objective here is simply to find a method of measuring image data complexity. Unfortunately there is no broadly accepted unique way of associating a fractal dimension with a set of experimental data so empirical methods are required. As

---

[1] A Markov Process is a stochastic process in which the future is determined by the present and is independent of the past. For example, in a random walk the current location does depend on the past but the next step direction is independent of the past. A record of the results of a Markov process is known as a Markov chain [Papo 84].

noted in Section 3.2, the general method of measuring the fractal dimension of a set is based on drawing a regression line of a log-log plot of the property being measured against the scale at which it is measured. Practical schemes usually follow the outlines devised by Barnsley [Barn 88b] which lead to a fractal dimension measure called the *box dimension*.

If A is a set in a Euclidean space and $N_\varepsilon(A)$ denotes the smallest number of sets with diameter no larger than ε needed to cover A, then the box dimension $\mathfrak{D}_B$ of A is given by:

$$\mathfrak{D}_B = \lim_{\varepsilon \to 0} \left\{ \frac{\ln(N_\varepsilon(A))}{\ln(1/\varepsilon)} \right\} \qquad \text{.....(3.2)}$$

For example, Fig. 3.2 shows three sets and the number of ε boxes required to cover them.

(a) A curve of length l may be covered by l/ε boxes of size ε and $2l/\varepsilon$ boxes of size $\varepsilon/2$. Thus for the curve the box dimension $\mathfrak{D}_B$ is 1, in agreement with the geometric dimension of a simple curve.

(b) A region of area A may be covered by A/ε^2 boxes of size ε and $2^2 A/\varepsilon$ boxes of size $\varepsilon/2$. Thus for the region of the plane the box dimension $\mathfrak{D}_B$ is 2, again in agreement with the geometric dimension.

(c) The Sierpinski triangle [Sier 12] is formed by recursively removing the middle inverted triangle from an equilateral triangle partitioned into four equilateral triangles of half the side length. At each stage only 3/4 of the area is left so the limit set has an area of $\lim_{n\to\infty}(3/4)^n = 0$. If the Sierpinski triangle is covered with N boxes of size ε then it requires $3N$ boxes of size $\varepsilon/2$ to cover it. Thus the box dimension of this set is $\ln 3/\ln 2 \approx 1.58$. This fits well with the concept of a set which has no area but is 'larger' or has 'more to it' than a line.

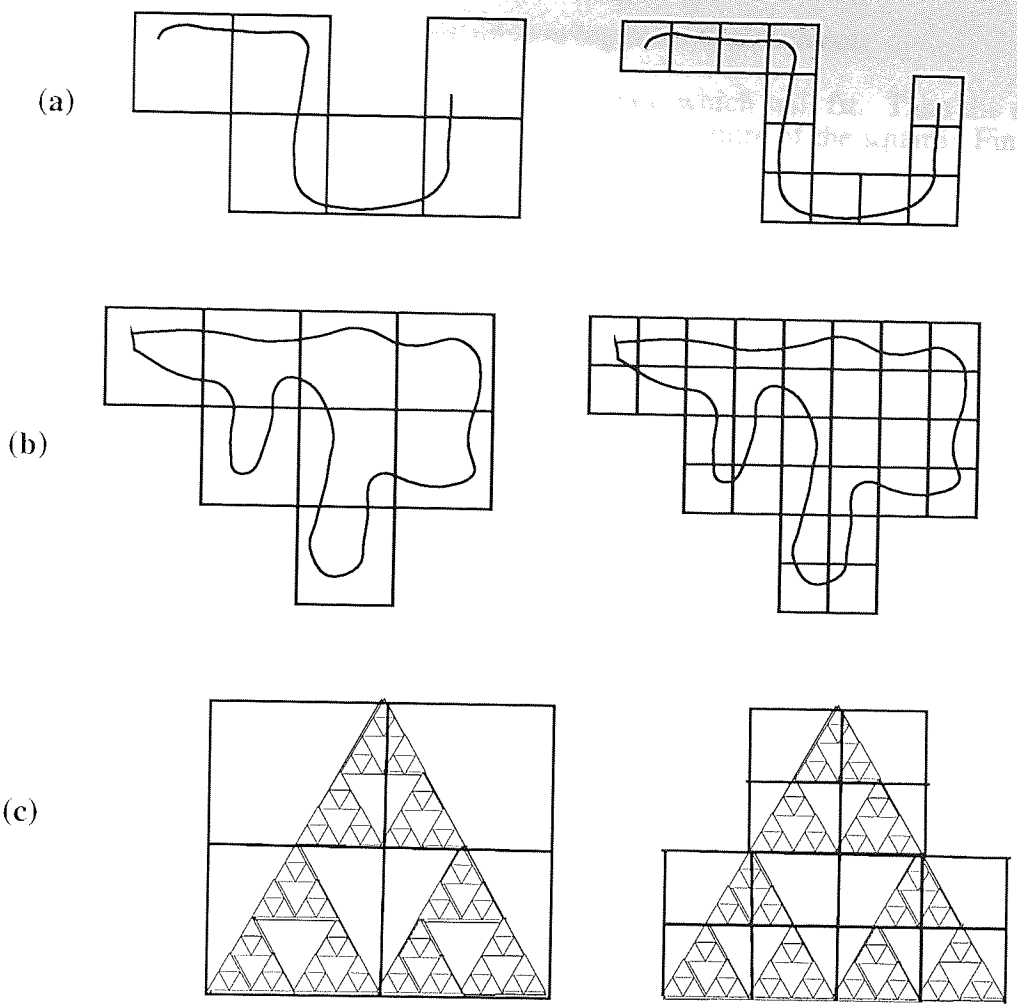


Fig. 3.2. Examples of sets and their box dimensions

Direct application of the box dimension requires some form of automated determination of how the image data is 'covered' by the ε boxes.

Barnsley described a non-automated method of measuring the fractal dimension of an image which resembles the theoretical definition of box dimension. Regions of the image are covered by disks of radius ε for a range of ε-values and in each case the number of disks required is counted to give values for $N_\varepsilon(A)$. If $\ln(N_\varepsilon(A))$ is then plotted against $\ln(1/\varepsilon)$ then the gradient of the best-fitting straight line approximation gives a value for $\mathfrak{D}_B$. The best fit is calculated by least-squares linear regression.

Since an automated method is required here, a method derived by Clarke [Clar 86] for use in geological mapping has been adapted for use with images by treating pixel intensities as contour levels. The algorithm is shown as Algorithm 3.1.

Algorithm 3.1. Clarke's method for measuring fractal dimension

1. Overlay the image with the largest square which will fit. Take the mean intensity of the four corners and assign it to the centre of the square. Find the total area of the four surfaces of the resulting pyramid.

2. Overlay four smaller squares on the same surface and measure their areas in the same way. This smaller scale yields a larger surface area since it includes more of the surface details.

3. Continue this process to the lowest level of scaling (two pixels by two pixels).

4. Plot the log of total area against the log of the area of an overlaid square for each size of unit square

5. The fractal dimension is given by the slope of this plot.

Edge size	Square Area	ln Square Area	Pyramid Area	ln Pyramid Area
256	65536	11.083	266658	12.495
128	16384	9.688	289121	12.575
64	4096	8.286	364151	12.805
32	1024	6.868	458180	13.035
16	256	5.416	799110	13.591
8	64	3.892	1401432	14.153
4	16	2.197	2303565	14.650
2	4	1.386	4054873	15.215

Table 3.1. Evaluation of fractal dimension of Lena by Clarke's method

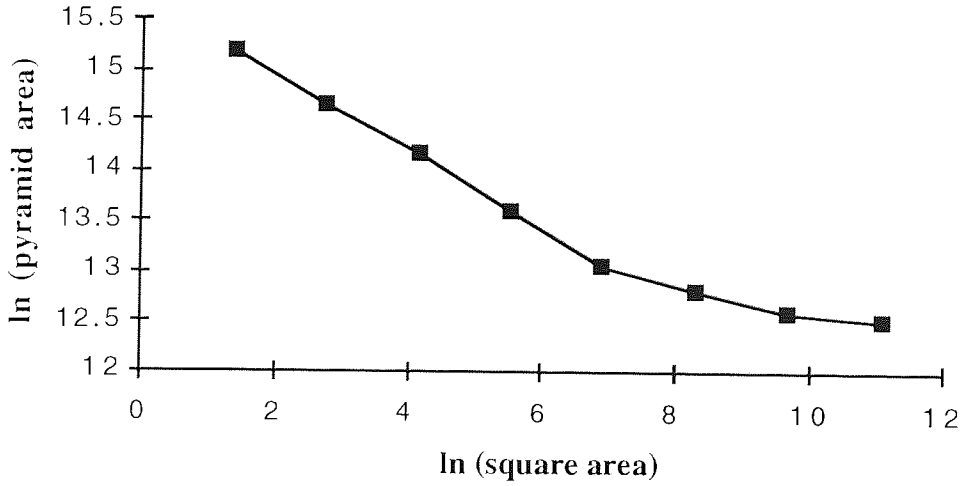


Fig. 3.3. Linear regression of data from Clarke's method for Lena image

From Fig. 3.3 it may be seen that there is an excellent linear relationship for the smaller square sizes but this fails for the larger square sizes. Since these larger sizes include only a very small number of samples, it has been chosen to use only square sizes from 64 down to 2 pixels.

The fractal dimension of the three primary test images has been estimated using Clarke's method with results as shown in Table 3.2. Also included is the zero'th order image entropy (calculated from Eqn. 2.1) together with the image compression ratios achieved by a general purpose data compressor (gzip, see Section 2.5) and the standard JPEG image compression algorithm.

Image	Zero'th order entropy/ bits per pixel	Clarke fractal dimension	gzip compression ratio	JPEG compression ratio (Q=75)
Lena	7.273	2.220	1.230	9.220
Barbara	7.511	2.299	1.129	6.131
Boats	7.074	2.271	1.318	7.532

Table 3.2. Fractal dimension, image complexity and compression ratio

The correlation of the image compression achieved with the various coding methods and the complexity as measured by entropy and fractal dimension has been calculated and is shown in Table 3.3. The correlation measure used here is the Pearson product-moment correlation coefficient r of paired measures x and y. This is given by Eqn. 3.3 where σ_x and σ_y are the standard deviations of x and y and μ_x and μ_y are the means of x and y. For a perfect correlation, $r = +1$, and a perfect negative correlation, $r = -1$.

$$r = \frac{1}{n\sigma_x\sigma_y}\left(\sum_{i=1}^{n} x_i y_i - n\mu_x\mu_y\right) \quad \text{.....(3.3)}$$

Complexity measure	gzip compression	JPEG compression	FWC compression
Entropy	-0.752	-0.574	-0.537
Fractal dimension	-0.684	-0.863	-0.909

Table 3.3. Correlation of complexity and compression measures

The correlation coefficients in Table 3.3. are all negative, since more complex data sets compress less. It is apparent that entropy provides a better complexity measure than fractal dimension for general purpose data compression (which is based on statistical coding). The fractal dimension measure is only slightly worse for predicting general purpose data compression performance and is much better at predicting the performance of JPEG compression which is intended purely for images.

For comparative purposes the final column in Table 3.3 shows the correlation of the complexity measures and the compression achieved by a novel image compression method called fractal waveform coding (FWC) which is the subject of Chapter 5. For now, it is simply noted that the fractal dimension provides an excellent predictor of the effectiveness of this form of image data compression.

3. 5 Space-filling curves

Some of the best-known examples of fractals are the space-filling curves. Chapter 4 is devoted to the use of two-dimensional forms of these curves for scanning images and higher dimensional forms of the curves are used in Chapters 6 and 7 for mapping vectors into one-dimensional forms. In this chapter only a few of the elementary properties of the space-filling curves are outlined sufficiently to provide insight into fractals.

The first space-filling curve was described by Peano in 1890 [Pean 90]. He showed that a continuous curve can pass through all the points of a two (or more) dimensional region. Another such curve was later found by Hilbert [Hilb 91] who viewed such curves as the limiting cases of sequences of polygons. At the time mathematicians regarded these curves as 'monsters' [Saga 91] since they were so difficult to analyse with the body of mathematics then known. For example, the curves are nowhere differentiable.

Because of the practical difficulties in drawing them, these curves remained mathematical curiosities until the advent of computers and suitable plotters in the 1960's. After Mandelbrot's pioneering work [Mand 82], space-filling curves were soon recognised as good examples of fractals. The curves have the distinctive properties of fractals listed in Section 3.2. In particular:

They exhibit self-similarity at different scales.

They can be generated by relatively simple rules.

Three of the most well known of these curves, the Hilbert and Peano curves and a form of the Sierpinski curve, are shown in Fig. 3.4.

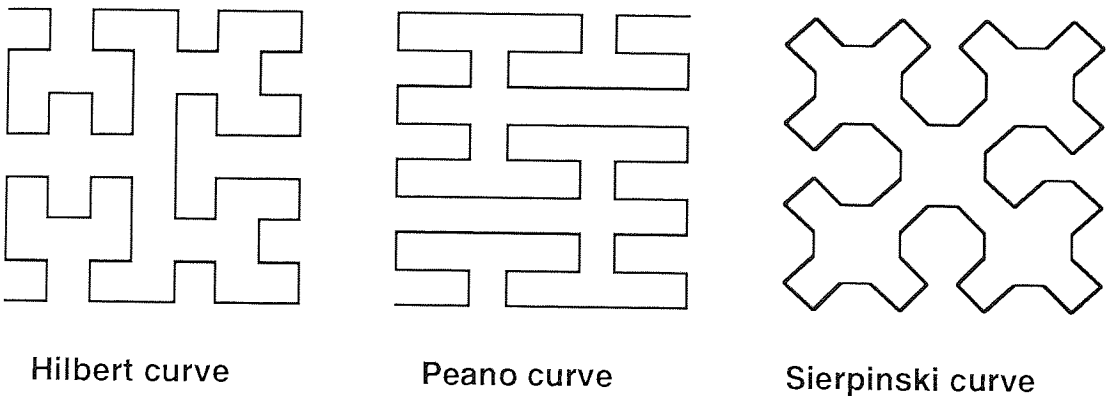


Fig. 3.4. Some space-filling curves

3.5.1 Order of space-filling curves

Space-filling curves are typically defined by a recursive formula using an order n which indicates the level of recursion. The order can be considered to be a formalised indication of the scale of the curve, or to what level of detail the curve is drawn. The curves are only truly space-filling when $n \to \infty$. However, the curves soon appear to completely cover the plane for even small values of n. In this thesis the curves are used in the analysis of digital images whose resolution is clearly finite so only curves of finite order need be used in their processing. Fig. 3.5 illustrates a Hilbert curve of different orders. An n'th order Hilbert curve, H_n, is composed of four appropriately rotated copies of the Hilbert curve of order $n-1$ with the endpoints being joined by three line segments which themselves form a copy of H_1.

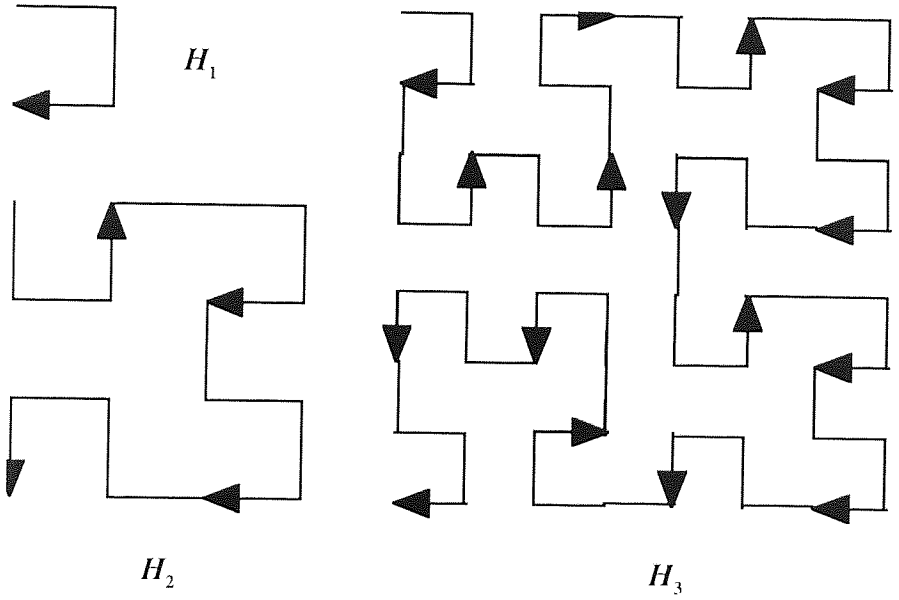


Fig. 3.5. Different order Hilbert curves

It is simple to show that H_n covers 2^n points. Hence a 9'th order Hilbert curve suffices for scanning the 512 by 512 pixel images used in this thesis.

3.6 Summary

Fractals provide an interesting and effective tool for analysing problems which have proved intractable using other methods. Fractal data arises from many physical sources and fractal behaviour is exhibited by many natural objects. Images of natural scenes capture this data and thus also contain fractal data. The success of fractal modelling for synthetic imaging suggests that there is considerable potential for using fractals to analyse the much greater complexity of real images.

It has been shown that the concept of a fractal dimension can be applied to real images and that it provides a natural and effective complexity measure which can be used to predict the performance of both fractal and non-fractal image compression methods. If lossless compression, based on conventional statistical methods from information theory, is to be applied then entropy is an appropriate measure. However, if lossy compression is to be applied, then advantage can be taken of knowledge of the complexity of the data at different scales which is provided by measurement of the fractal dimension of the data.

In addition to their uses for fractal modelling of image data, fractals such as space-filling curves offer an explicit mechanism for applying 'divide-and-conquer' methods to complex problems such as image compression when iterative or recursive operations are required at a range of scales.

In this thesis the well-established problem of image data compression is approached using a variety of fractal-based methods.

Chapter 4. Fractal scanning and space-filling curves

4.1 Introduction

In Chapter 2 the general principles of image scanning were introduced but no assumptions were made about the sequence in which the image pixels are scanned. The convention for the scanning sequence is so well established that this issue is often ignored and the scan is assumed to be a simple raster. However, there are many possible ways of scanning an image and, particularly when predictive coding is being used, the scan sequence has a significant influence on the compressibility of the resultant stream of pixels.

The possibility of achieving greater image compression by re-ordering data with space-filling curves, which were introduced in Chapter 3 as some of the best-known examples of fractals, was first suggested 25 years ago [Laem 67, Bial 69, Butz 71] but reports since then have been contradictory [Quin 89, Mogh 91]. In this chapter, new results are presented which provide a convincing case in support of the conjecture. In addition, while previously published work has concentrated on just two of the many possible space-filling curves, experimental results are presented here over a much wider range of curves. The idea of using space-filling curves for scanning is extended with the concept of hierarchies of space-filling scans together with an associated notation which concisely identifies such scans.

Space-filling curves have also been proposed [Cole 90, Cole 91] for use in the dithering of images for display on devices such as laser printers, which support only a limited set of intensity levels. Experimental work has been performed here to confirm this idea.

4.2 Types of image scan

An image scan is an ordering of the pixels in an image for sequential processing such as image compression. Of the three main classes of image compression schemes listed in Section 2.6, both waveform coding and transform coding entail an implicit assumption that the image data is processed in a particular order. In the case of waveform coding, the individual pixel ordering determines the compression. For transform coding, the pixel order in each block is not usually significant but the order in which the blocks are presented is relevant. Scans are classified here as *random-access* if there are no restrictions on where consecutively scanned pixels lie in the image or as *sequential* if consecutively scanned pixels are adjacent in the image.

4.2.1 Random-access scanning

Modern devices such as charge-coupled-device (CCD) cameras provide a true two-dimensional scanning ability (i.e. the image can be scanned element by element in any order [Flor 85]). The sequence in which the image data is read out is determined by the device fabrication and could, in principle, be in any desired order. For such devices random-access scanning can be applied but existing serial standards are usually used for reasons of compatibility. It is not clear that any advantage would be gained by scanning every pixel in an image with a random-access scan, but using such a method for statistically selecting a sub-set of the pixels is useful for purposes such as histogram generation. Powerful parallel systems which can scan many, or all, of the pixels concurrently might also be described as random-access. However, since only sequential processing is considered here, the work is restricted to sequential scans.

4.2.2 Sequential scanning

In sequential scanning the objective is to select consecutive pixels which are adjacent in the image so as to preserve pixel correlation. The fundamental problem is that the scanned data is only a one-dimensional set of values but it needs to represent the two-dimensional correlation present in the image. Thus, it is preferable to try and identify scans which maintain two-dimensional continuity through the image.

Sequential image scans may be further divided into raster scans which are essentially one-dimensional and non-raster scans which can traverse the image in both the x and y directions simultaneously.

4.2.3 Raster scanning

In raster scanning the signal is formed by scanning in a series of parallel lines, either horizontally or vertically. In broadcast television and in most other applications the scan starts at the top left hand corner and finishes at the lower right hand corner (see Fig. 4.1). Typically each line is scanned in the same direction so the scan must return to the beginning of each line in a very short time known as the flyback. This format dates back to the earliest developments in television such as the Vidicon tube and has led to several standard video signal formats such as RS-170, RS-343 and CCIR 601. Raster scanning has the advantage of being simple but there are several disadvantages:

throughout the resulting one dimensional array. Hence a raster scan results in high-frequency periodic components in the one-dimensional scanned form.

- The image resolution is locked to a standard (the number of lines in the image). The display must adopt the same standard or convert the input signal to its own standard.

- If the bandwidth of an analogue channel used to carry the video is reduced, only the horizontal resolution of the image is reduced, not the vertical. Therefore loss of resolution is not distributed evenly, giving lower image quality.

- Flyback which is non-useful time occupies more and more of total frame time as the line rate increases.

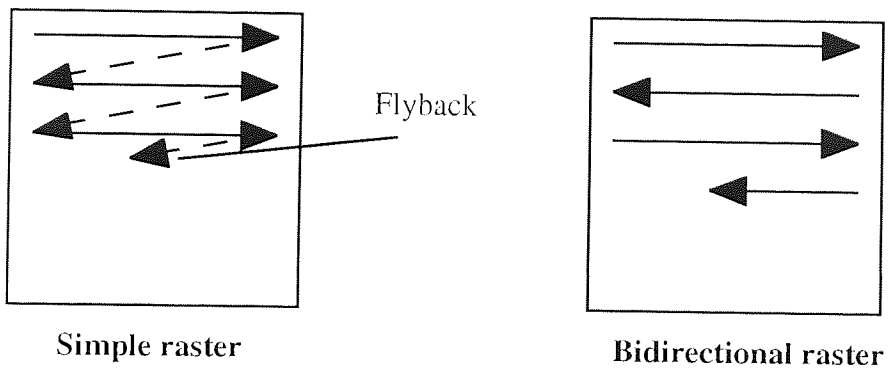


Fig. 4.1. Simple and bidirectional raster scanning

Flyback can be avoided if bidirectional scanning is adopted. In this case alternate lines are scanned from left to right and right to left (see Fig. 4.1). The recent HDTV standard uses a bidirectional (also known as boustrophedonic) scan. This alone gives a substantial improvement on unidirectional raster scanning in terms of spatial disruption for only a small increase in complexity because it avoids the discontinuities at the end of each line.

4.2.4 Non-raster scanning

There are $(n^2)!$ possible sequences in which the pixels of an n by n image can be scanned of which only a small fraction are likely to give any advantages over a bidirectional raster. Unfortunately, it is not obvious how to compare the scans except by an impractical exhaustive search. In addition, the size of the binary word needed simply to identify the scan, S_{scan}, would be huge.

$$S_{scan} = \log_2\left(n^2!\right) \quad \text{.....(4.1)}$$

If a much smaller subset of the possible scans containing just those which are likely to be useful can be identified, then both the time to search for the optimum scan and the storage of the identification of the scan can be significantly reduced.

Clearly, an optimum scan must exist for any given coding method. If, for example, run-length coding is to be used, the optimum scan would create a sequence of pixels where all pixels of the same intensity were consecutive. Under such circumstances the worst-case storage requirement, $S_{\max}$, for the encoded image would be the product of the number of possible grey levels, L, and the storage for each run. The maximum possible run length is n^2. Thus:

$$S_{\max} = L\left(\log_2(L) + \log_2\left(n^2\right)\right) \text{ bits} \quad \text{.....(4.2)}$$

Substituting realistic values of $L = 256$ and $n = 512$ in Eqn. 4.2 gives:

$$S_{\max} = 256\left(\log_2 2^8 + \log_2 2^{18}\right) = 256(8+18) \text{ bits} = 832 \text{ bytes}$$

This would represent an outstandingly good compression ratio of over 300:1 - and this is the worst-case compression ratio for such an optimal scan. Naturally, the problem with this is the size of the representation of the scan, S_{scan}. With $n = 512$, then, for numbers as large as n^2, Stirling's factorial approximation (Eqn. 4.3) is valid. For large n,

$$\ln n! \approx n \ln n - n \quad \text{.....(4.3)}$$

Converting to base 2 logarithms using $\log_2 n = \dfrac{\ln n}{\ln 2}$, then

$$S_{scan} \approx \frac{1}{\ln 2}\left(n^2 \ln n^2 - n^2\right) = \frac{n^2}{\ln 2}\left(\ln n^2 - 1\right) \text{ bits} \quad \text{.....(4.4)}$$

Hence, for $n = 512$, $S_{scan} \approx 542$ kbytes which is approximately double the size of the original image.

The purposes of examining non-raster scans here is to find a good, but not necessarily optimal, value for $S_{\max}$ which requires a much smaller value for S_{scan}.

This problem was addressed by Memon *et al* [Memo 95] who tried to decorrelate images by taking differences of adjacent pixels along a scan. Given an image, an optimal scan that minimises the absolute sum of the differences encountered can be computed efficiently. However, the scan is inevitably irregular and the number of bits required to encode it turns out to be prohibitive. They proposed a prediction

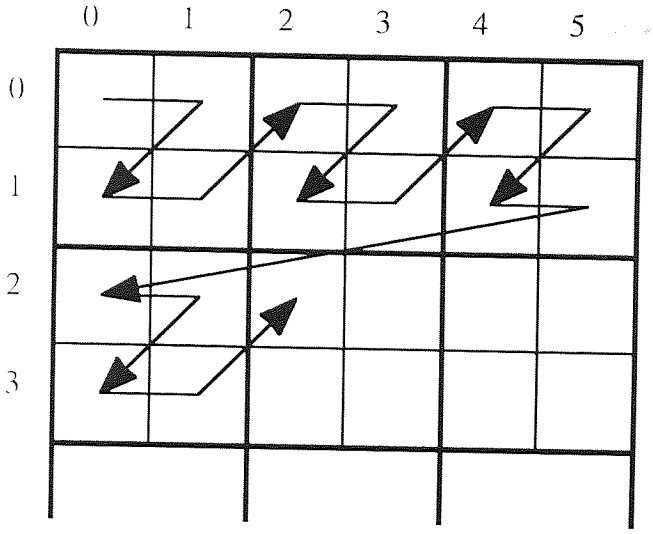
scheme which partitions an image into blocks and, for each block, selects a scan from a codebook of scans such that the resulting prediction error is minimised. Unfortunately this has similar problems to vector quantisation (see Chapter 6) in that considerable processing time is required to form a codebook and, unlike vector quantisation, the codebooks vary considerably between different images.

The approach taken here is to consider only regular, hierarchical scans which have concise descriptions. It should be recalled from Chapter 3 that a complex appearance requiring only a simple definition is a fundamental property of space-filling curves and of fractals in general. Intuitively it can be seen that the space-filling curves have appropriate properties for use as non-raster scan generators. They provide a sequential scanning mechanism which uniformly traverses the image in two dimensions yet they can be generated reasonably easily and require only concise descriptions.

The idea of a fractal (space-filling) scan has been proposed for broadcast TV [Drew 91]. The main advantage was seen to be the possibility of creating a system with built in upwards and downwards compatibility with different resolution broadcasting stations and receivers. This is not possible with a raster scan because the line rate has to be fixed. In a fractal scan it can be arranged that corresponding parts of the image are scanned at corresponding times by different resolution (order) scanning curves. In other words, all that needs to be fixed by the standard is the form of the scan (e.g. a Hilbert scan) and the orientation (choice of starting and finishing points).

4.2.5 Interleaving

In a practical implementation of an image compression system it should be considered how the decompressor might overlap the decompression of the data stream and the display of the data. The solution is to *interleave* the data components from the source image. For example, the JPEG standard defines an entity called a 'data unit' which is a single pixel in the lossless predictive mode and a block of pixels in the lossy DCT-based mode. When two or more components are interleaved, each component C_i is partitioned into H_i by V_i data units. For example, Fig. 4.2 shows interleaving with H=2, V=2. Regions are ordered within a component in row-column order. In JPEG terminology the minimum code unit (MCU) is the smallest group of interleaved data units.



JPEG MCU interleaving with H=2, V=2

$\text{MCU}_1 = \{d_{00}, d_{01}, d_{10}, d_{11}\}$

$\text{MCU}_2 = \{d_{02}, d_{03}, d_{12}, d_{13}\}$

etc.

$\text{MCU}_4 = \{d_{20}, d_{21}, d_{30}, d_{31}\}$

etc.

Fig. 4.2. Interleaved data unit ordering

Interleaving is not considered further in this chapter but it should be noted that the principle can be applied even if the scan sequence is not a simple raster.

4.3 Space-filling curves for scanning

The earliest suggestions that space-filling curves might provide a mechanism for bandwidth reduction appear to be by Laemmel [Laem 67] and Bially [Bial 69] who proposed a scheme called sampling mapping. They described how uniformly spaced samples of a continuous input signal could be grouped into sets of N samples with each set being interpreted as the coordinates of a point in an N-dimensional cube. They suggested that, if cube coordinates are now transmitted instead of the original samples, then a bandwidth reduction of N:1 is achieved. However, although they recognised that reconstruction of the signal by the receiver now becomes much more susceptible to noise, they failed to mention that no bandwidth reduction is achieved unless the new samples are represented by fewer bits than were used for the original samples. Though it was not recognised at the time, a modern interpretation of this method might consider it to be analogous to arithmetic coding (Section 2.5.2.3).

More recent uses of space-filling curves for bandwidth reduction have concentrated on their uses for re-ordering image pixels before applying conventional coding schemes such as predictive coding and run-length encoding.

There has been some debate over whether any real benefits are gained from non-raster scanning methods. The inconsistencies in the arguments for and against relate primarily to the choice of data for analysis or simulation. Some workers have

considered only binary images while others have considered grey-scale images but only with synthetic scenes. The disadvantages of such restricted data sources were outlined in Section 2.2.5. In this section results are presented which provide a comparison of the Hilbert curve and a simple raster scan for grey-scale images of natural scenes. In addition, some useful results are determined which are independent of the source image.

4.3.1 Evidence against space-filling curves

Quin *et al* [Quin 89] concluded that there is no merit in any scan more complicated than a bidirectional raster. They proved that, for a random image (i.e. one consisting of completely uncorrelated pixels), there is no difference in the run length histogram whichever scanning curve is chosen but they were unable to prove this conjecture for real images. This result is not very useful since elementary information theory shows that random images are incompressible anyway. They also performed simulation on binary images which contained a single randomly positioned and sized ellipse and measured the run length distribution for raster and Hilbert scans. They concluded that there was no statistical difference.

Whilst the choice of a binary image for simulation is reasonable, they only explored the potential for lossless compression. Lossy compression using predictive coding is more appropriate for grey-scale images. The choice of such an artificial image with a single feature must be considered sceptically, particularly in the light of findings from Chapter 3 where the fractal nature of real images was considered.

4.3.2 Evidence for space-filling curves

Several authors have produced results which support the conjecture that space-filling curves, particularly the Hilbert curve, provide superior scans. Most results have been experimental with the exception of a formal proof by Lempel and Ziv [Lemp 86] that the best compression ratio attainable by any lossless two-dimensional compressor is equal to the finite-state compressibility of the sequence read from the image traced by a Hilbert curve. Whilst this result is interesting, it is only applicable to lossless coding. In this section evidence provided by various methods of comparing raster scans and space-filling curve scans for lossy data compression is evaluated.

4.3.2.1 The neighbourhood relationship

There is further quantitative evidence in favour of space-filling curve scans. Sorek and Zeevi [Sore 88] introduced a measure which they called the neighbourhood

relationship which quantifies how well the association between pixels which are close in the two-dimensional image is preserved in the one-dimensional scanned form.

If the points along the scan are denoted by $S_i (1 \le i \le N)$ where N is the number of points in the scan, then the neighbourhood relationship, which is defined as the mean Euclidean distance $\overline{D_i}$ between pixels which have a separation of i in the scan, is given by Eqn. 4.5:

$$\overline{D_i} = \frac{1}{N} \sum_{p=1}^{N-i} \sqrt{\left(x_p - x_{p+i}\right)^2 + \left(y_p - y_{p+i}\right)^2} \quad \text{.....(4.5)}$$

The neighbourhood relationship has been evaluated here for a simple raster, a bidirectional raster and a Hilbert scan with the results shown in Fig. 4.3. Both the raster and Hilbert scans give a periodic relationship but the period for Hilbert scans is much greater than for raster scans and is not apparent over the maximum neighbourhood length used here.

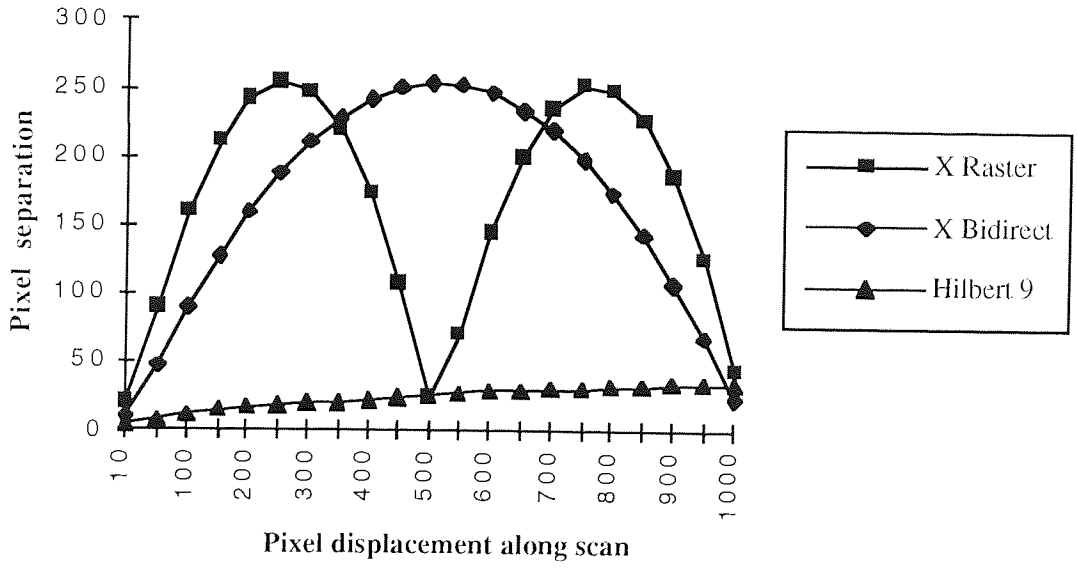


Fig. 4.3. Neighbourhood relationship for raster and Hilbert scans

Comparison of two scans involves comparing the values of $\overline{D_i}$ over all i up to a specified maximum neighbourhood length. Since it is unlikely that any scan will have a smaller $\overline{D_i}$ than any other scan for *all* values of i then a weighting function W_i may be assigned to the $\overline{D_i}$ to evaluate a single scan quality measure V. If this weighting function is inversely proportional to the separation i, then:

$$V = \sum_{i=1}^{N-1} \frac{\overline{D_i}}{i} \quad \text{.....(4.6)}$$

The quality measure V defined in Eqn. 4.6 has been evaluated here for simple raster, bidirectional raster and Hilbert scans of 512 by 512 arrays. The results are shown in Table 4.1 for a maximum neighbourhood length of 1000. Longer neighbourhood lengths are not considered since pixels any further apart are unlikely to be related. The improvement achieved by the bidirectional scan is only about 18% whereas the Hilbert scan improves the measure by an order of magnitude.

Scan	Simple raster	Bidirectional raster	Hilbert scan (9'th order)
V	626.09	510.65	69.23

Table 4.1. Neighbourhood preservation of various scan paths

In their analysis Sorek and Zeevi also presented evidence which showed that reconstruction errors after compression along a Hilbert scan were subjectively better. They attributed this to the Hilbert scan giving no preferred directivity in the restored image whereas a raster scan can cause more accurate sampling of edges in the scan direction giving a false orientational emphasis in the output. Furthermore, the reconstruction error is correlated in the direction perpendicular to the scan creating false contours known as Moiré patterns. Removal of Moiré patterns is the primary motivation for applying space-filling scans to the dithering of images - a concept discussed in Section 4.9.

4.3.2.2 Lossless coding

Lossless coding along a Hilbert curve has also been investigated by Provine and Rangayyan [Prov 94] who, in tests on a small set of 24-bit colour images, achieved compression ratios of between 3:1 and 5:1 using differential predictive coding and an entropy coder. Skarbek *et al* [Skar 89a] applied the Hilbert scan to lossless coding of dithered binary images and claimed it gave slightly better compression than raster scanning but it is difficult to relate their findings to other work since it is not usual to try to compress the dithered form of binary images.

4.3.2.3 Run-length coding

Cole [Cole 87] described run-length coding along a Hilbert curve covering the whole image. However, he only used binary images and these were synthetic and thus not very representative of real image scenes. Although his experimental arrangement was essentially the same as Quin's (noted in Section 4.3.1) his findings were entirely contrary. i.e. He found the Hilbert curve significantly better.

It would seem reasonable that the real test of any superiority of space-filling curve scans for run-length coding is the scanning of grey-scale images of natural scenes.

Fig. 4.4 shows two of the primary test images, 'Lena' and 'Barbara', scanned along a Hilbert curve with the 262,144 pixel one-dimensional images redrawn in raster form for convenience.

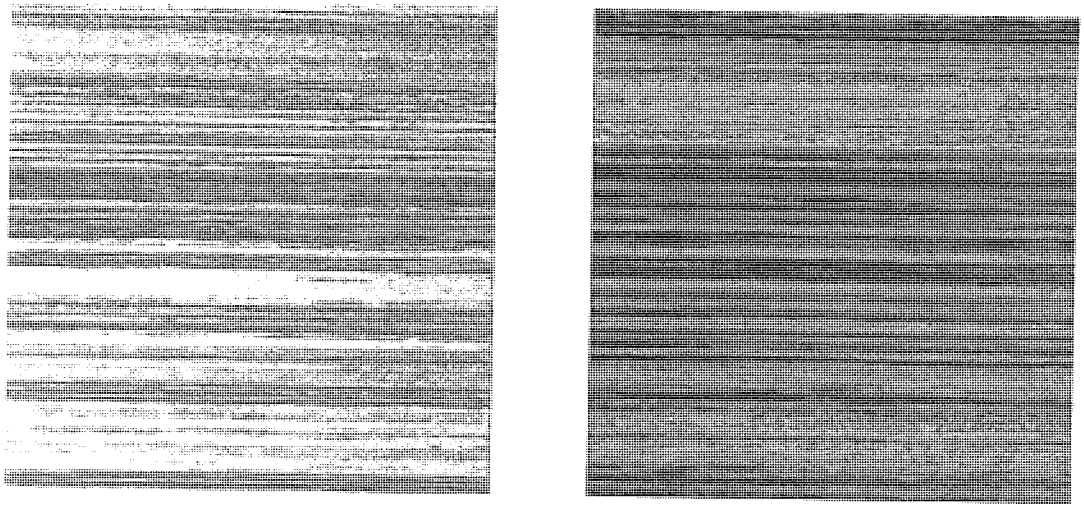


Fig. 4.4. 'Lena' and 'Barbara' images scanned along Hilbert curve

It is intuitively very clear from Fig. 4.4 that this data will compress well with run-length compression. Quantitative evidence is provided by the run length histograms shown in Fig. 4.5 for the three primary test images scanned with a raster scan and a Hilbert scan.

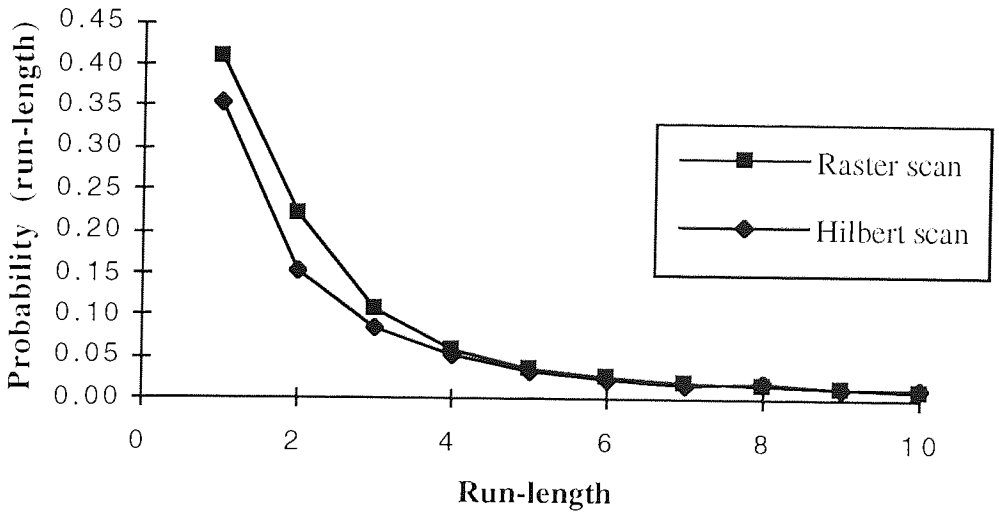


Fig. 4.5. Run-length distribution for raster and Hilbert scan paths

Although Fig. 4.5 may not appear dramatic, the run length distribution makes compression considerably easier with a Hilbert scan. The fraction of pixels in the very shortest runs of one or two pixels has fallen from 65% of the total to only 49%.

This alone has a considerable influence on the final compression ratio. Furthermore, though it is difficult to show an a graph of any reasonable scale, the Hilbert scan creates many more runs of the very longest lengths (64-512) pixels than the raster scan.

4.3.2.4 Other coding methods

Other coding methods such as vector quantisation [Samp 93], wavelet decomposition [Ansa 92a,] and lapped orthogonal transforms (LOT) [Ansa 92b] have been applied along Hilbert curves. The results have generally been in favour of the space-filling curves but, in these more sophisticated coding methods, the contribution to the compression ratio from the difference in scanning is greatly reduced.

4.3.3 Experimental comparison of Hilbert scan and raster scan

From the preceding discussion it is clear that, while the bulk of the evidence is broadly in favour of the Hilbert curve over the simple raster curve as a scan, there is some difficulty in directly comparing experimental results. For clarification, a variety of compression methods have been applied here to the larger, secondary set of test images. The compression methods shown are GIF (a lossless compression method based on arithmetic coding), lossy JPEG (see Section 2.8.4.1), FAX Group 3 (see Section 2.8.3.1) and fractal waveform coding (FWC) which is the subject of Chapter 5. The comparison between the scans is based on ΔC_r, the percentage change in compression ratio achieved by the Hilbert scan $C_{hilbert}$ over that achieved by the raster scan C_{raster}. i.e.

$$\Delta C_r = \frac{C_{hilbert} - C_{raster}}{C_{raster}} \times 100\% \quad \text{.....(4.7)}$$

The results are summarised in Table 4.2. As expected, there is little difference in the compression achieved for GIF compression since arithmetic coding makes little use of the data ordering but there is still a 4% improvement achieved with the Hilbert scan. JPEG compression gives consistently worse results with a Hilbert scan since the two-dimensional correlation present in the 8 by 8 pixel blocks processed by the JPEG process is lost by the Hilbert scan. However, in the case of compression techniques based on one-dimensional predictive coding, the superiority of the Hilbert scan is clear. Both Group 3 FAX compression and FWC achieve better compression with the Hilbert scan for all of the test images. For Group 3 FAX the mean improvement is 10% and for FWC it is 18%.

Image	ΔC_r % GIF	ΔC_r % Lossy JPEG (Q=75)	ΔC_r % FAX Group 3	ΔC_r % FWC
announcer	-1.19	-45.64	8.78	2.71
beachgirl	9.28	-42.35	4.37	22.53
bluegirl	8.88	-45.21	6.75	30.57
cablecar	0.73	-36.09	7.53	7.55
cornfield	-4.04	-31.00	7.58	2.72
flower	3.87	-40.66	11.53	24.55
fruits	6.56	-30.73	4.80	22.00
goldhill	1.38	-25.87	9.11	9.75
jewels	8.42	-39.86	5.77	17.49
kids	2.49	-37.42	10.04	11.19
koala	7.36	-29.98	18.60	32.07
leopard	2.20	-34.04	11.21	17.79
model	4.48	-43.46	4.05	27.28
narcisus	5.37	-24.34	15.99	19.74
nicki	7.87	-39.21	8.34	29.57
or-002	1.77	-31.74	4.38	21.26
orca7	4.07	-34.97	18.65	21.65
pens	3.99	-37.84	13.99	26.81
plumehat	6.00	-14.67	6.60	12.76
rabbit3	3.84	-19.21	14.12	7.39
reba08	10.33	-32.89	11.38	26.73
soccer	-3.18	-34.39	12.11	4.70
vi-002	1.99	-25.43	13.66	12.11
yacht	-0.42	-43.72	9.74	3.40
zebras	2.87	-27.27	20.45	15.11
zelda	7.00	-37.15	9.50	46.68
Mean	3.92	-34.20	**10.35**	**18.31**

Table 4.2. Comparison of raster and Hilbert scans

4.3.4 Space-filling curves and quad-trees

Further support for Hilbert curves as scan generators is provided by the relationship between the Hilbert curve and quad-trees [Same 84] which are a well-established method for image representation based on successive sub-divisions of the initial array into quadrants. Each node of the tree is a leaf if all of the pixels in the quadrant have the same colour (intensity for grey-scale images) or has four child nodes. This labelling is repeated until quadrants (possibly single pixels) are reached of a single colour (or grey-level).

The link between Hilbert curve and quad-tree is that each order of the Hilbert curve maps directly to the next level in the quad-tree. This relationship is shown in Fig. 4.6.

Different data structures may be used to hold the quad-tree data yielding different levels of data compression. For example, Samet showed that the minimum storage for a pointerless quad-tree of a 2^m by 2^m binary image is 2^{m+2}. However, regardless

of the data structure used, quad-trees typically yield a data compression ratio of 2-3:1 for binary images.

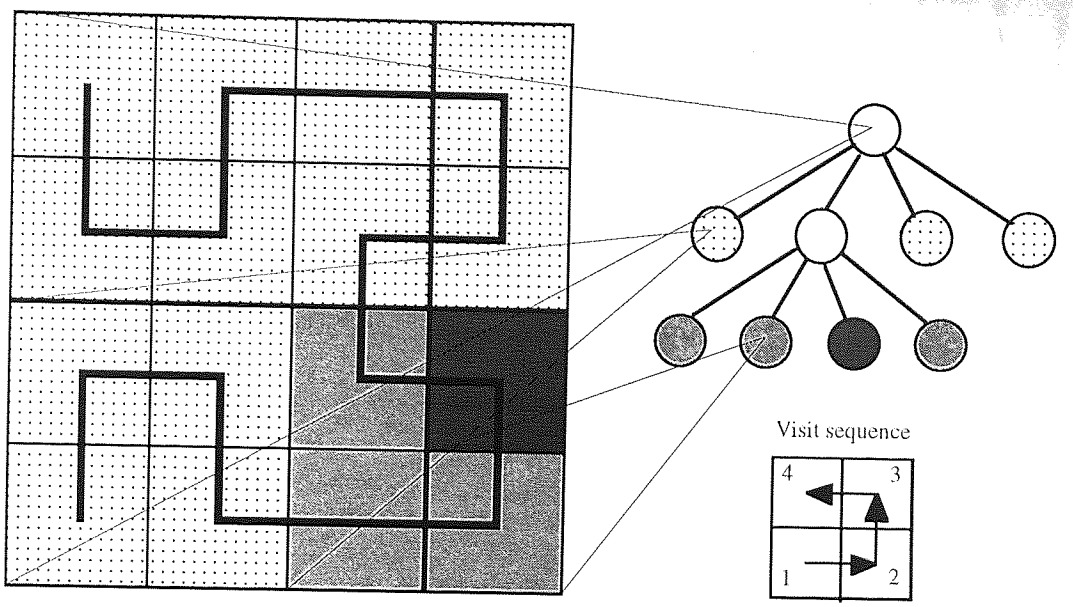


Fig. 4.6. Relationship between Hilbert curve and quad-tree

Kamata *et al* [Kama 93] compared the sizes of quad-tree representations of Hilbert scanned and raster scanned binary images. They found that the quad-tree formed from the Hilbert scan was consistently smaller than the quad-tree representation from a raster scan.

Similar tests have not previously been performed for grey-scale images but have been measured here. To code a grey-scale image as a quad-tree it is usual to define a mean distortion level at which it is acceptable to replace all of the pixels in the square by the mean grey level of the square. Fig. 4.7 shows the mean compression ratio resulting from a quad-tree representation, with various MSE acceptance levels, of the raster and Hilbert scanned versions of the three primary test images. The distortion figures are not shown since each MSE acceptance level yields approximately the same image distortion, irrespective of the scan. However, the compression ratio varies significantly.

The storage mechanism for the quad-tree components is not optimised here. The compressed size is simply taken to be proportional to the number of squares in the quad-tree, with an approximate requirement of 2 bytes per square. It does not take advantage of any savings which could be made by entropy coding based on the distribution of the square sizes.

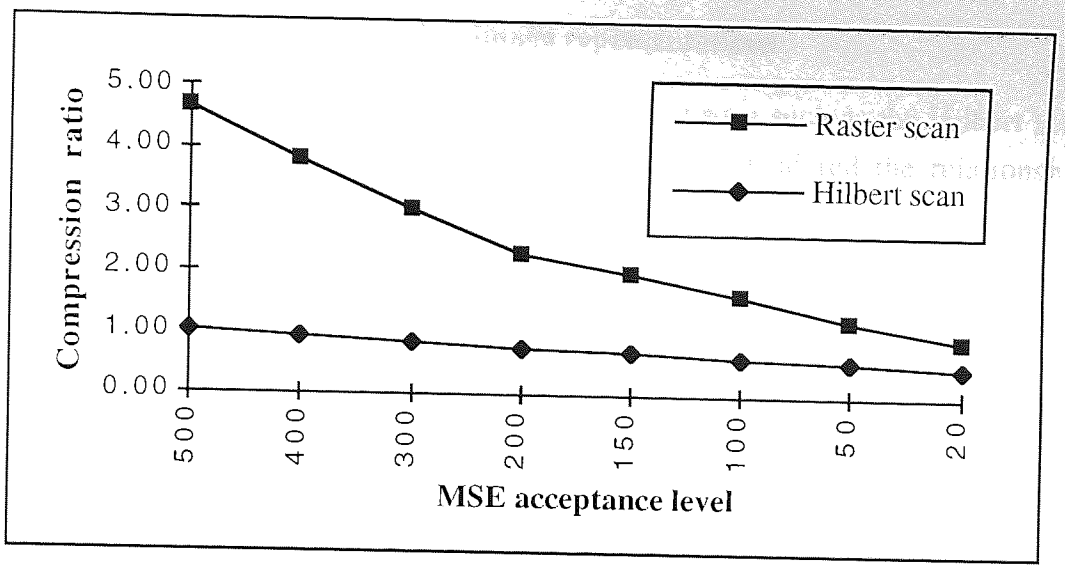


Fig. 4.7. Quad-tree compression for raster and Hilbert scans

In contrast to the finding of Kamata *et al* it is clear from Fig. 4.7 that the Hilbert scan gives a consistently *worse* compression ratio than a raster scan for quad-tree coding. However, this is taken as confirmation that the Hilbert scan has indeed reduced the two-dimensional correlation of the image by increasing the one-dimensional correlation along the scan path. The connection between quad-trees and Hilbert scans is presented to help explain how the Hilbert curve re-orders the image data and not as a suggestion that Hilbert scanning, or any other space-filling scanning, is necessarily appropriate for subsequent compression with quad-trees.

4.3.4.1 Accessing data from quad-trees

In spite of the negative comments made earlier about Hilbert scanning for quad-tree data compression, there is one task associated with quad-trees where the Hilbert scan offers a performance improvement. A major problem with image data stored in quad-trees is the increased complexity of accessing pixel values for processing. Laurini [Laur 85] first suggested that typical operations which need to be performed on stored quad-tree data (such as retrieving the intensity of a pixel, extracting a sub-image etc.) may be performed efficiently if the quad-tree elements are stored in Hilbert scan order. It has been shown that the problem of displaying image data stored as pointerless depth-first (DF) quad-trees [Mill 92] and breadth-first (BF) quad-trees [Skar 90] may be achieved with very efficient algorithms of computational complexity $O(n)$ by scanning along a Hilbert curve.

4.3.5 Space-filling curves and other image representations

A further insight into the properties of a space-filling scan such as the Hilbert scan was given by Moghaddam *et al* [Mogh 91]. They considered the relationship between Hilbert scan and an orthogonal expansion. By mapping the basis functions for a one-dimensional transform such as a DCT to two-dimensional basis matrices, using an inverse Hilbert curve mapping, they identified basis images similar to the basis functions of two-dimensional Fourier, Walsh and DCT transforms. However, the Hilbert scan matrices are subscripted by only one index giving a representation in which the spatial frequency increases in all directions simultaneously.

It is intriguing to consider that there may be a fundamental connection between the two-dimensional Fourier transform of an image and the one-dimensional transform taken along a Hilbert scan. A separable two-dimensional transform such as the Fourier transform is usually calculated by taking one-dimensional transforms of each row and then taking one-dimensional transforms of the results down each column. However, the natural definition of a true two-dimensional transform would be in terms of a two-dimensional transform of each quadrant of the image followed by a recursive subdivision, at each level taking the transform of the four values to produce the overall transform. The order in which the pixels are visited by this algorithm is the same as the visit order of a Hilbert curve. However, while several authors [Mogh 91, Witt 83] have commented on this, the exact relationship remains unclear.

4.4 Construction of space-filling curves

In order to use space-filling curves for image scanning, algorithms have to be found for plotting the curves. For image scanning only two-dimensional curves are required and only such constructions are analysed in this section. In Chapter 6 space-filling curves are applied to vector quantisation where far more complex curves in higher dimensions are required. It will be seen there that different algorithms can be applied since only a subset of the curve points are needed whereas in this chapter *all* of the curve coordinates need to be generated.

4.4.1 Ad hoc and systematic methods

Soon after graphics plotters became available some early methods for plotting space-filling curves [Bial 69, Butz 71, Null 71] were derived but these were over-complicated and inefficient. The title of Null's paper, "How to waste time with a plotter", gives some indication that, at the time, the curves were not pursued for entirely serious purposes. More elegant methods based on recursion have been

proposed [Wirt 76, Gold 81, Witt 83, Grif 85] which are much faster and have found uses in computer graphics, art and animation. Wirth [Wirt 76] even used a construction of the Hilbert curve in his programming textbook to illustrate the appropriate use of recursive programming. However, these methods are not helpful for the problems of sequentially scanning rectangular grids and mapping directly between grid points and scan sequence numbers.

Current methods for generating space-filling curves can be split into three categories: direct mappings, table-based methods and recursive methods. These are all examined in detail in this section. The choice of a method for curve generation depends on a trade-off between speed and storage requirements.

Direct mappings and tabular methods have been found for the most well known curves. Later, in Sections 4.5 and 4.6, it will be shown that there are so many possible curves that recursive generation of curves is more practical. Instead of trying to create a simple method of drawing the curve from a potentially complex set of tables, it is better to provide a simple definition of the first order version of the required curve and then use a more complex algorithm to plot the higher order version.

4.4.2 Direct mapping algorithms

Efficient use of space-filling curves for image processing requires algorithms to map directly between pixel coordinates and sequence number for a scan of arbitrary order (recall from Chapter 3 that the order of a curve indicates the level of recursion). Such a mapping has been found for the Peano curve by Cole [Cole 85a]. Cole showed that if $n > 2$ is an odd integer then there is a simple direct mapping between the first n^{2p} base n Gray code integers and the sequence of points on the p'th order Peano curve.

4.4.2.1 Gray codes

It is not always recognised that Gray codes are merely one example of a large set of cyclic progressive codes. These are number systems such that successive integers differ in only one bit. The reason for considering these number systems for generating space-filling curves follows from the observation that the points on the curves are always one 'step' apart in the x or y direction, but not both.

Given a base n integer $a = a_1 a_2 \ldots a_m \left(0 \le a_i < n, i = 1,2,\ldots,m\right)$ let the Gray code base n integer b corresponding to a be given by $b = b_1 b_2 \ldots b_m$. The conversion rules are different for odd and even based numbers but, in each case, set $p_j = \left(\sum_{i=1}^{j} a_i \right) \bmod 2$.

If n is odd then $b_1 = a_1$ and, for $i = 2,3,\ldots,m$, $b_i = \begin{cases} a_i & p_{i-1} = 0 \\ n-1-a_i & p_{i-1} = 1 \end{cases}$.

If n is even then $b_1 = a_1$ and $i = 2,3,\ldots,m$, $b_i = \begin{cases} a_i & \text{if } a_{i-1} \text{ is even} \\ n-1-a_i & \text{if } a_{i-1} \text{ is odd} \end{cases}$.

For the use of Gray codes to calculate Peano curve coordinates an important theorem was proved by Cole [Cole 85a] that, for odd bases, the operations of conversion to Gray code and reduced radix complementation are commutable. i.e. denoting the n reduced radix complement of a by $a^* = c_1 c_2 \ldots c_m$, where $c_i = n-1-a_i$, for $i = 1,2,\ldots,m$, and, denoting the base n Gray code equivalent of a by a', then

$$(a')^* = (a^*)' \qquad \ldots\ldots(4.8)$$

4.4.2.2 Mapping from Peano curve sequence to image coordinates

Cole defined this mapping in an n-dimensional space but in two dimensions there is a simpler mapping as described here. Let P_i be the i'th order Peano curve in two dimensions, $n > 2$ be an odd integer and m be a positive integer. Given a base n integer $a = a_1 a_2 \ldots a_{2m} (0 \le a_i < n, i = 1,2,\ldots,2m)$ and its Gray code equivalent $b = b_1 b_2 \ldots b_{2m}$ then, if $x = b_2 b_4 \ldots b_{2m}$ and $y = b_1 b_3 \ldots b_{2m-1}$, then the point (x,y) relative to integer Gray code scale axes is the a'th point on the Peano curve P_m.

For example, the 50'th point on P_3 may be found by converting 50 to base 3 giving 1212, with Gray code equivalent 1012. The coordinates of the point are then (02,11) in base 3 or (2,4) in decimal (see Fig. 4.8).

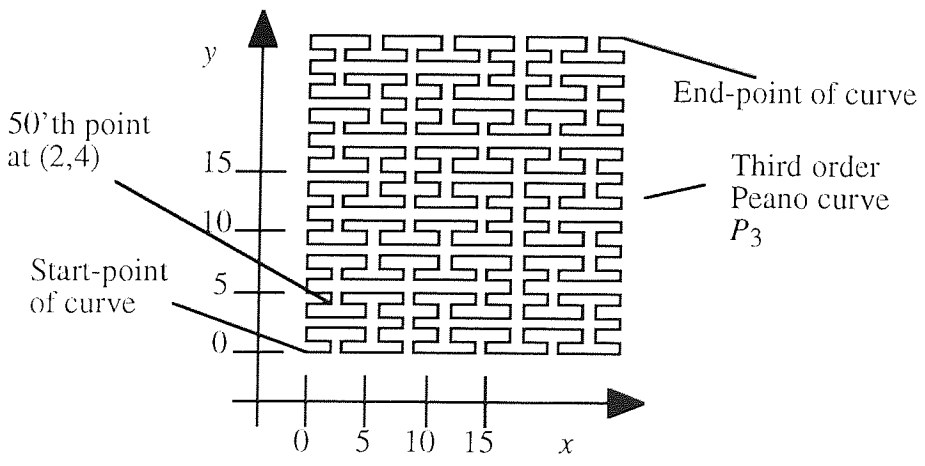


Fig. 4.8. Mapping from third order Peano curve to image coordinates

4.4.2.3 Mapping from image coordinates to Peano curve sequence

The inverse mapping is readily derivable from the forward mapping of Section 4.4.2.2 but extra care needs to be taken to ensure the coordinate values are padded to the correct size before the mapping takes place.

Let (x,y) be any point in a two-dimensional space whose coordinates are expressed as base 3 numbers of m_x and m_y bits respectively. i.e. $x = x_1 x_2 \ldots x_m \left(0 \le x_k < 3, k = 1,2,\ldots,m_x\right)$, $y = y_1 y_2 \ldots y_m \left(0 \le y_k < 3, k = 1,2,\ldots,m_y\right)$. Set $m = \max\left(m_x, m_y\right)$ and let $X = X_1 X_2 \ldots X_m$ where $X_k = \begin{cases} x_k & 1 \le k \le m \\ 0 & \text{otherwise} \end{cases}$ and $Y = Y_1 Y_2 \ldots Y_m$ where $Y_k = \begin{cases} y_k & 1 \le k \le m \\ 0 & \text{otherwise} \end{cases}$. If $a' = Y_1 X_1 Y_2 X_2 \ldots Y_m X_m$ then (x,y) is the a'th point on P_m expressed as a Gray code number.

For example, consider the point $(x,y) = (02,11)$ in base 3 used in Fig. 4.8. This gives X=02, Y=11. Hence a' = 1012, or a=1212 which is the 50'th point as before.

4.4.3 Table-based mappings

Unfortunately the direct mapping identified for the Peano curve cannot be extended to the Hilbert curve (the commutability of the conversion to and from Gray codes is only true for odd bases and so is not valid for the Hilbert curves which depend on base 2 arithmetic). For Hilbert curves. table-based transformations have been found [Cole 86a, Fish 86, Grif 86]. These are less elegant mathematically but are suitable for computer implementation.

Cole's algorithm describes a mapping from the first 2^{2p} integers to the ordered vertices of the *p'th* Hilbert curve H_p. Using the table approach, to convert a Hilbert curve sequence number to an image coordinate, its number is expressed as a binary number with all leading zeroes. The sequence numbering is assumed to start from zero. The binary digits are grouped into pairs. The most significant pair is used to index one of 4 tables A1-A4 (Table 4.3) which yields the most significant bit of each of the x and y coordinates expressed in binary and the number of the next table. This process is repeated p times for an image of 2^{2p} pixels.

Table A1				Table A2			
Digit pair	***x***	***y***	**Next table**	**Digit pair**	***x***	***y***	**Next table**
00	0	0	A2	00	0	0	A1
01	1	0	A1	01	0	1	A2
10	1	1	A1	10	1	1	A2
11	0	1	A4	11	1	0	A3
Table A3				**Table A4**			
Digit pair	***x***	***y***	**Next table**	**Digit pair**	***x***	***y***	**Next table**
00	1	1	A4	00	1	1	A3
01	0	1	A3	01	1	0	A4
10	0	0	A3	10	0	0	A4
11	1	0	A2	11	0	1	A1

Table 4.3. Translation from Hilbert scan sequence to pixel coordinates

For example, to find the coordinates of the 122'nd point on H_4, it is first necessary to convert the number 121 (sequence numbering starts from zero) to binary form, giving 01111001. The most significant digit pair is 01, which from Table A1 gives 1 and 0 as the most significant digits of x and y respectively and specifies Table A1 as the next table. This process is repeated for the next three pairs of digits to give the coordinates of the 122'nd point on H_4 as (1001,0100) in binary or (9,4) in decimal format.

The inverse translation from Hilbert curve vertices to integers can be performed in a similar way using the four tables B1-B4 in Table 4.4.

Table B1			Table B2		
x,y pair	**Integer pair**	**Next table**	**x,y pair**	**Integer pair**	**Next table**
00	00	B2	00	00	B1
10	11	B4	01	01	B2
10	01	B1	10	11	B3
11	10	B1	11	10	B2
Table B3			**Table B4**		
x,y pair	**Integer pair**	**Next table**	**x,y pair**	**Integer pair**	**Next table**
00	10	B3	10	10	B4
01	01	B3	11	11	B1
10	11	B2	01	01	B4
11	00	B4	11	00	B3

Table 4.4. Translation from pixel coordinates to Hilbert scan sequence

Other tabular methods of curve generation have been described by Skarbek *et al* [Skar 89b] and Kamata [Kama 93] each based on two-level look-up tables but requiring varying additional storage.

4.4.4 On-line and off-line curve generation

The choice of an algorithm for generating the curve depends on a trade-off between the storage required for look-up tables (off-line generation) and the computation time to generate coordinates as required (on-line generation). In general it seems better to create a mapping between the raster scan and the space-filling scan for all points and then use this to transform points as required. This strategy has been adopted for experimental work reported in this chapter since it provides a way of testing space-filling curves for existing image compression tools. The computational complexity of building a curve from a concise description is low but the off-line storage required by the coder and decoder for the forward and inverse transformation matrices is considerable. The forward transformation table for an n by n pixel scan requires $n^2\mathrm{INT}(\log_2 n)$ bits and the inverse lookup-table requires $2n^2\mathrm{INT}(\log_2 n)$ bits. Since the overheads of unpacking integers which do not fit exactly into bytes is considerable, $\mathrm{INT}(\log_2 n)$ is set to 8 for $n \le 256$ and 16 for $256 < n \le 2^{16}$. For example, for a 512 by 512 pixel scan the forward transformation look-up table requires 512 kB and the inverse look-up table requires 1 MB.

4.5 Generalised space-filling curves

Previously all of the authors investigating space-filling curves for image compression have selected the basic Hilbert curve or Peano curve (some have even selected the Hilbert curve but called it a Peano curve). There has been no comparative analysis of other curves, the order of the curves or the effectiveness of a single pre-processing scan compared with scanning while compressing. The Peano and Hilbert curves are the most regular curves and the easiest to construct but there are many such curves. Sagan [Saga 91] gave a proof that there are infinitely many space-filling curves but it can really only be applied to curves defined over continuous images and not to images defined on a discrete grid.

A large but finite number of curves can be defined for a discrete grid but these different curves have received little attention and authors even differ in the number of distinct curves which can be derived. However, if there are well-defined rules for their production, it is possible to label all such curves. Image compression algorithms can then be evaluated over many different scans and the optimum scan can be chosen for any given image. The extra information to pass to the decoder to identify the scan is small compared to the image size.

4.5.1 Searching for new curves

In order to decide whether a curve is space-filling a set of criteria is required for acceptance. All reported methods are based on the notion of a basic square grid or tile upon which the lowest order member of each family of curves is inscribed. A space-filling curve is considered to be a tessellation of square tiles containing different orientations of the same basic design which all join up to form a continuous open path. Identification of new curves involves finding new basic patterns which obey a prescribed set of rules. Various sets of these rules have been proposed such as requirements for:

- Self-similarity. The curve can be constructed by the recursive application of a set of rules for connecting components.
- Space-filling. The curve passes through all squares of the grid.
- Self-avoiding. The segments of the curve do not touch or intersect.
- Simplicity. The curve can be drawn with a single stroke of a pen, without lifting the pen or drawing a segment more than once.

Griffiths [Grif 86] introduced an extra restriction on the allowed rotations of tiles which, while in the spirit of mathematical elegance, reduces the number of distinct curves which can be used for image scanning. He observed that, for odd degrees, the copies of the basic tile may be grouped in pairs so that the entry and exit points are similarly placed on corresponding diagonals. Hence any tile on the completed board may be replaced by its partner, giving $2n^2$ variations of each basic design. Griffiths proposed selecting just one of these based on some arbitrary criterion such as requiring adjacent tiles to be either only rotations or only reflections of each other.

Prusinkiewicz *et al* [Prus 90] described a method of generating new curves based on production rules in context-free grammars. They adopted the four criteria listed previously (i.e. space-filling, self-avoiding, simple and self-similar) and adopted an acronym FASS to denote such curves. However, their method also accepts irregular curves, where each tile of the n'th order curve could be composed of different tiles of order n-1.

A method has been adopted here for curve generation here which is most similar to that used by Griffiths but without the restrictions on tile rotations. It is implicit in the definition of a space-filling curves that it is only necessary to find the first order

version of the curve, here called a basic curve, since this contains all the information necessary to recursively generate higher order versions.

Procedure for identifying new basic curves

Each tile is divided into an n by n grid. The path S_1^n is formed so that it enters and leaves the tile in corner grid squares and passes through the centres of each grid cell once only. S_1^n is said to be of degree n.

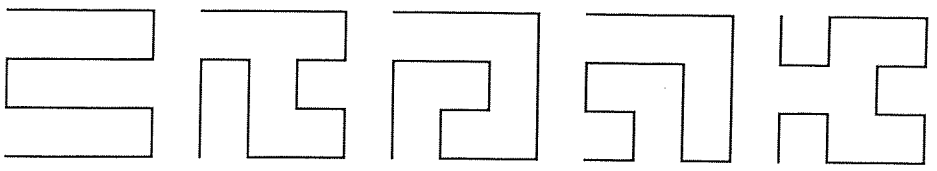
n^2 copies of the tile are fitted together to form a new tile S_2^n. The tiles may have 8 different orientations (rotations through 90° and reflections of 180°). The joining steps between the n^2 copies of S_1^n to form S_2^n must themselves form a path S_1^n.

If n is odd then the start and end points of S_1^n are at opposite ends of a tile diagonal and if n is even they are at the ends of a tile edge.

The tile is rejected if it is a mirror image of a previously accepted tile.

If these rules are met, it is possible to form S_3^n from n^2 copies of S_2^n and so on. The curve S_k^n is said to be of order k.

Using this systematic approach, there is found to be only one curve of degree 2 which is the Hilbert curve and only one curve of degree 3 which is the Peano curve. It will be seen in Section 4.6.2 where hierarchical scans are developed that the fourth order curves are particularly useful. The five fourth order curves are shown in Fig. 4.9. The last of these five basic curves is identical to the second order of the Hilbert curve. i.e. $S_1^4(5) \equiv S_2^2(1)$



$S_1^4(1)$ $S_1^4(2)$ $S_1^4(3)$ $S_1^4(4)$ $S_1^4(5) \equiv S_2^2(1)$

Fig. 4.9. Basic fourth degree space-filling curves

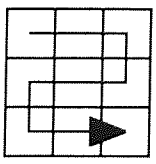
For higher degrees the number of basic curves of degree n, C_n, increases rapidly, as shown in Table 4.5. The numbers of curves identified by other workers are also included in Table 4.5. The correspondence between methods is good for curves of low degrees but worsens for higher degrees. The correspondence is also lower for odd degrees since these provide far greater possibilities for symmetric or mirrored solutions which meet different acceptance criteria.

Curve degree, n	No. of basic curves, C_n, reported for each method		
	Griffiths	Prusinkiewicz	Wilton
2	1	1	1
3	1	1	1
4	5	5	5
5	14	43	18
6	not reported	897	897
7	not reported	not reported	3364
8	not reported	not reported	>20000

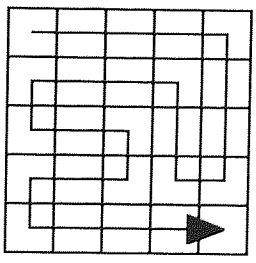
Table 4.5. Numbers of space-filling curves of each degree

The distinguishing feature of curves accepted here is their regularity which leads to a concise description. Other authors were not necessarily concerned with compact representations of the scans and accepted scans where each of the order n-1 tiles making up a tile of order n could be different members of the family of curve at that degree.

For higher degrees, where there is more than curve in each family it is necessary to assign a unique label, or *index* number, to each member of the family. This requires that there should be a unique ordering of the member of each set of curves S^n. There are two ways of achieving this. All of the basic scan patterns can be exhaustively generated and held by the coder and decoder or the basic scan pattern can be included in the overall scan description. In either case, the basic scan pattern can be represented by a canonical ordering (in row-column order) of the sequence in which the n^2 cells of the basic degree n tile are visited. For example, in Fig. 4.10, the degree 3 tile is defined by the sequence $\{0,1,2,5,4,3,6,7,8\}$, and the degree 5 tile by the sequence: $\{0,1,2,3,4,13,12,11,10,5,14,15,16,9,6,19,18,17,8,7,20,21,22,23,24\}$.



Degree 3 tile



Degree 5 tile

Fig. 4.10. Canonical definitions of space-filling curves

Each basic scan pattern represented this way incurs a storage requirement of $n^2 \log_2\left(n^2\right)$ bits. For example, there are 897 basic curves of degree 6 but each only requires $6^2 \log_2\left(6^2\right) = 216$ bits = 27 bytes to define.

In addition, the basic scan pattern requires a listing of the orientations of the basic pattern which are required to create the second and higher order curves. For two-dimensional curves there are only 8 orientations so the orientation data has a storage requirement of $3n^2$ bits per scan.

Thus a complete description of all the basic curves up to the sixth degree has a storage requirement of:

$$\sum_{n=2}^{6}\left\{3n^2 + C_n n^2 \mathrm{INT}\left(\log_2\left(n^2\right)\right)\right\} \text{ bits.}$$

It is straightforward to substitute the numbers of curves of each degree from Table 4.5 to determine that the total requirement is 193,636 bits or approximately 24.6 kB, of which most is taken up by the description of the sixth order curves. This is not excessive and could easily be stored by the coder and the decoder.

4.6 Hierarchical scans

To provide a greater variety of scans a new class of mixed scans is introduced here. These take the form of hierarchies of scans where each order of the scan may use a curve of any degree, order or index. A new notation is defined here to describe these complex scans.

The notation has similarities with that developed by Bourbakis and Alexopolous [Bour 92]. They were interested in defining a large subset of the possible scans with useful properties for image data encryption rather than compression but they required the same key feature, namely that it should be possible to give a concise description of the scan. They created a context free language called SCAN which could be used to identify a subset of the possible scans of an n by n image. They identified 15 basic scan patterns based on a 4 by 4 grid with useful cryptographic properties and assigned to them letters in the set $\Sigma = \{R, C, D, E, A, I, O, L, S, H, Y, W, Z, B, X\}$ where the letters can, with some imagination, be associated with the appearance of the scan patterns. Some examples are shown in Fig. 4.11.

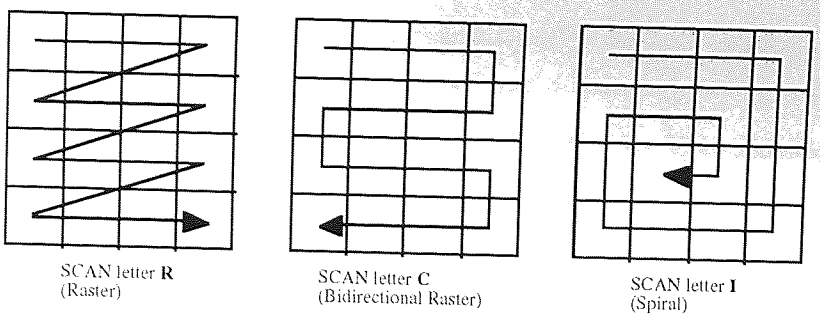


Fig. 4.11. Graphical representation of some of the SCAN letters

A word in the SCAN alphabet is of the form: $L_1 n_1 \# L_2 n_2 \# \ldots \# L_i n_i$ where $L_i \in \Sigma$, n_i is a power of 2, $\prod n_i = n$ and # is a separator. Each word then uniquely describes a scan of the complete image. For example, Fig. 4.12 shows the scan of an 8 by 8 image given by the SCAN word $I4\#R2$ which gives a 4 by 4 spiral (SCAN letter I) of 2 by 2 rasters (SCAN letter R) covering an 8 by 8 grid.

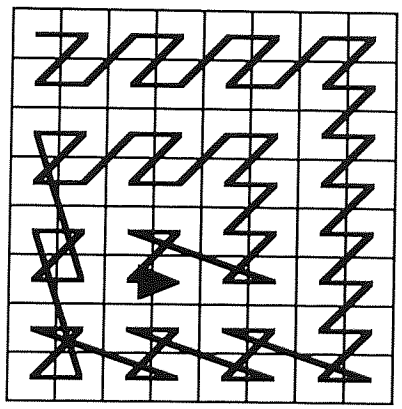


Fig. 4.12. Scan generated by SCAN word $I4\#R2$

The scan patterns selected by Bourbakis are not all suitable as patterns for space-filling curves and are defined on a fixed grid size (corresponding to curves of degree 4) but the same approach can be used to give a hierarchical notation for valid space-filling scans.

In this thesis, hierarchical space-filling scans are represented with the notation given in Eqn. 4.9. The term level is used to denote stages of the scan. In the case of a scan consisting of l levels then:

$$\text{Scan} = S_{k_1}^{n_1}(m_1)\#S_{k_2}^{n_2}(m_2)\#\ldots\#S_{k_i}^{n_i}(m_i)\ldots\#S_{k_l}^{n_l}(m_l) \qquad \ldots\ldots(4.9)$$

where $S_k^n(m)$ is the k'th order instance of the m'th member of the set of space-filling curves of degree n.

The sequence of the levels is reversed compared with the cryptographic scans used by Bourbakis. In this implementation of the scans (due to the recursive way in which the scans are generated) it is easier to reverse the order of the levels so that the first stage of the scan description is the 'innermost' or final stage of the scan

For example, the structure of a two level scan $S_1^4(3)\#S_1^2(1)$ is shown in Fig. 4.13. The scan consists of four copies of the third member of the set of fourth degree basic curves drawn at order one. The four copies are placed in a sequence with orientations defined by the single member of the set of second degree basic curves drawn at order one.

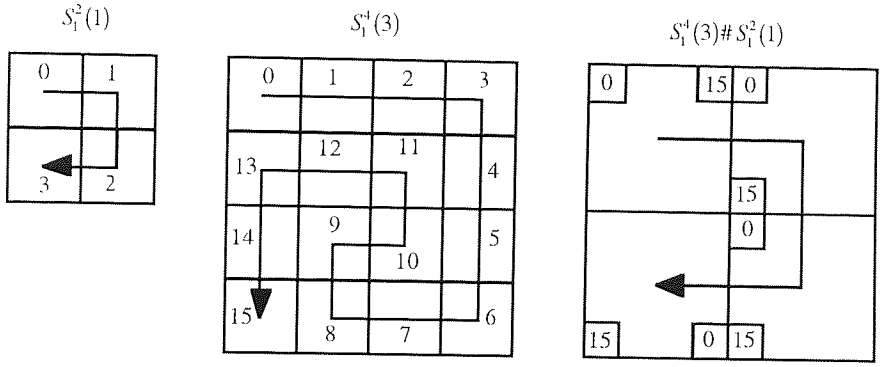
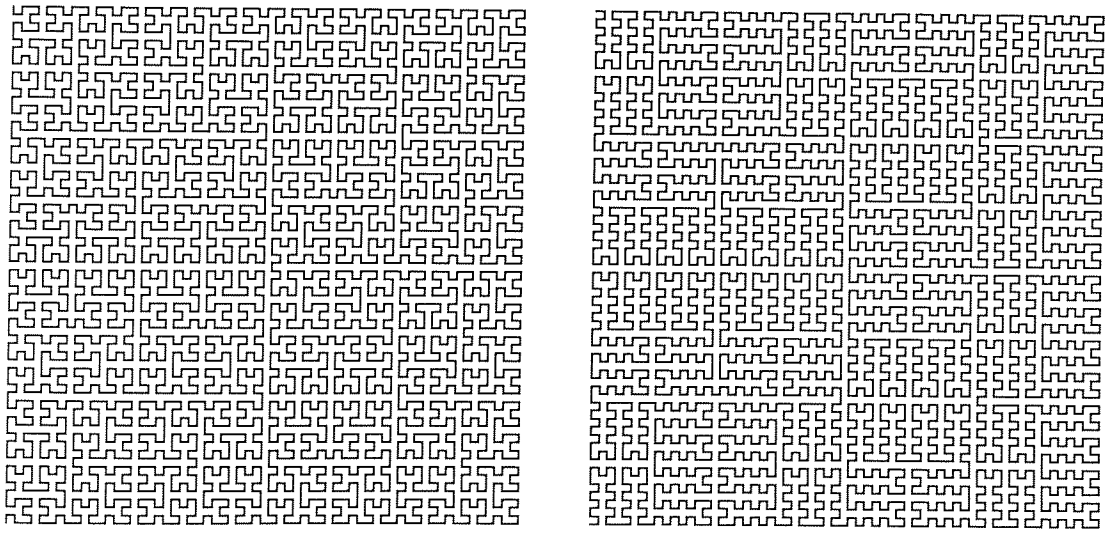


Fig. 4.13. Example of structure of hierarchical scan

Fig. 4.14 shows the details of two scans of a 64 by 64 image. Fig 4.14(a) is the simple one-level scan $S_6^2(1)$ (which is the Hilbert scan) and Fig. 4.14(b) is the three-level scan $S_1^2(1)\#S_1^4(1)\#S_3^2(1)$.



(a) **(b)**

Fig. 4.14. Examples of hierarchical scans

4.6.1 Continuity of curves

It is readily apparent from the way in which the space-filling curves are defined that a hierarchical curve can only be continuous if all its levels are of odd degree or all are of even degree. Without this restriction the start and end points of the curves at each level would not be at the correct corners. This reduces significantly the number of potentially useful hierarchical curves which can be used to scan images of the required size but, as is shown in Section 4.6.2, some image sizes still provide considerable variety in the possible choices of hierarchical curves.

4.6.2 Exact scans of 512 by 512 images

It is possible exhaustively to list all of the scans which exactly cover a grid of 512 by 512 pixels. These can be found by considering all of the 30 possible factorisations of the number 512 which are listed in Table 4.6. The number of permutations is calculated differently if any of the factors are even powers of two since these factors can be represented by two different degree curves. (i.e. factors which are the same even power of two are distinguishable).

Factorisation	Permu-tations	Factorisation	Permu-tations	Factorisation	Permu-tations
2^9	1	$2^2.2^3.2^4$	6	$2.2.2.2^3.2^3$	10
2.2^8	2	$2^3.2^3.2^3$	1	$2.2.2^2.2^2.2^3$	60
$2^2.2^7$	2	$2.2.2.2^6$	4	$2.2^2.2^2.2^2.2^2$	120
$2^3.2^6$	2	$2.2.2^2.2^5$	12	$2.2.2.2.2.2^4$	6
$2^4.2^5$	2	$2.2.2^3.2^4$	12	$2.2.2.2.2^2.2^3$	30
$2.2.2^7$	3	$2.2^2.2^2.2^4$	24	$2.2.2.2^2.2^2.2^2$	120
$2.2^2.2^6$	6	$2.2^2.2^3.2^3$	12	$2.2.2.2.2.2.2^3$	7
$2.2^3.2^5$	6	$2^2.2^2.2^2.2^3$	24	$2.2.2.2.2.2^2.2^2$	42
$2.2^4.2^4$	6	$2.2.2.2.2^5$	5	$2.2.2.2.2.2.2.2^2$	8
$2^2.2^2.2^5$	6	$2.2.2.2^2.2^4$	20	$2.2.2.2.2.2.2.2.2$	1

Table 4.6. Factorisations of 512

To form the scans from the factorisation there are four issues to take into account:

- For a given factorisation, each *different* permutation of the factors gives a different scan. For example, consider a two-level factorisation of 512 such as $2^4.2^5$. This could yield a scan based on a 32 by 32 pixel grid such as S_5^2 replicated 256 times on a 16 by 16 grid defined by the curve S_4^2. The factorisation $2^5.2^4$ would yield a completely different scan based on the curve S_4^2 replicated 1024 times as defined by S_5^2.

- Factors which are even powers of 2 may be tiled by appropriate orders of second and fourth degree curves. For example, S_2^2 and S_1^4 both tile a 4 by 4 grid. Thus each even power of 2 in each factorisation yields two different scans.

- All of the exact scans of a 512 by 512 grid will consist of hierarchies of curves of degrees 2^d, for $d = 1,2,3,....$ However, it has already been seen (Table 4.5) that the number of curves of degree 8 becomes unmanageable so, in practice, the scans will consist of hierarchies of just second and fourth degree curves.

- Although there is only one second degree curve, the Hilbert curve, there are five curves of the fourth degree. Thus each even power of 2 in each factorisation yields five different scans. There is a complication here since one of the five degree 4 curves is identical to the second order degree 2 curves but this is taken into account when counting the curves.

Table 4.7 shows which of the second and fourth degree curves can be used to give each factor size.

Factor size	Possible curves	Factor size	Possible curves
2	S_1^2	2^5	S_5^2
2^2	S_2^2 or S_1^4	2^6	S_6^2 or S_3^4 (or S_2^8)
2^3	S_3^2 (or S_1^8)	2^7	S_7^2
2^4	S_4^2 or S_2^4	2^8	S_8^2 or S_4^4

Table 4.7. Possible curves for each factor

Combining Tables 4.6 and 4.7 to give Table 4.8 gives a total of 100144 possible hierarchical scans which exactly cover a 512 by 512 pixel image. If the duplications caused by the equivalence of scans consisting entirely of Hilbert scans (all of which are the same as a simple scan of H_9) and the equivalence of S_2^2 to one of the five types of S_1^4 are removed there are still 3401 different scans.

To calculate the numbers of possible scans for each number of levels in Table 4.8 it should be noted that sub-scans covering a grid whose size is an even power of two can always be achieved in five different ways (i.e. with the appropriate order of S^2 or the four different types of S^4). Therefore the number of possible curves for each factorisation listed in Table 4.8 is equal to the number of permutations of each factorisation multiplied by (five raised to the power of the number of even powers of two in each factorisation).

To determine how many of the possible scans in Table 4.8 are unique it can be observed that portions of scans consisting of consecutive levels which are just different orders of S^2 may be replaced by a higher order version of S^2 since, for $p,q = 1,2,3....$ $S_p^2(1)\#S_q^2(1) \equiv S_{p+q}^2(1)$. This applies to the families of second and third degree curves which have only one member.

In Table 4.8 the sub-totals for number of scans column are for each factorisation as listed in Table 4.6.

No. of levels in curve	No. of scans	No. of unique scans
1	1	1
2	10+10+10+10=40	8+8+8+8=32
3	3+150+6+150+150+150+1=610	0+104+0+48+48+96+0=296
4	20+60+60+3000+60+3000=6200	0+0+0+1536+0+1536=3072
5	5+500+10+1500+75000=77015	0+0+0+0=0
6	30+150+15000=15180	0+0+0=0
7	7+1050=1057	0+0=0
8	40	0
9	1	0
Totals	100144	3401

Table 4.8. Numbers of possible hierarchical curves

Table 4.8 shows that:

- The vast majority of possible scans come from the factorisation $512 = 2.2^2.2^2.2^2.2^2$ but most of the unique curves come from the four-level factorisations. To a good approximation, the number of scans is proportional to the number of fourth degree curves in the scan description
- The number of unique scans is zero for all levels above five since these are always equivalent to factorisations with fewer levels.

An example of a complex exact scan of a 512 by 512 pixel array is shown in Fig. 4.15. This is the four-level scan $S_1^4(3)\#S_1^2(1)\#S_1^4(1)\#S_2^4(2)$ which includes a second degree scan and 3 different fourth degree scans including one of the second order.

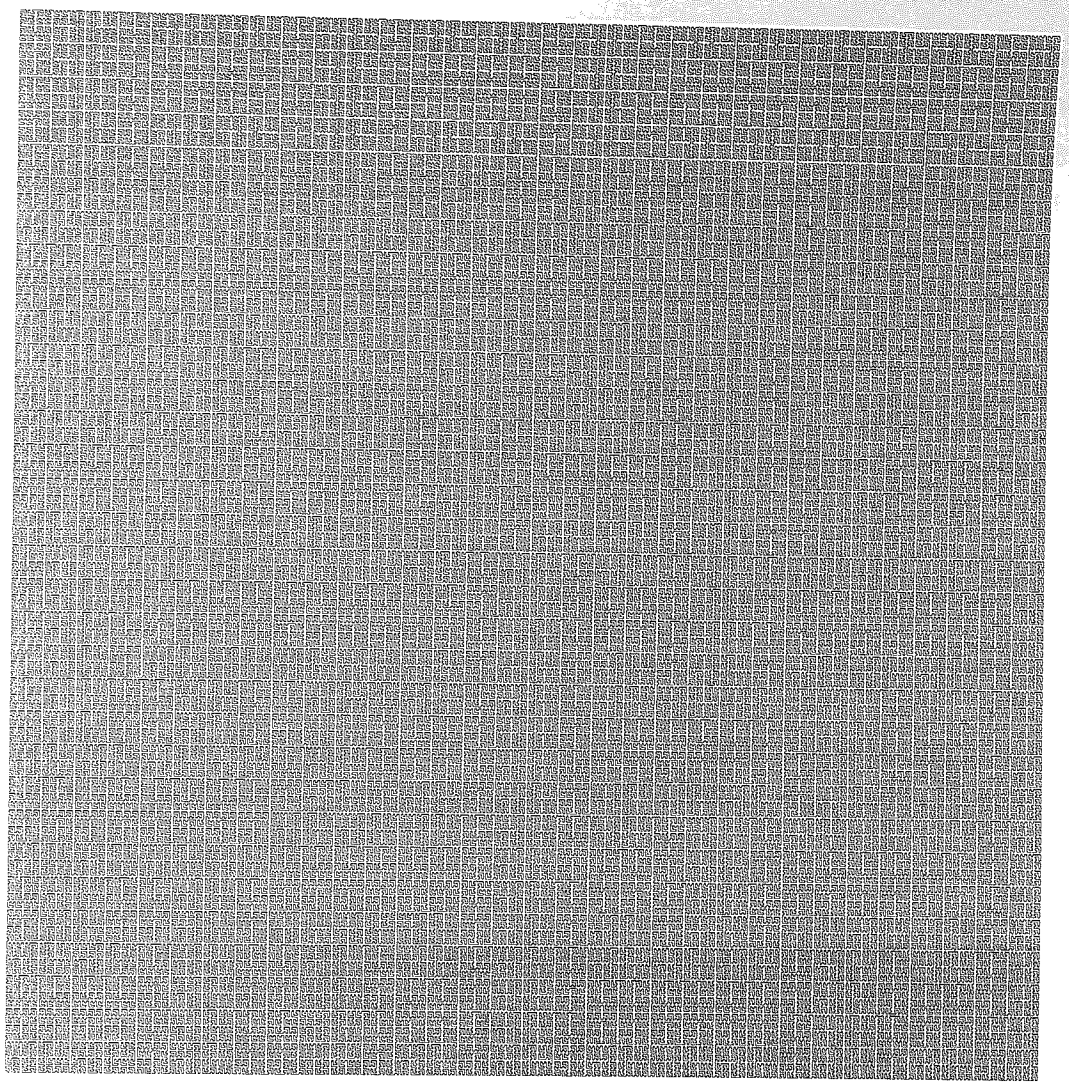


Fig. 4.15. Example of a four-level hierarchical 512 by 512 pixel scan

4.7 Comparison of space-filling scans for image compression

It was established in Section 4.3 that the Hilbert curve is superior to both the simple raster curve and the bidirectional raster curve for pixel re-ordering prior to run-length compression. Using the results from Sections 4.4 to 4.6 it is now possible to perform a comparative study of all the space-filling curves which exactly cover images of 512 by 512 pixels.

Although the Hilbert curve performs better for fractal waveform coding and Group 3 FAX it appears from Section 4.3 that the greatest advantage from space-filling curves comes from any compression process which exploits run-length encoding. To compare the many different hierarchical scans a simple grey-scale run length coder was constructed. This coder is optimised for speed rather than compression since the objective is a comparison between scans rather than the absolute compression ratio achieved.

The three primary test images were scanned in the sequence of each of the 3401 scans identified in Section 4.6 and compressed with this run length coder. Table 4.9 gives a summary of the results and Fig. 4.16 shows a histogram of the mean changes in compression ratio achieved by the scans over the standard ratio which is taken to be that achieved by a raster scan. This is the same measure ΔC_r which was used in Section 4.3.3 to establish the superiority of the Hilbert scan over the raster scan except that in this case the numbers are not expressed as percentages.

No. of levels in scan	No. of scans	Mean ΔC_r	Hilbert ΔC_r	Best ΔC_r
1	1	0.327	0.327	0.327
2	32	0.226	N/A	0.246
3	296	0.262	N/A	0.410
4	3072	0.279	N/A	0.705
All levels	**3401**	**0.277**	**0.327**	**0.705**

Table 4.9. Summary of effects of hierarchical scans on compression ratio

This experiment required considerable computation time (more than a week on a modern workstation) but it was felt that, in the interests of completeness, all possible curves should be tested.

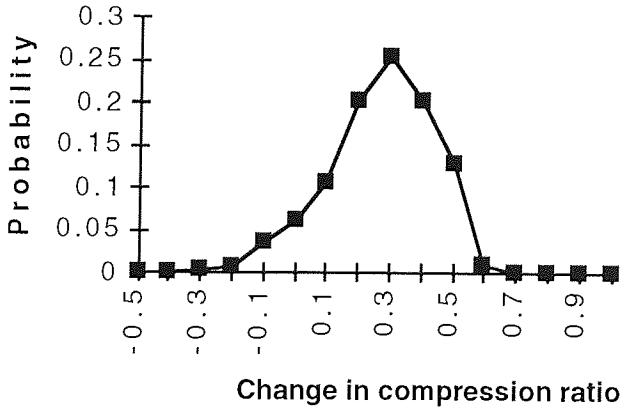


Fig. 4.16. Histogram of compression ratio improvement for exact hierarchical space-filling scans

It may be seen from Table 4.9 and Fig. 4.16 that:

- The mean of the distribution is positive and is at an improvement in compression ratio of 0.277. i.e. a space-filling curve scan will typically improve the compression ratio achieved by a run-length coding method by 28%.

- The distribution is very approximately normal. Standard tables could be used to estimate the probable effect of an arbitrary space-filling scan on the compression but it seems that there is currently no effective solution other than trying many curves until a good compression ratio is achieved.

- The mean improvement is approximately the same however many scans are considered. However, if a large number of scans are tested a small number of curves with markedly better compression are found.

- The simple Hilbert scan performs remarkably well. In this experiment it is better than all of the two-level curves and was improved upon by only three of the three-level curves and 72 of the four-level curves (though these included the curves giving greatly superior compression ratios).

4.8 Extensions to hierarchical scans

4.8.1 Non-exact scans and curve clipping

In the tests on 512 by 512 pixel images many exact scans were identified. However, for some image sizes, there are only a small number of hierachical scans which exactly fit the image. In such cases a strategy of curve clipping may be adopted where the specified scan is clipped uniformly around the image. For example, all scans of the form $S_2^3(m_1)\#S_3^6(m_2)$ define a region $3\times 216 = 648$ pixels square. This may be clipped to scan a 640 by 640 pixel image as shown in Fig 4.17.

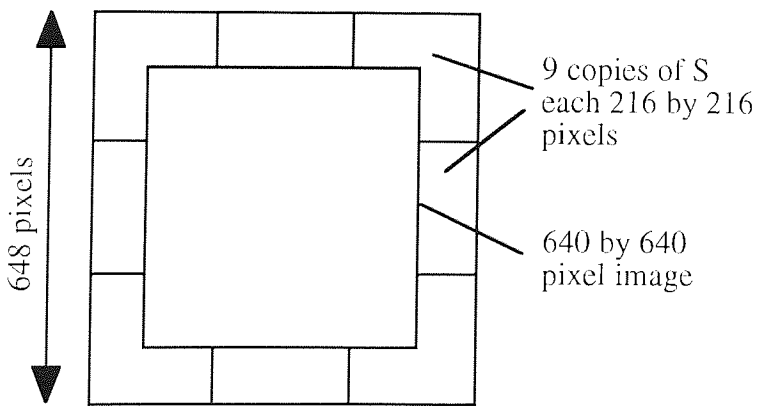


Fig. 4.17. Clipped scan

Most common computer images have standard resolutions such as 640 by 480 pixels, 800 by 600 pixels, 1024 by 768 pixels etc. This is because the video memories which are used to store the images are fabricated with DRAM devices with sizes which are

multiples of 64 kbytes. Thus the image side lengths will frequently have many different factorisations as has been shown to be the case for 512 by 512 images.

4.8.2 Non-square images

It may appear that space-filling curves can only be applied to square images. For example, the *n'th* Peano curve P_n fits in a square of side 3^n and the *n'th* Hilbert curve H_n fits in a square of side 2^n. Clearly, not all images are square. For example current UK television screens have 625 lines with an aspect ratio of 4:3 and the new HDTV standard has 1125 lines with an aspect ratio of 5:3. Drewery [Drew 91] suggested that certain aspect ratios (2:1, 3:1, 4:1, 3:2, 4:3) could be achieved by starting and finishing at selected points of a Hilbert curve. He also considered that small deviations from these ratios could be obtained by modifying the pixel aspect ratio. However, it is preferable to consider how space-filling curves might be fitted exactly into arbitrary rectangular regions.

4.8.2.1 Rectangular curve clipping

A simple strategy which could be adopted is to scan a rectangle by clipping a region from a larger square scan. This has the disadvantage that when the curve next crosses the required region it will not necessarily be adjacent to the previous pixel. However, it would be simple to implement and readily adaptable to different shaped images. In Section 4.8.1, where curve clipping was applied to square scans it was found that the resulting curve discontinuities as the scan passes outside the clipped region reduce the advantages of a space-filling scan but not significantly. Here also, experiments have been performed which show that, for a wide variety of rectangular regions clipped from a larger space-filling scan, typically less than 1% of the scan steps are between non-adjacent pixels. This fraction also decreases as the size of the rectangle increases and as the aspect ratio of the rectangle decreases.

4.8.2.2 Murray curves

As an alternative to curve clipping, Cole [Cole 85b, Cole 86b] introduced the concept of multiple radix (murray) arithmetic and showed how this could be used to define murray curves. These are similar to the Peano curves but, instead of being restricted to square regions, they can be inscribed in a rectangle of sides m, n where m and n are both odd.

A multiple radix (murray) integer is sequence of digits $d_n d_{n-1} \ldots d_1$ together with a sequence of integers $r_n, r_{n-1}, \ldots, r_1$ where r_i defines the radix associated with d_i for $i = 1,2,\ldots,n$. Naturally, $0 \le d_i < r_i$. The usual arithmetic operations can be defined

on murray integers, noting that carry takes place from digit i to digit $i+1$ when the sum in the i'th place exceeds $r_i - 1$. Also, the transformation to Gray code and reduced radix complement can be defined as for normal integers.

Cole showed that, if the side lengths m and n of a rectangle can be factorised into odd factors, to give a pair of odd-based murray integers which are transformed in the same way as base 3 integers to form Peano curves then new curves are formed very much like Peano curves but inscribed within rectangles.

If m and n are not prime then different curves can be constructed for each factoring simply by changing the order of the factors. Fig. 4.18 shows a murray curve inscribed in a 45 by 35 rectangle having x radices 5, 3, 3 and y radices 5, 1, 7.

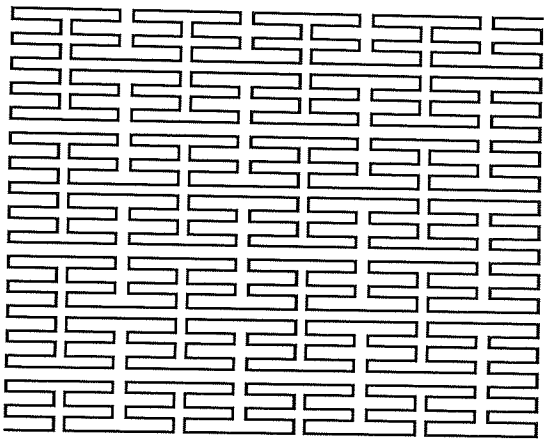


Fig. 4.18. Murray curve

For reasons explained in Chapter 2, only square images have been used for experimental work in this thesis. However, the existence of the murray curves means that most of the analysis may be applied to rectangular images provided that the image side lengths can be appropriately factored. Where no appropriate factoring exists, the curve clipping strategy could be applied.

4.9 Halftoning using space-filling curve scans

Apart from the use of space-filling curves for grey-scale image compression there has been considerable interest in the use of these scans for reducing false image features such as Moiré patterns which occur as a result of half-toning grey-scale images for display on bilevel devices. Claims for the effectiveness of space-filling curve scans have been contradictory. Results presented here suggest that space-filling scans can provide a compromise between the best features of some established halftoning methods.

4.9.1 Half-toning

Photographs and CRTs provide sources of continuous-tone images. For these sources the intensity (grey-level) of each pixel on the display is proportional to the intensity of the pixel. However, there are many forms of display where the output can only be represented by two levels (black or white). Although binary monitors are now rare, virtually all printed images are produced by processes (lithography, letterpress, gravure, laser printing) which have binary output levels. These processes use halftoning to approximate grey-scale appearance at normal viewing distances.

Halftoning involves creating a microstructure of dots which vary locally over a much smaller scale than the image as a whole. This process is also known as spatial dithering. The human visual system spatially integrates the fine patterns of black and white dots to create a perceived grey level. In developing halftoning patterns the objective is to determine a mapping which is subjectively pleasing, free from any false textural contouring or any other artefacts. There are various halftoning strategies such as grey-scale character fonts, ordered dithering and error diffusion dithering.

4.9.2 Grey-scale character fonts

For each grey-level pixel in the input image an n by n pixel region in the output image contains an n by n binary dot pattern (or character) chosen from a character set. For example a 3 by 3 pattern can be used to obtain 10 different grey levels (Fig. 4.19). This approach is conceptually and practically simple but usually gives rise to false contours in the output. The character must be chosen to try and avoid directional emphasis. For example, the dots should be arranged without any obvious pattern. Also if a pixel is black for a particular level it should be black for all greater input levels (See Fig. 4.19).

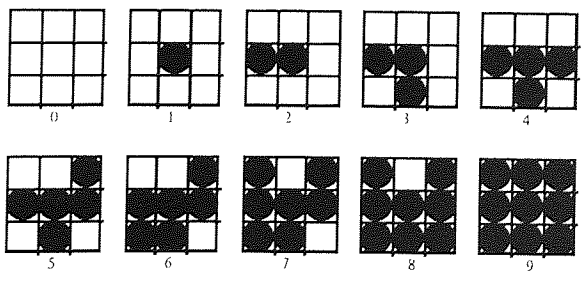


Fig. 4.19. Halftone character maps for 10 grey-levels

Grey-scale maps are rarely used now since dithering strategies usually produce far better results.

4.9.3 Pseudo-random thresholding and dithering

The false contouring generated by the halftoning process can be reduced by introducing random noise to the image. The perceptual effect is to decrease the high-frequency image information. This causes the observer to integrate better the intensity in a region. The noise may be added before or after the halftoning since the addition of pseudorandom noise followed by fixed thresholding is equivalent to thresholding the signal with a pseudorandom threshold. There are two main classes of dithering methods: ordered dither and error diffusion [Ulic 87] together with a hybrid method called dot diffusion [Knut 87].

4.9.3.1 Ordered dither

This is an image independent method commonly used in laser printers. An n by n dither matrix D^n is assumed to be replicated across the whole input image. The output image is formed by pixel-by-pixel comparison of the input intensity and the corresponding dither matrix. Each output pixel is determined by the threshold T_{xy} selected on the basis of the current pixel (x, y) as:

$$T_{xy} = D^n(x \bmod n, y \bmod n) \quad \text{.....(4.10)}$$

Hence the n by n dither matrix allows $n^2 + 1$ distinct grey-levels to be represented. The dither matrix dimension is analogous to a spatial sampling rate. Selection of this dimension must avoid oversampling with too large a dither matrix in smooth areas or undersampling with too small a dither matrix in areas with high frequency content. 4 by 4 matrices are common but larger matrices have been recursively defined [Jarv 76].

Ordered dithering is further sub-divided into dispersed dot dithering and clustered dot dithering, depending on the design of the dither matrices. Dispersed dot dithering gives less artificial texture but clustered dot dithering is most used because of physical limitations in output devices such as laser printers which are not good at producing isolated dots.

4.9.3.2 Error diffusion

Error diffusion dithering methods such as the Floyd-Steinberg [Floy 75] method are image dependent. They are inherently sequential, requiring a scan along the image comparing pixels with a threshold. As each pixel is quantised, the quantisation error is propagated (diffused) along the path. This alleviates some of the artificial texturing cause by ordered dithering. Floyd-Steinberg error diffusion may be modified to

diffuse the error with various weightings among different groups of neighbouring pixels. For example, Stucki [Stuc 81] diffused the error from each pixel to 12 neighbouring pixels with weightings as shown in Fig. 4.20.

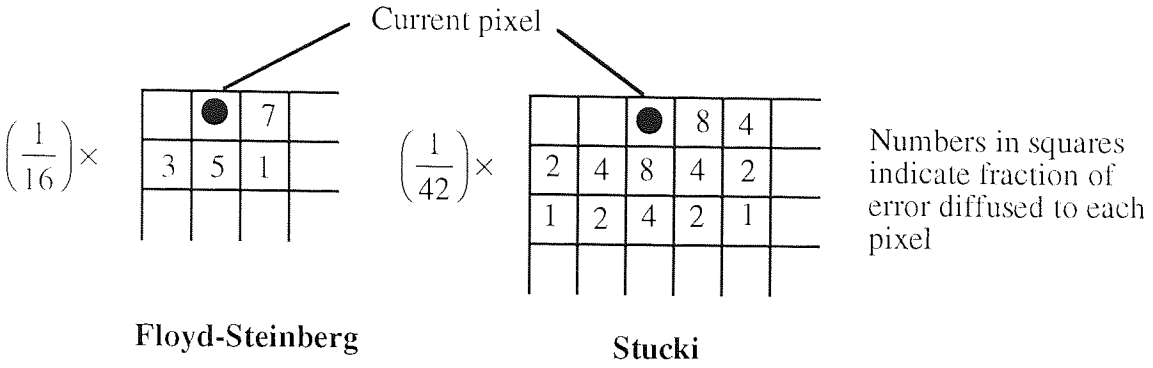


Fig. 4.20. Error diffusion dithering methods

4.9.3.3 Dot diffusion

Dot diffusion [Knut 87] provides a trade-off between ordered dithering and error diffusion. A threshold matrix, similar to that used in clustered dot dithering is applied to the image repeatedly. However, in this case, the matrix determines the order in which the quantisation error in each pixel is diffused among its neighbours.

4.9.4 Error diffusion along space-filling curves

Wyvill and McNaughton [Wyvi 91] proposed a particularly simple method of error diffusion which involves tracing a space filling curve over the image, accumulating intensity at each pixel. When the accumulated intensity is high enough they draw a dot. This leads to artificial texturing mainly in regions of low intensity where there are few dots.

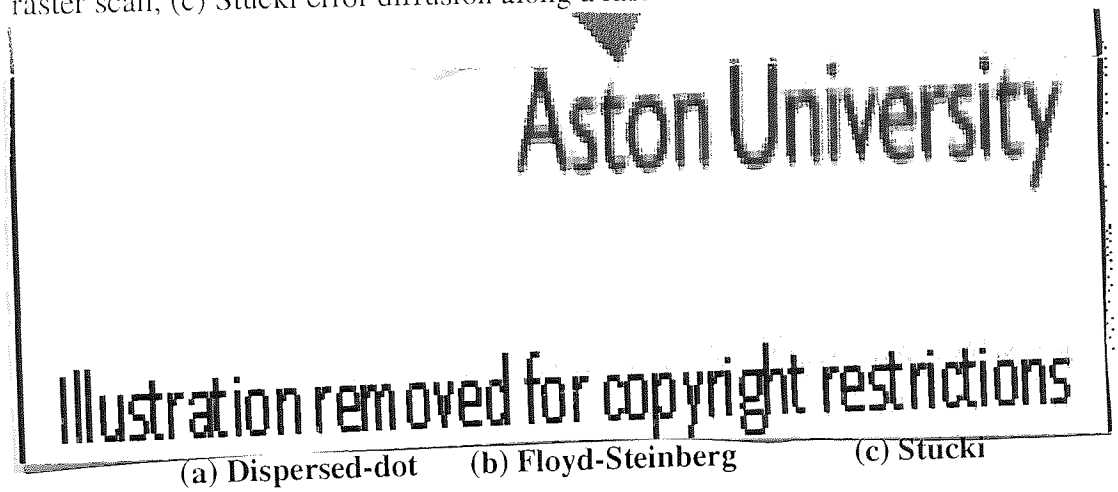
It has been suggested [Chen 92] that error diffusion along the path of a space-filling curve produces subjectively better results. Linnainmaa [Linn 88] tried error diffusion along a Hilbert curve covering the whole image. Velho and Gomes [Velh 91] also used error diffusion along a Hilbert curve but introduced clustering of pixels to produce a scheme equivalent to Knuth's dot diffusion. A parallelisable version of their method was later added by Zhang and Webber [Zhan 93].

Cole [Cole 90, Cole 91] described a method of halftoning rectangular images based on the use of murray curves. He introduced an extra low level of tiling of the output image, similar to the use of grey-scale fonts, where the tile patterns are determined by the intensity value itself along with the previous value of the cumulated sum and the

orientation of the low level tile. He claimed that this method performs significantly better for edges in the image.

4.9 Experimental comparison of halftoning methods

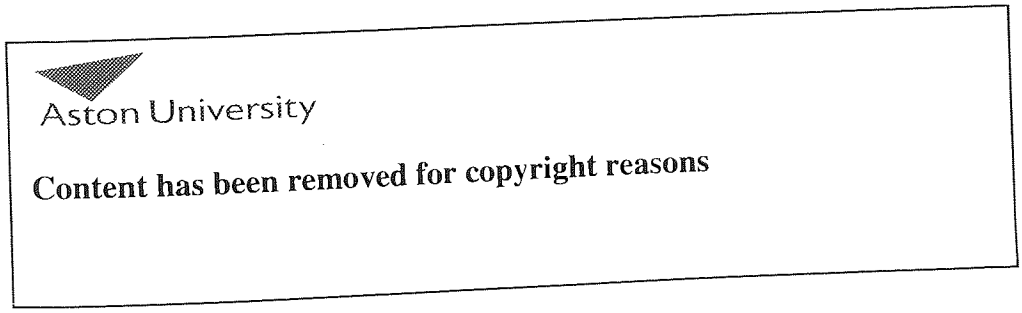
Objective comparison of dithered images is difficult. Measures such as SNR cannot be applied on a pixel-by-pixel basis since different dithering methods do not necessarily place dots at the same coordinates. Subjective comparisons suggest that the preferred dithering method depends greatly on the viewing medium. Results for dithered images assessed on a computer screen show marked differences from images printed with a laser printer. Fig. 4.21 shows detail from the Lena test image dithered using (a) dispersed-dot ordered dithering, (b) Floyd-Steinberg error diffusion along a raster scan, (c) Stucki error diffusion along a raster scan.



(a) Dispersed-dot (b) Floyd-Steinberg (c) Stucki

Fig. 4.21. Raster dithering of the Lena test image

By comparison Fig. 4.22 shows the results of diffusing the error along a Hilbert curve with the error diffused to (a) the next scanned pixel, (b) diffused to the next 4 scanned pixels and (c) diffused to the next 12 scanned pixels.



(a) 1 pixel (b) 4 pixels (c) 12 pixels

Fig. 4.22. Hilbert scan error-diffusion dithering of the Lena test image

The error diffusion lengths of 4 and 12 chosen for illustration in Fig 4.22 correspond to diffusion over the same number of pixels as the Floyd-Steinberg and Stucki error diffusion methods. Subjective tests using student volunteers indicate that most viewers prefer the Hilbert scanned error diffusion when all of the error is diffused to the next scanned pixel. Longer diffusion lengths create a less 'grainy' image than either of the raster scanned error diffusion methods but this is achieved with a loss of some fine detail.

Overall there is little to choose between the more sophisticated raster scan error diffusion methods and error diffusion along a space-filling curve. In subjective tests of both the laser printed images and screen displays of the dithered images the Stucki method (Fig. 4.21(c)) and Hilbert scan error diffusion to a single pixel (Fig. 4.22(a)) received equal top ranking.

4.10 Summary

A considerable volume of experimental data has been generated from which it may be stated with confidence that space-filling curves do provide a superior scanning mechanism whenever an essentially one-dimensional compression method is to be applied to the scanned data. This includes run-length encoding and predictive coding but excludes methods such as quad-tree coding and block-transform methods which already make some use of the two-dimensional nature of the source data.

It has been shown that there are many more space-filling curve than the few well-known examples. A systematic classification scheme has been introduced for these curves such that a space-filling scan of an image may readily be generated.

Some image resolutions such as 512 by 512 pixels have been shown to be particularly suitable for decomposition into many different, regular curves. Most popular computer image resolutions have factorisations which yield moderate numbers of different regular scans. Alternative, but less elegant, scanning solutions have been presented for rectangular images and for images whose horizontal and vertical resolutions are prime numbers.

It has been shown that, if an image is scanned with many different space-filling curves, it is usually possible to find a scan which leads to significantly better compression. However, identification of the optimum scan is extremely computationally demanding and, at present, requires an exhaustive search. Since simulation has shown that the improvement in compression ratio is unlikely to be more than a factor of two, then the search could stop as soon as a curve is found with

reasonable properties (say an improvement of more than 50%). However, the general problem of finding the optimum curve remains unsolved.

Although space-filling curve scans have been shown to be effective in improving the compression ratio achievable by some coding methods, they do not necessarily produce the dramatic increases in compression ratio which are desired. It is therefore sensible to attempt to apply the space-filling curve scans to more sophisticated coding methods which can better exploit them. In Chapter 5, a method known as fractal waveform coding, is presented which benefits considerably from these scans.

Finally, space-filling curves have been shown to provide an interesting alternative method to the most sophisticated current error-diffusion dithering algorithms. However, superiority of the curves for dithering has not been conclusively demonstrated.

Chapter 5. Fractal Waveform Coding

5.1 Introduction

In Chapter 2 the three main classes of image coding schemes (waveform coding, transform coding and model coding) were introduced. In Chapter 4 novel scanning mechanisms were identified which are particularly appropriate for waveform coding systems such as predictive coding and run-length encoding. In this chapter a waveform coding scheme called fractal waveform coding (FWC) is developed and discussed in detail. This is a form of lossy data compression having similarities with conventional techniques such as run length encoding, delta modulation and adaptive delta modulation. The essential difference lies in the use of fractal geometry to decide when to transmit a new predicted signal value. In order to highlight the features of the new scheme the related conventional methods are described first. Then the FWC scheme is subjected to a detailed analysis from which optimum settings of the various parameters which control the compression are determined. Although similar schemes to the FWC scheme presented here are known, previous descriptions have identified neither the practical difficulties nor their solutions which are presented here. Furthermore, it is shown here that the FWC scheme makes an excellent adjunct to the use of space-filling curves for image data compression.

5.2 Conventional waveform coding schemes

The term waveform coding is applied to schemes where the image intensity itself or some simple variation of it (such as the difference between two consecutive pixels) is coded. Previously it has been generally accepted that data compression techniques based on waveform coding are comparatively simple to implement but have a lower efficiency than those based on transform coding or model coding [Jain 81, Lim 90, Nels 92].

It will be assumed here that the input to the coding scheme is an image which is already digitised (i.e. the image has been obtained from a real image source by passing it through a quantiser). This process is the essence of the basic pulse code modulation (PCM) scheme represented in Fig. 5.1.

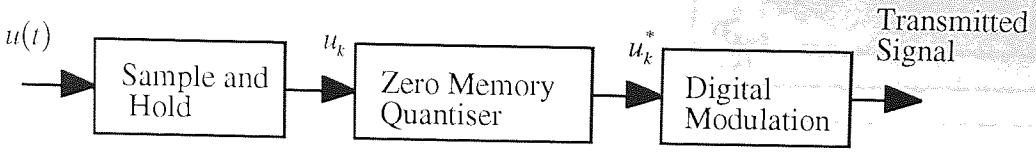


Fig. 5.1. Basic PCM system

In PCM an input analogue signal $u(t)$ is sampled to give a continuous variable u_k. This is quantised to a discrete variable u_k^* which is modulated before transmission. In principle any quantisation and codeword assignment methods can be used for waveform coding but, for simplicity, scalar quantisation and uniform length codeword assignment are predominantly used.

5.2.1 Predictive quantisation

In the basic PCM scheme shown in Fig. 5.1 the quantiser has zero-memory (i.e. each input sample is treated independently). However, almost all real signals will have some statistical dependency between samples. This is particularly true for image data where the sum of the samples must represent a meaningful image to a human observer.

In order to see the potential for compression that this gives, consider the variance of the intensities of adjacent pixels in an image. If these adjacent pixel intensities are modelled by scalar random variables X and Y, then the variance of a linear combination of X and Y is given by:

$$\sigma^2_{aX+bY} = a^2\sigma^2_X + b^2\sigma^2_Y + 2ab\sigma^2_{XY} \quad \text{.....(5.1)}$$

Since, for adjacent pixels, $a=1$, $b=-1$, and $\sigma^2_X = \sigma^2_Y$, the variance of the difference in adjacent pixels is given by:

$$\sigma^2_{X-Y} = 2\left(\sigma^2_X - \sigma^2_{XY}\right) \quad \text{.....(5.2)}$$

Thus, for highly correlated images (large σ^2_{XY}), the variance of the difference in adjacent image pixel intensities σ^2_{X-Y} can be much less than the variance of the pixel intensities σ^2_X. Using basic results from information theory [Shan 48], this can be exploited in differential coding since a signal with a smaller variance is easier to compress than the raw image data.

This has been verified experimentally for the three primary test images with results as summarised in Table 5.1.

Image	Pixel Variance σ_X^2	Pixel Correlation σ_{XY}^2	Interpixel Variance σ_{X-Y}^2
Lena	2406	2351	109
Barbara	2841	2478	726
Boats	3405	3281	248

Table 5.1. Correlation of pixels in test images

5.2.2 Differential pulse code modulation (DPCM)

Prediction can be added to the basic PCM scheme to give differential pulse code modulation (DPCM). The overall system is shown in Fig. 5.2.

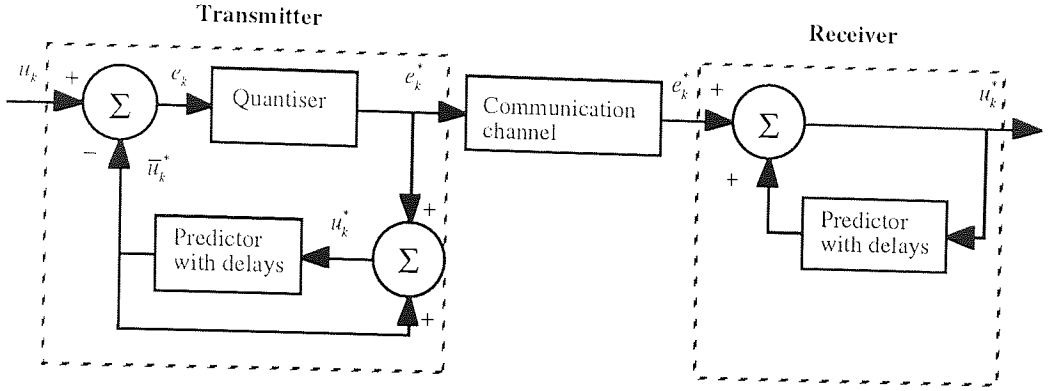


Fig. 5.2. Basic DPCM system

As for PCM, the input to the system is a sequence of samples $\{u_n\}$. In considering the processing of sample u_k it is assumed that all previous samples have already been transmitted and that both the transmitter and receiver can make use of the information contained in these samples. The reproduced value of u_n is denoted by u_n^*. A quantity $\overline{u}_k^*$ is formed which is an estimate of u_k predicted from the previous samples and a predicted error e_k which is defined by:

$$e_k \equiv u_k - \overline{u}_k^* \quad \ldots\ldots (5.3)$$

Then, for transmission, it is sufficient to quantise only e_k instead of u_k. Hence, the reproduced value of u_k is given by :

$$u_k^* = \overline{u}_k^* + e_k^* \quad \ldots\ldots (5.4)$$

DPCM is widely used in this form. It gives a gain in MSNR over simple PCM which is proportional to the ratio of the variances of the sample and error sequences.

5.2.3 Delta modulation (DM)

Delta modulation (DM) (sometimes termed differential modulation or linear modulation) is a simplified DPCM scheme which may be considered to be the limiting case of DPCM where just one-bit quantisation is used. The predictor is simply a one-step delay function so, although the system signal flow is the same as in Fig. 5.2, the estimate of the next value and the predicted error are now given by:

$$\overline{u}_k^* = u_{k-1}^*, \text{ and } e_k = u_k - u_{k-1}^* \quad \ldots\ldots (5.5)$$

DM suffers from three main limitations: slope overload, granularity noise and instability. Slope overload occurs when the sampling rate is too low, allowing the signal to change too rapidly for the sampled steps to keep up. The coder can only respond to this by making several delta steps. The overload may be reduced by using a higher sampling rate but this reduces the compression. Granularity noise is the step-like nature of the output when the input signal is almost constant. Both of these problems (which are shown in Fig. 5.3) can be reduced by low-pass filtering the input and output signals or by using a tristate-DM scheme [Paz 76].

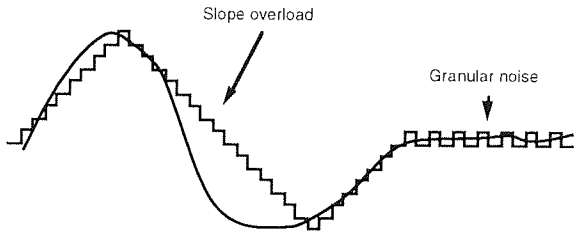


Fig. 5.3. Problems with delta modulation

A compromise is necessary between slope overload and granular noise for a given bit rate. A method of controlling this compromise is provided by the step-size Δ which is an important design parameter The difference signal e_k has values $\pm\Delta/2$. When Δ is small the granular noise is reduced but the slope overload is increased (leading to blurring in the case of an image). When Δ is large the slope overload distortion is reduced but the graininess in smooth regions increases. DM results in a lower bit rate than logarithmic quantisation for a given signal-to-noise ratio if that ratio is low but, as the desired quality increases the bit rate grows faster than for logarithmic PCM. Δ may be adjusted, leading to adaptive delta modulation (ADM) [Lei 77]. Typically Δ is increased or decreased by a constant factor which depends on whether the new transmitted bit will be equal to or different from the last one.

$$\Delta_{n+1} = \begin{cases} 2\Delta_n & u_{n+1} < u_n < u_{n-1} \text{ or } u_{n+1} > u_n > u_{n-1} \\ \Delta_n/2 & u_{n+1}, u_{n-1} < u_n \text{ or } u_{n+1}, u_{n-1} > u_n \end{cases} \quad \ldots\ldots (5.6)$$

The third problem with DM, instability, is more difficult to solve. The transmitter does not have to include the receiver structure to compute $\overline{u}_k^*$ but this means the quantisation noise can accumulate. Attempts have been made to alleviate this problem by attenuating the predicted value by a constant called *leak* [Cutl 71] but this has not proved satisfactory.

Although DM systems have been found useful in the past for applications such as telephony, they have been largely superseded by modern DPCM systems. Recently, though, there have been proposals to improve the performance of DM systems by adopting non-uniform sampling [Zhu 94, Leung 94]. The FWC method described in this chapter provides a strategy for selecting the sample points required by such non-uniform sampling schemes.

5.2.4 Run length encoding (RLE)

Run-length encoding is a well-established technique for encoding images [Prat 78, Scha 89] which is routinely used in a one-dimensional form for Group 3 FAX transmission and in two-dimensional form for Group 4 Fax transmission [Urba 92]. The principle is simple; instead of transmitting each pixel intensity, runs of pixels at the same intensity are identified and only the intensity and the number of pixels in the run need be sent. There are numerous variations and enhancements on this basic principle [Warn 90]. The main difficulty is in setting the maximum run-length since the word size for the run-lengths has to be carefully chosen to accommodate typical rather than maximum lengths.

In addition, most natural scenes contain only very short run-lengths at a single intensity or colour, and are poorly compressed by lossless RLE. Run-length coding of grey-scale images is usually achieved in a modified way by only transmitting a new run when the difference in pixel intensities exceeds a certain threshold T.

Knowledge of typical run length statistics is required before the maximum run-length L and threshold T can be determined. Experimental results are shown in Fig. 5.4 for the three principal test images. Since the effect of varying the threshold is clearly dependent on the number of quantisation levels in the image, a new quantity, relative threshold, is formed (Eqn. 5.7) by pre-quantising the images to a smaller number of intensity levels q to give a more meaningful representation of the effect of the threshold.

$$\text{Relative threshold} = T/q \qquad (5.7)$$

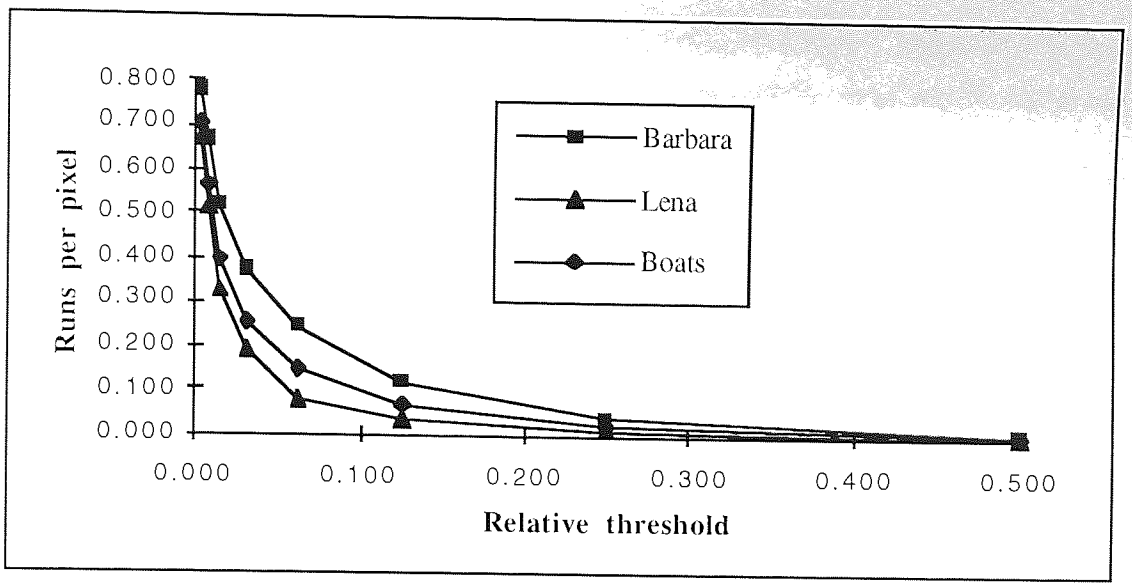


Fig. 5.4. Run-length statistics for test images

The results are also normalised by plotting the ratio of runs to pixels (instead of the number of runs) against the relative threshold. Intuitively, for an image consisting of random pixel values, one might expect a rectangular hyperbolic relationship of the form of Eqn. 5.8.

$$\text{Average Run-length} \propto 1/T \qquad (5.8)$$

This can be deduced from a simple argument. If the probability P_T of the interpixel difference exceeding T is constant along a line, then $P(\text{Run of length } 1)=P_T$. The probability of a run of length 2 is $(1-P_T)P_T$ and the probability of a run of length L is given by $P(L)=(1-P_T)^{L-1}P_T$. If P_T is proportional to T then $P(L) \propto 1/T$.

For real images the compression curves lie below the rectangular hyperbola by an amount which gives a measure of the 'quality' of the compression.

The results achieved are highly dependent on how the threshold is chosen. The requirements on the threshold are conflicting; small threshold values improve the quality of the reconstructed image while minimising compression whereas large values introduce significant distortion but give much better compression. This difficulty is overcome by the fractal waveform coding scheme introduced in this chapter.

5.3 Fractal waveform coding (FWC)

The key modification to RLE introduced by fractal waveform coding is the determination of the predicted approximation to the next pixel level. Unlike other lossy modifications to RLE, this approximation is based on the original ideas of Mandelbrot

in his discussion of the length of the coastline of Britain [Mand 67, Mand 82]. It may be recalled from Chapter 3 that a generalisation of Mandelbrot's ideas may be stated as:

$$(\text{measurement of variable}) \propto (\text{scale at which variable is measured})^{-\mathfrak{D}} \quad \ldots\ldots\ldots\ldots (5.9)$$

where $\mathfrak{D}$ is the fractal dimension of the process.

In fractal waveform coding a 'one-dimensional signal' waveform is extracted from the image data and this is treated as a two-dimensional object with dimensions time and amplitude. Using Eqn. 5.9 predictive coding can be used by 'measuring' this waveform at different scales to generate simpler versions of the image data and thus compress the data.

The first published discussion of a method of this type for image compression was by Walach and Karnin [Wala 86]. They also used the analogy of measuring the length of a coastline by setting dividers to a prescribed length, termed the yardstick length Y, and walking the coastline, with each new step starting where the previous step left off. The coastline is constructed by drawing a line through the grey levels of the pixels of the image under consideration and considering this to be a two-dimensional object. The issue of how this line should be drawn is discussed later in Section 5.5 but, for simplicity, it may be assumed to be straight. This is illustrated in Fig. 5.5, where the yardstick length is Y and the image intensity or grey level at x_i is g_i. One end of the yardstick is placed at the first grey level $P1 = (x_i, g_i)$ and the other end intersects the waveform at $P2 = (x_{i+1}, g_{i+1})$.

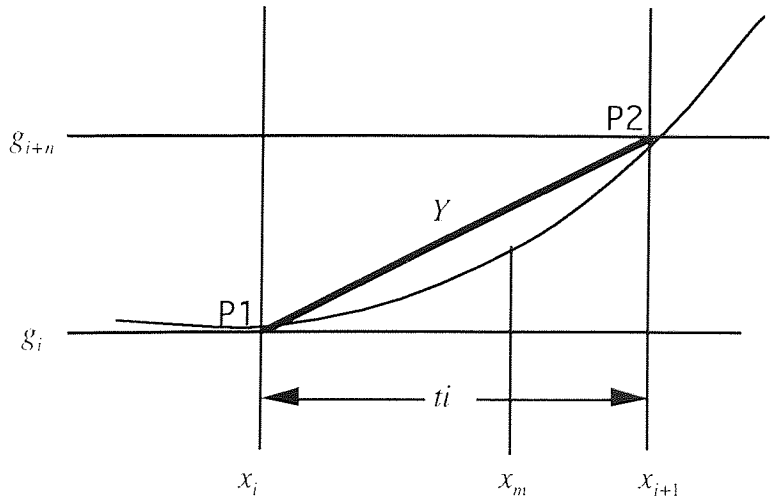


Fig. 5.5. Fractal waveform coding

Various decisions may be made about the point P2. Should x_{i+1} be restricted to integer values? Which point is accepted if the crossing is not unique? These two issues are

related. If non-integer values for x_{i+1} are accepted such as x_m shown in Fig. 5.5, then there will be an infinite number of possible values for P2. The solution adopted here is to accept only integer values but to allow more complex interpolation methods when restoring the waveform (see Section 5.2.6).

The projected distance t_i traversed by the yardstick on the i'th step is then simply $t_i = x_{i+1} - x_i$. The grey level at x_{i+1} is

$$g_{i+1} = g_i + sign_i\sqrt{Y^2 - t_i^2} \quad \text{.....(5.10)}$$

where $sign_i$ distinguishes the cases of ascent and descent.

In Fig. 5.6 the resultant set of transmitted points is shown for an image line.

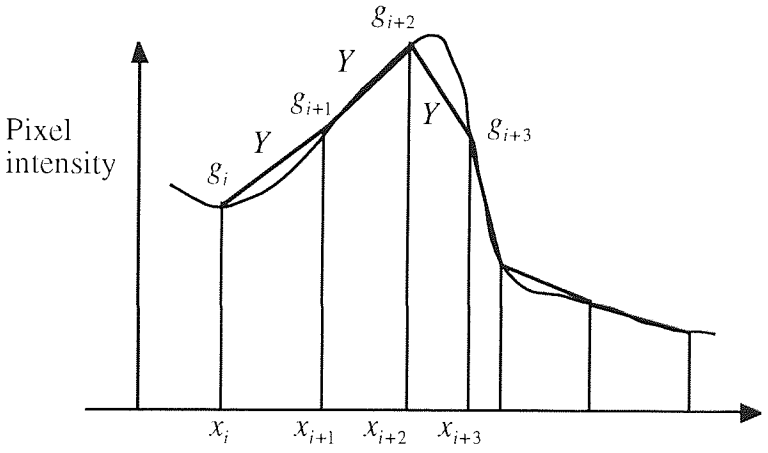
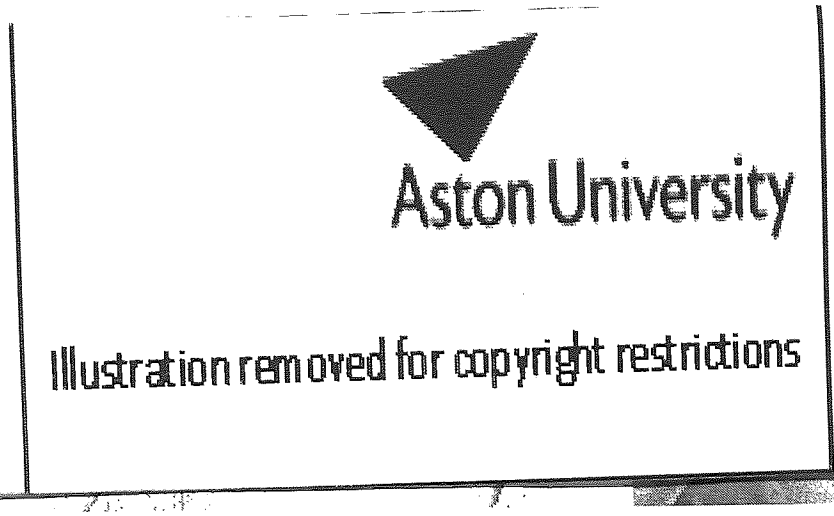


Fig. 5.6. FWC in operation along a line of pixels

A typical example of the FWC scheme in use at a compression ratio of approximately 10:1 is shown in Fig. 5.7 for the Lena test image. Fig. 5.7(a) shows the pixels selected for transmission by running a yardstick of length 31 along the image data. Fig. 5.7(b) shows the restored image after interpolation. Note that in Fig. 5.7(a) the grey level is correct at each of the selected points. In Fig. 5.7(b) simple linear interpolation has been applied between the points. More sophisticated interpolation schemes are considered later in Section 5.2.6.



(a) (b)

Fig. 5.7. Stages in fractal waveform coding

5.4 Choice of yardstick length

In the basic version of the coding scheme, the yardstick length Y is the critical design consideration which controls the trade-off between the compression ratio and the quality of the restored image. A smaller value for Y will reduce the level of compression and improve the reconstructed image quality. The length must be chosen to give the required level of detail. The perceived effect depends on factors such as the resolution of the display and its distance from the viewer but the primary influence on the useful range for Y is the number of quantisation levels in the image data.

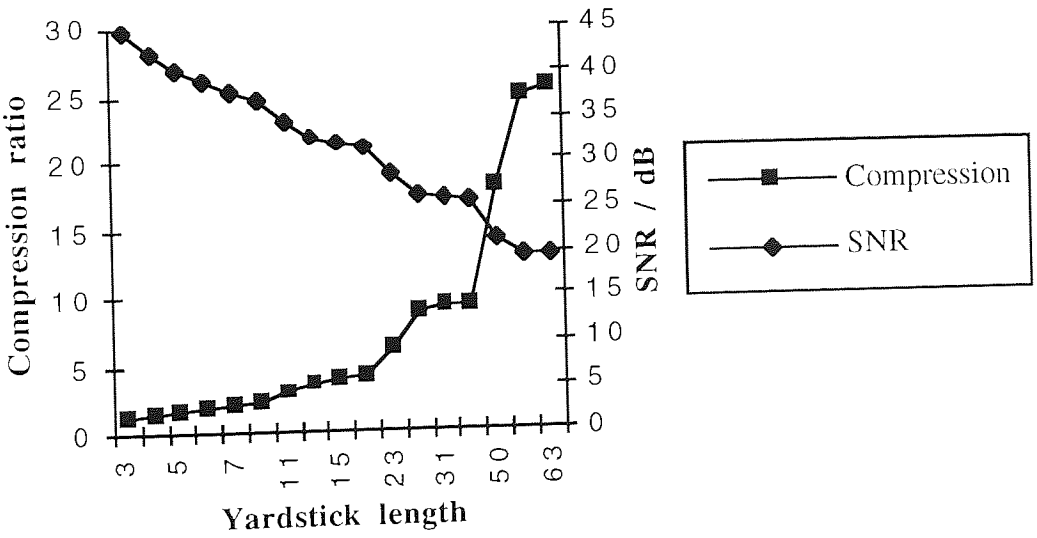


Fig. 5.8. Effect of varying yardstick length

The effect of varying Y is demonstrated for the set of three primary test images. The mean results are shown in Fig. 5.8, from which it may be seen that good compression ratios are achieved at objectively good SNR values. However, the results are not always subjectively acceptable, particularly for edge features. Methods of coding the edges are examined in Section 5.7.

The 'steps' in the compression curve are not significant. They are caused by the process of allocating storage for the step lengths up to a maximum dictated by the yardstick length Y. Particularly wasteful lengths occur at $Y = 4,8,16,32,...$ since these correspond to cases where n bits are used to represent only $2^{n-1}+1$ different step lengths. Alternatively it could be viewed that yardstick lengths of the form $Y = 2^n - 1$ are good lengths since step sizes in the range $[0, Y]$ are represented without waste.

Yang, Wu and Mills [Yang 88] considered the possibility of making the yardstick length adaptive to the local image characteristics. Their system could switch between two yardsticks but the criterion for switching yardsticks was simply that a coded length had been encountered which was greater than or less than a pre-set threshold. They did not discuss how this threshold might be chosen or the overheads inherent in this process. A more general method of adaptively switching yardsticks is presented here in Section 5.6.1.

5.5 Vertical expansion factor

Although the fractal coding scheme automatically produces a trade-off between resolution and quantisation errors, the relative impact of each may be set by scaling all grey levels by a fixed parameter. Walach and Karnin termed this the vertical expansion factor (VEF) [Wala 86]. This transformation is applied before transmission and the reverse transformation is applied after reconstruction. A high value for the VEF increases the relative importance of the vertical direction since it leads to low quantisation errors while a low value for the VEF emphasises the horizontal direction (i.e. accurate resolution).

This produces slightly different results from varying the yardstick length because VEF may be varied continuously whereas Y is restricted to integer values. The effect of varying VEF (with Y set to 15) is shown in Fig. 5.9 This should be compared with Fig. 5.8 which showed the effect of varying Y.

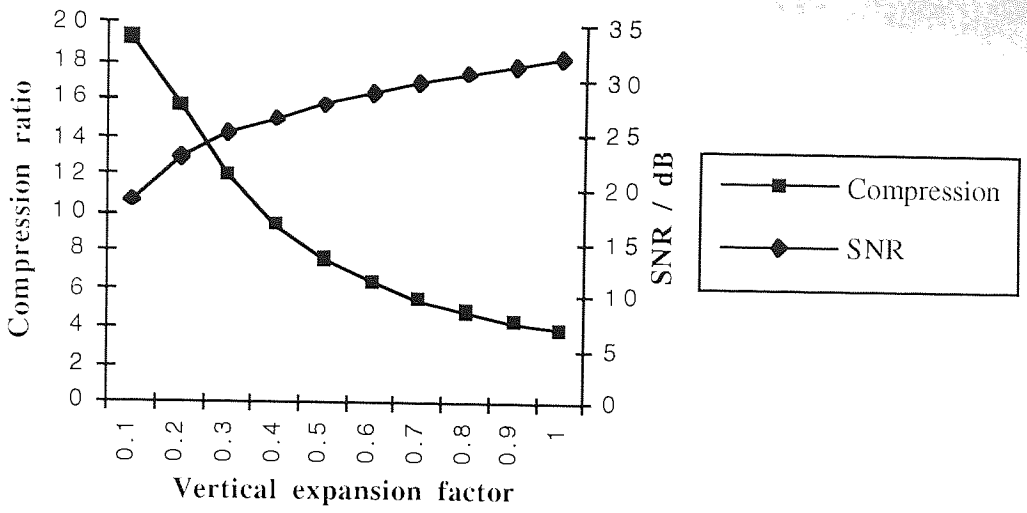


Fig. 5.9. Effect of varying vertical expansion factor

It may be recalled from Section 5.3.1 that, if only Y is varied, there is little advantage in choosing values other than of the form $Y = 2^n - 1$. It may be seen from Fig. 5.9 that similar compression and image quality may be obtained, but without the 'steps' in the compression ratio, by varying VEF as well. In effect, intermediate values of Y may be simulated without the wastage of using n bits to represent up to only $2^{n-1} + 1$ different step lengths by transferring the lost accuracy along the scan to more accurate resolution along the intensity axis.

5.5.1 FWC quality factor

It is possible to form a single constant Q_{fwc}, analogous to the quality setting in JPEG compression, which controls the trade-off between compression ratio and quality by making Y and VEF 'hidden variables'. For $G = 256$, experimental results indicate that the useful ranges of values for Y and VEF are approximately 3 - 63 and 0.5 - 1.0 respectively. Thus Q_{fwc} is given a range of 1-100 having an approximately linear effect on compression and quality, with 1 giving the best compression and poorest quality and 100 giving the lowest compression and highest quality, by defining Y and VEF in terms of Q_{fwc} as in Eqns. 5.11 and 5.12.

$$Y = \begin{cases} 63 & Q_{fwc} \le 5 \\ 31 & 5 < Q_{fwc} \le 35 \\ 15 & 35 < Q_{fwc} \le 65 \\ 7 & 65 < Q_{fwc} \le 95 \\ 3 & Q_{fwc} > 95 \end{cases} \quad \text{.....(5.11)}$$

$$\text{VEF} = \begin{cases} 0.5 + 0.5\frac{Q_{fwc}}{5} & Q_{fwc} \le 5 \\ 0.5 + 0.5\frac{(Q_{fwc}-5)}{30} & 5 < Q_{fwc} \le 35 \\ 0.5 + 0.5\frac{(Q_{fwc}-35)}{30} & 35 < Q_{fwc} \le 65 \\ 0.5 + 0.5\frac{(Q_{fwc}-65)}{30} & 65 < Q_{fwc} \le 95 \\ 0.5 + \frac{(Q_{fwc}-95)}{5} & Q_{fwc} > 95 \end{cases} \quad \text{.....(5.12)}$$

The performance of the FWC system as a function of the new quality setting is shown in Fig. 5.10.

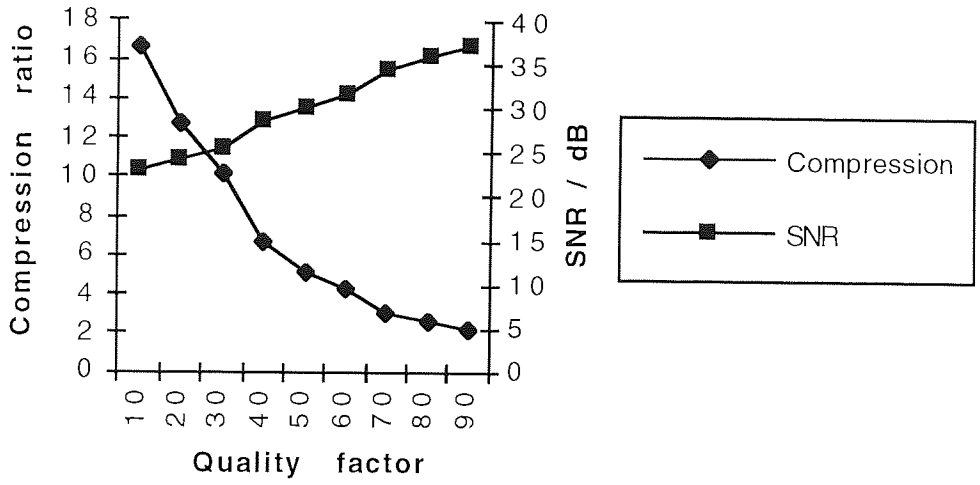


Fig. 5.10. Effect of varying FWC quality setting

All subsequent experiments using the FWC system are performed using Q_{fwc} as a variable rather than Y and VEF.

5.6 Trigger function

Again referring to Fig. 5.5, a decision is required on how to compute the horizontal distance t_i covered in the current step. As previously described, this is conceptually equivalent to fixing one end of the yardstick at P1 and allowing it to rotate about this point until the other end meets the waveform. This does not, however, provide a convenient mechanism for calculating the required distance. A more suitable method is to calculate the absolute difference between the current level g_i and the Y subsequent data samples $(g_{i+1},\ldots,g_{i+Y})$ and to compare these values to a *trigger function* $TF[n]$, defined over the range $1 \le n \le Y$, representing the arc drawn by the non-fixed end of the yardstick. The yardstick will be stopped at the first location for which the absolute difference in levels exceeds the trigger function. i.e.

$$|g_i - g_{i+n}| \geq TF[n]; 1 \leq n \leq Y \quad \text{.....(5.13)}$$

A trigger function determined in this way is here termed a *circular* trigger function. For example, for the waveform shown in Fig. 5.11, using a circular trigger function, a step of length 6 would be sent.

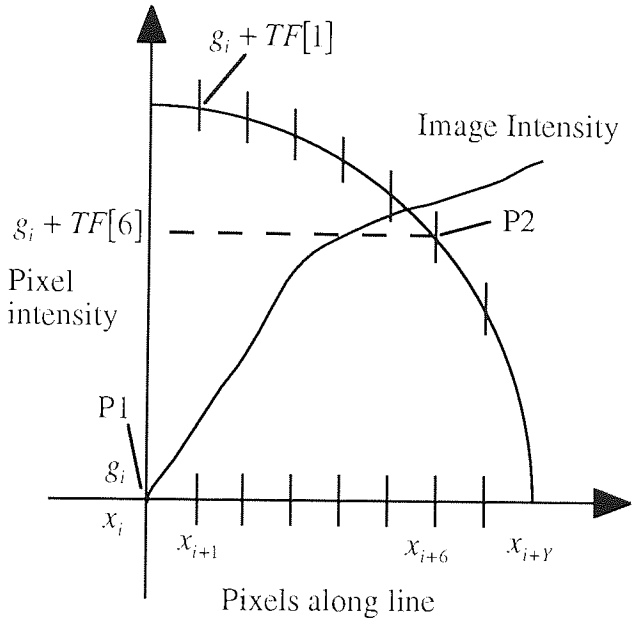


Fig. 5.11. The circular trigger function

Since the choice of the next transmitted point is determined not by an idealised form of the FWC process, where the yardstick rotates until it meets a continuous waveform, but by comparison of subsequent data samples with a discrete trigger function, then the possibility arises of choosing some other, arbitrary mathematical function to form the trigger function. Since only the magnitude of the pixel differences is compared with the trigger function, a monotonically decreasing set of values gives a smaller error in regions where the pixel differences are changing only in sign.

e.g. for $Y = 15$, $TF[n] = [15\ 15\ 14\ 13\ 13\ 11\ 10\ 9\ 8\ 7\ 5\ 3\ 1\ 0]$; $1 \leq n \leq 15$

Walach and Karnin [Wala 86] chose an exponential function for which they reported good experimental results. However, their trigger function did not fall to zero at $n = Y$. Experiments here found that trigger functions where $TF[Y] \neq 0$ were unworkable since step lengths of Y then yield indeterminate pixel intensities. This is examined further when the stability of the FWC scheme is discussed in Section 5.8.

There is no *a priori* reason why any particular function should be superior to the original circular function but it does seem reasonable to choose a function which is related to the interpolation technique used for the restored image. Functions which lie 'above' the linear trigger function will improve step accuracy for shorter steps and become increasingly inaccurate for long steps. Here, a greater variety of trigger functions which lie 'above' and 'below' the linear trigger function have been tested. These functions, whose generating formulae are listed in Table 5.2, are shown in Fig. 5.12.

Trigger Function Name	Generating Formula
Linear	$TF[n] = Y - \left(\frac{Y(n-1)}{Y-1}\right)$ where $2 \le n \le Y-1$
Circular	$TF[n] = \sqrt{Y^2 - n^2}$ where $2 \le n \le Y-1$
Exponential (with parameter α)	$TF[n] = Y\exp\left(-\left(\frac{n-1}{\alpha}\right)\right)$ where $2 \le n \le Y-1$; default value $\alpha = \ln(Y)$
Negative Exponential (with parameter α)	$TF[n] = Y - \exp(-\alpha(n-1))$ where $2 \le n \le Y-1$; default value $\alpha = \ln(Y)$
Hyperbolic	$TF[n] = \frac{Y}{n} - 1$ where $2 \le n \le Y-1$

Table 5.2. Trigger functions

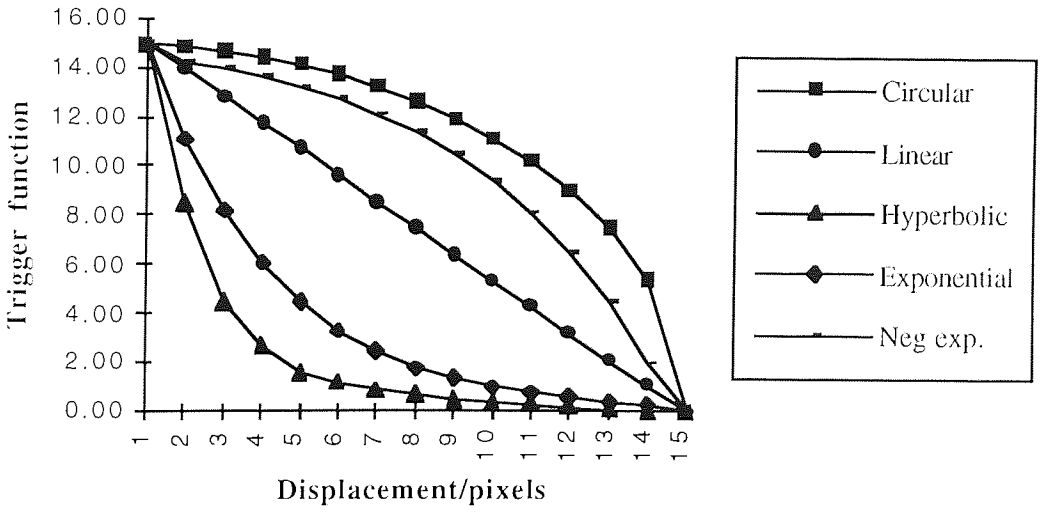


Fig. 5.12. Trigger functions for yardstick length of 15

Each function is subject to two constraints:

$TF[1] = Y$ This identifies edges (i.e. steps greater than the yardstick length)

$TF[Y] = 0$ This ensures a match is found in smooth areas where there has been no intensity change over a distance equal to the yardstick length.

The different trigger functions have all been applied to the FWC system to give compression performances shown in Fig. 5.13 and distortion performances shown in Fig 5.14. The SNR values in Fig. 5.14 are achieved using linear interpolation (see Section 5.9) during image decompression.

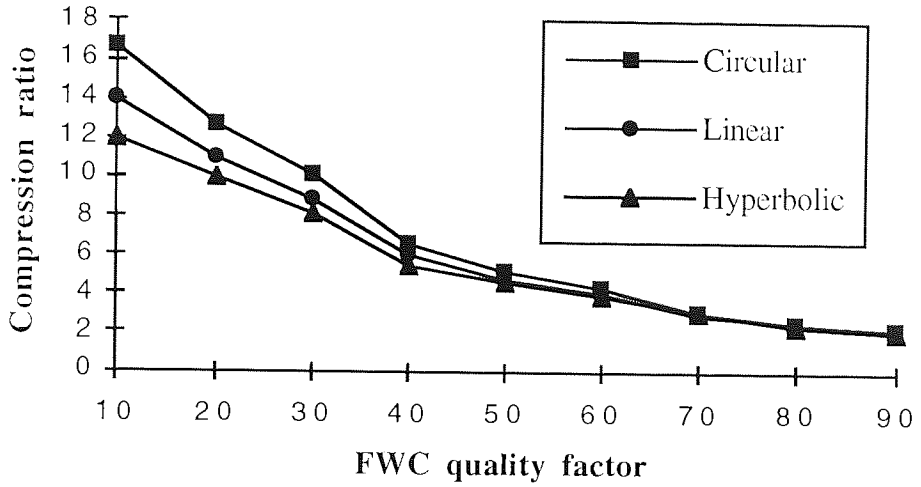


Fig. 5.13. FWC compression with different trigger functions

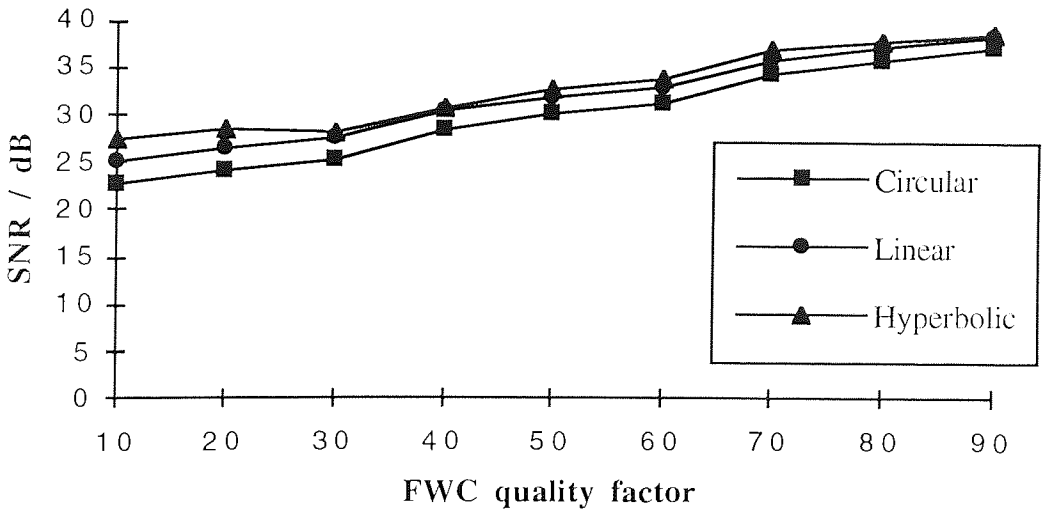


Fig. 5.14. FWC distortion with different trigger functions

From Figs. 5.13 and 5.14 it may be seen that the effect of the different trigger functions is to exaggerate the effect of the quality factor. This may be understood by considering the graphs of the trigger functions in Fig. 5.12. A trigger function such as the

hyperbolic trigger function maintains accuracy of the pixel intensities over a wide variation in steps lengths whereas a trigger function such as the circular function tends to give longer steps and thus increases the compression ratio.

The differences between the trigger functions are quite small so, in view of its better compression ratio, the circular trigger function is used unless otherwise indicated.

5.6.1 Multiple trigger functions

Goel and Kwatra [Goel 88] used a state machine approach to determine both the trigger function and the yardstick length. This is essentially an extension of the adaptive yardstick proposed by Yang, Wu and Mills [Yang 88]. A more general model has been tried here by leaving the yardstick length controlled only by the quality factor and using the extra parameter α required by the exponential trigger functions to form families of trigger functions among whose members the coder (and decoder) can swap depending on the local image properties. This has been modelled by setting upper and lower thresholds in the step sizes and moving to another trigger function table when the previous step size is above or below the upper and lower thresholds respectively. Using this approach, the parameter α is set to be proportional to the number of trigger functions in the table.

This has not been found to be effective because transitions between trigger tables take place infrequently. The most important local image property is whether or not an edge is to be processed. It will be seen in Section 5.7 that edges have to be handled completely differently and it is not appropriate simply to change to a slightly different trigger function.

5.7 Treatment of sharp edges

It is known that edges play a most important role in human perception of images [Marr 82]. In the FWC scheme edges are defined as image intensity transitions greater than the yardstick length. These will not necessarily correspond to edges as perceived by a human observer, particularly for small yardsticks, but it is still important to be able to code these edges accurately. The basic FWC scheme provides no method for handling such edges so additional processing is required. The options are:

1) Provide an additional method of coding when a sharp edge is encountered. For example, actual grey levels could be transmitted using a suitable quantiser. This method will be called *delta-transmit*. Within this basic scheme a variety of different quantisers have been tested.

2) A step of length zero could be sent. This method will be called *zero-step*. The sign bit then indicates to the decoder that a vertical (i.e. grey-level) step of maximum length (i.e. yardstick length multiplied by vertical expansion factor) should be taken in the positive or negative direction before the next step is decoded. This is particularly simple to implement but creates a considerable overhead in terms of the number of steps which need to be coded. For example, with a small yardstick a single, very abrupt edge might require several zero-steps.

5.7.1 Delta-transmit edge processing

Given the importance of edges in human perception of images it would seem sensible to encode edges with some accuracy. Since an edge can create a maximal intensity change this implies that edges should be encoded with the full resolution available from the source image. This presents a problem since non-edge step magnitudes are encoded with sufficient bits only to represent step sizes up to the yardstick length. The number of bits B_s to encode each step magnitude and the number of bits B_e to encode each edge magnitude are given by:

$$B_s = \mathrm{INT}(\log_2 Y),\ B_e = \mathrm{INT}(\log_2 G), \quad \text{.....(5.14)}$$

So that the decoder can distinguish them, the encoder must provide an extra data bit B_{type} to indicate whether the next sample is a step or an edge. A further sign bit B_{sign} is required for the steps and edges. Hence the total number of bits B_{total} to encode an image which has N_e edges and N_s non-edge steps is given by:

$$B_{total} = N_s\left(B_s + B_{sign} + B_{type}\right) + N_e\left(B_e + B_{sign} + B_{type}\right) = N_s\left(B_s + 2\right) + N_e\left(B_e + 2\right) \text{.....(5.15)}$$

5.7.2 Quantisation of edge steps

Before deciding which quantiser is appropriate for delta-transmit processing of edges, the probability distribution of the edge magnitudes, which is shown in Fig. 5.15, should be considered. This figure shows the mean distribution of edge magnitudes over the three primary test images for three different yardstick lengths.

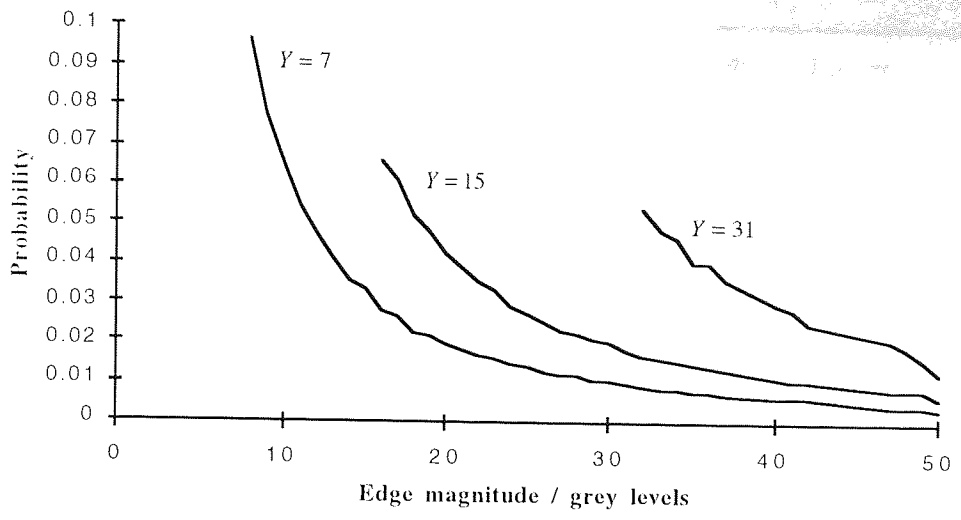


Fig. 5.15. Edge magnitude distributions for test images

It is clear from Fig. 5.15 that most edges are very small. There are just a few with large magnitudes (such as at the start of the scan of the image). A quantiser must be chosen which can code these edges accurately without wasting storage for the far greater numbers of small edge magnitudes. From Section 2.4 it may be determined that a logarithmic quantiser might be expected to be appropriate. For comparison, both linear and logarithmic quantisers have been tested here.

Although the presence or absence of an edge is a key factor in determining whether a viewer can perceive the continuity of shapes in an image, the exact magnitude of the intensity change at the edges is less significant. This presents the possibility of achieving a more efficient coding scheme by quantising edge magnitudes into the same number of bits as are used for the normal steps.

It is possible to show the effect of the edge step quantisation by performing a modified compression/restoration process where edge steps are coded with a fixed number of bits. Figs. 5.16 and 5.17 show the performance of the FWC system with linear and logarithmic quantisers for $Q_{fwc} = 50$. It may be recalled from Section 5.5.1 that, at this value of Q_{fwc}, $Y = 15$, so 4 bits would normally be allocated for coding non-edge steps. Since the source images are quantised to 8 bits per pixel, both quantisers have no quantisation errors at 8 bits per pixel.

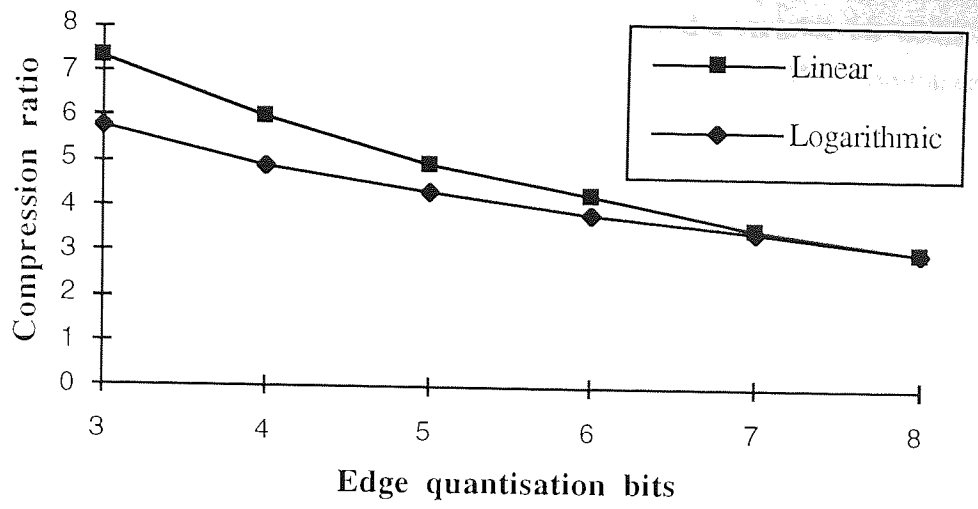


Fig. 5.16. FWC Compression ratio using different edge quantisers

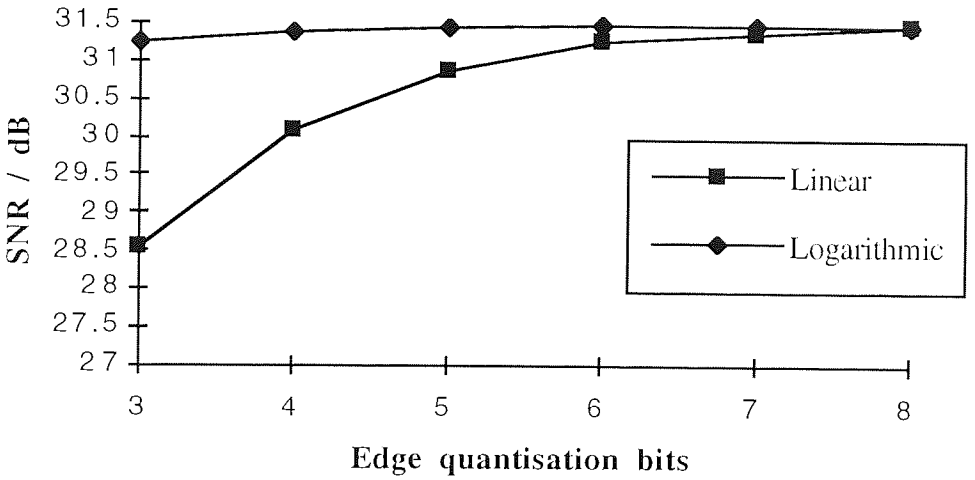


Fig. 5.17. FWC SNR using different edge quantisers

From Figs. 5.16 to Fig 5.17 it can be concluded that:

- The SNR is not significantly improved by using more than 5 bits for edge quantisation.
- For edges quantised with less than 5 bits the logarithmic quantiser gives a worthwhile improvement of about 2 dB in the SNR compared with the linear quantiser.
- The compression is inversely proportional to the number of bits used for quantisation as a result of the high proportion of edges with small yardstick lengths.

If a logarithmic quantiser is adopted, then it only remains to compare the overall FWC system performance using a fixed 5 bit edge quantiser and a variable bit-width edge quantiser (whose width is determined by the yardstick length). This comparison is shown in Figs. 5.18 and 5.19.

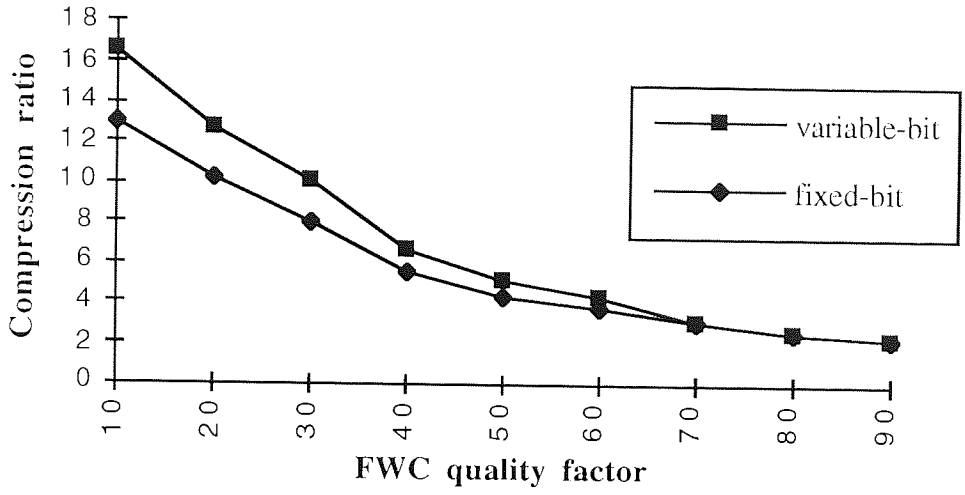


Fig. 5.18. FWC compression ratio with fixed and variable width logarithmic edge quantisation

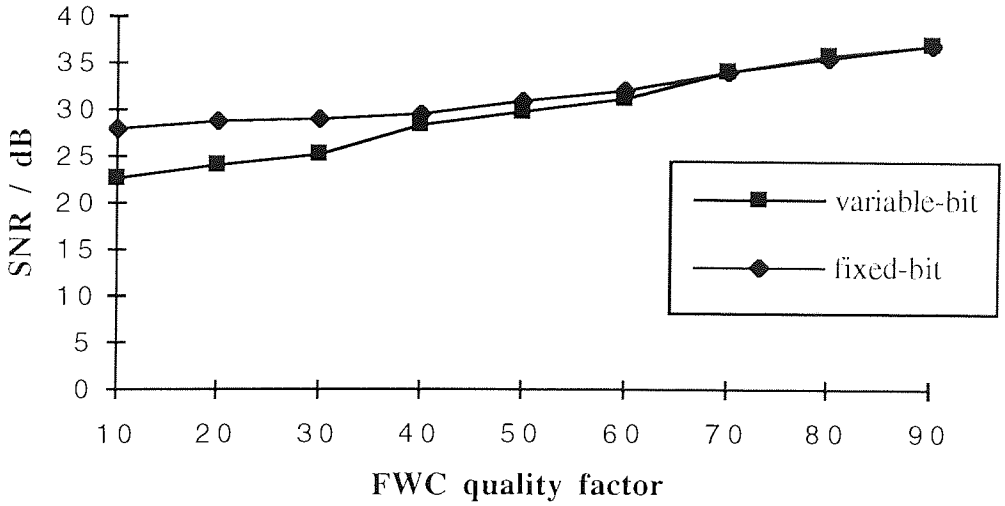


Fig. 5.19. FWC SNR with fixed and variable width logarithmic edge quantisation

It is concluded from Figs. 5.18 and 5.19 that over the useful range of values for Q_{fwc}, a variable bit-width edge quantiser gives only slightly better compression for a slightly worse distortion. Thus, for efficient representation of the coded data, the edges will be

coded in as many bits as are allocated for the non-edge steps (i.e. as many bits as are required to represent the yardstick length).

Variable numbers of edge quantisation bits make the coded data stream slightly more complicated but are feasible as long as the bit(s) denoting edge/not-edge (and sign) are sent *before* each step.

5.7.3 Zero-step edge processing

Processing the edges using a zero-step algorithm is simpler than using the delta-transmit algorithm since no edge quantiser is required. The performance depends on how often multiple zero steps are required for a single edge. If multiple zero steps are not required then the method becomes the same as delta-transmit but without the quantisation error at each edge. The distribution of edge magnitudes has already been shown in Fig. 5.15. The majority of edges are small so most zero steps will be single but as the yardstick is reduced the number of multiple zero-steps increases. The performance of the zero-step implementation of the FWC algorithm is shown in Fig. 5.20. This can be compared with the performance using delta-transmit with variable bit-width logarithmic edge quantisation in Figs. 5.18 and 5.19.

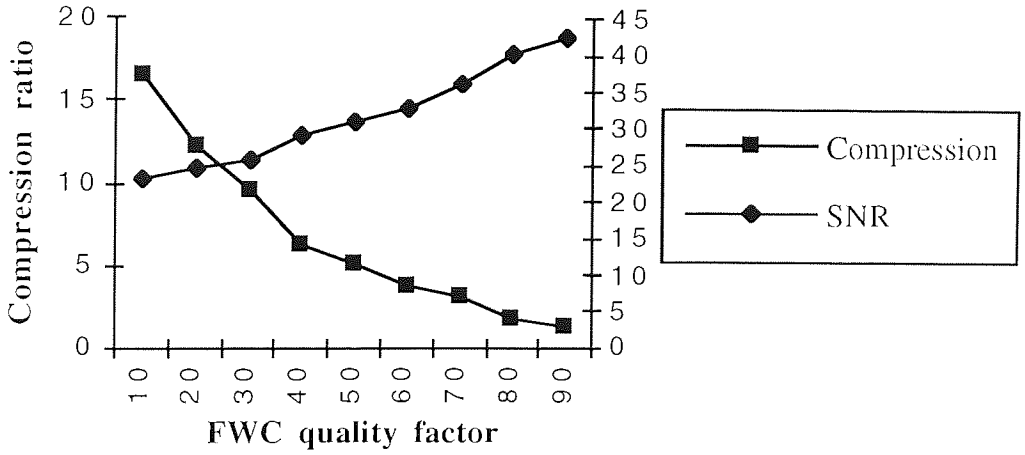


Fig. 5.20. Performance of FWC with zero-step edge treatment

The conclusion is that delta-transmit processing of edges is significantly better than zero-step processing. For lower quality factors (larger yardsticks) there is little difference but, at higher quality factors, the number of edges encountered was such that coding requirements for the extra steps rapidly exceed the requirements for the non-edge steps. For higher quality settings the zero-step distortion is slightly better but this does not make up for the loss in compression.

5.8 Stability of the coding scheme

With any coding scheme it is important to consider the stability due to errors in the communication channel and to inherent accumulated errors (rounding errors) in the scheme itself. It is important to note that in all predictive schemes the intention is to remove the mutual redundancy between successive samples and quantise only the new information. Since the prediction is based on the output rather than past input samples the predictor has to be in the feedback loop around the quantiser. Hence quantisation noise at a given step is fed back to the quantiser input at the next step. This stabilises the system by preventing accumulation of errors in the reconstructed signal.

In their basic FWC scheme Walach and Karnin assumed that each transmitted/received point would be exact. Preliminary experiments here indicated that this rapidly accumulates errors and is unworkable. It has been found that a predictor/corrector method works well. i.e. The grey level used to determine the next step is offset by the known error Δg in the previous step before applying the trigger function. This is illustrated in Fig. 5.21. The ends of the yardstick at the n'th step are denoted by P1(n) and P2(n).

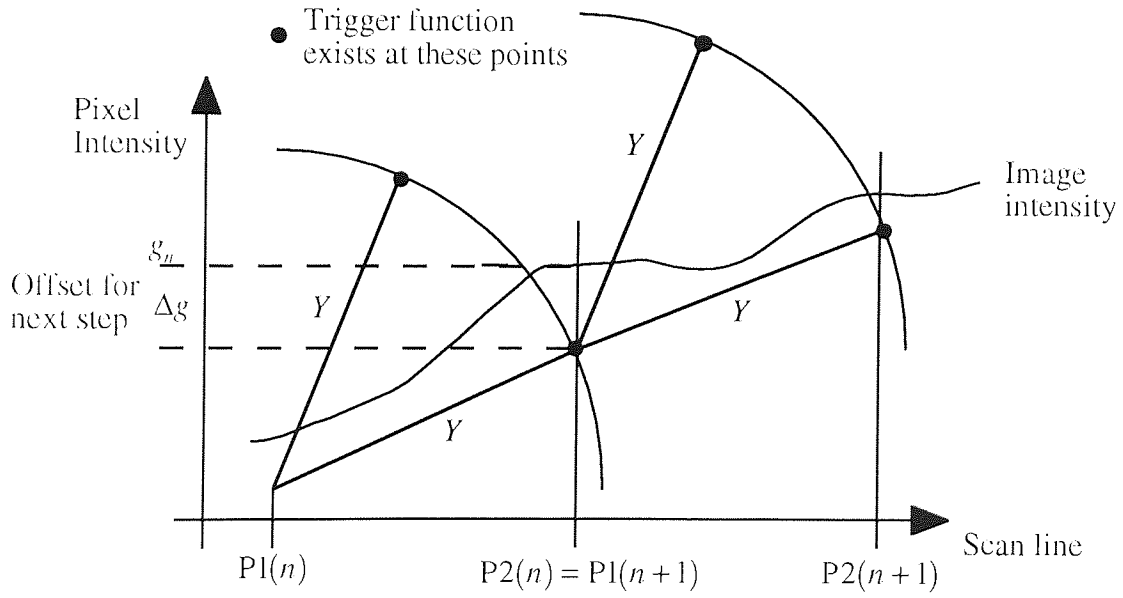


Fig. 5.21. Step offset to maintain stability of FWC

Although the trigger function at the n'th step meets the image intensity function at intensity g_n, the value used in the n+1'th step is $g_n - \Delta g$. The same principle applies for non-edge steps and edge steps, except that for edge steps the value of Δg is determined from the *quantised* value of the edge magnitude.

5.9 Interpolation of the restored image

When an image is reconstructed from the transmitted set of (t_i, $sign_i$) values, all that has been obtained is a set of key points as was shown in Fig. 5.7(b). The grey levels of the intermediate pixels must be obtained by some interpolation procedure. In previous work [Wala 86, Goel 88, Yang 88] on FWC only simple (one-dimensional) linear interpolation has been used. However, more sophisticated interpolation can be used. In particular, schemes can be adopted which make use of knowledge about how the reconstructed points were derived and of the fractal nature of the image data. The choices of interpolation scheme and encoding processes are related. In particular, the method used to process sharp edges in the source image places constraints on the reliability of the interpolation.

5.9.1 Interpolation principles

The principle of interpolation is shown (in one dimension) in Fig. 5.22. The samples $f(x_1)$, $f(x_2)$, $f(x_n)$ ... represent a discrete set of points on the signal $f(x)$. The problem is one of determining $f(x)$ at points other than $\{x_i\}$.

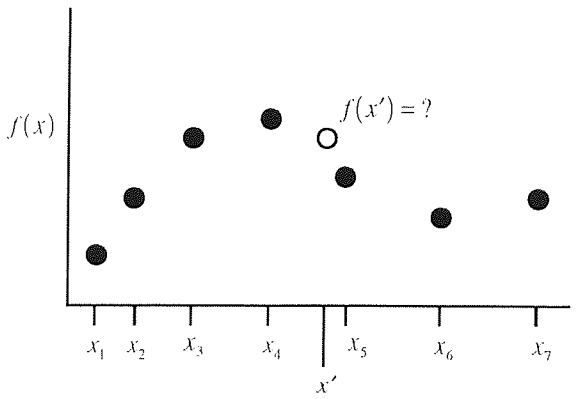


Fig. 5.22. Principle of interpolation

The simplest method, called nearest neighbour or zero-order interpolation is to set $f(x)$ to $f(x_i)$ for the x_i closest to x. Linear (or first-order) interpolation can be used between x_i and x_{i+1} where $x_i \le x \le x_{i+1}$. Higher order interpolation involves fitting a curve to $f(x_i)$ for the x_i near x and then evaluating $f(x_i)$ at x. In practice, rather than specifying the curve, it is conventional to specify a weighting function $h(s)$ which gives the weight to assign to the neighbouring pixels as a function of their distance s from x. Then, the reconstructed value at x is given by:

$$f(x) = \sum f(x_i) h(x_i - x) \qquad \text{.....(5.16)}$$

These interpolation methods can be extended to two dimensions to give bilinear interpolation and cubic convolution. Eqn. 5.16 now becomes:

$$f(x,y) = \sum_i \sum_j f(x_i, y_j) h(x_i - x, y_j - y) \qquad \text{.....(5.17)}$$

The interpolation functions and weighting functions are given in Table 5.3 [Nibl 86].

Method	Interpolation function	Weighting function
Nearest neighbour	$f(x) = f(x_i)$ for x_i closest to x	$h(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$
Linear	$f(x) = bi(f(x_i), f(x_{i+1}), x - x_i)$ where: $bi(y_1, y_2, d) = y_1(1-d) + y_2 d$	$h(x) = \begin{cases} 1-x & 0 \le x < 1 \\ 0 & x \ge 1 \\ h(-x) & x < 0 \end{cases}$
Cubic Convolution	$f(x) = cc\begin{pmatrix} f(x_{i-1}), f(x_i), f(x_{i+1}), \\ f(x_{i+2}), x - x_i \end{pmatrix}$ where: $cc(y_1, y_2, y_3, y_4, d) = y_2 + d\begin{pmatrix} (-y_1 + y_3) + \\ d(2y_1 - 2y_2 + y_3 - y_4) + \\ d(-y_1 + y_2 - y_3 + y_4) \end{pmatrix}$	$h(x) = \begin{cases} 1 - 2x^2 + x^3 & 0 \le x < 1 \\ 4 - 8x + 5x^2 - x^3 & 1 \le x < 2 \\ 0 & x \ge 2 \\ h(-x) & x < 0 \end{cases}$

Table 5.3. Interpolation functions

It was shown in Section 2.3 that optimal interpolation of an image may be achieved using a sinc weighting function. i.e.

$$h(x,y) = \text{sinc}(x)\text{sinc}(y) = \frac{\sin(\pi x)}{\pi x}\frac{\sin(\pi y)}{\pi y} \qquad \text{.....(5.18)}$$

This is extremely computationally intensive to evaluate since it requires a weighted sum over *all* the image samples, but, if just pixels in the neighbourhood are considered, then cubic convolution is a good approximation to the sinc function.

The effect of different interpolation schemes for uniformly sampled data is shown in Fig. 5.23. In each case the starting point was the 'Lena' test image sub-sampled to 128 by 128 pixels. The images were interpolated back to 512 by 512 pixels by (a) nearest neighbour, (b) bilinear and (c) cubic interpolation. The improvement in the image with the increasingly sophisticated interpolation methods is clear.

(a) (b) (c)

Fig. 5.23. Effect of interpolation on uniformly sampled image.

5.9.2 Interpolation of FWC-compressed images

There is a more fundamental difference in interpolation required for the FWC scheme; the samples are not uniformly spaced. For a simple raster scan, samples along a row may be spaced between 1 and the yardstick length, Y, pixels apart but there may be columns where there are no samples between lines. For example, Fig. 5.24 shows a grid representing a region of the image to be reconstructed.

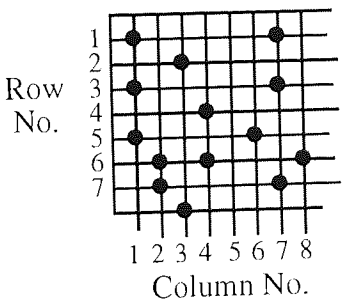


Fig. 5.24. Interpolation positions for FWC

The dots represent pixels whose intensities are known (i.e. they are at the end of yardstick steps or either side of edges). Interpolation can be performed on every row and on some columns such as column 2 and column 7 but no interpolation is possible on column 5. In addition there will be columns where the only known pixels are so far apart that interpolation is almost certainly unjustified since the pixels will lie in different regions of the image. Faced with this difficulty it is reasonable to perform just one-dimensional interpolation along the scan (i.e. along each row).

To show the differences caused by the non-uniform sampling grid produced by FWC compression, in Fig. 5.25, the image was interpolated with (a) nearest-neighbour, (b) one-dimensional linear and (c) one-dimensional cubic interpolation.

(a) (b) (c)

Fig. 5.25. Effect of interpolation on FWC compressed image.

It may be seen from Fig. 5.25 that interpolation from the points produced by FWC is particularly effective since the samples generated by the FWC process are optimally selected. There cannot be any large variations in the pixel intensities of the original image between the sampled points or they would have been matched by additional steps or edges during the coding process.

5.9.3 Interpolation taking into account the trigger function

None of the interpolation methods described so far make direct use of knowledge of the encoding scheme. e.g. If the trigger function is circular then constraints can be placed on the maximum difference between the received points and the interpolated points.

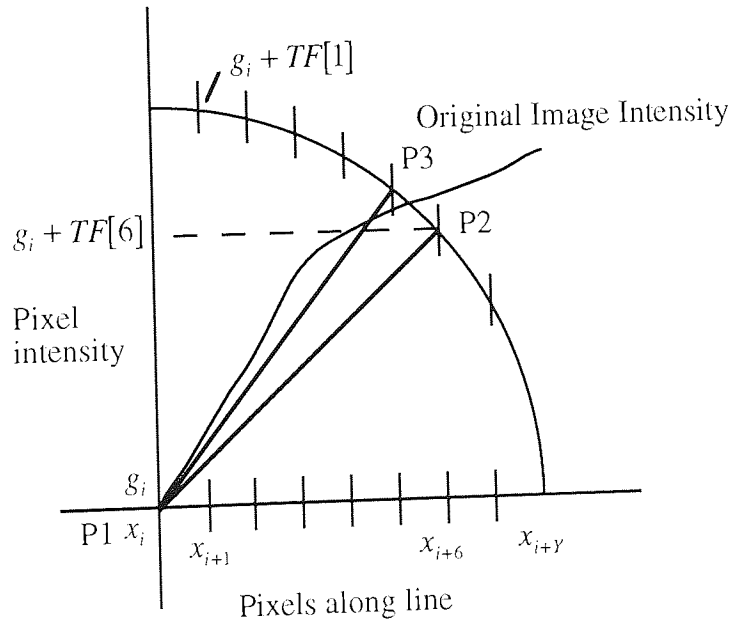


Fig. 5.26. Interpolation using circular trigger function

Fig. 5.26 shows a pair of restored pixels (based on Fig. 5.9 which was used to illustrate the trigger function) at x_i and x_{i+6}. The interpolation task is to determine the grey levels of the intermediate points. The restored intermediate points must lie in the triangular region bounded by (P1, P2, P3) or the waveform would have been matched elsewhere. A straight line approximation (linear interpolation) is not necessarily best and points on the median of the triangular region should be better. The appropriate modification is to perform linear interpolation but, instead of generating points along P1 - P2, points are generated along the line from P1 to the midpoint of P2-P3. i.e.

$$g_{i+n} = \frac{1}{2}\left\{f\left(g_{j-1},n\right)+f\left(g_j,n\right)\right\} \quad \text{.....(5.19)}$$

where these are the n points interpolated between the pixels $\left(x_i,g_i\right)$ and $\left(x_j,g_j\right)$ and $f\left(g_k,n\right)$ represents the value of the radius vector from $\left(x_i,g_i\right)$ to $\left(x_k,g_k\right)$ at x_{i+n}

But $f\left(g_k,n\right)=n\frac{g_k}{k}$, so

$$g_{i+n} = \frac{n}{2}\left\{\frac{g_{j-1}}{j-1}+\frac{g_j}{j}\right\} \quad \text{.....(5.20)}$$

Table 5.4 shows quantitative measures of the effect of the different interpolation methods on the FWC-compressed Lena image.

		SNR / dB		
Quality factor	**Compression ratio**	**Modified one-dimensional linear**	**Bilinear**	**Cubic convolution**
20	12.7	24.2	24.1	24.8
30	10.1	25.3	25.3	25.6
40	6.6	28.6	28.7	28.7
50	5.2	30.2	30.3	30.4
60	4.3	31.6	31.8	31.7

Table 5.4. Effect of different interpolation methods

Taking into account the additional computational complexity and the modest increases in SNR, cubic convolution is not considered appropriate. Both bilinear interpolation and the modified form of one-dimensional linear interpolation described in this section produce similar results. Since it is computationally more efficient, one-dimensional interpolation is preferred.

5.10 FWC using space-filling curve scans

It was determined in Chapter 4 that the potential for improvement in image compression by using a space-filling curve as a scan generator is greatest for one-dimensional coding methods such as those discussed in this chapter. FWC has been tested here along a Hilbert scan. The FWC parameters have been selected to be the same as were used in the raster scan. The overall performance of FWC along a Hilbert scan is shown in Fig. 5.27 which may be compared with Figs. 5.18 and 5.19 which gave the optimum performance of FWC with a raster scan.

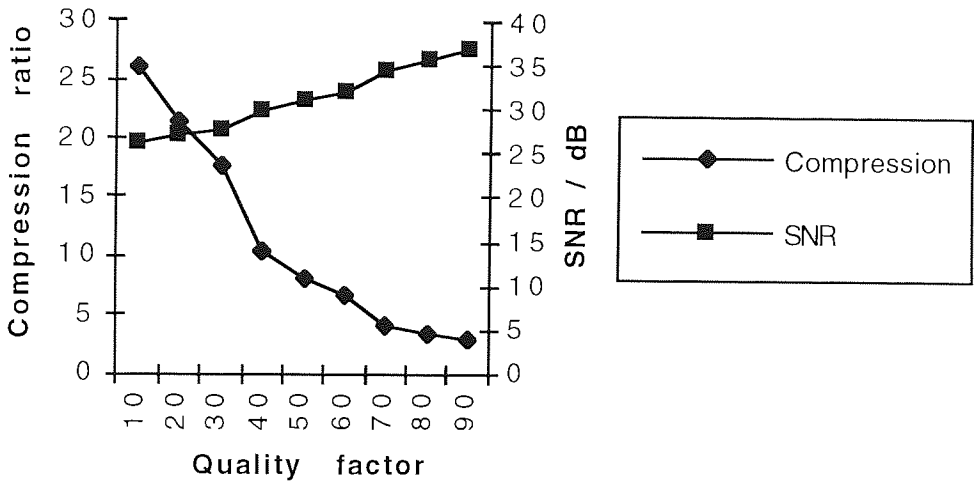


Fig. 5.27. Performance of FWC using Hilbert scanning

Fig. 5.28 shows a magnified region of the Lena test image compressed using FWC along a raster and a space-filling scan with the FWC quality factor set to 30. This low quality setting increases the errors to make the comparison clearer.

Raster scan **Hilbert scan**

Fig. 5.28. FWC compression of raster and Hilbert scanned images

In Fig. 5.28 the raster scanned image is compressed at a ratio of 10:1 with a SNR of 25 dB whereas the Hilbert scanned image is compressed, using the same FWC parameters, at a ratio of 18:1 with a SNR of 28 dB. At lower compression ratios, and when viewed on a screen, the subjectively better performance using the Hilbert-scanned images is even clearer. With a more realistic FWC quality factor of 50, using the three primary test images gives the performance shown in Table 5.5. These results show such improvement using the space-filling scan that the experiment has been repeated for the larger image test set with results also given in Table 5.5.

	Raster scan		Hilbert scan	
Image	**Compression ratio**	**SNR / dB**	**Compression ratio**	**SNR / dB**
Lena	5.18	30.21	8.15	31.09
Barbara	2.25	28.19	2.82	27.98
Boats	4.30	29.56	5.09	29.53
Large test set	6.35	30.57	7.98	30.68

Table 5.5. FWC compression using raster and Hilbert scans

Over this larger set FWC using a space-filling scan gives a 26% improvement in compression ratio with no detrimental effect on the distortion.

5.11 Summary of FWC

Fractal waveform coding has been demonstrated to be an effective method for image compression. FWC may be considered to be a lossy version of run-length encoding where variations in the image intensity up to a maximum run length are neglected according to a relationship which takes into account the fractal nature of the data.

The FWC scheme has numerous parameters which may be set. Extensive experimentation has determined that the two most significant variables, the yardstick length and the vertical expansion factor may be replaced by a single parameter which can be used to control the quality (i.e. compression ratio and distortion) of the process.

The mechanism for deciding when to encode a step along the image data depends on a trigger function. Several trigger function have been tested with the most effective being the circular trigger function. Analysis indicates that this is because this trigger function best matches linear interpolation during image restoration. The possibility of using multiple trigger functions has been tested and has been shown to provide a significant improvement, particularly if a separate method is provided for processing the abrupt intensity transitions caused by edges.

The method of handling sharp edges in images has been found to be important. It has been shown that the best method is to use a separate logarithmic quantiser for edges.

Several interpolation methods have been tested for the restoration phase and have been found to give small improvements in reconstructed image quality but it is debatable whether they justify the extra complexity. One-dimensional linear interpolation, with a minor modification to take into account knowledge of the encoding process has been shown to be simple and effective.

The primary reason for adopting waveform coding is often its simplicity of implementation. FWC is simple to implement and has a low computational complexity. The prototype system which is unoptimised and includes error detection and statistics collection runs very quickly. This advantage is lost if some of the enhancements described are employed. In general, adding complexity to FWC does not help. An overall comparison of all the image compression methods presented in this thesis is given in Chapter 8. The simple conclusion here is that, when used in conjunction with a space-filling scan, fractal waveform coding is the best of the low-complexity waveform coding methods.

Chapter 6. Vector Quantisation

6.1 Introduction

In Chapter 2 the process of quantisation was introduced as a fundamental component of any practical image compression system. Members of a set of scalar variables such as the pixel intensities of an image may be quantised separately using a scalar quantiser. Alternatively, the set of scalar values may be divided into blocks. Each block may then be viewed as a unit and the scalar values in the unit can be jointly quantised. This is termed vector quantisation (VQ) or block quantisation. In this chapter VQ is examined in detail. It is shown that the performance of practical VQ systems depends on the efficiency of vector lookup in the VQ codebook. Two new methods of greatly reducing the computational cost of vector lookup are presented. One of these approaches exploits space-filling curves which provide a mapping from multidimensional data into one dimension.

It should be noted that while compression schemes such as transform coding do operate on vectors, the quantisation of the transform coefficients is usually a scalar process. In this chapter only systems where the quantisation process operates on vectors are considered.

6.2 Justification for VQ

Vector quantisation is, in principle, superior to scalar quantisation. This follows from Shannon's rate-distortion theory [Berg 71], in particular the noiseless coding theorem. Shannon showed that vector quantisation is, in theory, always preferable to scalar quantisation, *even if the scalars have been pre-processed to make them uncorrelated.* Shannon's theory states that it is possible to code, without distortion, a source of entropy H bits per symbol (entropy was defined in Section 2.7.1.1) using $H + \varepsilon$ bits per symbol, where ε is an arbitrarily small quantity. Thus the maximum achievable compression C is defined as

$$C = \frac{\text{Average bit rate of original data}}{\text{Average bit rate of encoded data}} = \frac{B}{H+\varepsilon} \approx \frac{B}{H} \qquad \text{.....(6.1)}$$

For N by M image blocks of B bits per pixel there are $L = 2^{BNM}$ possible patterns which can occur. If the probabilities of each of these patterns were known then the entropy of N by M image blocks with B bits per pixel could be calculated. All L possible image blocks could be stored in a table and images could be encoded simply

with their addresses in this table, requiring approximately H bits per image block or H/NM bits per pixel.

Direct application of this approach is impractical to say the least. Even for small values of N and M, L becomes prohibitively large. For example, with B=8 and N=M=4, $L = 2^{8*4*4} = 2^{128} \approx 10^{35}$. However, if instead of storing all L possible patterns, a small subset containing only the most probable patterns is used, then practical VQ systems can be designed.

6.3 Practical VQ systems

6.3.1 Notation

In practical systems small blocks, typically 4 by 4 pixels or 8 by 8 pixels, are quantised. The set of patterns to be stored is generally termed the codebook and may be derived in a variety of ways. Codebook design is discussed later in Section 6.4.

It is convenient to introduce here some notation to describe the VQ systems addressed in this chapter.

Let $\mathbf{f} = [f_1, f_2, ..., f_N]$ represent an N-dimensional vector of N discrete scalar values f_i. $\mathbf{f}$ is mapped by a vector quantiser Q to another N-dimensional vector $\mathbf{r} = [r_1, r_2, ..., r_N]$. The vector $\mathbf{r}$ is chosen from a set of L possible reconstruction or quantisation levels. If $\hat{\mathbf{f}}$ represents $\mathbf{f}$ after quantisation then $\hat{\mathbf{f}} = Q(\mathbf{f}) = \mathbf{r}_i$ for $\mathbf{f} \in C_i$, where $\mathbf{r}_i$ for $1 \le i \le L$ denotes the L reconstruction levels and C_i is the i'th cell in the vector space containing $\mathbf{f}$. If $\mathbf{f}$ is in the cell C_i then $\mathbf{f}$ is mapped to $\mathbf{r}_i$.

Thus the codebook contains just one vector corresponding to each reconstruction cell. The reconstruction vectors are usually chosen to be the geometric centroids of the cells.

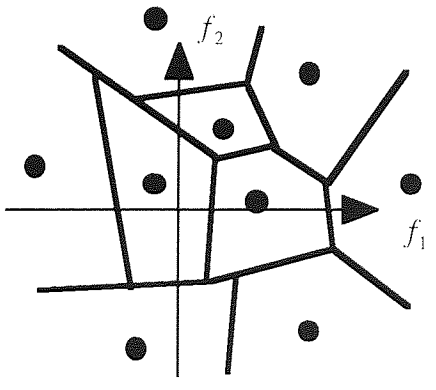


Fig. 6.1. VQ cells in 2 dimensions (9 reconstruction levels)

This is easy to visualise in the two dimensional case. Fig. 6.1 shows an example where N=2 and L=9. The dots are the reconstruction levels and the lines are the cell boundaries. The key difference between vector and scalar quantisation is that in VQ the cells can have arbitrary shapes as well as sizes. If Fig. 6.1 is compared with Fig. 2.4 in the discussion of scalar quantisation in Chapter 2 it may be seen that designing a vector quantiser is qualitatively different from designing a scalar quantiser. Higher-dimensional VQ systems are much harder to visualise but represent a straightforward generalisation of the two-dimensional case.

6.3.2 Implementation

Implementation of a VQ coding scheme is then conceptually simple as shown in Fig. 6.2. The encoder determines which of the codebook vectors is 'closest' to the source vector (using some suitable criterion) and transmits the address of the selected vector to the decoder. The decoder then allocates the same code vector to be output vector in the restored image.

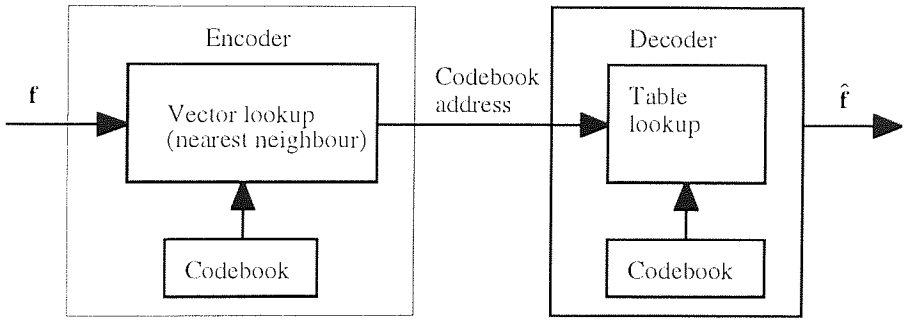


Fig. 6.2. Implementation of a VQ system

6.3.3 Compression ratio of VQ systems

The compression ratio achieved by a VQ system is simple to determine. If the VQ system codes N-dimensional vectors whose components can take 2^B values using a codebook with L entries, then the compression ratio is given by $C = \frac{NB}{\log_2 L}$. Compression ratios for reasonable values of N and L with a typical value of 8 for B (corresponding to the 256 grey-level images used in this thesis) are shown in Fig. 6.3. Assuming square blocks are used, $N = (\text{block size})^2$.

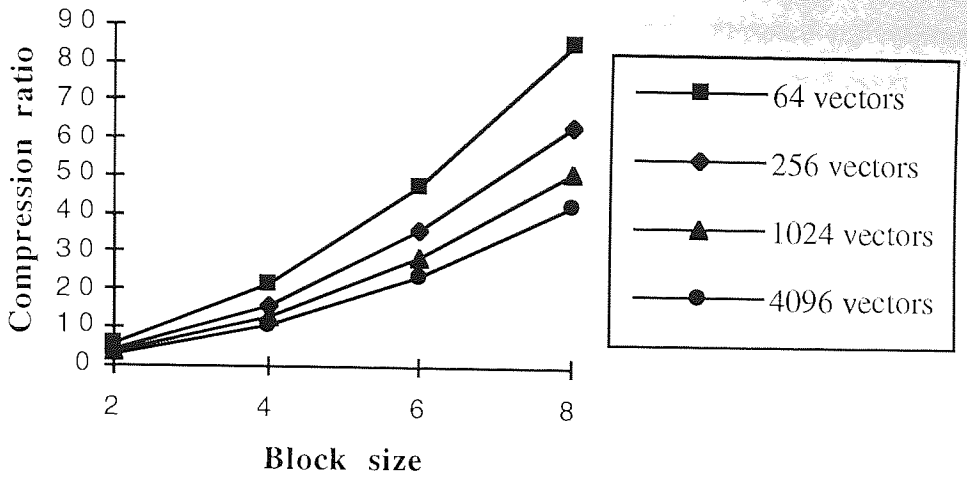


Fig. 6.3. Compression ratios for practical VQ systems

Clearly, the compression ratios achievable with VQ are very good compared with other systems (see Section 2.8). Furthermore, the values in Fig. 6.3 are the worst-case compression ratios. In practice, further improvement can be gained by entropy coding the transmitted values of the codebook addresses of the vectors. The difficulties with VQ lie in achieving this compression without introducing excessive distortion and without requiring excessive processing time.

6.3.4 Distortion caused by VQ systems

The distortion caused by the VQ process depends on the quality of the codebook and how thoroughly it is searched for the closest vector. These factors are considered in Sections 6.4 and 6.5 respectively.

In order to quantify the performance of a VQ system (or any other system) a distortion measure is required. This is the cost $d(\mathbf{x},\hat{\mathbf{x}})$ of reconstructing an input vector $\mathbf{x}$ as a reproduction vector $\hat{\mathbf{x}}$. The system performance can be described by the average distortion $E\big(d(\mathbf{x},\hat{\mathbf{x}})\big)$. In practice, the usual goal is to minimise the long term sample average given by:

$$\overline{D} = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} d(\mathbf{x},\hat{\mathbf{x}}) \qquad (6.2)$$

One of the simplest, and the most commonly used distortion measures is the squared error:

$$d(\mathbf{x},\hat{\mathbf{x}}) = \|\mathbf{x}-\hat{\mathbf{x}}\|^2 = \sum_{j=0}^{N-1} \left(x_j - \hat{x}_j\right)^2 \qquad (6.3)$$

It is neither necessary nor desirable to minimise an RMS measure since minimising the squared error gives the same results but without the added computational requirements of calculating square roots during the codebook design and coding processes.

6.3.5 Complexity of VQ systems

There are two main computationally complex tasks involved in a VQ system: (1) codebook design and (2) vector lookup. Various methods have been proposed to reduce the complexity of one or both of these tasks. The most critical task is vector lookup since this is an inherent component of most codebook design algorithms. Whether the complexity of designing the codebooks is considered to be a problem depends on whether a static or dynamically changing codebook is to be used. Static codebooks can be optimised over very large training sets and can contain many reproduction vectors since the one-off cost of transmitting the codebook to the receiver can be discounted. Dynamic codebooks must be small or the overhead in transmitting new codebook values will counteract the gains achieved by using a VQ system. In Sections 6.5 and 6.6 novel approaches to using static codebooks are described. These codebooks are large both in terms of the block size and the number of reproduction vectors in the codebook. Reference to Fig. 6.3 shows that significant compression gains may be achievable with large codebooks.

6.4 VQ codebook design

An optimal vector quantiser is one which employs a code book that yields the least average distortion among all possible codebooks. Although it is not, in general, possible to design a globally optimal code book, there is a well-known clustering algorithm, termed the LBG algorithm after its authors Linde, Buzo and Gray [Lind 80] which leads to locally optimal codebooks. Codebooks developed by this algorithm are optimal for small perturbations of the codevectors but are not optimal over all possible codebooks. The LBG algorithm is an extension of the Lloyd-Max optimal scalar quantiser [Max 60] discussed in Chapter 2.

6.4.1 Distortion measures and source statistics

The requirement for a distortion measure has already been noted in section 6.3.4. However, optimising a codebook using the mathematical expectation of the long term distortion cannot be calculated unless the probability distribution of the input source is known. For images this would require knowledge of the probabilities of all possible image blocks (or at least all those which might be encountered by the image coder). A theoretical approach to codebook design would be to use a data source whose statistics

best matched this distribution. This has been tried using data sources with, for example, Gaussian and Laplacian distributions and has proved successful for VQ coding of speech [Fisc 84, Fisc 89, Sabi 84] for which there are well-developed models for the probability distributions of the coded elements. Unfortunately there is no general agreement over a source distribution which is a good model for image data.

For VQ coding of images a more pragmatic approach is usually adopted. Usually the source is taken to be a long sequence of training data (for example, a large set of vectors from images representative of the type to be compressed) and an attempt is made to design codes which minimise the distortion for these sequences. It is then hoped that the same codes can be used on future data to yield approximately the same average distortion. It has been shown [Font 81] that for memoryless (i.e. non-predictive) VQ systems, using reasonable distortion measures such as the squared error, this is true if the source is asymptotically mean stationary.

The sequence of steps in a clustering algorithm such as the LBG algorithm is illustrated in Algorithm 6.1. The steps are intuitively fairly obvious but the computational cost of step 2 - the determination of the minimum distortion partition - should not be overlooked.

Algorithm 6.1. LBG codebook design

1. Assume an initial codebook and a training sequence.

   Let L be the number of quantisation levels. Set the distortion threshold $\varepsilon \geq 0$. Assume an initial L-level codebook $\hat{A}_0$ and a training sequence of n vectors $\left(x_j; j = 0,1,\ldots,n-1\right)$ with m, the number of iterations, set to 0.

2. Encode the training sequence into a sequence of channel symbols using the given decoder minimum distortion rule.

   Given a codebook $\hat{A}_m = \left(y_i; i = 0,1,\ldots,L\right)$, find the minimum distortion partition (Voronoi or Dirichlet partition) $P\left(\hat{A}_m\right) = \left(S_i; i = 1,\ldots,L\right)$ of the training sequence. Each training vector x_j belongs to the set S_i if $d\left(x_j, y_i\right) \leq d\left(x_j, y_l\right)$ for all l.

3. Calculate the average distortion:

$$D_m = D\left[\hat{A}_m, P\left(\hat{A}_m\right)\right] = (n-1)\sum_{j=0}^{n-1} \min_{y \in A_m} d\left(x_j, y\right)$$

4. If the average distortion is small enough, quit.

If $\frac{D_{m-1} - D_m}{D_m} \leq \varepsilon$ stop with A_m as the final codebook, otherwise continue.

5. Find the optimum codebook

$$\hat{x}\left(P\left(\hat{A}_m\right)\right) = \hat{x}\left(S_i; i = 1,\ldots,L\right) \text{for } P\left(\hat{A}_m\right) \text{ where } \hat{x}\left(S_i\right) = \frac{1}{\|S_i\|} \sum_{j: x_j \in S_i}^{m} x_j$$

6. Replace the old reproduction codeword of the decoder for each channel symbol v by the centroid of all training vectors which mapped into v and repeat.

Set $\hat{A}_{m+1} = \hat{x}\left(S_i\right)$, increment m to m+1 and go to Step 2.

In the literature there is some inconsistency in distinguishing methods of searching for vectors and the subsequent extraction of the minimum distortion partitions from the iteration technique itself. Gradient search methods can be applied to try and improve the convergence rate of the algorithm. In addition, optimisation techniques such as simulated annealing [Kirk 83] can be used to avoid local minima by introducing 'noise' irrespective of the distortion measure. Improvement of the convergence rate becomes increasingly important for the design of larger codebooks such as those considered in Sections 6.5 and 6.6.

In the simulated annealing approach to optimisation problems a small probability is assigned to accepting a solution at each iteration which is worse than the previous solution. This probability is large during the initial stages to ensure coverage of the whole solution space but decreases exponentially as the optimisation continues to ensure convergence

Simulated annealing cannot be used directly with geometric centroid partitioning since each iteration always gives a lower distortion partition. However, when used in conjunction with gradient search the distortion improvement is not monotonic so simulated annealing does give a much better chance of finding the true global minimum distortion. This increases the number of iterations and hence the processing time since LN gradients have to be evaluated for each iteration (the gradients are the rates of change in the total distortion with changes in each component of each codebook vector).

Extensive computation here (several weeks processing time on high-performance computer workstations) has shown that, while the combination of gradient search and simulated annealing does produce good results, codebooks of similar low distortion can

be found much more quickly with the conventional centroid partitioning by trying many different initial codebooks.

6.4.2 Choice of initial codebook.

A strategy is required to form the initial code book $\hat{A}_0$. Two basic approaches are used: starting with a simple codebook of the correct size; starting with a simple, small codebook and recursively generating larger ones.

6.4.2.1 Simple initial codebooks

In principle, the first L vectors from the training sequence could be used but there is no guarantee that these will be sufficiently widely spaced or in any way representative of the whole training set. This can be modified by selecting widely spaced vectors from the training set. This is termed random code generation. Better methods of creating simple initial codebooks can be found by using uniform scalar quantisers on each dimension of the vectors and forming the codebooks as Cartesian products of these uniformly spaced scalars.

Product codes of VQ codebooks are defined [Gray 84] in a similar way to conventional vector products. Given a collection of codebooks C_i, $i = 0,1,...,m-1$, each consisting of L_i vectors of dimension k_i, then the product codebook C is defined as the collection of all $L = \prod_i L_i$ possible concatenations of m words drawn successively from the m codebooks C_i. i.e.

$$C = \underset{i=0}{\overset{m-1}{\times}} C_i = \{\text{all vectors } (\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_{m-1}); \hat{\mathbf{x}}_i \text{in } C_i; i = 0,1,...,m-1\} \quad \text{.....(6.4)}$$

For example, Abut *et al* [Abut 82] suggested starting with a scalar quantiser codebook C_0 and then forming successive codebooks with the iteration

$C^2 = C_0 \times C_0$, $C^k = C^{k-1} \times C_0$, when $k > 2$. Each C^{k-1} represents a good $k-1$ dimensional VQ codebook.

6.4.2.2 Splitting algorithms

Instead of forming higher dimensional codebooks from smaller dimensional codebooks it is possible to start with a single vector of the required number of dimensions and repeatedly split this into pairs of vectors until the desired L level initial codebook is reached. This method was used by Linde *et al* [Lind 80] in their original paper on the LBG algorithm. The initial vector is typically the centroid of the entire training

sequence. The splitting can be achieved by adding and subtracting a fixed perturbation vector.

6.4.3 Experimental results for codebook design

For testing the novel VQ lookup algorithms described in this chapter, codebooks have been designed for various vector sizes with different numbers of codebook vectors.

6.4.3.1 Initial codebooks

The initial codebooks were formed in different ways depending on the number of vectors in the codebook.

For $L \leq 2^B$ the initial codebooks were formed using the splitting technique described in section 6.4.2.2.

For $2^B \leq L \leq 2^{B+1}$ the additional initial codebooks vectors were formed from training vectors which were randomly extracted from the large set of test images (Appendix A).

For $2^{B+1} \leq L$ the additional initial codebook vectors were randomly chosen. It was found that, particularly for large codebooks, this gave better coverage of the whole vector space.

In each case the number of training vectors was much greater than the codebook size (see Table 6.1).

Codebook Size	No. of training vectors
64	5000
256	10000
1024	20000
4096	40000

Table 6.1. Number of training vectors used for each codebook

The ratio of training vectors to codebook size is greater for small codebooks, firstly because a large ratio is impractical for large codebooks and secondly because the smaller codebooks are much more dependent on the training vector selection.

6.4.3.2 Codebook quality

The distortions yielded by the codebooks over the whole of the large set of training images are shown in Table 6.2.

Block size	Vector Size	Mean RMS distortion / dB			
		L=64	L=256	L=1024	L=4096
2	4	30	33	36	38
4	16	26	27	29	30
6	36	24	25	26	27
8	64	23	24	25	26

Table 6.2. Codebook quality for image training set

The codebooks were then tested on the smaller test set of three images (which were not members of the training set). Fig. 6.4 shows the reconstructed images using a block size of 4 with (a) a very small codebook of only 64 entries and (b) with a large codebook of 4096 entries.

(a) 64 vector codebook **(b) 4096 vector codebook**

Fig. 6.4. VQ compressed image with different codebook sizes

The problems with small codebooks are clear. The image appears to be coarsely quantised since there are too few codebook vectors to represent even the mean grey levels of the blocks adequately. Large codebooks give subjectively excellent results but the computation time is very high as is discussed in Section 6.5.

The distortion results using an exhaustive full search of the codebook are shown in Fig. 6.5. These results are in excellent agreement with the distortion figures for the training sets indicating that the training set is sufficiently diverse to generate codebook vectors for a wide range of images.

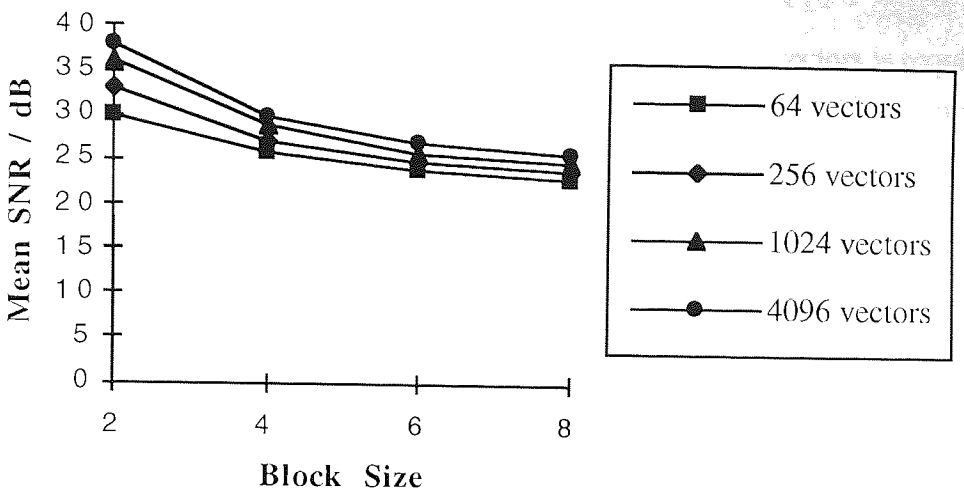


Fig. 6.5. Distortion performance of VQ system on test images

6.5 Vector lookup

In the design of VQ codebooks and in the implementation of VQ coders there exists a vector lookup problem. i.e. an algorithm is required to choose, from a codebook of size L, the 'closest' vector of size N to each source vector using some minimum distance (usually Euclidean) criterion. For an arbitrary codebook, the closest codevector may always be found by an exhaustive full search (EFS) of the codebook (See Algorithm 6.2) but this has an exponential computational complexity.

Algorithm 6.2. EFS vector lookup

To find the nearest codevector $\mathbf{y_i}$ to source vector $\mathbf{x}$

1. Calculate $d_j = d(\mathbf{x}, \mathbf{y}_j)$ for $j = 1,\ldots,L$

2. Find index i such that $d_i < d_j$ for $j = 1,\ldots,L,\ j \neq i$

3. $\mathbf{y_i}$ is the nearest codevector to $\mathbf{x}$

There are three strategies which may be used to try overcome the exponential complexity of EFS. Codebook reduction, codebook structuring and fast codebook-independent algorithms. Of these, most effort has conventionally been given to codebook structuring and codebook reduction which are briefly discussed in Sections 6.5.1 to 6.5.2. In this thesis, codebook-independent algorithms are preferred since they have more general applicability. Any conclusions drawn for these algorithms should also be valid for the smaller codebooks used in the codebook-dependent approaches.

6.5.1 Codebook reduction methods

If raw vectors from the input image are quantised a wide range of vectors is required in the codebook since raw vectors will exhibit the greatest diversity. Murakami *et al* [Mura 82] reduced the codebook size by normalising each vector in the codebook and the input sequence to have zero mean and unit standard deviation. This requires that the mean and variance of each block have to be (scalar) quantised and transmitted as well as the codebook address. This is a significant overhead for small codebooks but gives a substantial saving for large codebooks.

Baker and Gray [Bake 82] extended this approach with mean/shape VQ (MSVQ) by again transmitting the scalar quantised mean and vector quantising only the difference between the input vectors and the sample mean. Baker [Bake 84] then introduced mean/residual VQ (MRVQ) by subtracting the scalar quantised sample mean from the vector *before* vector quantising the residual. This reduces the blocking distortion caused by coarse quantisation of the sample mean in MSVQ.

While MRVQ performs well in quantising uniform vectors near the origin of the hyperspace containing the codebook vectors it still gives significant errors encoding edge blocks which lie far from the origin. Unfortunately these large amplitude vectors lead to the most visually sensitive distortion. Sabin and Gray [Sabi 84] introduced a modification called gain/shape VQ (GSVQ) where the codevectors, after mean removal are normalised so that they lie on the surface of a hypersphere with an arbitrary radius. Thus each vector is transformed into a 'gain' part representing the intensity of the block and a 'shape' part representing the spatial distribution of the vector elements. It may be recognised that, by analogy with waveform coding, VQ corresponds to sampling the waveform, MRVQ to removing the DC component and GSVQ to normalising the signal [Lee 86].

6.5.2 Structured codebook methods

The codebook may be structured during the design phase so that only smaller sub-codebooks have to be searched during the encoding phase. The most significant scheme adopting this approach is classified vector quantisation (CVQ) [Rama 86] but tree-search VQ (TSVQ) [Buzo 80] has proved popular as well as various hybrid schemes such as multi-step VQ [Juan 82] and lattice VQ [Nasr 88].

6.5.2.1 Classified VQ (CVQ)

In CVQ vectors are grouped into classes having distinctive perceptual features such as edges or uniform shading. Separate small codebooks are designed for each vector

class. When a source vector has to be matched it is necessary only to search the appropriate sub-codebook. This approach is effective for small vectors (up to 6 by 6 blocks) but has practical problems with larger vectors since the number of possible classes which can be defined grows rapidly.

Ramamurthi and Gersho [Rama 86] used four different classes:

- Shade blocks (those with no significant gradient)
- Midrange blocks (those with a moderate gradient but no edge)
- Edge blocks (with four orientations, horizontal, vertical and two diagonals)
- Mixed blocks (no definite single edge but significant gradient)

The distinction between midrange blocks and edge blocks is made on the basis of intensity changes less than or greater than the Weber fraction (see Section 2.4.3) respectively. Further structuring of the codebook may be made in terms of the polarity of the edges (i.e. intensity changes from high to low or vice versa). A block is classified as mixed if it contains positive and negative edge transitions.

Ramamurthi and Gersho's block classification provides a useful tool for comparing the training set, the test images and the codebooks described in Section 6.4. Fig. 6.6 shows the block classification for (a) the test set and the three primary test images, (b) Lena, (c) Barbara and (d) Boats. Fig. 6.7 shows the block classification for sixteen different codebooks. The codebooks are denoted by c*n_m,* where *n* is the block size and *m* is the number of vectors in the codebook.

From Figs. 6.6 and 6.7 it is apparent that, for a given block size and codebook size, the vector classification statistics are similar for the training set and the codebooks. This is not an obvious conclusion. The codebooks are designed to minimise the average distortion over all vectors. Only if the weighted contribution to the total error by each class of vectors is the same does this similarity in the vector classification statistics arise. This confirms the likely performance of the codebooks over a wide range of test images. However, it is clear that individual images may have quite different vector classification distributions. The Lena image, for example, has a particularly large fraction of shade blocks which is in accordance with its known ease of compression by most other schemes.

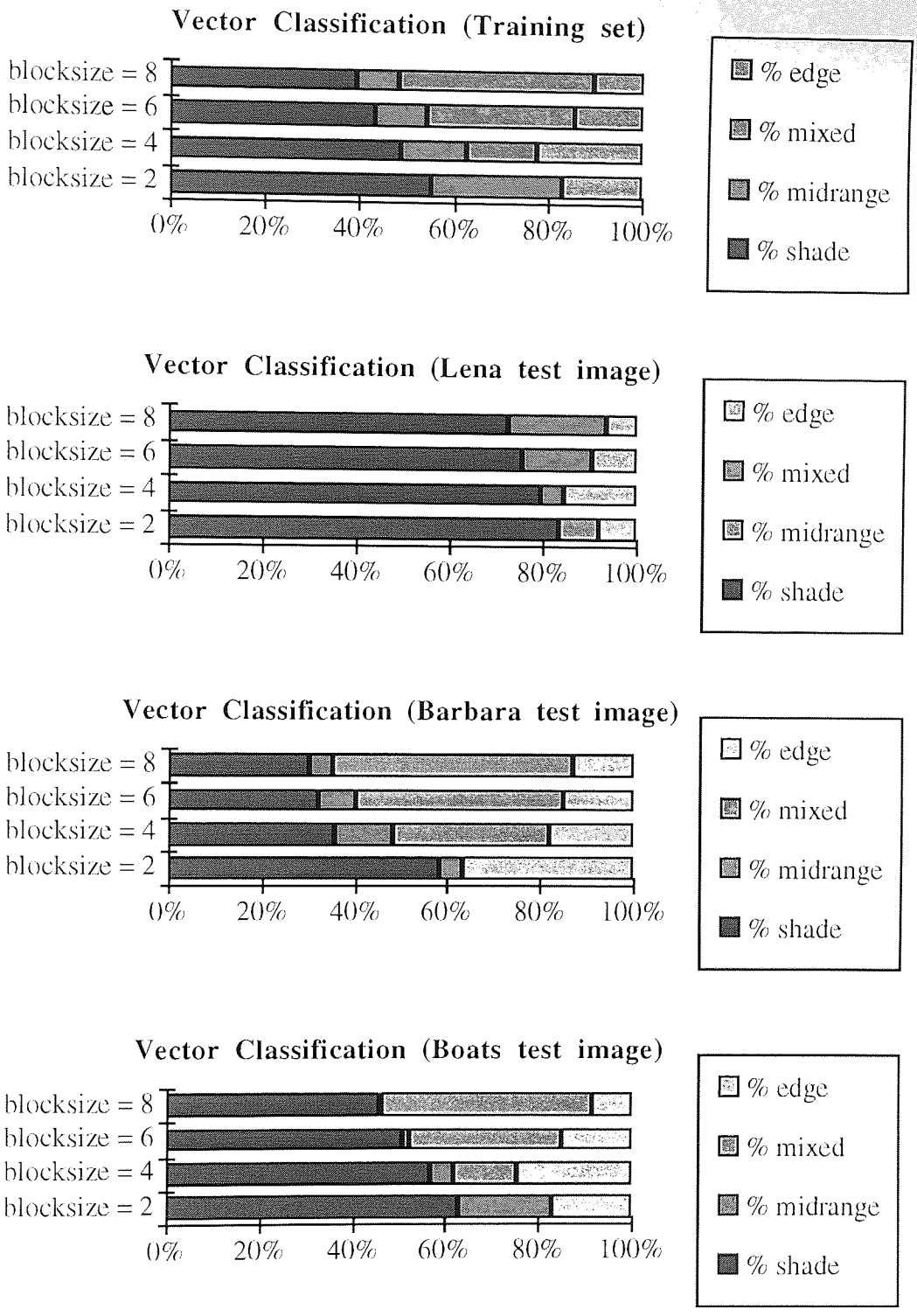


Fig. 6.6. Vector classification for the training set and test images (there are no mixed blocks of block size 2)

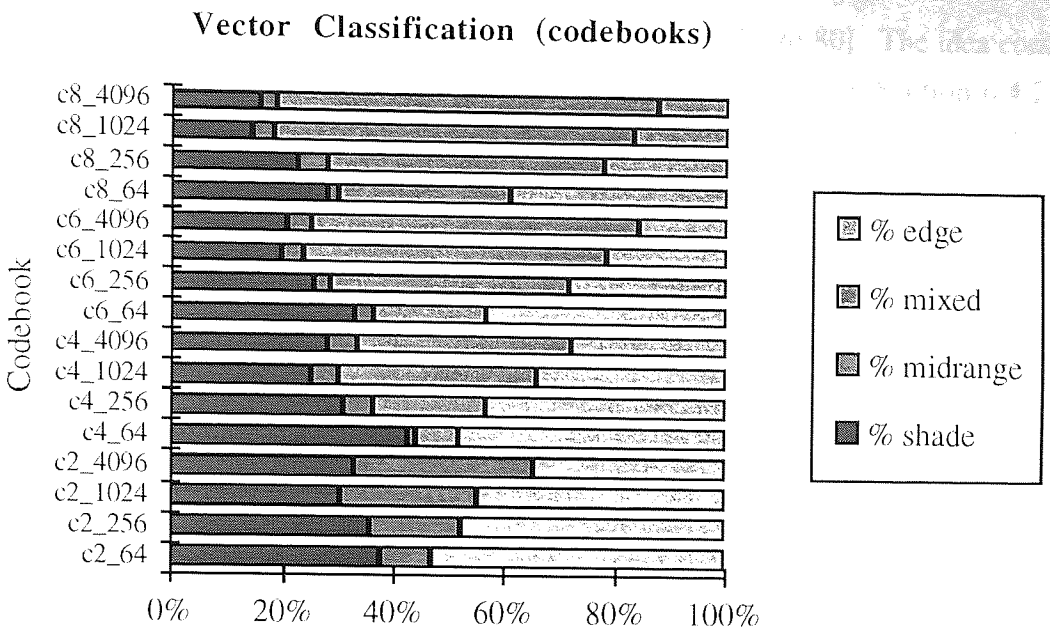


Fig. 6.7. Vector classification for VQ codebooks (there are no mixed blocks of block size 2)

There are some further observations about the block classifications in Figs. 6.6 and 6.7:

- The proportion of edge blocks in the codebooks is small but increases proportionally more for larger codebooks. This is expected since the first vectors to be allocated in a codebook will inevitably be the shade blocks which will do most to lower the total SNR, since they ensure the block means are correct. Edge blocks can only be 'afforded' once the shade blocks have been filled.

- There are, of course, no mixed blocks for a block size of 2, since they cannot contain two edges.

- The fraction of mixed blocks increases significantly for large block sizes such as 8. This shows the limitations of a crude classification scheme for the very large number of possible 8 by 8 blocks.

Although vector classification is not pursued further here for VQ, it is used again in Chapter 7 where the realisation that the edge classes, in particular, can be considered as transformations of each other leads naturally to affine transformation schemes such as Iterated Function System (IFS) coding.

6.5.2.2 Tree-search VQ (TSVQ)

Tree-search VQ systems were first proposed by Buzo *et al* [Buzo 80]. The idea comes from the splitting technique for initial codebook design discussed in Section 6.4.2.2. The sub-codebooks formed at each stage in the LBG algorithm can be stored. When coding a vector it is necessary only to look up the vector in a sequence of small codebooks instead of a single large codebook. The encoder is no longer optimal for the coder and the storage requirements for the tree of codebooks is greater than for the single codebook. However, the lookup complexity is now only linearly instead of exponentially dependent on the codebook size.

6.5.3 Codebook-independent algorithms

The search for vector lookup algorithms which avoid the exponential complexity of EFS has led to codebook-independent algorithms.

Some sophisticated approaches based on geometrical considerations have been proposed such as the Binary Hyperplane Test (BHT) algorithm of Chen and Gersho [Chen 86]. They made use of the fact that to identify the nearest code vector to **f** it is sufficient to identify the Voronoi region which contains **f**. By creating a binary search tree of hyperplanes together with a simple test to identify whether each Voronoi region is on the same side of the hyperplane as the test vector it is possible to quickly eliminate the code vectors of all regions which are not intersected by the hyperplanes. This leaves only a small subset (often only one) code vector to be tested by EFS. Experimental results indicate that this requires only 50% of the computation time of EFS for low bit rates (1-2 bpp) and as little as 25% of the computation time at higher bit rates (5-6 bpp).

However, more dramatic performance improvements have been achieved by simpler approaches such as the partial distance search and the Euclidean magnitude search proposed here.

6.5.3.1 Partial distance search

The EFS Algorithm 6.2 can be improved directly by the partial distance search (PDS) algorithm of Bei and Gray [Bei 85] shown as Algorithm 6.3 which allows codebook entries to be rejected as soon as the partially evaluated distance exceeds the currently-held minimum distance.

Algorithm 6.3. EFS with PDS truncation

To find the nearest codevector $\mathbf{y}_i$ to a source vector $\mathbf{x}$

1. Set minimum distortion to be arbitrarily large.

$$\Delta_{min} = \infty$$

2. Test each code vector

FOR each codevector $\mathbf{y}_i$, $i = 1,2,...,L$

2. Set partial distortion to be zero and vector component index to be zero.

$$\Delta = 0;\ k = 0$$

3. Sum contributions from each vector component to form partial distortion until partial distortion exceeds minimum distortion or all components have been added.

WHILE $\Delta \leq \Delta_{min}$ AND $k < N$

$$\Delta = \Delta + \left(y_{ik} - x_k\right)^2;\ k = k + 1$$

ENDWHILE

4. If partial distortion is less than previous minimum distortion, make this the minimum distortion.

IF $\Delta \leq \Delta_{min}$ THEN

$$\Delta_{min} = \Delta;\ c = i$$

ENDIF

ENDFOR

5. $\mathbf{y}_c$ is the closest vector to $\mathbf{x}$

It is apparent that the PDS truncation of the vector lookup is dependent on the ordering of the vectors in the codebook. This was first exploited by Paliwal and Ramasubramanain [Pali 89] who arranged the codebook vectors in descending order of their cluster sizes to eliminate as many vectors as possible at each stage. Other methods of sorting the codebook have been successful. For example, Hsieh *et al* [Hsie 91] improved vector lookup in the codebook generation phase only by sorting the codebook according to the arithmetic mean of the vectors. They tested only codebook vectors whose means were within a fixed threshold of the test vector and achieved a speed up of approximately 8 compared with EFS.

6.5.3.2 Euclidean magnitude search truncation

It has been found [Wilt 92] that sorting the codebook according to the Euclidean magnitude of the vectors, using a similar thesholding approach to vector lookup in both

the codebook generation and encoding phases can achieve a speed up of approximately 20 compared with EFS.

Experimental results indicate that, in many instances, the codevector with closest absolute magnitude to the source vector is a good candidate for the closest vector. It is thought that this is because typical codebooks are very small compared to the number of possible source vectors so the distribution of codevectors in any codebook and hence the distribution of codebook vector magnitudes is sparse. However, the code vector with closest magnitude is not usually precisely the closest vector, so a more pragmatic method has been derived. A small fraction, w, of the codebook is chosen from the codevectors whose magnitudes are close to that of the source vector. This subset which is termed the search 'window' is then searched for the closest vector instead of searching the whole codebook. This gives the new Algorithm 6.4:

Algorithm 6.4: Euclidean magnitude truncated search

To find the nearest codevector $\mathbf{y_i}$ to a source vector $\mathbf{x}$

During codebook design, Evaluate the (squared) magnitudes of the codebook vectors: $M_i = \sum_{j=1}^{N} \left(y_{i_j}\right)^2$ for $i = 1,\ldots,L$ and sort the codebook in ascending order of M

1. Calculate squared magnitude of source vector:

$$M_x = \sum_{j=1}^{N} \left(x_j\right)^2$$

2. Find index i of vector with closest magnitude:

$$\left|M_i - M_x\right| < \left|M_j - M_x\right| \text{ for } j = 1,\ldots,L,\ j \neq i$$

3. Calculate distortions of all code vectors within window centred on index i.

$$d_j = d\left(\mathbf{x}, \mathbf{y}_j\right) \text{ for } j = i - {}^{Lw}\!/_{2},\ldots,i,\ldots,i + {}^{Lw}\!/_{2}$$

4. Identify closest code vector within window centred on index i. i.e.

Find index c such that $d_c < d_j$ for $j = i - {}^{Lw}\!/_{2},\ldots,i,\ldots,i + {}^{Lw}\!/_{2}$

5. $\mathbf{y_c}$ is (probably) the closest vector to $\mathbf{x}$

The speedup achieved by this approach is thus $1/w$. Simulation has been used to verify this and to determine a suitable value for w. Initial simulations indicated that w is indeed the critical factor. A window size of 4 on a small codebook with 64 vectors

gives the same probability of matching the closest vector as a window size of 256 on a large codebook of 4096 vectors. This greatly simplifies illustration of the algorithm's effectiveness since w can be made constant. The simulated VQ system finds the closest vector using both EFS and Euclidean magnitude truncation with varying values of w set to 0.0625 which gives good results. The results presented in Fig. 6.8 are the mean values obtained for the three primary test images using the codebooks described in Section 6.4.

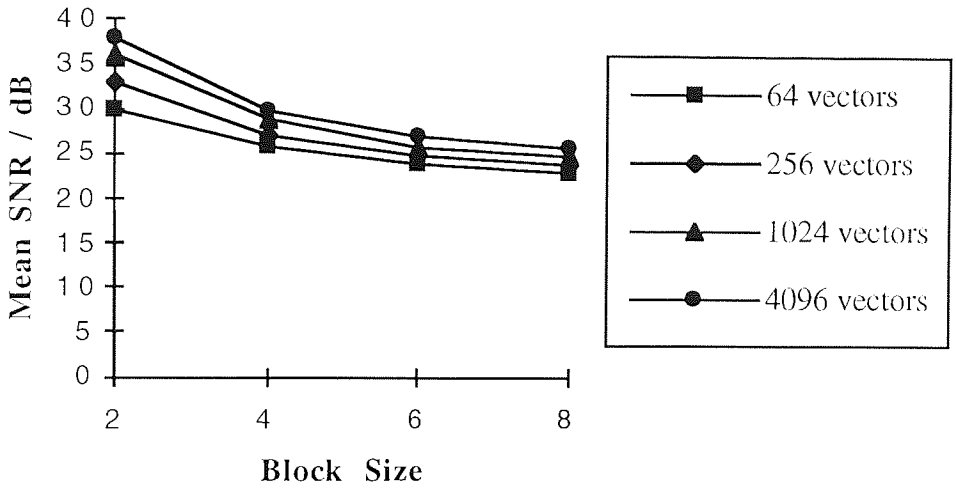


Fig. 6.8. Distortion performance of VQ system with Euclidean search (w=0.0625)

Comparison with Fig. 6.5 which gave the VQ distortion performance using exhaustive search for vector lookup shows that the distortion is almost unchanged even though only 1/16 of the codebook has been searched.

The effect on the total distortion due to matches of the true 'closest' vector is simply the quantisation distortion due to the quality and size of the codebook. The contribution to the total distortion due to the mismatches depends on how 'badly' they missed. This is the potential weakness in the method because the misses could, in principle, be far from the true closest vector. However, results indicate that this is rarely the case. To demonstrate this, define:

SNR_{EFS} = SNR with optimum reconstruction vectors

SNR_{MAG} = SNR with window search based on vector magnitudes

Extra distortion = SNR_{EFS} - SNR_{MAG}

The extra distortion can then be evaluated for each test image and each codebook. This is shown in Fig. 6.9.

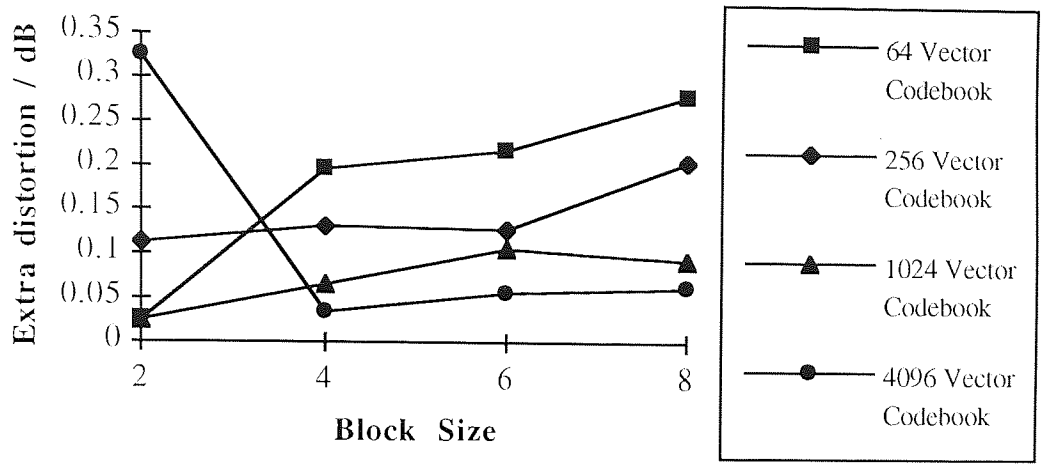


Fig. 6.9. Extra distortion introduced by Euclidean magnitude search

Neglecting the unrepresentative results for vector sizes of 4 (2 by 2 vectors) since vectors so small give little practical benefit (compression ratio only 4:1) then it can be seen that the extra distortion is small and almost independent of the vector size. Although the extra distortion is inversely proportional to the codebook size the dependence is so small that for realistic codebook sizes the extra distortion is approximately constant.

This finding is both surprising and significant since it means that one can truncate the search of a VQ codebook considerably and yet introduce only a small and predictable extra distortion to the reconstructed image. This means that much larger codebook sizes can be considered than are usually feasible. This, in turn, introduces the possibility of quantising larger vectors which, without large codebooks, are too coarse to be useful.

6.5.3.2 Vector magnitude distributions

Further improvements in the efficiency of VQ lookup may be made by taking into account the probability distribution function (p.d.f.) of the vector magnitudes.

Consider first the simple case of an N-dimensional vector $\mathbf{r}$ whose components are uniformly (rectangularly) distributed in the range $[0,T]$ Then, from first principles, the p.d.f $\phi(r)$ is given by:

$$\phi(r) = \text{constant} = \frac{1}{T} \qquad (6.5)$$

Then the mean μ and the variance σ^2 are given by Eqns. 6.6 and 6.7.

$$\mu = \int_0^T r\phi(r)dr = \left[\frac{r^2}{2T}\right]_0^T = \frac{T^2}{2T} = \frac{T}{2} \quad \ldots\ldots (6.6)$$

$$\sigma^2 = \int_0^T (r-\mu)^2 \phi(r)dr = \int_0^T r^2\phi(r)dr - \mu^2 = \frac{1}{T}\left[\frac{r^3}{3}\right]_0^T - \mu^2 = \frac{T^2}{3} - \frac{T^2}{4} = \frac{T^2}{12} \quad \ldots\ldots (6.7).$$

Analysis of functions of random variables [Papo 84] shows that if a random variable x is uniformly distributed over the range $[c,d]$, then the random variable $y = ax^2$ has as p.d.f given by Eqn. 6.8.

$$\phi(y) = \frac{1}{2(d-c)\sqrt{ay}} \quad \ldots\ldots (6.8)$$

In the VQ lookup problem, it is required to find the p.d.f. of the vector Euclidean magnitudes. Thus taking an N-dimensional vector $\mathbf{v}$ it is required to find the p.d.f of $Z = \|\mathbf{v}\| = \sum_{i=1}^{N} v_i^2$ $\ldots\ldots (6.9)$

It may be shown that, as a consequence of the Central Limit Theorem , the p.d.f. $\Phi(Z)$ of the squared Euclidean magnitudes $Z = \|\mathbf{v}\| = \sum_{i=1}^{N} v_i^2$ of such vectors is approximately Gaussian over the range $[0, NT^2]$ with mean μ, variance σ^2 and p.d.f.:

$$\Phi(Z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-(Z-\mu)^2}{2\sigma^2}\right\} \quad \ldots\ldots (6.10)$$

From Eqn. 6.8, for each component v_i of $\mathbf{v}$, the mean value of its square $\mu_{v_i^2}$ is given by:

$$\mu_{v_i^2} = \int_0^{v_i = T} v_i^2 \phi\left(v_i^2\right) dv_i^2 = \int_0^{T^2} \frac{v_i^2}{2T\sqrt{v_i^2}} dv_i^2 \quad \ldots\ldots (6.11)$$

which gives:

$$\mu_{v_i^2} = \frac{1}{2T}\int_0^{T^2} \sqrt{v_i^2}\, dv_i^2 = \frac{1}{2T}\left[\frac{2\left(v_i^2\right)^{\frac{3}{2}}}{3}\right]_0^{T^2} = \frac{\left(T^2\right)^{\frac{3}{2}}}{3T} = \frac{T^2}{3} \quad \ldots\ldots (6.12)$$

and the variance $\sigma_{v_i^2}^2$ is given by:

$$\sigma_{v_i^2}^2 = \int_0^{v_i=T} \left(v_i^2 - \mu_{v_i^2}\right)^2 \phi\left(v_i^2\right) dv_i^2 \qquad \text{.....(6.13)}$$

Substituting $y = v_i^2$ and $t = \sqrt{y}$ so $\frac{dt}{dy} = \frac{1}{2\sqrt{y}}$. Then

$$\sigma_{v_i^2}^2 = \sigma_y^2 = \int_0^{T^2} \left(y - \frac{T^2}{3}\right)^2 \frac{1}{2T\sqrt{y}} dy = \frac{1}{2T} \int_0^{y=T^2} \frac{\left(t^2 - T^2/3\right)^2}{t} 2t dt$$

$$= \frac{1}{T} \int_0^T \left(t^2 - T^2/3\right)^2 dt = \frac{1}{T}\left[\frac{t^5}{5} - \frac{2T^2 t^3}{9} + \frac{T^4 t}{9}\right]_0^T = \frac{4T^4}{45} \qquad \text{.....(6.14)}$$

To summarise, the p.d.f. of the (squared) Euclidean magnitudes of N-dimensional uniformly distributed vectors uniformly distributed over the range $[0,T]$ is a Gaussian distribution over the range $\left[0, NT^2\right]$ having mean $\mu = T\sqrt{\frac{1}{3}}$ and variance $\sigma^2 = \frac{4T^4}{45N}$.

Since the maximum magnitude of a vector is $\sqrt{NT^2}$ then the peak of this distribution is at $1/\sqrt{3}$ ($\approx 58\%$) of the maximum magnitude. Fig. 6.10 illustrates the distribution for N= 16, T = 255. Clearly for realistic vector sizes this peak is most pronounced. The analysis may be generalised for vectors whose components have other distributions but the sharp peak at the same location is always expected as it is a result of the convolution of N functions rather than the properties of the functions themselves.

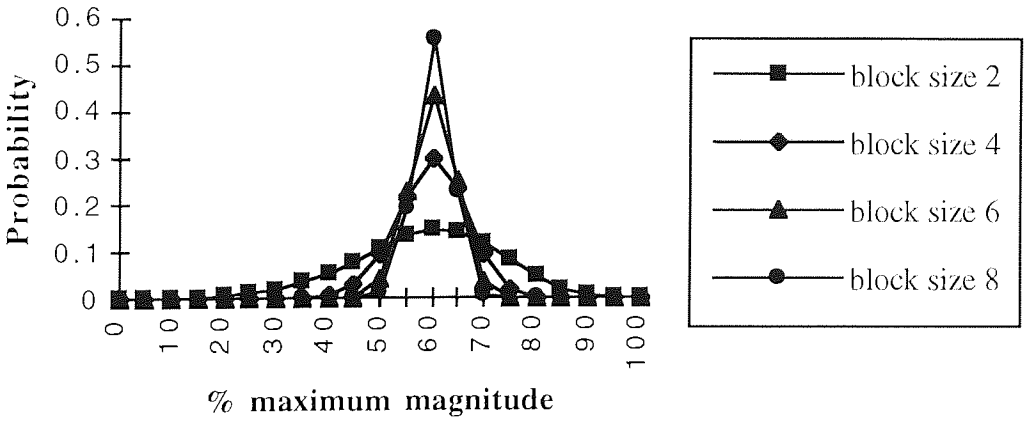


Fig. 6.10. p.d.f. of random vector magnitudes

The distribution of image squared vector magnitudes has been evaluated for various test images. For example, Fig. 6.11 shows the distribution for the 'Boats' test image. Distributions for real images are naturally much more complex than that generated by random vectors but in most cases a peak at $1/\sqrt{3}$ ($\approx 58\%$)of the maximum magnitude is visible superimposed on the distribution. This is true both for absolute vector

magnitudes and for inter-block difference magnitudes used in MS/VQ. However, the distributions for image data do not vary for different block sizes. This is because the components of each vector for image data are correlated.

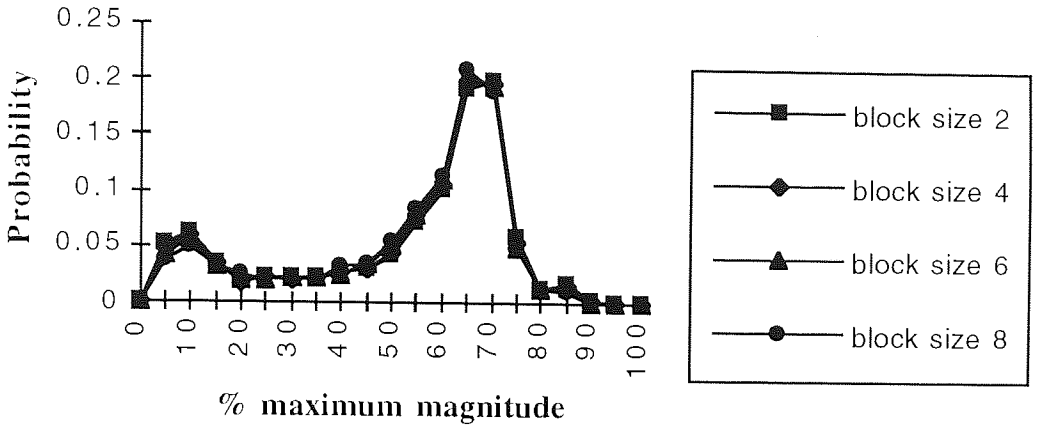


Fig. 6.11. p.d.f. of 'Boats' image vector magnitudes

This peak in the vector magnitude distribution can be exploited by scaling the search window described previously in Section 6.5.3.2 so its width is proportional to the Gaussian distribution either to reduce w and hence the lookup time for a given distortion or to improve the distortion for a given lookup speed. This has been tried experimentally here but has not been found to be particularly effective. It has already been shown that the extra distortion introduced by the truncated Euclidean magnitude search is very small, so increasing the window size around the peak in the distribution is not helpful. Conversely, reducing the window size away from the peak does provide an additional speed up but the distortion increases rapidly for very small window sizes.

6.6. Space-filling curves and VQ

In Chapter 4 it was observed that space-filling curves have the useful property of mapping from n-dimensional spaces to a one-dimensional space. It is shown here that the advantages of VQ can be preserved together with the simplicity and efficiency of scalar quantisation by using space-filling curves in both VQ codebook design and vector lookup.

As described in the previous sections the fundamental difficulty with VQ is the lookup process. However, scalar quantisation does not suffer from this drawback since optimal Lloyd-Max quantisers can be designed for various input sequences. If the elements of a vector space could be uniquely ordered such that the ordering corresponded in some way to the 'closeness' of the vectors, then the n-dimensional

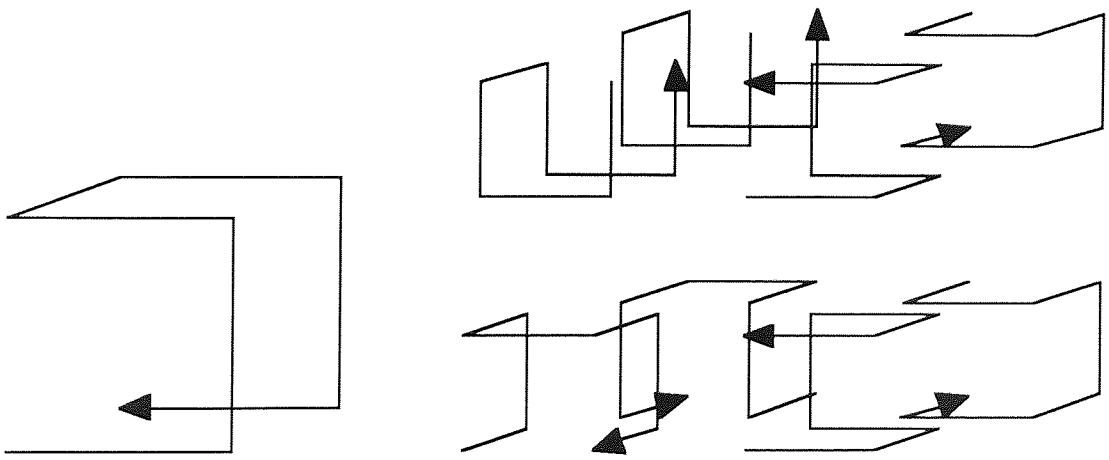
lookup problem would be reduced to scalar (one-dimensional) lookup which can be performed very efficiently with a binary search.

Let the n-dimensional vector space containing the image vectors be denoted by R_m^n, where each of the coordinates has discrete values in the range $[0, 2^m - 1]$. A mapping M is required such that: $M: R_m^n \rightarrow R_{nm}^1$.

6.6.1 Space-filling curves in higher dimensions

In Chapter 4 space-filling curves such as the Hilbert curve were used to map between two-dimensional data and a one-dimensional scanned from of the data. The definitions of space-filling curves may readily be generalised for higher dimensions. For example, Fig. 6.12 shows the first order Hilbert curve H_1^3 drawn in three dimensions and the second order curve H_2^3 formed from eight appropriately rotated copies of H_1^3 placed in the sequence defined by H_1^3.

N.B. A different convention to Chapter 4 is used here for denoting the space-filling curves. In this chapter only second degree curves (i.e. instances of the Hilbert curve) are considered but they are drawn in n dimensions instead of two dimensions. Thus H_m^n denotes the m'th order Hilbert curve drawn in n dimensions.



First order three-dimensional Hilbert curve

Second order three-dimensional Hilbert curve

Fig. 6.12. Hilbert curve in three dimensions

The mapping $M: H_m^n \rightarrow R_{nm}^1$ provides the one-to-one mapping required for VQ lookup but the practical difficulties involved in generating the mapping are considerable. For example, H_8^{16}, which could be used for 4 by 4 vectors with pixels quantised to 8 bits, contains $2^{128} \approx 3x10^{38}$ points so simple lookup table is impractical. The mapping of

each vector has to be performed as required so the issues raised in Chapter 4 regarding direct generation of Hilbert scan coordinates need to be re-examined.

6.6.2. Generation of large Hilbert curves

One early method of Hilbert coordinate generation was described by Butz [Butz 71]. It was noted in Chapter 4 that this method is excessively complicated for generating two-dimensional curves. However, his method for coordinate generation from the index of a point on a Hilbert curve made use of a bit-slice strategy which enabled the problem to be solved by splitting it in the pixel quantisation domain rather than in the spatial domain. This strategy makes possible generation of Hilbert curves in higher dimensions. Butz's algorithm is described in detail as Algorithm 6.5. Butz's method was revised by Oliveri *et al* [Oliv 86] who were able to reverse his dataflow diagram to create a process for deriving the Hilbert curve index from the coordinates (i.e. to perform the mapping $M:H_m^n \rightarrow R_{nm}^1$). Oliveri's algorithm is described in detail as Algorithm 6.6.

Neither Butz nor Olivieri gave examples for Hilbert curves of greater than three dimensions. However, the algorithms are valid for n-dimensions and have been implemented here to provide efficient Hilbert curve transformations for the large curves in higher dimensions required by vector quantisers.

Algorithm 6.5. Butz algorithm for Hilbert curve coordinate generation

The key idea introduced by Butz is that of performing the calculations for each of the n dimensions of the vector separately. A Hilbert curve index number can be expressed as a nm bit binary number r which can be split into m words (Butz used the term byte but this will not be used here since bytes are invariably considered to be of 8 bits today) with ρ^i representing the i'th word. Thus $r = \rho_1^1 \rho_2^1 \ldots \rho_n^1 \rho_1^2 \ldots \rho_n^m$.

The Hilbert address is an n-dimensional vector $\mathbf{a} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ whose n components are expressed as m-bit binary numbers $\alpha_i = \alpha_i^1 \alpha_i^2 \ldots \alpha_i^n$, $j = 1, \ldots, n$.

A sequence of bit operations (Boolean and shift operations) is then applied to each ρ^i to give values for α^i. The result of the operations is such that each word is replaced by its reflected binary Gray code (RBGC) [Pere 91]. Any cyclic progressive number system could be used but it is convenient to use the most common system which is often simply called *the* Gray code. Rules for integer to Gray code and Gray code to integer conversion were given in Chapter 4.

Bit operations for Butz algorithm

Butz expressed his algorithm in terms of the following sequence of bit operations which has to be applied to each word of the curve index number.

1. Form J_i. This is an integer between 1 and n equal to the subscript of the principal position of ρ^i. The principal position of ρ^i is the last position in ρ^i such that $\rho^i_j \neq \rho^i_n$. If all bits of ρ^i are equal then the principal position is the n'th.

2. Form σ^i. This is an n-bit word such that $\sigma^i_1 = \rho^i_1$, $\sigma^i_n = \rho^i_n \oplus \rho^i_{n-1}$, where $\oplus$ denotes the bit-wise exclusive-or operation.

3. Form τ^i. This is an n-bit word obtained by complementing σ^i in the n'th position and then, if and only if, the resulting word is of odd parity, complementing in the principal position.

4. Form S^i. This is an integer equal to $\sum_{k=1}^{i-1}(J_k - 1)$.

5. Form $\tilde{\sigma}^i$. This is an n-bit word formed by shifting σ^i right circular by S^i places. There is no shift in σ^1.

6. Form $\tilde{\tau}^i$. An n-bit word formed by shifting τ^i in the same way as σ^i.

7. Form ω^i. This is an n-bit word where $\omega^i = \omega^{i-1} \oplus \tilde{\tau}^{i-1}$, $\omega^1 = 0_1 0_2 \ldots 0_n$ (an n-bit word with all bits set to zero).

8. Form α^i. This is an n-bit word where $\alpha^i = \omega^i \oplus \tilde{\sigma}^i$.

9. Then the Hilbert address $\mathbf{a} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$.

The interactions between the n words are best shown in the form of a dataflow diagram as shown in Fig. 6.13 where the rectangular entities are the terms formed at each stage and the circular entities are the bit operations defined in Table 6.3.

Operation	Description
$v \oplus w$	The bitwise exclusive OR of vectors **v** and **w**
v ROTR a	Vector **v** rotated right by a bits
v COMP a	Vector **v** complemented in the n'th position and then if and only if the resulting vector has odd parity, complemented in a'th position.
MAIN **v**	A scalar equal to the last position j in vector **v** where $v_j \neq v_n$.
EXTR **v**	A vector **w** given by $w_1 = 0$, $w_i = w_{i-1} \oplus v_i$, for $2 \leq i \leq n$

Table 6.3. Bit operations for Butz and Oliveri algorithms

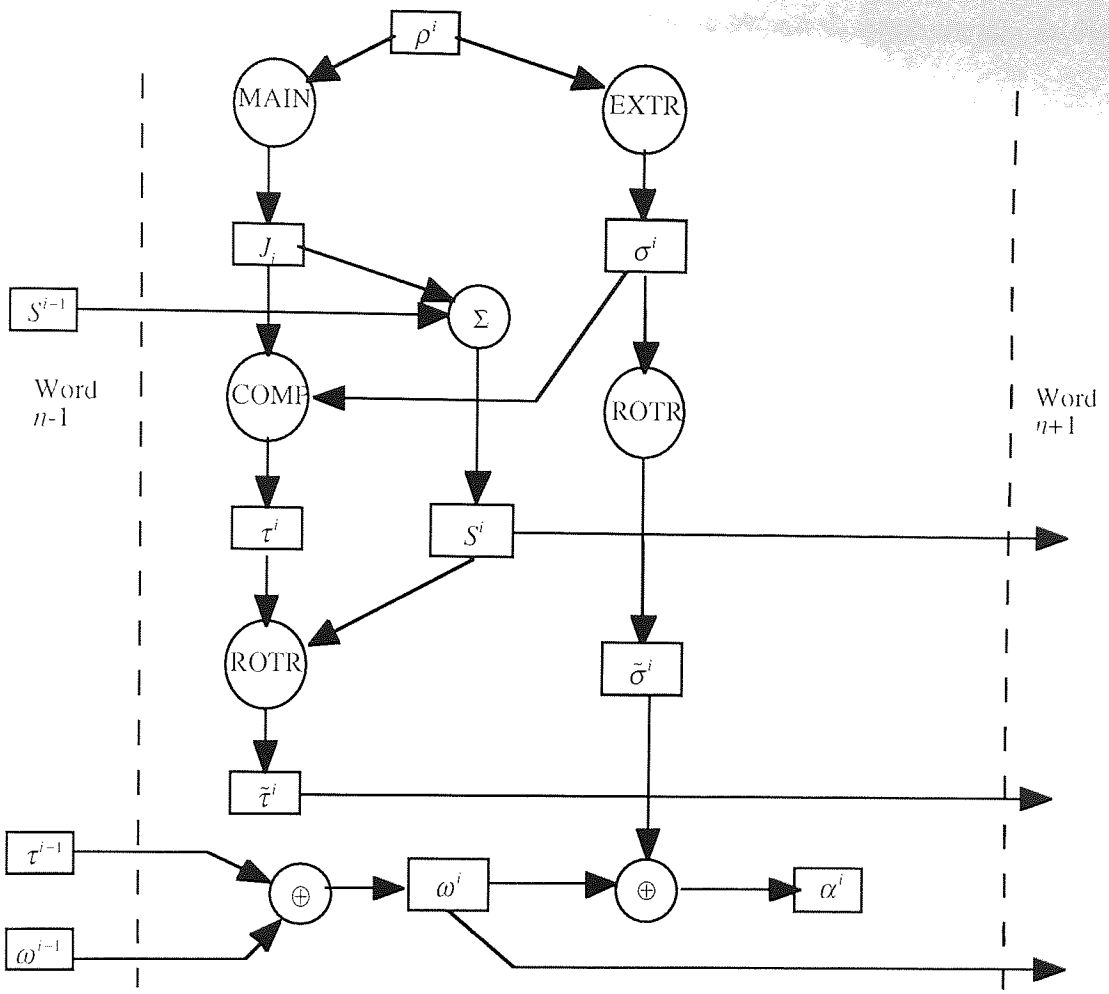


Fig. 6.13. Dataflow of Butz algorithm for Hilbert curve address generation.

Algorithm 6.6. Oliveri algorithm for Hilbert index calculation

Oliveri's algorithm for determining the sequence number of a point on a Hilbert curve is essentially a reversal of the Butz algorithm.

Given a vector $\mathbf{f} \in R_m^n$ (R_m^n is the space whose n components are expressed as m-bit binary numbers), it is required to form an integer r which is the index of $\mathbf{f}$ in H_m^n (the m'th order Hilbert curve in n-dimensions).

Express $\mathbf{f}$ as $\mathbf{f} = (c_1, c_2, \ldots, c_n)$, where $c_i = c_i^1 c_i^2 \ldots c_i^m$. Then form the m-vector $\mathbf{a}$ where $\mathbf{a} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ with $\alpha_i = \alpha_i^1 \alpha_i^2 \ldots \alpha_i^n$ and $\alpha_j^i = c_i^j$.

Let $r = \rho_1^1 \rho_2^1 \ldots \rho_n^1 \rho_1^2 \ldots \rho_n^m$, set $\omega^0 = 0_1 0_2 \ldots 0_n$ and $S^0 = 0$

Calculation of the Hilbert index of $\mathbf{f}$ now proceeds by iterating a set of bit operations for each of the m components of $\mathbf{a}$. The operations used are the same as those given

for the Butz algorithm in Table 6.3. The dataflow is shown diagramatically in Fig. 6.14.

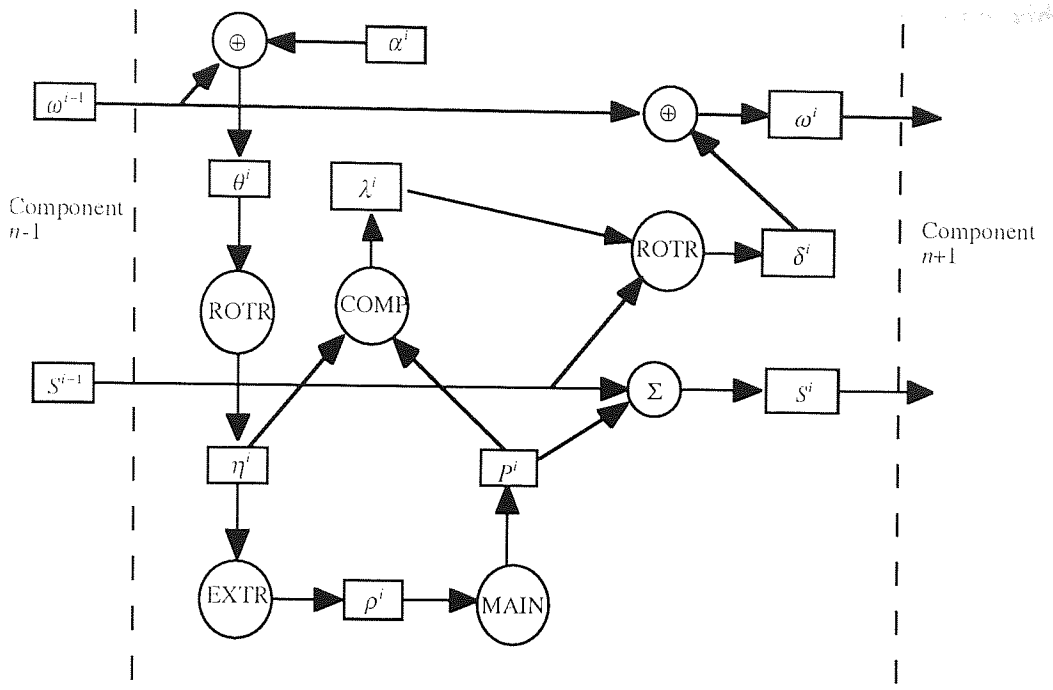


Fig. 6.14. Dataflow of Oliveri algorithm for Hilbert curve index generation

This may appear to be a rather complicated scheme but the computational complexity of the algorithm is only $O(nm)$.

For two and three dimensional curves it is straightforward to verify the algorithms (and their implementations) by exhaustively generating all the Hilbert indices of curves of order two to eight and comparing with the results obtained by tabular [Cole 87, Grif 86] or recursive [Gold 81, Wirt 76] methods described in Chapter 4. However, for higher dimensional curves the number of points is prohibitive. Also creation of the tables necessary for Cole's or Griffith's methods is not practical for higher dimensions. Some confidence in the implementations has been obtained by:

- Verifying that the two algorithms are indeed the inverses of each other for a large sample of Hilbert curve addresses and index numbers.
- Confirming that, for each dimension, a sample of points on the $(m+1)$'th order curve lie in the same hypercube as in the m'th order curve.

6.6.3 Hilbert curves in fast vector lookup

As described in Section 6.5, a good approach for improving the speed of vector lookup is to impose an ordering on the codebook so that vectors may be looked up with a simple binary search. For this to be successful the ordering must have the property that vectors which are closely ordered are 'near' to each other in the original vector space. Since a persistent theme of this thesis is this particular property of the Hilbert curves, it is appropriate to use the Hilbert indices as an ordering relation in VQ codebook lookup.

Simulations have been performed using the same codebooks and test images as for Euclidean lookup in Section 6.5 together with an efficient 16-dimensional eighth order Hilbert index generator based on the Oliveri algorithm. Initial simulations indicated that the variation in the lookup performance followed a similar relationship to that resulting from Euclidean lookup with only a small fraction of the codebook needing to be searched to achieve a good match. Thus a similar windowing strategy has been adopted.

The window centre in the codebook is given by the codebook entry with closest Hilbert index to each test vector. However, since the codebook vectors are not uniformly spaced, instead of checking vectors within a codebook address displacement proportional to the window size, vectors are checked within a Hilbert index displacement proportional to the window size. Fig. 6.15 shows the distortion performance with w set to $1/16$ to permit direct comparison with the results for Euclidean search truncation. This corresponds to searching $4/64$, $16/256$, $64/1024$ and $256/4096$ vectors respectively for the different sized codebooks.

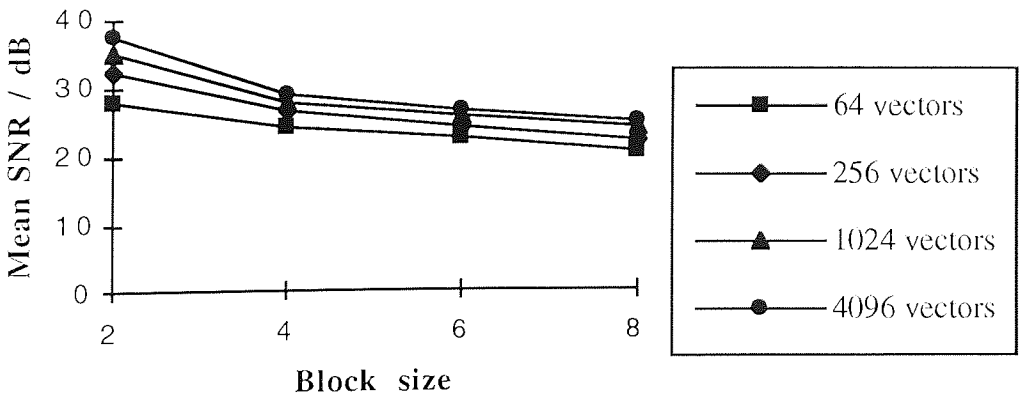


Fig. 6.15. Distortion performance of VQ system with Hilbert search (w=0.0625)

The results are again good but it is not clear from Fig. 6.15 how the Hilbert search compares with the Euclidean magnitude search. Fig. 6.16 shows the extra distortion over EFS lookup caused by the Hilbert search.

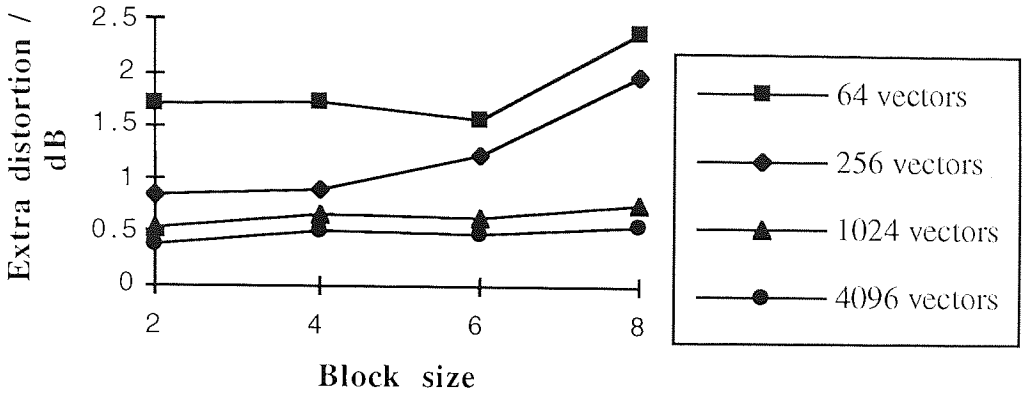


Fig. 6.16. Extra distortion caused by Hilbert lookup for different codebook sizes

Comparison of Fig. 6.16 with Fig. 6.9 shows that, while fast lookup using the Hilbert addresses of the vectors is effective, it is not superior to the fast lookup using the Euclidean vector magnitudes. It is, however, almost as effective for the largest vector sizes with the largest codebooks. If extrapolated to extremely large ($> 2^{12}$ vectors) codebooks of large vectors (> 8 by 8 blocks), it is possible that the Hilbert curve search may prove superior. This has not been tested but codebooks of this size would at least be feasible using the fast lookup algorithms proposed here but would be totally impractical using conventional algorithms.

6.7 Conclusions

Vector quantisation has been shown to provide the potential to yield substantial image data compression. The distortion resulting from VQ data compression depends on the size of the vectors and the number of vectors in the codebook. To achieve useful data compression the vectors need to have at least 16 components (corresponding to 4 by 4 pixel blocks). For vectors of this size the codebook needs to contain at least 256 and preferably more entries to achieve an acceptable SNR of greater than 30 dB. Unfortunately, the computational complexity of both the codebook design and the vector lookup during the encoding phase depends exponentially on the size of the vectors and the size of the codebook. Since it is also the cause of the computational cost of codebook design, vector lookup is considered to be the primary problem with VQ.

In this chapter two new methods for reducing the computational complexity of vector lookup have been proposed. The first, searching only a subset of the codebook determined by the Euclidean magnitude of the vectors has been shown to be particularly simple yet very effective. Typically, distortion within less than 0.5 dB of that attained using exhaustive full search is achieved by searching only $1/16$ of the codebook which gives a speedup factor of 16. The second method proposed uses Hilbert curves drawn in a number of dimensions equal to the vector size to generate an ordering of the vectors which also allows lookup to be performed on just a subset of the codebook. This method has also proved effective, but only for the largest vectors and codebooks does it match the performance of the Euclidean magnitude lookup.

For both the Euclidean magnitude and Hilbert lookup algorithms, partial search improvements identified by other workers may still be applied to the pre-selected subset of the codebook to attain a further speedup of approximately 50%. The extra distortion is acceptably small since an inherently lossy system such as VQ would not be considered as a coding scheme unless some distortion was acceptable. Users must decide where to make the trade-off in speed against distortion by setting the window size in both algorithms to match their own requirements.

Chapter 7. Iterated Function Systems

7.1 Introduction

This thesis is concerned with applications of fractals and fractal-like methods for image coding. One approach which has attracted great attention and which is perhaps closest in spirit to the use of mathematical fractals is based on the use of Iterated Function Systems (IFS) [Barn 85a]. As with any image compression system the goal is to identify succinctly and then remove the redundancy present in an image. Practical systems based on IFS achieve this through self-transformations of the image on a block-by-block basis. After a finite number of iterations of the image transformations an approximation to the original image may be derived. The description of the transformations is generally known as a fractal code. The image compression method itself is also known as fractal block coding [Jacq 92].

IFS coding incorporates concepts which differ radically from conventional image compression techniques. This chapter, of necessity, contains an introductory review of mathematical material specific to IFS coding before presenting a comparative study of practical approaches.

In general, the main criterion for the usefulness of an IFS-based image compression system is how well it is automated (i.e. how the fractal code is derived). Early methods required substantial manual intervention. In addition, all of the variations of IFS coding are extremely computationally demanding. Three different methods have been implemented here to provide a framework within which performance-enhancing techniques can be investigated.

It is shown here that there are a number of similarities between IFS coding and Vector Quantisation. The performance enhancing techniques for VQ identified in Chapter 6 have also been successfully applied here and it is shown that IFS coding can provide significant image data compression while maintaining good image quality.

7.2 History of IFS image compression

Iterated Function Theory was first developed by Hutchinson [Hutc 81] but was later popularised by Barnsley with his book 'Fractals Everywhere' [Barn 88b]. Barnsley observed that real images are rich in affine-redundancy (i.e. under suitable affine transformations, large parts of images look like smaller parts of the same image). He derived the important Collage Theorem [Barn 85b] which places significant restrictions

on an IFS if it is to represent an image. There is near-universal acceptance that an IFS or other fractal description can generate natural looking images (the word 'natural' is important; visual artefacts caused by the compression process are the downfall of many other methods). However, there has been considerable scepticism about the practicality of 'the inverse IFS problem' - i.e. how to generate the IFS for a given image.

Barnsley believed his Collage Theorem (see Section 7.4.4.3) solved the problem and was granted a patent [Barn 91] based on its application which he has attempted to exploit commercially though his company Iterated Systems Inc. Unfortunately, the manner in which he announced his success [Barn 88a] with extravagant claims of 10,000:1 data compression meant that the elegance of the theory was overlooked leading to a rather derisive response from the image processing community. Kominek [Komi 94] suggested the term 'Graduate Student Algorithm' for the process of locking a student in a room with a computer and an image until the student had derived a good IFS to represent the image!

Certainly, Barnsley's original scheme required many hours processing time on a powerful workstation to compress a single image even with a human guiding the software. However, he invented a modified scheme called Partitioned Iterated Functions Systems (PIFS) which was implemented by Jacquin [Jacq 89] as the first practical IFS scheme. Far more modest compression was claimed but the algorithm was amenable to full automation. Current IFS developments all stem from Jacquin's work.

7.3 Basic idea of IFS image compression

This section contains an overview of Barnsley's original presentation of the IFS coding method. It is not at all suitable for compressing real images but it is helpful for clarifying the ideas which are explored in a more sophisticated way in Section 7.4.

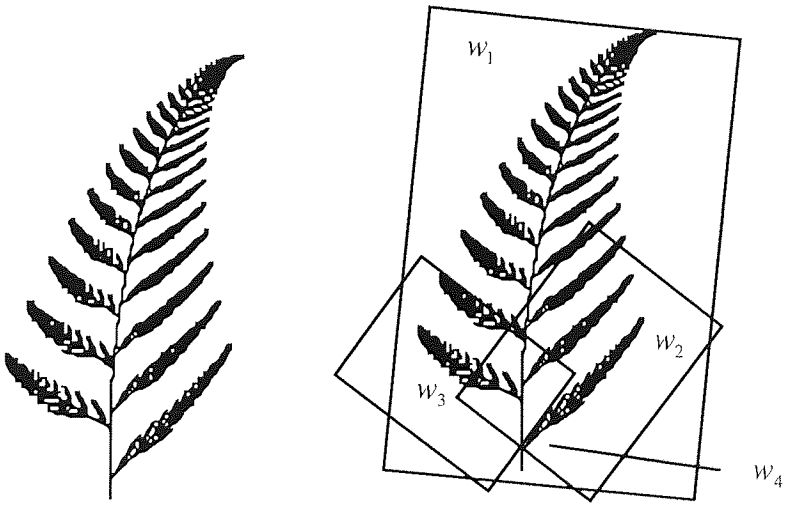
An IFS is a collection $W = \{w_k : k = 1,\ldots,N\}$ of k contractive affine transformations which may be described as combinations of rotations, scalings and translations of coordinates. It is required that each transformation is contractive (i.e. it moves points closer together). Each transformation may be written in matrix form as:

$$w_k(x,y) = \begin{bmatrix} a_{11}(k) & a_{12}(k) \\ a_{21}(k) & a_{22}(k) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1(k) \\ b_2(k) \end{bmatrix} \quad \ldots\ldots (7.1)$$

If the region of the plane to be transformed is an image, then the transformation moves points from the image to a distorted copy of the image.

Barnsley and his co-workers discovered that, starting with arbitrary values for x and y, if each of the transformations in turn is iteratively applied then the set of points traced in the plane converges to a fixed pattern or image which depends on the choice of coefficients in the IFS. If probabilities P_k are assigned to each transformation in the IFS and just one transformation is used at each iteration, then the rate at which the pattern converges to the final image increases considerably, but the final pattern is unchanged.

Barnsley's most famous example is the black spleenwort fern for which he found a striking likeness could be achieved with an IFS consisting of only four transformations. (w_1 maps the fern to the upper part of itself; w_2 and w_3 map the fern to its lower right and left leaves; w_4 maps the fern to the lower part of the stalk). Fig. 7.1(a) shows the whole fern. Fig. 7.1(b) shows the transformations which produce the fern . The IFS coefficients are shown in Table. 7.1.



(a) Complete fern **(b) Transformation set**

Fig. 7.1. Barnsley's fern

w	a11	a12	a21	a22	b1	b2	P
1	0	0	0	0.16	0	0	0.01
2	0.2	-0.26	0.23	0.22	0	1.6	0.07
3	-0.15	0.28	0.26	0.24	0	0.44	0.07
4	0.85	0.04	-0.04	0.85	0	1.6	0.85

Table 7.1. IFS Coefficients for Barnsley's fern

The motivation for using IFS for compressing images is that the table specifying the IFS coefficients has a very low storage requirement compared with the image which results from iteration of the IFS. The fundamental problem is finding the IFS coefficients which will produce a specific image.

7.4 Theory of IFS image compression

At first sight it is not at all obvious that an image can be described better with a process which produces the image from an arbitrary starting image than by a description of the image itself. However, analogous problems with real numbers [Dett 92, Wait 90] make this clearer. For example, consider the problem of finding the square root of 2 using Newton's method. It is simple to show that the iteration sequence $x_{n+1} = \frac{x_n^2 + 2}{2x_n}$ leads to $\sqrt{2}$. Indeed, even with a very poor initial value $x_0 = 100$, a result is obtained accurate to three decimal places after only ten iterations. It is the quality of the iteration or transformation which is important, not the initial value. In IFS coding, this same principle is applied to image blocks instead of numbers by iteration of contractive mappings. The mappings are expressed as combinations of isometric (i.e. rotation, reflection and translation) and grey-level scaling transformations. Each isometric transformation must be spatially contractive but will not alter the block grey levels. An additional grey-level transformation is required to achieve convergence of the grey levels.

To code a whole image, the image must first be partitioned into a set of blocks which together tile the whole image. For each block the rest of the image is searched for a region which is 'similar'. The most similar block is termed the subject block's match. A code is then generated for each block specifying the mathematical function which maps its match contractively onto itself, while minimising the difference between the result of the transformation and the original subject block.

Fractal image compression has been justified with various degrees of mathematical sophistication. Unfortunately the more rigorous derivations [Jacq 89, Barn 85a, Barn 88b] lead to the least practical implementations. Since fractal image compression is a lossy technique it is not strictly necessary that a proof should exist for its validity, but it is important that some bounds can be placed on the probable fidelity of any reconstructed images. In this section an appropriate notation and a non-rigorous explanation are developed which show how fractal image coding works. It is first necessary to describe metric spaces, affine transformations - in particular self-affine transformations - and to demonstrate Barnsley's Collage Theorem which is the main justification for IFS image coding.

7.4.1 Metric Spaces

Fractal image coding is derived from the use of metric spaces to represent images. A metric space (X,d) is a space (i.e. a set of points) X together with a real-valued

function d, which measures the 'distance' between pairs of points x and y in X. If d obeys the four axioms:

(1) $\forall x, y \in X: d(x,y) = d(y,x)$ (2) $\forall x, y \in X, x \neq y: 0 < d(x,y) < \infty$

(3) $\forall x \in X: d(x,x) = 0$ (4) $\forall x, y, z \in X: d(x,y) \leq d(x,z) + d(z,y)$

then d is a metric for the space X.

If continuous (i.e. non-digitised) images are available for coding then the images may be considered as *measures* in the two-dimensional plane. IFS theory has been developed for this model by Barnsley and Hurd [Barn 92] but in most cases the images which are to be coded either already exist in digitised form or are produced by an inherently discrete device such as a CCD camera. Either way, the digitised form is the most amenable to computer processing. It is therefore more appropriate to consider models where images are represented by discrete pixels.

For IFS coding there are two equivalent ways of modelling a digital image. When partitioning the image, as will be required in Section 7.5.1, it is convenient to consider a grey-scale n by n pixel image to be a function $p: G \rightarrow R^{n^2}$ on the set G of all ordered pairs (i,j) (pixels) with $1 \leq i \leq n$, $.1 \leq j \leq n$.

Alternatively, when considering contractive transformations in Section 7.4.3 it is more convenient to represent the image by a vector $\mathbf{p} \in R^{n^2}$. This implies the existence of a suitable mapping of the elements p_k of the vector $\mathbf{p}$ to the values $p(i,j) = p_{ij}$ of the function p. Any one-to-one mapping is valid such as the simple raster scan which gives $p_{k=n(i-1)+j} = p_{ij}$ but the more complex space-filling curve mappings of Chapter 4 could be applied.

7.4.2 Distance metrics

There are many valid metrics which could be used to represent the distance or distortion, d, between a source image and its reconstructed coded image. The term 'metric' is preferred to 'distance' since it is more general; the distance between two sets may not be a meaningful concept. Barnsley assumed that the Hausdorff fractal metric would be used to measure the 'similarity' of images but such measures are difficult to calculate. The Hausdorff metric $h(\mathbf{x},\mathbf{y})$ is defined by:

$$h(\mathbf{x},\mathbf{y}) = \max\left\{\max_{1\leq i\leq m}\left[\min_{1\leq j\leq m}\left(d\left(x_i,y_j\right)\right)\right], \max_{1\leq i\leq m}\left[\min_{1\leq j\leq m}\left(d\left(x_j,y_i\right)\right)\right]\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7.2)$$

The distinctive feature of this metric is that is provides a better value for the 'closeness' of sets in shape and geometry while remaining insensitive to slight shifts. It is therefore considered better for geometrical modelling [Barn 88c]. The Hausdorff metric is also called the min-max metric since it equals the longest minimal distance between all of the pairs of points (taken in each direction) of two sets. For example, in Fig. 7.2 the Hausdorff distance is shown as a bold line between three different pairs of sets **x** and **y**. In Figs. 7.2(a) and 7.2(b) the Hausdorff distance is measured from **x** to **y**; in Fig. 7.2(c) it is measured from **y** to **x**.

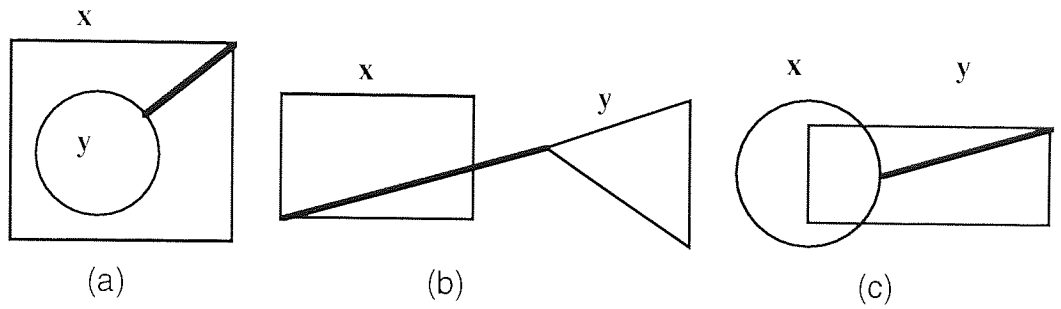


Fig. 7.2. Hausdorff metric

Since images are frequently represented by real valued vectors it is reasonable to consider distance measures based on vector p-norms, i.e.

$$d_p(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_p \qquad (7.3)$$

The p-norms of the vector $\mathbf{x} = (x_1, x_2, ..., x_m)$ are defined by:

$$\|\mathbf{x}\|_p = \sqrt[p]{\left(\sum_{i=1}^{m} |x_i|^p\right)} \qquad (7.4)$$

with the most common values for p being 1 (Manhattan metric), 2 (Euclidean or RMS metric) and ∞ ($\|\mathbf{x}\|_\infty = \max_{1\le i\le m} |x_i|$).

There is also a practical reason for using simpler geometric metrics. It is only necessary to use a norm in which iterated transformations (see Section 7.4.4) are contractive. It can be shown [Wait 90] that, provided at least one norm can be found in which the transformations are contractive, then the iteration will converge in *any* norm so, for practical purposes, the simpler metrics are more appropriate.

7.4.3 Transformations on metric spaces

If (X,d) is a metric space then a transformation on X is a function $f: X \to X$ which assigns exactly one point $f(x) \in X$ to each point in X. If f is *one-to-one* (i.e.

$f(x) = f(y)$ implies $x = y$) and *onto* (i.e. $f(X) = X$) then f is also invertible and an inverse transformation $f^{-1}: X \to X$ can be defined.

Using the same terminology as standard function theory, a transformation on a metric space is said to map *from* a *domain to* a *range*.

For IFS image coding use is made of the self-similarity of images which is apparent on a global and local scale. In both cases, but particularly for small image blocks, the similarity takes the form of (range) image blocks appearing to be distorted versions of other (domain) blocks. The image scene may, for example, consist of many copies of a given three-dimensional object, which appear different due to positional translation, perspective, rotation in the plane of the image, rotation in the third dimension of the image scene or some combination of these. All of these distortions can be caused by a class of geometric transformations known as *affine* transformations (Section 7.4.3.1) which are the basic component of IFS image coding systems.

In addition, variations in the illumination of the objects lead to similarity of image blocks under grey-scale transformations. These can be described either by a modification of the affine transformations to three dimensions or, more simply, by a separate class of grey-scale or *massic* transformations discussed in Section 7.4.3.2.

7.4.3.1 Affine transformations

The transformations used in IFS coding are the familiar geometric transformations in the Euclidean plane $\mathbf{R}^2$. A transformation $w: \mathbf{R}^2 \to \mathbf{R}^2$ of the form shown in Eqn. 7.5 is a two-dimensional affine transformation.

$$w(x, y) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad \ldots\ldots (7.5)$$

Since w can always be re-written as Eqn. 7.6,

$$w(x, y) = \mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{t} \quad \ldots\ldots (7.6)$$

where $\mathbf{A}$ is a 2 x 2 real matrix and $\mathbf{t}$ is a column vector, then such a transformation represents a linear transformation $\mathbf{A}$ which deforms space relative to the origin followed by a translation or shift by the vector $\mathbf{t}$.

The matrix $\mathbf{A}$ can also be re-written as in Eqn. 7.7:

$$\mathbf{A} = \begin{bmatrix} r_1 \cos\theta_1 & -r_2 \sin\theta_2 \\ r_1 \sin\theta_1 & r_2 \cos\theta_2 \end{bmatrix} \quad \ldots\ldots (7.7)$$

where (r_1, θ_1) and $(r_2, \theta_2 + \pi/2)$ are the polar coordinates of the points (a_{11}, a_{21}) and (a_{12}, a_{22}) respectively. If $r_1 = r_2 = r$ and $\theta_1 = \theta_2 = \theta$ then **A** is known as a similitude transformation which applies a rotation through an angle θ and a scaling of r.

For example, Fig. 7.3 shows the affine transformation $w(triangle)$ of the small triangle with vertices (x_1, y_1), (x_2, y_2) and (x_3, y_3) to the large triangle with vertices (X_1, Y_1), (X_2, Y_2) and (X_3, Y_3).

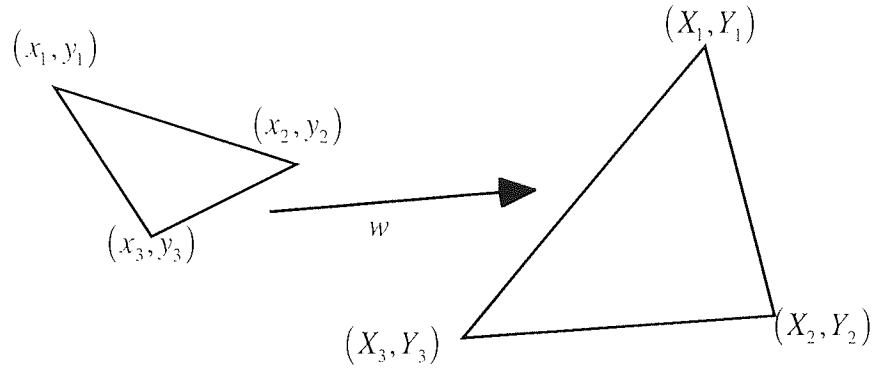


Fig. 7.3. Affine transformation of triangle

Although the general affine transformation allows for scaling and skewing, a further simplification is often applied. The majority of practical schemes use only fixed scaling and do not use skewing at all. In addition, instead of the infinite set of rotations allowed by the general affine transformation, typically a much smaller set of only eight rotations is used. This simplification of the transformations has implications for the storage of the IFS coefficients since they are effectively quantised and can be stored far more economically than the real numbers required for the general affine transformation coefficients.

7.4.3.2 Massic transformations and grey-scale images

The affine transformations described in Section 7.4.3.1 are only suitable for binary images since they do not produce a grey-level scaling. However, it is possible to extend the affine transformations into three dimensions by adding a third dimension which represents the grey-scale intensity.

In three dimensions the affine transformation described by Eqn .7.5 becomes:

$$w(x, y, z) = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \text{.....(7.8)}$$

with z representing the grey-scale intensity and the new coefficients a_{33} and b_3 representing a grey-scale scaling factor and offset respectively. However, it is more convenient to split this transformation into a two-dimensional affine transformation as before together with a separate one-dimensional grey-scale or massic transformation.

$$z = a_{33}z + b_3 \quad \text{.....(7.9)}$$

This term 'massic' was introduced by Jacquin [Jacq 89] to emphasise that the transformation affects the pixel values of the block by changing the intensity or mass distributed over a cell. This simple transformation provides all the grey scale operations which are required in practice. i.e.:

- Absorption at grey level g_0. All block pixels are set to a single grey level g_0.
- Intensity shift by Δg. All block pixels are changed by a level offset Δg.
- Contrast scaling by value α. All block pixel intensities are multiplied by a scaling factor α.
- Combinations of scaling and translation.

7.4.3.3 Contractive transformations

A contractive transformation moves points in a metric space closer together. The contractivity factor is a measure of how much closer the points are after the transformation. Contractivity can be formally defined by the Lipschitz factor.

Given a metric space (X,d), a transformation $w: X \to X$ is Lipschitz with Lipschitz factor s if there exists a positive real value s such that: $d\big(w(x),w(y)\big) \le sd(x,y)$ for every $x,y \in X$. If $s<1$ then w is contractive.

Under an affine transformation parallel lines are preserved but angles are not so the transformation maps all the points (x,y) in one domain parallelogram to the points (X,Y) in another range parallelogram. The transformation is said to be contractive if the area of the range parallelogram is less than the area of the domain parallelogram. The contractivity factor s of an affine transformation is equal to the determinant of the affine transformation matrix **A**. i.e.

$$s = a_{11}a_{22} - a_{12}a_{21} \quad \text{.....(7.10)}$$

For the massic transformation defined in Eqn. 7.9 the contractivity factor is simply the intensity scaling coefficient a_{33}.

7.4.4 Iterated transformations

The general three dimensional transformation of Eqn. 7.8 may be iterated by repeatedly applying the transformation to the output of the previous transformation. Some important theorems for IFS coding concerning iterated transformations are restricted to metric spaces with certain properties. In particular the metric spaces must be *complete*.

A metric space (X,d) is complete if the limit point of every sequence $\{x_n\}$ of points in the space is also in the space X. Formally, for any sequence of points $\{x_n\}$ in the metric space (X,d), for any $\varepsilon > 0$,there exists an integer N such that, for all $n,m > N$, $d(x_m, x_n) < \varepsilon$ and $\left(\lim_{n\to\infty} x_n\right) \in X$.

Completeness implies that the space has no 'missing values'. For example, the set of rational numbers with the metric $d(x,y) = |x-y|$ is not complete since it is easy to create sequences which converge to irrational numbers. For IFS image coding completeness implies that iterated transformations of the space defined for the images must converge to another image.

7.4.4.1 Contractive transformation fixed-point theorem

It is now possible to state the contractive transformation fixed-point theorem which is the first of the two key theorems which underpin IFS coding. If (X,d) is a complete metric space and $w: X \to X$ is a contractive transformation then, denoting by w^n the n'th iteration of w, there exists a unique fixed point or attractor $x_f \in X$ such that for any point $x \in X$:

$$x_f = w(x_f) = \lim_{n\to\infty} w^n(x) \quad \text{.....(7.11)}$$

It can also be shown [Fish 94a] that, for $n > m$:

$$d\left(w^m(x), w^n(x)\right) \le \frac{s^m}{1-s} d(x, w(x)) \quad \text{.....(7.12)}$$

Since the contractivity factor $s < 1$, by choosing large values for n and m, the left hand side of the equation, the difference between iterations, can be made arbitrarily small.

7.4.4.2 Iteration of a set of transformations

An iterated function system (IFS) is a collection $W = \{w_k : k = 1,\ldots,N\}$ of k contractive affine transformations with:

$$w_k(x,y)=\begin{bmatrix} a_{11}(k) & a_{12}(k) \\ a_{21}(k) & a_{22}(k) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}+\begin{bmatrix} b_1(k) \\ b_2(k) \end{bmatrix} \quad \text{.....(7.13)}$$

If W is a contraction mapping, then there is a unique attractor set A, which satisfies the invariance condition:

$$A=\bigcup_{k=1}^{N} w_k(A) \quad \text{.....(7.14)}$$

i.e. in the same way that a single contractive transformation has a fixed point, the union of a set of such mappings has an attractor set. This implies that iteration of W on *any* set will result in the sequence of sets converging to A with a rate of convergence dependent on the contractivity factors of the separate transformations.

Whether the set of transformations in Eqn. 7.13 as a whole is contractive under iteration depends on the choice of metric. Since all of the metrics described in Section 7.3.2 depend only on the intensity (i.e. on the z axis), it is not strictly necessary that the two-dimensional affine transformation should be contractive. It is only required that the massic contractivity $a_{33}<1$ for each transformation. However, it has been found by Fisher [Fish 94a] that even this restriction can be relaxed. He found that allowing *some* of the a_{33} coefficients to be as large as 1.2 still ensured eventual contraction since iteration of the set of transformations for each image block 'mixes' the scalings.

In spite of this it is sensible to try and make both the affine and the massic transformation contractivity factors as small as possible since this increases the rate at which the overall set of transformations converges.

7.4.4.3 The Collage Theorem

The original IFS encoding scheme proposed by Barnsley and Sloan [Barn 88a] concentrated on trying to find an IFS for the whole image. Not surprisingly, finding transformations which could approximate a real world image by transformed copies of itself proved difficult and was only achieved for a very restricted category of images such as Barnsley's fern (see Section 7.3).

It is now generally accepted that solving the inverse IFS problem for a complete image by matching a transformed version of a *whole* image to itself is probably impractical since real image scenes are diverse and do not necessarily obey fractal geometry. Real ferns do not branch down to infinitesimally small leaves; they are distorted, discoloured and torn and they are probably surrounded by other plants and ground which looks very different. However, some parts of an image often exhibit great similarity with

other parts. For example, in a landscape one grassy region looks very much like another. If the image is examined at higher magnifications this self-similarity becomes even more apparent. Anyone who has ever assembled a jigsaw will be familiar with this effect!

It is reasonable to consider segmenting the image according to features such as edges, texture, intensity etc. and 'looking up' each image segment in a library of pre-computed IFS codes. In principle this would greatly simplify the problem of finding the IFS codes. However, a very large library would be required to contain a usefully wide selection of IFS types. In addition it is a non-trivial problem to determine to which class of IFS codes a given image segment belongs. While it is easy for a human observer to say that a certain image region looks, for example, like a cloud, this is not yet practical for an automated system.

Instead, the source image can be used to form its own IFS library by partitioning the image into smaller sub-image blocks which are matched with transformed versions of other blocks. The codes then form a self-affine system (SAS) with the IFS codes now mapping from large parts of the image to smaller parts of the *same* image. The global transformation of the image onto itself is then constructed from a combination of local block transformations. How the image is partitioned into domain and range blocks is considered in Section 7.5 as an issue to be addressed by practical IFS coding systems.

If infinite iteration of Eqn. 7.12 is considered then the Collage Theorem can now be stated as:

$$d\left(x, x_f\right) \leq \frac{1}{1-s} d\left(x, w(x)\right) \qquad \text{.....(7.15)}$$

The image encoding problem (or inverse problem) is finding the IFS whose attractor is the image to be encoded. For the inverse problem to be soluble it is required that the distance between the original image and the once-transformed image $d\left(x, w(x)\right)$ should be as small as possible. The once-transformed image is known as the image collage (since the process of covering an image with transformed copies of itself is similar to creating an artwork collage). Barnsley's Collage Theorem implies that to find an IFS whose attractor is close to a given set, it is necessary only to find a set of transformations such that the union (or collage) of the images of the given set under the transformations is *near* to the given set.

7.4.5 Summary of IFS coding theory

Starting with a source image μ_{orig} to be encoded the inverse problem of iterated function theory is the optimisation problem:

> Find the contractive image transformation τ defined from the metric space (M,d) to itself for which μ_{orig} is an approximate fixed point.

It follows from the Collage Theorem that some immediate requirements may be placed on τ. Taking a scalar contractivity factor s then it is required that: $\exists s < 1$ such that $\forall \mu, \nu \in M,\ d\big(\tau(\mu), \tau(\nu)\big) \leq sd(\mu, \nu)$. In addition $d\Big(\mu_{orig}, \tau\big(\mu_{orig}\big)\Big)$ should be 'as close to zero' as possible (this is not as vague as it seems; the Collage Theorem only requires that the collage should be approximate).

The transformation τ is known as a fractal code for μ_{orig} which is said to be approximately self-transformable under τ. Provided that τ has a lower complexity than the original image then τ can be viewed as a lossy image code. Different practical implementations of IFS coding (see Section 7.5) lead to different numbers of coefficients in τ but, in each case, the coefficients are simply discrete numerical values to which conventional bit assignment and entropy coding (see Chapter 2) can be applied.

7.5 Practical IFS image coding

IFS coding proceeds by means of a search of a 'virtual codebook' obtained from a pool of domain blocks and a pool of block transformations. For each range block, the 'closest' transformed domain block is found and the transformation coefficients required to achieve this match are stored as the IFS code for that range block.

The issues to be addressed by practical IFS coding schemes may be summarised as:

(1) How to partition the image into domain and range blocks.

(2) Specification of a class of discrete contractive image transformations which will be considered in searching for the best match.

(3) Choice of search strategy and matching condition (which implies choice of a specific distortion measure).

(4) Optimisation of storage of IFS codes

(5) How to iterate the codes so as to generate the decoded image.

7.5.1 Image partitioning

There are various ways in which the partitioning may be performed such as colour separation, edge detection and texture variation and spectrum analysis. Such context-dependent schemes are hard to apply since it requires considerable storage simply to describe the partition itself. If non-context-dependent partitioning is used a decision has to be made on whether to use regular partition sizes and shapes or to allow irregular partitions.

The simplest practical partitioning method involves dividing the image into a number of square blocks of equal size to form the range blocks and a number of larger, but still square domain blocks. A domain block may be at any position in the image and may overlap other domain blocks. The range blocks form a uniform tiling of the image whereas the domain blocks may not even cover the image. This is illustrated in Fig. 7.4.

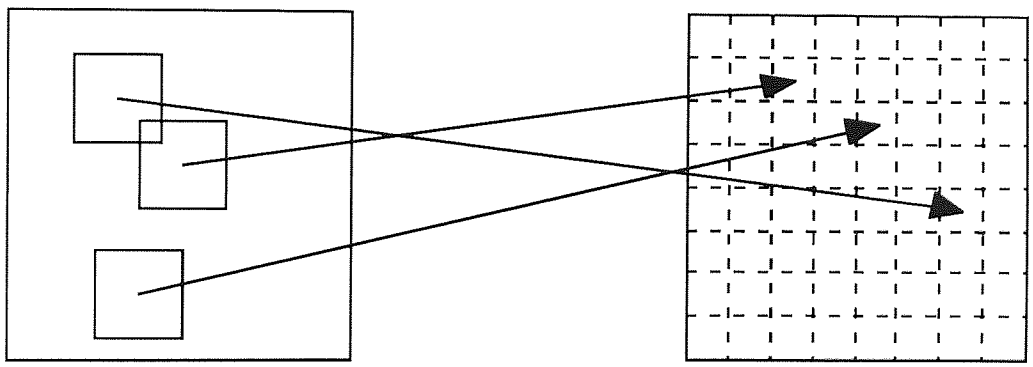


Fig. 7.4. Domain to range block mapping

For every range block there is a corresponding domain block, the position of which is an unknown to be determined in the coding process. The global transformation is then constructed by taking each range block in the image and specifying a domain block and a domain block to range block transformation.

7.5.1.1 Non-rectangular partitioning

There is no *a priori* reason why the domain or range blocks should be square or even rectangular. Since affine transformations map between parallelograms it would seem that parallelograms or triangles would be the natural partitioning element. For example, consider the rather contrived example image in Fig. 7.5 which contains five triangular regions with the same texture on an irregular background texture.

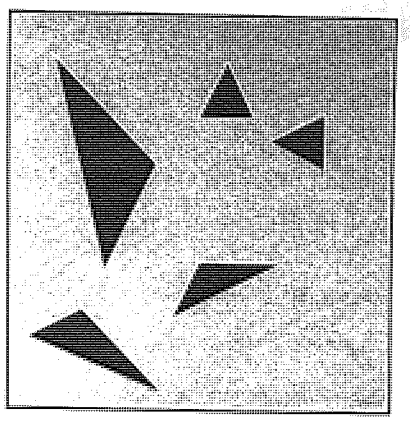


Fig. 7.5. Non-square image partitioning for IFS

Clearly the largest triangle could be chosen as a domain block and each of the other four triangles could be formed by single transformations of it. In this example much of the image is coded by just the few parameters needed to describe the triangle vertices and their texture. However, the remaining part of the image has a most irregular shape and would require many more parameters to describe. Although there are images consisting of many copies of a single fixed shape region, most natural images would exacerbate this problem greatly. In addition there is a significant difficulty caused by the conventional definition of images on a rectangular grid. If a non-rectangular partition is employed then there are difficulties in deciding to which partition each pixel belongs after transformation.

There are few published accounts of non-rectangular partitioning. Jacobs *et al* [Jaco 92] used rectangular domain blocks but allowed 45° block orientation and found this led to increased coding time and reduced compression but better reconstructed image quality.

7.5.1.2 Rectangular partitioning

The majority of IFS-based systems described in recent reviews [Jacq 93, Saup 94, Fish 94a] have used square grids to define the domain and range blocks. A regular block grid makes IFS coding much easier to automate since it avoids diversion by a few obviously self-similar features. Making the grid square adds efficiency since only a single pair of coordinates is required to specify each domain block.

The grid sizes are an important issue. For practical reasons the range blocks should be fairly small since small blocks are more likely to be similar to other blocks in the image. Also, smaller blocks fall more readily into categories which can be used to limit the search for block matches. Unfortunately the use of small range blocks places severe limitations on the potential compression achievable. For example if 4 by 4 range blocks

are used, each requiring an IFS code, then, depending on how the IFS coefficients are stored (see Section 7.5.4), but assuming 24 bits per IFS code, the maximum compression would be approximately 5:1.

7.5.1.3 Hierarchical partitioning

A solution to the poor compression achieved by small, square block partitioning is to consider multi-level or hierarchical partitioning schemes. Jacquin [Jacq 92] used two-level grids: 16x16 and 8x8 for the domain blocks and 8x8 and 4x4 for the range blocks but this provides not much improvement in compression. Quad-trees (described in Section 4.3.4) offer a far superior strategy for partitioning and were first used for IFS coding by Jacobs *et al* [Jaco 92].

For quad-tree partitioning the sizes of the root and leaf blocks need to be determined. Ideally, to reduce the number of blocks in the tree, the leaf cells should be as large as possible. For example, Jacobs *et al* used quadtree partitioning from 32x32 for the root block down to 4x4 for the leaf blocks.

The advantage of using a quad-tree to define the range blocks can be shown by an example. Given a source image of 512 by 512 pixels, if 4 by 4 square range blocks are used then the image will always require 16384 IFS codes. Using a quad-tree with a minimum square size of 4 by 4 pixels will require a maximum of 21845 IFS codes but few images will require anywhere near the this number of codes. For the large image test set experiments indicate that, on average, only 3000-4000 codes are required for each image. Fig. 7.6 shows an enlarged portion of a quad-tree partition of the Lena test image. There are several regions where range blocks as large as 32 by 32 pixels could be used.

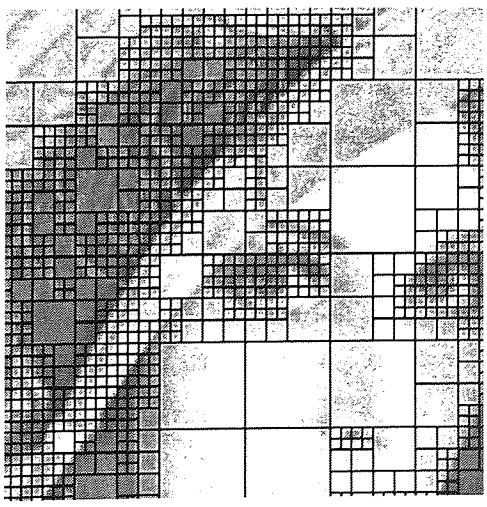


Fig. 7.6. Detail of Lena image showing quad-tree partitioning

Each code for the quad-tree requires slightly more storage than a fixed range block size. Depending on how the quad-tree is represented the additional storage is only 1 or 2 bits per IFS code. Therefore, using the quad-tree for partitioning can typically lead to image data compression by a further factor of 4.

Fisher and Menlove [Fish 94b] used a partitioning scheme called horizontal/vertical (HV) partitioning with rectangular range blocks and having each rectangular region of the image recursively divided, either horizontally or vertically, to form two new rectangular domain blocks. This has similar properties to the quad-tree but has the advantage of being adaptive to the image features. However, it does require twice the storage for the IFS data to define the domain block locations.

7.5.1.4 Implicit partitioning

Kaouri [Kaou 91] proposed a method in which the domain partitioning is implied by the range partitioning. Each small range block is transformed from the large domain block which contains it. He claimed that this did not affect picture quality and gave a considerable reduction in coding rate but he did not say which images he used. It is reasonable to assume that his simplification would work for images without much self-similarity but would achieve far less compression for images with self-similarity distributed across the image and at different scales

7.5.2 Specification of possible transformations and block classification

The general IFS transformation permits an extremely large number of possible transformations of each domain block which may yield each range block. Practical IFS coding methods must reduce this variety either using some theoretical justification or, more typically, a pragmatic choice is made simply to reduce the complexity.

The most common method of reducing the number of possible transformations is to restrict the classes of affine transformations. For example, most methods exclude skew transformations. Kaouri [Kaou 91] went as far as excluding rotation and reflections leaving only translational transformations.

The massic transformations may also be reduced to a small set. For example, Jacquin [Jacq 92] restricted the massic scaling parameter a_{33} to have only four possible values chosen from the set $\{0.7, 0.8, 0.9, 1.0\}$.

Each reduction in the possible types of transformation has the added advantage of reducing the number of coefficients required to specify the IFS (see Section 7.5.4) and therefore increases the achievable compression.

7.5.3 Search strategy and matching condition

As was discussed in Section 7.4.2 there seems to be little practical merit in adopting a more complicated distortion measure than the simple Euclidean metric. This leaves open the issue of how many domain blocks are searched for a match and in which sequence they are tested.

The search strategy is partly determined by the selection of allowed transformations as discussed in Section 7.5.2. If a wide variety of transformations are permitted, then to code an image in a reasonable time only a relatively small subset of the possible transformed domain blocks can be compared with each range block.

To determine the appropriate transformation for each IFS block is computationally expensive because it requires a extensive or possibly exhaustive search of all candidate domains. For each domain each possible transformation must be applied. It is desirable to reduce the list of candidate domains in some way. Similar problems have been encountered in VQ as described in Section 6.5 and the same solution, block classification, can be applied in IFS coding to simplify the search for optimal domain blocks.

The well-regarded system based on classification into shade blocks, edge blocks, and midrange blocks introduced by Ramamurthi and Gersho [Rama 86] has been used by Jacquin [Jacq 92], Beaumont [Beau 91] and Kaouri [Kaou 91]. Boss and Jacobs [Jaco 92, Boss 94] used a scheme similar to VQ codebook design but based on *archetype* blocks rather than specific codebook blocks. Using a set of 72 archetypes they found the scheme sufficiently flexible to permit the range block size to be increased to 8 by 8 and hence achieved a typical compression ratio of 16:1. They also reported that this was much faster than using a conventional classifier but the reconstructed image quality was only slightly worse.

A simpler but effective method has been used by Fisher [Fish 91] in which square image blocks are divided into four quadrants each of which has an associated mean and variance for the pixel intensity. The blocks can then always be oriented into one of 3 major classes and a further 24 minor classes depending on the relative means of the quadrants.

Jacquin chose the essentially 'brute-force' method of exhaustively testing all allowed transformations of all domain blocks. He modified this slightly by dividing the image into four quadrants and only testing domain blocks from the same quadrant as the range block to achieve a speed-up by a factor of four. Unfortunately, this is a most artificial restriction and, inevitably, excludes blocks which may have provided a good match.

Beaumont [Beau 91] chose a larger (12 x 12) domain blocks but still with 4 x 4 range blocks and tested domain blocks in a expanding spiral about the domain block centred on the range block. This is also an exhaustive search strategy but he truncated the search when a match was found to within a set distortion threshold.

The searching process is very similar to the problem of identifying the closest codebook vector in a VQ coder. This comparison and possible analogous solutions to the problem are considered in Section 7.7.

7.5.4 Storage of IFS codes

The storage requirement for an IFS-encoded image is essentially the product of the number of IFS codes (i.e. the number of range blocks) and the storage required for each code. Reduction of the number of codes is achieved by the partitioning strategy (see Section 7.5.1). However, the decoder has to be able to reconstruct the partitioning purely from the IFS-encoded data so additional information is required. This must either be implicit in the ordering of the IFS codes or provided explicitly in the form of domain position information.

In the most general form, from Eqn. 7.8, each IFS code would require at least eight coefficients, six to specify the affine transformation (a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2) and two to define the massic transformation (a_{33}, b_3) together with further values to define the domain block position (d_x, d_y). Still more values would be required if the domain blocks were allowed irregular sizes and shapes. In the theoretical derivation in Section 7.4 all of the coefficients were assumed to be real numbers but for an image defined on a discrete grid much more compact representations can be used. Using integer or fixed point number representations for 512 by 512 pixel images with 256 grey-levels the storage requirements for the coefficients are given in Table 7.2.

Coefficient	Storage / bits
Affine transformation coefficients a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2	16
massic transformation coefficients a_{33}, b_3	8
Domain block position d_x, d_y	8
Entire code	128

Table 7.2. General IFS code storage requirements

The total requirement of 128 bits for each code implies that such a scheme would have to use very large (at least 8 by 8) range blocks to achieve any useful compression. Practical schemes can achieve compression using moderate range block sizes by having much lower storage requirements for each IFS code. For example, in schemes such as Jacquin's [Jacq 92] the size of domain blocks is always some fixed multiple of the size

of the range blocks so the spatial contraction is constant and need not be stored. It is also usual to greatly restrict rotational transformations to just a small set of reflections and orthogonal rotations. Taken together these simplifications reduce the storage requirement for each IFS code to 16-24 bits.

Furthermore, experimental results derived here (see Section 7.7) indicate that the distribution of the values for all of the code coefficients is non-uniform so conventional entropy coding can be applied the coefficients to achieve a further storage reduction. The most significant saving is for the domain block positional information; many of these coefficients are small since the matching domain block is often found in the neighbourhood of the range block.

7.5.5 IFS image decoding

Most early work on IFS image compression assumed that Monte Carlo methods would be used for decoding (or *rendering* as it is frequently termed) the image from the IFS coefficients. For example, Barnsley described the random iteration algorithm (RIA) [Barn 88a] which is that used in the introduction to IFS in Section 7.3. Starting with an arbitrary initial point, randomly chosen transformations from the IFS are repeatedly applied to produce points which are drawn to the fixed point of the IFS, eventually rendering the whole of the attractor. While Monte Carlo rendering is extremely easy to implement it is most inefficient and being non-deterministic (i.e. there is no guarantee when and even if a particular pixel will be rendered) it is not suitable for use in decoding compressed images.

For decoding grey-scale images which have been IFS coded on a block-by-block basis, deterministic rendering is most appropriate. Since the range block partitioning ensures that the whole image is uniformly covered, and there is only one set of IFS coefficients per block, then each IFS code may simply be iterated in turn on its range block.

A practical method of speeding up the decoding process was proposed by Beaumont [Beau 91]. Instead of starting with an arbitrary image he chose to encode also a small image consisting of the mean grey levels of each range block. At the decoder this small image was interpolated to full size to give a starting image for the IFS iteration.

Kaouri [Kaou 91] modified the decoding process by using the results of those blocks which have already been transformed in the current iteration rather than substituting all the transformed blocks at once. This speeds up the convergence considerably.

Whichever rendering algorithm is used the decoding process is always much quicker than the coding. This is seen as one of the major virtues of IFS image coding.

7.5.6 Other IFS coding methods

More than 30 different IFS-based compression schemes have been published between 1990 and 1995. It is not feasible to cover them all here. In a recent review Saupe [Saup 94] identified several basic approaches which have met with varying degrees of success. Most of the published schemes are variations of the themes outlines in Section 7.4. However, there are some schemes, which while still being based on IFS coding, include more novel variations. For example, the Bath fractal transform requires no searching at all and the Beaumont method operates as a conventional IFS scheme but with the image data first transformed into the frequency domain.

7.5.6.1 Bath fractal transform

Monro and Dudbridge [Monr 92a, Monr 93a] devised a scheme called the Bath fractal transform (BFT) which avoids the computational difficulty of searching for block matches by self-tiling the blocks with reduced copies of themselves. This requires no search at all; instead the optimal mapping for each block is determined with a least-squares criterion. This is closer in spirit to classical self-similar fractals. The independence of the blocks leads to a coding scheme which is provably optimum per block but is inevitably non-optimum for the whole image.

7.5.6.2 Beaumont scheme

Beaumont [Beau 91] described a practical scheme with a frequency domain transformation to the image before coding since it is well known that the human visual system is far less sensitive to coding errors at some frequencies. Since there are already significant computational problems with IFS coding, Beaumont chose not to use the usual FFT or DCT transforms; instead he used the Hadamard Transform [Scha 89] which is much faster to implement but still extracts the frequency components of the block data. If the total coding time of the Beaumont algorithm is considered, the decision to use the Hadamard transform is questionable since this occupies only a small fraction of the total coding time. As was discussed in Chapter 2 there are many frequency-based transforms which could be used.

After the Hadamard transformation the block coefficients are re-ordered in a zig-zag path (see Fig. 7.7) so as to sort them in order of 'energy'. This is the same technique as is used in many transform coding methods such as the DCT coefficients in JPEG coding (See Section 2.8.4).

$$\begin{bmatrix} & 1 & 5 & 6 \\ 2 & 4 & 7 & 12 \\ 3 & 8 & 11 & 13 \\ 9 & 10 & 14 & 15 \end{bmatrix}$$

Fig. 7.7. Scan order of Hadamard coefficients

The error between the range and domain blocks is taken as the sum of the errors along this zig-zag path. The domain block is discarded as a match when the errors exceeds a pre-set threshold. The sorting ensures that poorly matching blocks are rejected early, thus reducing up the mean search time. The error thresholds are determined from tables derived from the DCT scaling coefficients for intensity in the JPEG standard [Penn 93].

7.6 Comparison of IFS coding and VQ

There is a clear similarity between IFS coding and VQ. In each case the source image is decomposed into a regular grid of blocks which are coded according to how they match a 'codebook' of blocks. The differences may be summarised as:

- In VQ the domain and range blocks are the same size; in a IFS the domain blocks are always larger.
- In VQ the range blocks are copies of the domain blocks; in an IFS each domain block undergoes a spatial scaling followed by an intensity scaling and offset to form the range block.
- In VQ the codebook is stored independently of the image to be coded; in an IFS the codebook is 'virtual'. It is not explicitly stored and only exists during the iterations of the decoder.
- In VQ the codebook is shared among many images; in an IFS each image has its own virtual codebook.

Considering more refined versions of VQ such as mean/shape VQ (See Section 6.5.1) the similarity becomes even greater leaving only the distinction between real and virtual codebooks.

The most significant problem with VQ and IFS coding methods is the computational complexity. Even after the potential virtual codebook of domain blocks is greatly reduced by only allowing isometric transformations and coarsely quantised massic transformations, it still comprises a large set of vectors which has to be tested to find a match for each range block.

7.7 Experimental comparison of IFS coding methods

Here, three coding methods have been implemented to form a comparison of the IFS compression strategies which may be adopted as outlined in Section 7.5.

The first method is based on the original scheme proposed by Jacquin. It uses fixed 4 by 4 range blocks and fixed 8 by 8 domain blocks. A block classification scheme based on Ramamurthi's method is used and block matching uses an exhaustive search of the entire domain block pool.

The second method is based on Beaumont's suggestions. It again uses fixed 4 by 4 range blocks but the domain blocks are increased to 12 by 12 pixels. All blocks are frequency transformed using the Hadamard transform. Block matching takes place in an expanding spiral centred on the range block and is potentially exhaustive, but may be truncated as soon as a 'close' match is found.

The third method uses code placed in the public domain by Fisher [Fish 94a]. It uses a quad-tree partition for the range blocks with each domain block size being twice the size of its range block. Block matching uses a simple classification scheme based on dividing each block into four quadrants and ordering the blocks by the relative means and variances of each quadrant.

The comparative compression and distortion performance of the three methods on the image test sets are shown in Table 7.3.

	Jacquin		Beaumont		Fisher	
Image	**Compres -sion ratio**	**SNR / dB**	**Compres -sion ratio**	**SNR / dB**	**Compres -sion ratio**	**SNR / dB**
Lena	7.4	31.5	8.0	32.1	9.2	36.4
Barbara	5.6	23.8	6.8	25.5	6.5	27.2
Boats	8.0	26.5	8.3	32.0	7.6	33.1
Mean of large image set	8.6	30	8.8	32.1	7.8	32.5
					13.7	30.1
					20.1	28.7

Table 7.3. Performance of representative IFS coding schemes

Neither Jacquin's nor Beaumont's methods which are based on fixed range block partitioning provide any reasonable parameters which may be used to trade off compression against distortion. However, Fisher's method which is based on quad-tree partitioning permits variation of the error threshold during quad-tree formation so that a mean performance over the primary image test set may be formed as shown in Fig 7.8.

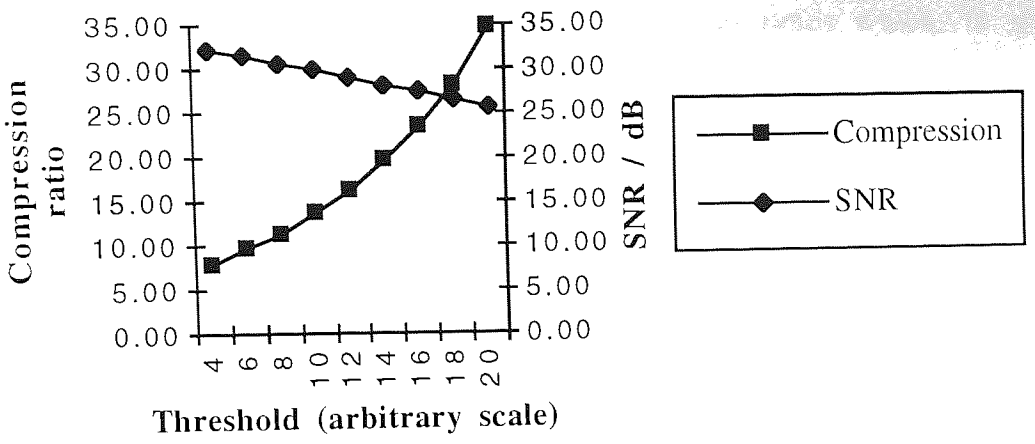


Fig. 7.8. Overall performance of Fisher's IFS coding scheme

Both Jacquin's and Beaumont's methods achieve only modest compression. It is inevitable from the choice of fixed 4 by 4 range blocks that compression ratios greater than 16:1 are unachievable but the mean compression ratios of 8.6:1 at only 30 dB SNR for Jacquin's method and 8.8:1 at 32 dB SNR for Beaumont's method are disappointing.

The main advantage of Beaumont' method appears to be the speed of decoding. Since the starting point of the decoding iterations is an image composed of the range block means, the algorithm converges very quickly. Typically only one or two iterations are required. Fig. 7.9 shows detail from the Lena image during decompression after (a) zero iterations (i.e. the range block means), (b) one iteration and (c) three iterations. There is little difference after the first iteration.

(a) 0 iterations **(b) 1 iteration** **(c) 3 iterations**

Fig. 7.9. IFS Convergence with Beaumont's method

Jacquin's method requires typically 8-10 iterations to converge. However, the decoding times are small compared with the coding times which, in these implementations, were excessive at over an hour per image.

In contrast, from these experimental results, the superiority of Fisher's method based on quad-tree partitioning is clear. Subjectively the results from Fisher's method are also good. Fig. 7.10 shows detail from the Lena image compressed at (a) 20:1 with a SNR of 30 dB and (b) compressed at 10:1 with a SNR of 35 dB. This performance compares favourably with JPEG compression.

(a) (b)

Fig. 7.10. Detail from Lena image compressed with Fisher's method

The computation time of Fisher's method is still high, requiring several minutes per image but is substantially less than for either of the other two methods.

7.7.1 Fast lookup algorithms for IFS coding

Here, the fast lookup techniques identified in Chapter 6 for vector quantisation lookup have been applied to all three of the IFS coding methods which have been implemented by classifying and sorting the range and domain blocks. It is not obvious that this is appropriate since the fast algorithms depend on a sorting of the codebook being performed in advance of the image compression. For IFS coding the codebook is only determined during image compression so the codebook sorting has to take place as part of the compression process.

It has been found that, for Jacquin's and Beaumont's methods, sorting by either Euclidean magnitude and Hilbert address achieves similar results to those achieved in VQ speed-up. i.e. the compression ratio and the SNR are essentially unchanged but a

considerable speed-up is achieved though this is not as pronounced as for VQ because of the overhead in sorting the virtual codebook.

Using these techniques both the Jacquin and Beaumont methods were speeded up by a factor of between 10 and 20. However, this only reduces the time taken to approximately the same as that required by the Fisher algorithm and their compression and distortion performance remains worse. Fisher's method is not significantly speeded up since the block classification scheme used results in relatively small numbers of domain blocks being searched for each match.

7.8 Summary

A method of compressing images using sets of fractal transformations known as iterated function systems has been presented. This scheme exploits the self-affine redundancy present in images of natural scenes. The transformations between areas of the image are combined into a set which represents a single mapping of the image onto itself. It is a fundamental result of iterated transform theory that if the transformation set is contractive then iteration of the set will result in the same image irrespective of the starting image. A second fundamental result known as the Collage Theorem states that the final image resulting from the iteration may be made as close as required to the original image simply be ensuring that the first iteration of the IFS codes (the collage) is close to the required image.

It has been shown that the key design issue for practical IFS coding schemes is the method by which the image is partitioned into domain and range blocks. In particular, the size of the range blocks determines the possible compression which can be achieved.

Three different methods of IFS compression have been implemented and compared. Experimental results indicate that the key to achieving useful compression with IFS coding is the use of an adaptive algorithm for selecting the range blocks. The quad-tree partitioning method has been shown to be effective and yields compression ratios and distortion figures which are comparable with the JPEG standard.

IFS coding has also been found to be one of a very small group of image compression methods which is capable of giving acceptable image quality at very high compression ratios (above 40:1 for example).

IFS coding has the unique property of display resolution independence. An IFS-compressed image can be displayed on any display without the need for interpolation. For example, a small part of an image may be scaled to fill the whole display without

the 'blockiness' which would result from any other method of image data storage. Whether the detail which is then 'revealed' was ever present in the original image is debatable. In many applications the fact that the scaled image appears superior subjectively is sufficient but in many applications the consequences of making financial or safety-critical decisions based on the images will make this unacceptable.

IFS image compression is inherently asymmetric in terms of computation for coding and decoding. In some applications this is not a disadvantage particularly if the decompression is so optimised that it is quicker than competitive methods. What is more significant is that if more time is spent on coding so as to achieve better quality it does not increase the decoding time.

While JPEG and MPEG have become *de facto* standards for still and motion image compression respectively, the attractive potential compression ratios offered by IFS, coupled with the subjectively superior images in the presence of considerable distortion mean that the future for IFS compression may still be promising. IFS coding includes some ideas which are radically different from conventional methods and this has led to some resistance to its adoption. Surveys [Saup 94, Jacq 93] indicate an ever-increasing interest in IFS coupled with an expanding publication base which could still lead to a preference for IFS coding in applications where the inherent flexibility of scaling with 'naturalness' of the reconstructed images is of paramount importance. Recently Microsoft have licensed ISI's fractal image compression software for use in their Encarta multimedia products because it offered the 'highest usable compression' on the market (i.e. compression ratio was considered more important than reconstructed image quality).

Chapter 8. Conclusions

8.1 Summary of thesis

This thesis has addressed the problem of compression of image data. It has been shown that, while considerable compression of image data is theoretically possible, it is difficult to achieve great compression of real data. It is clear that lossless compression is not very effective on image data but, since digital images are approximations of analogue data to start with, it is appropriate to apply lossy compression to such data. This requires an objective measure of the reconstructed image fidelity but it has been shown here that objective measures are not always consistent with subjective measurements.

Demand for a working standard has resulted in the JPEG lossy image compression standard which has been widely adopted and has been used here as a benchmark against which the novel schemes have been tested. The JPEG standard performs well at modest compression ratios but has some drawbacks.

- It is computationally demanding. There is still a requirement for conceptually and computationally simple compression techniques.
- At higher compression ratios the reconstructed image quality falls off rapidly. This is inherent in transform based schemes once the compression ratio is high enough to leave only a very small number of transform coefficients.
- It is resolution dependent. Any attempt to display the image at a resolution higher than the highest coded resolution still results in blockiness from the required pixel replication.

The existence of a standard has not reduced the research being applied to the problem of image data compression since there is still a demand for significantly better compression ratios and better reconstruction quality. In this thesis, fractals, which have proved to be an interesting and effective tool for analysing intractable problems in other domains, have been applied to image compression. Specifically, contributions have been made in the following areas.

8.1.1 Fractal dimension and complexity

It has been shown that the concept of a fractal dimension can be applied to real images and that it provides a natural and effective complexity measure which can be

used to predict the performance of both fractal and non-fractal image compression methods. Specifically, it has been demonstrated that entropy is an appropriate measure for lossless compression using conventional statistical methods, but for lossy compression, the fractal dimension of the data is a superior measure since advantage can be taken of knowledge of the complexity of the data at different scales.

8.1.2 Space-filling curve scanning

It has been shown that space-filling fractal curves provide a mechanism for applying 'divide-and-conquer' methods to complex problems in which iterative or recursive operations are required at a range of scales. The problem approached here is scanning of images so as to re-order the pixels for subsequent processing. Substantial experimental data has been generated to indicate that these curves do provide a superior scanning mechanism whenever one-dimensional compression methods such as run-length encoding or predictive coding are to be applied to the scanned data.

The few well-known space-filling curves have been generalised here to create many new curves together with a systematic classification scheme including hierarchical combinations of the curves. Using this system, space-filling scans may readily be generated for images of any size but it has been shown that some image resolutions, such as 512 by 512 pixels, are particularly suitable for decomposition into many different, regular curves. It has been shown that, if an image is scanned with many different space-filling curves, it is usually possible to find a scan which leads to significantly better image compression. Unusually, this increase in compression does not lead to a loss in image quality.

Space-filling curves have also been shown to provide an interesting alternative method to the most sophisticated current error-diffusion dithering algorithms. Superiority of the curves for dithering has not been conclusively demonstrated but some subjective testing has indicated a preference for images dithered in this way.

8.1.3 Fractal waveform coding

A method of waveform coding has been described which makes use of fractal geometry to approximate an image by a sequence of steps taken by a 'yardstick' in a process analogous to measuring a feature on a map with dividers. This technique has numerous parameters which may be set but, through extensive experimentation, the optimum settings for most of these have been determined (including the method of treating sharp edges and the method of interpolating the restored images). The

remaining parameters have been merged into a single control parameter similar to the JPEG quality value.

FWC has been shown to be simple to implement and to have a low computational complexity. It has been found that, in general, adding complexity to FWC does not improve its performance significantly. When used in conjunction with a space-filling scan, fractal waveform coding has been demonstrated to be an excellent low-complexity waveform coding method.

8.1.4 Fast lookup in vector quantisation

Vector quantisation is known to possess the potential to yield substantial image data compression but the distortion resulting from VQ data compression depends on the size of the vectors and the number of vectors in the codebook. High compression requires large vectors and low distortion requires large codebooks but the computational complexity of both codebook design and the vector lookup during the encoding phase depends exponentially on the size of the vectors and the size of the codebook.

In this thesis two new methods for reducing the computational complexity of vector lookup have been proposed. The first, searching only a subset of the codebook determined by the Euclidean magnitude of the vectors has been shown to be particularly simple yet very effective. Typically, distortion within less than 0.5 dB of that attained using exhaustive full search is achieved by searching only $1/16$ of the codebook, giving a speed-up factor of 16. The second method proposed uses Hilbert curves drawn in a number of dimensions equal to the vector size to generate an ordering of the vectors which also allows lookup to be performed on just a subset of the codebook. This method has also proved effective, but only for the largest vectors and codebooks does it match the performance of the Euclidean magnitude lookup.

8.1.5 Iterated function system coding

A comparative analysis of current developments in IFS image compression has been presented. It has been shown that the key design issue for practical IFS coding schemes is the method by which the image is partitioned into domain and range blocks. In particular, the size of the range blocks determines the possible compression which can be achieved.

Three different methods of IFS compression have been implemented and compared. Experimental results indicate that useful compression requires an adaptive algorithm

for selecting the range blocks. The quad-tree partitioning method has been shown to be effective, yielding compression ratios and distortion figures which are comparable with the JPEG standard.

8.2 Comparison of coding methods

It is difficult to encapsulate the relative merits of disparate image compression schemes in a single diagram. There is a fundamental trade-off between compression ratio and image quality which is displayed by almost all practical schemes. However, even within this framework, most methods include parameters which may be used to trade the time or cost of compression against one or both of compression and quality.

Furthermore, there are pitfalls in comparing experimental schemes with established schemes which have been highly optimised. Selection of the coefficients or values by the 'data modelling' stage of a compression process is usually followed by a 'back-end' coding stage based on entropy coding In a typical JPEG-compressed image as much as 20-30% of the storage saving is achieved by means of the highly optimised entropy coding of the DCT coefficients [Wall 91]. If the modelling stage is not fully effective in removing redundancy then this may be counteracted by entropy coding whereas a very effective modelling method will benefit little from 'back-end' entropy coding.

Taking into account these difficulties, Fig. 8.1 gives a general view of the relative performances of the image compression schemes discussed in this thesis.

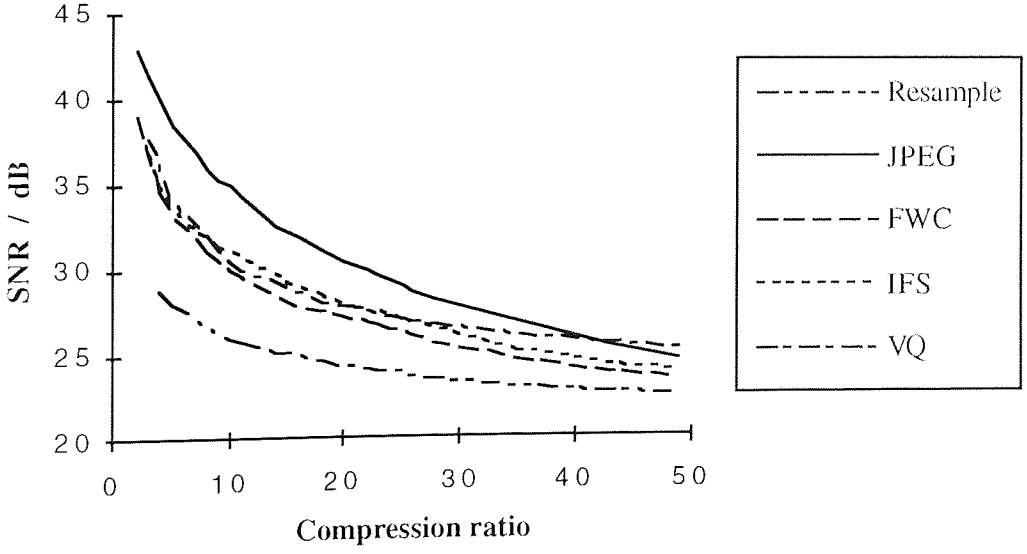


Fig. 8.1. Comparison of image compression methods

Viewed entirely dispassionately, it is remarkable how little difference there is between the simplest methods and the most complex. However, it should be noted that the distortion axis of Fig. 8.1 is logarithmic and that this graph does not take into account subjective assessment.

If low computational complexity is required, then fractal waveform coding together with a space-filling scan performs creditably. If computation time is relatively unimportant then, at the very highest compression ratios, vector quantisation is superior. This is achieved with large codebooks which, without the performance increasing methods introduced in this thesis, would not normally be feasible.

Also, such a blunt comparison does not give credit to features such as the unique display resolution independence property of IFS coding.

8.3 Conclusions

The long term goal of a practical image compression method which achieves significant (i.e. order of magnitude) improvement over existing methods has not been discovered here. However, results broadly comparable with existing methods have been achieved with unoptimised implementations of novel methods.

The problem of 'matching' or pattern recognition is central to this work. Comparing objects transformed in strange ways into different domains crops up repeatedly. It is a fundamental problem and one which requires vast computing power. Thus the efficiency of the matching process as well as the quality has to be examined. Inevitably this leads to a compromise. In this work the issue of speed in the matching process has been addressed in some detail. Central to the argument is the issue of examining or covering a very large 'space' of possibilities uniformly and of mapping between spaces of many dimensions to spaces of fewer dimensions (typically only one dimension). In one dimension many significant problems have well-known optimal solutions. It is appealing to consider how some of the daunting problems in many dimensions can be solved by first mapping to one dimension.

It is pleasing that the tool which provides the link between many dimensional spaces and one dimensional space turns out to be the space-filling curves first identified in the nineteenth century by some of the leading mathematicians of the time as pathological curves or 'monsters' not suitable for analysis by reasonable mathematicians. These are now recognised as some of the first fractal entities to be discovered.

Notwithstanding the positive comments made in each chapter about the improvements in data compression provided by the techniques identified here, it is apparent that, viewed entirely objectively, there is considerable convergence in the performance of all image compression methods. Given the general trade-off between compression ratio and distortion as exemplified by Fig. 8.1, it appears that the only truly dramatic increase in performance comes from the decision to adopt lossy rather than lossless data compression. The desired order-of-magnitude improvement in compression ratio with acceptable distortion does not seem to be achievable.

Since some extremely small image data sets, such as cartoon drawings of well-known sports stars and politicians, can still convey the required information, it becomes reasonable to consider precisely what is being compressed with an image compression system. Differences arise, depending on the nature of the image data. In some cases the meaning of the image is all-important and the subjective quality is immaterial. For example, is a military remotely piloted vehicle about to bomb the right target? In other cases, such as entertainment video, the information content may be vanishingly small but significant commercial reward or loss may rest on the perceived quality of the images.

8.4 Recommendations

Each of the topics discussed in this thesis could be explored further. Specific areas where development is required are:

- Identification of other computationally simple methods of measuring the fractal dimension of images to give accurate estimation of the compressibility of the images.
- Identification of methods for efficiently choosing the optimum space-filling scan of an image.
- Further development of the link between VQ and IFS coding. It may be possible to combine the best features of both methods by performing conventional VQ but using an optimum virtual codebook which is created during the compression process.
- Extension of the multi-dimensional mapping property of space-filling curves to other problems.

Looking further ahead, if the long-term objective of massive (100:1 or more) image compression ratios are to be achieved, it seems that significant developments are

required in image understanding and in the creation of coding models which relate more closely to physiological image perception. Greater understanding of the way in which humans perceive images should lead to methods of segmenting images according to features such as contours and texture. Features identified in this way could be coded as objects with a concise notations such as IFS codes.

8.5 Final comments

None of the methods described in this thesis has been developed to the level required for a standard method, yet comparable performance to the JPEG standard has been achieved. Future developments of any new scheme to compete with the standard will depend on the perceived potential for commercial exploitation. If a new technique is developed in secrecy, leading to a proprietary standard, the benefits of using it have to be substantial if it is to be adopted. On the other hand, unless the inventor is a major hardware or software vendor, the potential for profit from devising a new open standard is limited. For a proprietary standard such as ISI's fractal image compression to have been adopted by the most significant software vendor Microsoft in some of its major products is a remarkable achievement.

Without major development effort, it seems probable that the image compression schemes devised here will remain unexploited. However, some of the ideas, particularly the re-ordering and multi-dimensional mapping properties of the space-filling curves, are sufficiently general that they may find uses in other subject domains where large, complex data sets require analysis.

This thesis has brought together a number of distinct contributions applying different aspects of fractals to image data compression. It is hoped that future work will continue to explore the issues raised and help to integrate the abstract, mathematical view of fractals with the pragmatic demands made by modern image processing systems for powerful, flexible and efficient data compression.

References

Abut 82 — Abut H., Gray R. M., Rebolledo G., 'Vector quantisation of speech and speech-like waveforms', IEEE Trans. ASSP, Vol. ASSP-30, Jun. 1982, pp. 423-435.

Ahme 74 — Ahmed N., Natarajan T., Rao K. R., 'Discrete cosine transform', IEEE Trans. Comput., Vol. C-23, Jan. 1974, pp. 90-93.

Aho 95 — Aho A. V., Ullman J. D., 'Foundations of computer science', Computer Science Press, New York, 1995.

Ahum 93 — Ahumada A., 'Computational image-quality metrics: a review', SID '93, Dig. Tech. Papers, (Seattle, WA., May 1993), pp. 305-308.

Andr 75 — Andrews H. C., 'Two dimensional transforms', in 'Topics in applied physics: picture processing and digital filtering', Vol. 6, T. S. Huang (Ed.), Springer-Verlag, New York, 1975.

Ansa 92a — Ansari A. C., Gertner I., Zeevi Y. Y, Tanenhaus M. E., 'Image compression: wavelet-type transform along a generalized scan', Proc. SPIE, Vol. 1630, 1992, pp. 99-107.

Ansa 92b — Ansari A., Fineberg A., 'Image data ordering and compression using Peano scan and LOT', IEEE Trans. Consumer Electronics, Vol. 38, No. 3, Aug. 1992, pp. 436-445.

Arps 94 — Arps R. B., Truong T. K., 'Comparison of international standards for lossless still image compression', Proc. IEEE, Jun. 1994, Vol. 82, No. 6, pp. 889-899.

Atha 94 — Athanas P. M., Abbott A. L., 'Real-time image processing on a custom computing platform', Computer, Vol. 28, No. 2, 1994, pp. 16-25.

Bake 82 — Baker R. L., Gray R. M., 'Image compression using non-adaptive spatial vector quantization', Proc. 16th Asilomar Conf. Circuits, Syst., Comput., Oct. 1982, pp. 55-61.

Bake 84 — Baker R. L., 'Vector quantization of digital images', Ph. D. dissertation, Dept. of Electrical Engineering, Stanford University, Stanford, CA., June 1984.

Barn 85a — Barnsley M. F., Ervin V., Hardin D., Lancaster J., 'Solution of an inverse problem for fractals and other sets', Proc. National Academy of Science, Vol. 83, Apr. 1985, pp. 1975-1977.

Barn 85b — Barnsley M. F., Demko S., 'Iterated function systems and the global construction of fractals', Proc. Royal Soc. London, Vol. A399, 1985, pp. 243-275.

Barn 88a — Barnsley M. F., Sloan A. D., 'A better way to compress images', Byte Magazine, Jan. 1988, pp. 215-223.

Barn 88b — Barnsley M. F., 'Fractals everywhere', Academic Press, New York, 1988.

Barn 88c — Barnsley M. F., Jacquin A. E., 'Application of recurrent iterated function systems to images', Proc. SPIE, Vol. 1001, Visual Communications and Image Processing '88, pp. 122-131.

Barn 91 — Barnsley M. F., Sloan A., 'Method and apparatus for processing digital data', US Patent No. 5065447.

Barn 92 — Barnsley M. F., Hurd L., 'Fractal image compression', A. K. Peters, Wellesley, 1992.

Beau 90 — Beaumont J. M., 'Advances in block-based fractal coding of still pictures', IEE Coll. Digest, No. 171, 1990, pp 3/1-3/6.

Beau 91 — Beaumont J. M., 'Image data compression using fractal techniques', BT Technology Journal, Vol. 9, No. 4, Oct. 1991, pp. 93-109.

Bei 85 — Bei C. D., Gray R. M., 'An improvement of the minimum distortion encoding algorithm for vector quantization', IEEE Trans. Comm., Vol. COM-33, 1985, pp. 1132-1133.

Berg 71 — Berger T., 'Rate-distortion theory', Prentice-Hall, Englewood Cliffs, N. J., 1971.

Bial 69 — Bially T., 'Space-filling curves: their generation and their application to bandwidth reduction', IEEE Trans. Inf. Theory., Vol. IT-15, No. 6, 1969, pp. 658-664.

Boss 94 — Boss R. D., Jacobs E. W., 'Archetype classification in an iterated transformation image compression algorithm', in Fisher Y. (Ed.), 'Fractal Image Compression', Springer-Verlag, New York, 1995.

Bour 92 — Bourbakis N., Alexopolous C., 'Picture data encryption using scan patterns', Pattern Recognition, Vol. 25, No. 6, 1992, pp. 567-581.

Brig 74 — Brigham E. O., 'The fast Fourier transform', Prentice-Hall, Englewood Cliffs, N. J., 1974.

Butz 71 — Butz A. R., 'Alternative algorithm for Hilbert's space-filling curve', IEEE Trans. Computers, Vol. C-20, Apr. 1971, pp. 424-426.

Buzo 80 — Buzo A., Gray A. H., Gray R. M., 'Speech coding based upon vector quantisation', IEEE Trans. ASSP, Vol. ASSP-28, Oct. 1980, pp. 562-574.

Chen 86 — Cheng D., Gersho A., 'A fast codebook search algorithm for nearest-neighbour pattern matching', Proc. ICASSP 86, Tokyo, 1986, pp. 265-268.

Chen 92 — Chen J.-S., 'A comparative study of digital halftoning techniques', Proc. IEEE 1992 National Aerospace and Electronics Conference, NAECON 1992, Dayton, OH, Vol. 3, pp. 1139-45.

Cher 91 — Cherbit G. (Ed.), 'Fractals, non-integral dimensions and applications', John Wiley & Sons Ltd., Chichester, UK, 1991.

Clar 86 — Clarke K. C., 'Computation of the fractal dimension of topographic surfaces using the triangular prism method', Computers and Geosciences, Vol. 12, No. 5, 1986, pp. 713-722.

Cole 85a — Cole, A. J., 'A note on Peano polygons and Gray codes', Int. J. Comp. Maths, Vol. 18, 1985, pp. 3-13.

Cole 85b — Cole A. J., 'Multiple radix arithmetic and computer graphics', Bulletin of the Inst. of Mathematics and its Applications', Vol. 21, No. 11, Nov. 1985, pp. 431-442.

Cole 86a — Cole A. J., 'Direct transformations between sets of integers and Hilbert polygons', Int. J. Comp. Math., Vol. 20, No. 2, 1986, pp. 115-122.

Cole 86b — Cole A. J., 'Pure mathematics applied (scanning theory of bit-mapped monitors)' , University Computing, 1986, Vol. 8, pp. 2-7.

Cole 87 — Cole A. J. 'Compaction techniques for raster scan graphics using space-filling curves', The Computer Journal, Vol. 30, No. 1, 1987, pp 87-92.

Cole 90 — Cole A. J., 'Naive halftoning', Proc. CG International '90. Computer Graphics Around the World, Singapore, Jun. 1990, pp. 203-22.

Cole 91 — Cole A. J., Halftoning without dither or edge enhancement', Visual Computer, Vol. 7, No. 5-6, 1991, pp. 232-46.

Cosm 94 — Cosman P. C., Gray R. M., Olshen R. A., 'Evaluating quality of compressed medical images: SNR, subjective rating and diagnostic accuracy', Proc. IEEE, Vol. 82, No. 6, Jun. 1994, pp. 914-932.

Cutl 71 — Cutler C. C., 'Delayed encoding: stabiliser for adaptive coders'. IEEE Trans. Comm., Vol. COM-19, Dec. 1971, pp. 898-906.

Deni 91 — Denis T., 'Practical considerations in image processing', in Pearson D. (Ed.), 'Image processing', McGraw-Hill (UK), 1991.

Dett 92 — Dettmer R., 'Form and functions: fractal-based image compression', IEE Review, Sep. 1992, pp. 323-327.

Drew 91 — Drewery J. O., 'Image scanning using a fractal curve', BBC Research Digest, 1991/4.

Elli 83 — Elliot D. F., Rao K. R., 'Fast transforms, algorithms and applications', New York, Academic Press, 1983.

Falc 90 — Falconer K. J., 'Fractal geometry: mathematical foundations and applications', John Wiley & Sons Ltd, Chichester, 1990.

Fax3 — CCITT (1988) , Vol. VII, fascicle VII.3, Recommendation T.4.

Fax4 — CCITT (1988), Vol. VII , fascicle VII.3, Recommendation T.6.

Fisc 84 — Fischer T. R., 'Vector quantisation of memoryless Gaussian, gamma, and Laplacian sources', IEEE Trans. Commun., Vol. COM-32, Sep. 1984, pp. 1065-1069.

Fisc 89 — Fischer T. R., 'Geometric source coding and vector quantisation', IEEE Trans. IT, Vol. 35, No. 1, Jan. 1989, pp. 137-145.

Fish 86 Fisher A. J., 'A new algorithm for generating Hilbert curves', Software - Practice and experience, Vol. 16, No. 1, Jan. 1986, pp. 5-12.

Fish 91 Fisher Y., Jacobs E. W., Boss R. D., 'Fractal image compression using iterated function systems', Technical Report 1408, Naval Ocean Systems Centre, San Diego, CA, 1991.

Fish 94a Fisher Y. (Ed.), 'Fractal Image Compression', Springer-Verlag, New York, 1994.

Fish 94b Fisher Y., Menlove S., 'Fractal encoding with HV partitions', in Fisher Y. (Ed.), 'Fractal Image Compression', Springer-Verlag, New York, 1994.

Flor 85 Flory R. E., 'Image acquisition technology', Proc. IEEE, Vol. 73, Apr. 1985, pp. 613-637.

Floy 75 Floyd, R. W., Steinberg L., 'Adaptive algorithm for spatial grey scale', SID Int. Sym. Digest of Tech. Papers, 1975, pp. 36-37.

Font 81 Fontana R. J., Gray R. M., Kieffer J. C., 'Asymptotically mean stationary channels', IEEE Trans. IT, Vol. IT-27, May 1981, pp. 308-316.

Gell 94 Gell-Mann M., 'The quark and the jaguar: adventures in the simple and the complex', Little, Brown & Co., 1994.

Goel 88 Goel B. D., Kwatra S. C., 'Data compression algorithm for colour images based on run-length coding and fractal geometry', Proc. IEEE Conf. on Communications 1988: Digital Technology - Spanning the universe, Philadelphia, USA, Jun. 12-15 1988, pp. 1253-1256.

Gold 81 Goldschlager, L. M., 'Short algorithms for space-filling curves', Software - Practice and Experience, Vol. 11, 1991, p. 99.

Gonz 92 Gonzalez R. F., Woods R. E., 'Digital Image Processing', Addison-Wesley, Reading, MA, 1992.

Gray 84 Gray R. M., 'Vector Quantization', IEEE ASSP Magazine, Apr. 1984, pp. 4-29.

Grif 85 Griffiths, J. G, 'Table-driven algorithms for generating space-filling curves', Computer-aided Design, Vol. 17, 1985, pp. 37-41.

Grif 86 Griffiths, J. G ., 'An algorithm for displaying a class of space-filling curves', Software - Practice and Experience, Vol. 16, No. 5, May 1986, pp. 403-411.

Hilb 91 Hilbert D., 'Uber die stetige Abbildung einer Linie auf ein Flaechenstuck', ('On a continuous line which covers part of a surface'), Mathematische Annalen, 38 (1891), pp. 459-460.

Hsie 91 Hsiegh C., Lu P., Chang J, 'Fast codebook generation algorithm for vector quantization of images', Patt. Rec. Lett., Vol. 12, 1991, pp. 605-609.

Huff 52 Huffman D. A.., 'A method for the construction of minimum-redundancy codes', Proc. IRE, Vol. 40, No. 9, Sep. 1952, pp 1098-1101.

Hutc 81 Hutchinson J., 'Fractals and self-similarity', Indiana Univ. J. Math., Vol. 30, 1981, pp. 713-747.

Imag 94 Imagemagick User Guide, John Cristy, E. I. du Pont De Nemours & Co. Inc., 1994.

Jaco 92 Jacobs E. W., Fisher Y., Boss R. D., 'Image Compression: A study of the iterated transform method', Signal Processing, Vol. 29, No. 3, Dec. 1992, pp. 251-263.

Jacq 89 Jacquin A. E., 'A fractal theory of iterated Markov operators with applications to digital image coding', Ph. D. dissertation, Georgia Tech., USA, 1989.

Jacq 92 Jacquin A. E., 'Image coding based on a fractal theory of iterated contractive image transformations', IEEE Trans. Image Processing, Vol. 1, No. 1, Jan. 1992, pp. 18-30.

Jacq 93 Jacquin A. E. 'Fractal image coding: a review', Proc. IEEE, Vol. 81, No. 10, Oct. 1993, pp. 1451-1465.

Jain 81 Jain A. K., 'Image data compression: a review', Proc. IEEE, Vol. 69, No. 3, Mar. 1981, pp. 349-389.

Jain 89 Jain, A. K., 'Fundamentals of digital image processing', Prentice-Hall, Englewood Cliffs, NJ, 1989.

Jarv 76 Jarvis J. F., Judice C. N., Ninke W. H., 'A survey of techniques for the display of continuous tone pictures on bilevel displays', Computer Graphics and Image Processing, Vol. 5, 1976, pp. 13-40.

Jaya 84 Jayant N. S., Noll P., 'Digital Coding of waveforms', Prentice-Hall, Englewood Cliffs, N.J., 1984.

Juan 82 Juang B., Gray A. H., 'Multiple stage vector quantization for speech coding', Proc. ICASSP 82, Paris, May 3-5 1982, Vol. 1, pp. 597-600.

Kama 93 Kamata S., Perez A., Kawaguchi E. A, 'Computation of Hilbert's space-filling curves in N-dimensional space', Transactions of the Institute of Electronics, Information and Communication Engineers D-II, Vol. J76D-II, No. 3, 1993, pp. 797-801, (in Japanese).

Kaou 91 Kaouri H. A., 'Fractal coding of still images', Proc. 6th IEE Conf. on Digital Processing of Signals in Communications, Loughborough, U.K., Sep. 1991, pp. 235-239.

Kaye 89 Kaye B. H., 'A random walk through fractal dimensions', VCH Publishers, New York, 1989.

Kirk 83 Kirkpatrick S., Gelatt C. D., Vecchi M. P., 'Optimization by simulated annealing', Science, Vol. 220, No. 4598, 1983, pp. 671-680.

Knut 76 Knuth D. E, 'Big omicron and big omega and big theta', SIGACT News, Vol. 7, No. 2, Apr-Jun 1976, pp. 18-23.

Knut 87 — Knuth D., 'Digital halftones by dot diffusion', ACM Trans. on Graphics, Vol. 6, No. 4, Oct. 1987, pp. 245-273.

Komi 94 — Kominek J., 'Fractal image compression frequently-asked questions', comp.compression.research Usenet Newsgroup, 1994.

Kret 75 — Kretz F., 'Subjectively optimal quantisation of pictures', IEEE Trans. Comm., Vol. COM-23, Nov. 1975, pp 1288-1292.

Laem 67 — Laemmel A. E., 'Dimension reducing mappings for signal compression', Microwave Research Inst., Polytechnic Inst. Brooklyn, Rept. R-632-57, 1967.

Laur 85 — Laurini, R. L., 'Graphical data bases built on Peano space-filling curves', Proc. Eurographics 85, Nice, Sep. 9-13 1985, pp. 327-338.

Lee 86 — Lee H. J., Lee D. T., 'A gain-shape vector quantiser for image coding', Proc. ICASSP 86, Tokyo, 1986, pp. 141-144.

LeGa 91 — Le Gall D., 'MPEG: A video compression standard for multimedia applications, Comm. ACM, Vol. 34, No. 4, 1991, pp. 47-58.

Lei 77 — Lei T. R., Scheinberg N., Schilling D. L., 'Adaptive delta modulation systems for video encoding', IEEE Trans. Comm., Vol. COM-25, Nov. 1977, pp. 1302-1314.

LeMe 91 — Le Mehaute A., 'Fractals, materials and energy', in Cherbit G. (Ed.), 'Fractals, non-integral dimensions and applications', John Wiley & Sons Ltd., Chichester, UK, 1991.

Lemp 86 — Lempel A., Ziv J., 'Compression of two dimensional data', IEEE Trans. Information Theory, Vol. IT-22, 1986, pp. 75-81.

Leung 94 — Leung S. W., Zhu Y. S., Yim F., 'Implementation of non-uniform sampling delta modulation (NS-DM)', Proc. SICE '94, 33rd SICE Annual Conference, Tokyo, 26-28 Jul. 1994, pp. 817-819.

Lim 90 — Lim J. S., 'Two-dimensional signal and image processing', Prentice-Hall, Englewood Cliffs, N. J., 1990.

Lind 80 — Linde Y., Buzo A., Gray, R. M., 'An algorithm for vector quantiser design', IEEE Trans. Comm., Vol. COM-28, 1980, pp. 84-95.

Linn 88 — Linnainmaa S ,'New efficient representation of photographic images with restricted number of gray levels', Proc. 9th Int. Conf. on Pattern Recognition (IEEE), Rome, Nov. 1988, Vol. 1, pp. 143-145.

Liou 91 — Liou M., 'Overview of the px64 kbit/s video coding standard', Communications of the ACM, Vol. 34, No. 4, 1991, pp. 59-63.

Madd 86 — Maddox J., 'Gentle warning on fractal fashions', Nature, Vol. 322, Jul. 23, 1986, p. 303.

Mand 67 — Mandelbrot B. B., 'How long is the coastline of Great Britain, statistical self similarity and fractional dimension', Science, Vol. 155, 1967, pp. 636-638.

Mand 82 — Mandelbrot B. B., 'The fractal geometry of nature', W. H. Freeman and Co., San Francisco, 1982.

Marm 86 — Marmolin H., 'Subjective MSE measures', IEEE Trans. Syst., Man, Cyb., Vol. SMC-16, May/Jun 1986, pp. 486-489.

Marr 82 — Marr D., 'Vision. A computational investigation into the human representation of visual information', W. H. Freeman and Co., New York, 1982.

Max 60 — Max J. 'Quantizing for minimum distortion', IEEE Trans. Information Theory, Vol. IT-6, Mar. 1960, pp. 7-12.

Meak 86 — Meakin P., 'A new model for biological pattern formation', J. Theoretical Biology, Vol. 118, 1986, pp. 101-113.

Memo 95 — Memon N. D., Sayood K., Magliveras S. S, 'Lossless image compression with a codebook of block scans', IEEE J. on Selected Areas in Communications, Vol. 13, No. 1, Jan. 1995, pp. 24-30.

Mill 92 — Millar R. J., Nicholl, P. N., 'A Hilbert polygon directed raster scan display', Proc. International Conference on Image Processing and its Applications, Maastricht, NL, (7-9 April 1992), pp. 357-360.

Mogh 91 — Moghaddam B., Hintz K. J ., Stewart C. V., 'Space-filling curves for image compression', Proc. SPIE, Vol. 1471, Sadjadi F. A. (Ed.), Conf. on Automatic Object Recognition, Orlando, Fl., 1991, Ch. 42, pp. 414-421.

Monr 92a — Monro D. M., Dudbridge F., 'Fractal approximation of image blocks', Proc. ICASSP 92, San Francisco, 23-26 Mar. 1992, Vol. 3, pp. 485-488.

Monr 92b — Monro D. M., Dudbridge F., 'Fractal block coding of images', Electronics Letters, Vol. 28, May 1992, pp. 1053-1055.

Monr 93a — Monro D. M., 'A hybrid fractal transform', Proc. ICASSP 1993, Minneapolis, USA, 23-30 Apr. 1993, Vol. 5, pp. 169-172.

Monr 93b — Monro D. M., 'Class of fractal transforms', Electronics Letters, Vol. 29, No. 4, 18 Feb. 1993, pp. 362-363.

Mura 82 — Murakami T., Asai K., Zamakazi E., 'Vector quantiser of video signals', Electronics Letters, Vol. 7, Nov. 1982, pp. 1005-1006.

Nasr 88 — Nasrabadi N. M., King R. A., 'Image coding using vector quantization : a review' , IEEE Trans. Comm., 1988, Vol. COM-36, pp. 957-971.

Nels 92 — Nelson M., 'The data compression book', M&T Books, 1992.

Nibl 86 — Niblack W., 'An introduction to digital image processing', Prentice-Hall, Englewood Cliffs, NJ, USA, 1986.

Null 71 — Null A., 'Space-filling curves, or how to waste time with a plotter', Software - Practice and Experience, Vol. 1, 1971, pp. 403-410.

Oliv 86 — Oliveri F., Conte G., Guglielmo M., 'A technique using a one-dimensional mapping for vector quantisation of images', Proc. ICASSP 86, Tokyo, 1986, pp. 149-152.

Pali 89 — Paliwal K. K., Ramasubramanian V., 'Effect of ordering the codebook on the efficiency of the partial search distance algorithm for vector quantization', IEEE Trans. Comm., Vol. COM-37, No. 5, May 1989, pp. 538-540.

Papo 84 — Papoulis A., 'Probability, random variables and stochastic processes', McGraw-Hill, New York, 1984.

Patt 93 — Patterson D. A., Hennessey J. L., 'Computer organization and design: the hardware/software interface', Morgan Kaufmann, San Mateo, Calif., 1993.

Paz 76 — Paz I. M., Collins G. C., Batson B. H., 'A tri-state delta modulator for run-length encoding of video'. Proc. Nat. Telecomm. Conf. (Dallas, Texas), Vol. 1, Nov. 1976, pp. 6.3.1 - 6.3.6.

Pean 90 — Peano G., 'Sur une courbe, qui remplit toute une aire plaine', ('On a continuous line which covers part of a surface'), Mathematische Annalen, 36 (1890), pp 157-160.

Pear 90 — Pearson D. E., Hanna E., Martinez K., 'Computer-generated cartoons', in 'Images and understanding', Barlow H. B. *et al* (Eds.), Cambridge University Press, Cambridge, UK, 1990.

Peit 88 — Peitgen H. O., Saupe D., 'The science of fractal images', Springer-Verlag, New York, 1988.

Peit 92 — Peitgen H. O., Saupe D., Jurgens H., 'Chaos and fractals: new frontiers of science', Springer-Verlag, New York, 1992.

Penn 88 — Pennebaker W. B., Mitchell J. L., 'An overview of the basic principles of the Q-coder adaptive binary arithmetic coder', IBM Journal of research and development, Vol. 32, No. 6, Nov. 1988, pp. 771-726.

Penn 93 — Pennebaker W. B., Mitchell J. L., 'JPEG still image data compression standard', Van Nostrand Reinhold, 1993.

Pent 84 — Pentland A. P., 'Fractal-based description of natural scenes', IEEE Trans. Patt. Anal. and Mach. Int., Vol. 6, No. 6, 1984, pp. 661-674.

Pere 91 — Perez A., Kamata S., Kawaguchi E., 'Hilbert scanning arithmetic coding for multispectral image compression', Proc. Applications of digital image processing 14, Tescher A. G. (Ed.), San Diego, Calif., 22-26 Jul. 1991, pp. 354-361.

Piet 86 — Pietronero L., Toasatti E. (Eds.), 'Fractals in physics', Elsevier, Amsterdam, 1986.

Prat 69 — Pratt W. K, Andrews H. C., Kane J., 'Hadamard transform image coding', Proc. IEEE, Vol., 57, No. 1, 1969, pp. 58-68.

Prat 78 — Pratt W., 'Digital image processing', Wiley, New York, 1978.

Prov 94 Provine J. A., Rangayyan R. M., 'Lossless compression of Peano scanned images', Jnl. of Electronic Imaging, Vol. 3, No. 2, pp. 176-81.

Prus 90 Prusinkiewicz P., Lindenmayer A., Fracchia F. D., 'Synthesis of space-filling curves on the square grid', Proc. of the First IFIP Conference on Fractals in the Fundamental and Applied Sciences, Lisbon, 6-8 June 1990, pp. 341-366.

Quin 89 Quin A., Yanagisawa, Y., 'On data compaction of scanning curves', Computer Journal, Vol. 32, No. 6, Dec. 1989, pp. 563-566.

Rama 86 Ramamurthi B., Gersho A., 'Classified vector quantisation of images', IEEE Trans. Commun., Vol. COM-34, No. 11, 1986, pp. 1105-1115.

Robe 62 Roberts L. G., 'Picture coding using pseudo-random noise', IRE Trans. Inf. Theory, Vol. IT-8, No. 2, Feb. 1962, pp. 145-154.

Sabi 84 Sabin M. J., Gray R. M., 'Product code vector quantisers for waveform and voice coding', IEEE Trans. ASSP, Vol. ASSP-32. Jun. 1984, pp. 474-488.

Saga 91 Sagan H., 'Some reflections on the emergence of space-filling curves', J. Franklin Institute, 1991, Vol. 328, No. 4, pp. 419-430.

Same 84 Samet H., 'The quadtree and related hierarchical data structures', ACM Computer Surveys, Vol. 16, No. 2, Jun. 1984, pp. 187-260.

Samp 93 Sampath A., Ansari A. C., 'Combined Peano scan and VQ approach to image compression', Proc. SPIE, Vol. 1903, 1993, pp. 175-186.

Saup 89 Saupe D., 'Random fractals in image synthesis', in Cherbit G. (Ed.), 'Fractals, non-integral dimensions and applications', John Wiley & Sons Ltd., Chichester, UK, 1991.

Saup 94 Saupe D. 'A review of the fractal image compression literature', Computer Graphics, Vol. 28, No. 4, Nov. 1994, pp. 268-276.

Scha 89 Schalkoff R. J., 'Digital image processing and computer vision', Wiley, New York, 1989.

Shan 48 Shannon C. E., 'A mathematical theory of communication', Parts 1 and 2, Bell Syst. Tech. J., Vol. 27, July 1948, pp. 379-423, pp. 623-656.

Sier 12 Sierpinski W., 'Sur une nouvelle courbe qui remplit toute une aire plaine', ('On a new curve which fills part of a surface'), Bull. Acad. Sci. Cracovie, Serie A, 1912, pp. 462-478.

Skar 89a Skarbek W., Agui T., Nakajima M, 'Compression of dithered binary images using Hilbert scan', Transactions of the Institute of Electronics, Information and Communication Engineers, Vol. E72, No. 11, Nov. 1989, pp. 1235-1242.

Skar 89b Skarbek W., Agui T., Nakajima M., 'Software tools for Hilbert scan of large images', Trans. of the Institute of Electronics, Information and Communication Engineers, Vol. E72, No. 5, May 1989, pp. 561-564.

Skar 90 Skarbek W., Agui T., Nakajima M., 'Generating breadth-first expression for gray scale quadtree', Journal of Information Processing, Vol. 13, No. 3, 1990, pp. 355-360.

Sore 88 Sorek N., Zeevi Y. Y., 'On-line visual data compression along a one-dimensional scan', Proceedings of the SPIE - The International Society for Optical Engineering, Vol. 1001, pp. 764-770.

Stuc 81 Stucki P., 'MECCA - A multiple error correcting computation algorithm for bilevel image hardcopy reproduction', Research Report RZ1060, IBM Research Laboratory, Zurich, Switzerland, 1981.

Ulic 87 Ulichney R., 'Digital halftoning', MIT Press, Cambridge, Mass., 1987.

Urba 92 Urban S. J., 'Review of standards for electronic imaging for facsimile systems', Jnl. of Electronic Imaging, Vol. 1, No. 1, Jan. 1992, pp. 5-21.

Velh 91 Velho L., Gomes J. M. 'Digital halftoning with space filling curves', Graphics, Vol. 25, No. 4, Jul. 1981, pp. 81-90.

Wait 90 Waite J., 'A review of iterated function system theory for image compression', IEE Colloquium Digest No. 1990/171, Dec. 1990, pp. 1/1-1/8.

Wala 86 Walach E., Karnin E., 'A fractal-based approach to image compression', Proc. ICASSP 86, Tokyo, 1986, Vol. 1, pp. 529-532.

Wall 91 Wallace, G. K., 'The JPEG still picture compression standard', Communications of the ACM, Apr. 1991, Vol. 34 , No. 4, pp. 30-44.

Warn 90 Warner W. C., 'Compression algorithms reduce digitised images to manageable size', EDN, Jun. 21, 1990, pp. 203-212.

Welc 84 Welch T., 'A technique for high-performance data compression', IEEE Computer, Vol. 17, No. 6, Jun. 1984, pp. 8-19.

Wilt 92 Wilton A. P., Carpenter G. F., 'Fast search methods for vector lookup in vector quantisers', Electronics Letters, Vol. 28, No. 5, pp. 2311-2312.

Wirt 76 Wirth N, 'Algorithms + Data Structures = Programs', Prentice-Hall, Englewood Cliffs, N. J., 1976.

Witt 83 Witten I. H.. Wyvill B., 'On the generation and use of space-filling curves', Software - practice and experience, Vol. 13, 1983, pp. 519-525.

Witt 87 Witten I. H., Radford N., Cleary J, 'Arithmetic coding for data compression', Comm. of the ACM, Vol. 30, No. 6, 1987, pp. 520-540.

Wyvi 91 Wyvill G., McNaughton C. , 'Three plus five makes eight: a simplified approach to halftoning', Proc. Scientific Visualization of Physical Phenomena, 26-28 June 1991, Cambridge, MA, USA, pp. 379-392.

Wyze 67 Wyzecki G. W., Stiles W. S., 'Color Science', John Wiley, New York, 1967.

Yang 88 Yang K., Wu L., Mills M., 'Fractal based image coding scheme using Peano scan', Proc. IEEE Int. Symp. on circuits and systems, Espoo, Finland, 7-9 June 1988, Vol. 3, pp. 2301-2304.

Zhan 93 Zhang Y., Webber R. E., 'Space diffusion: an improved parallel halftoning technique using space-filling curves, Proc. Computer Graphics (Anaheim CA, USA, Aug. 1993), pp. 305-312.

Zhu 94 Zhu Y. S., Leung, S. W., 'A non-uniform sampling delta modulation technique', Proc. 1994 IEEE Region 10 9th Annual International Conf. Theme: Frontiers of Computer Technology, 1994, Vol. 2, pp. 708-712.

Zipf 49 Zipf G. K., 'Human behaviour and the principle of least effort', Cambridge MA, Addison-Wesley (Hefner reprint), 1949.

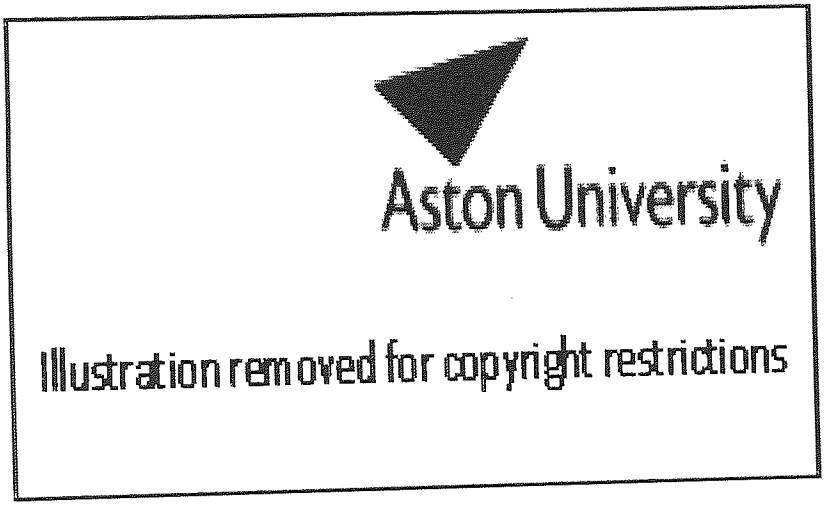
Ziv 77 Ziv J., Lempel A., 'A universal algorithm for sequential data compression', IEEE Trans. IT, Vol. 23, No. 3, May 1977, pp. 337-343.

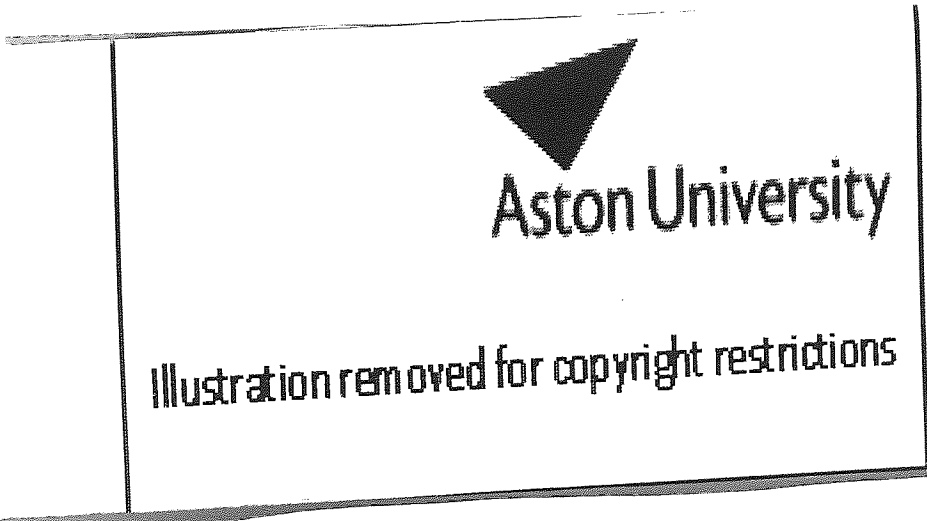
Ziv 78 Ziv J., Lempel A., "Compression of individual sequences via variable-rate coding', IEEE Trans. IT, Vol. 24, No. 5, Sep. 1978, pp. 530-536.

Appendix A - Test Images

A.1 Primary (small) image test set

Image Name	Notes
LENA	A 'classic' test image. Mostly easy with some regions of fine detail. This version is derived from the original RGB format.
BOATS	An intermediate difficulty image. The sharp edges of the boat masts need to be picked out from the plain background. Legibility of the boat name is also a good test.
BARBARA	A difficult image to compress. There are large regions of fine detail with sharp edges.

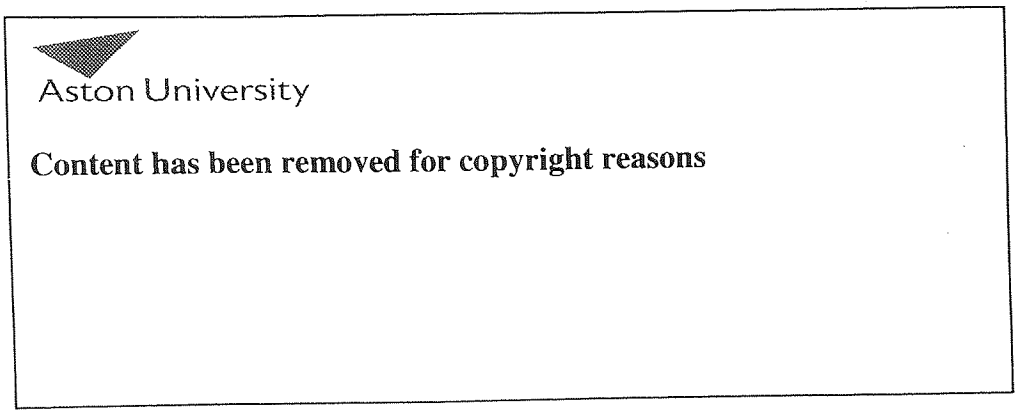


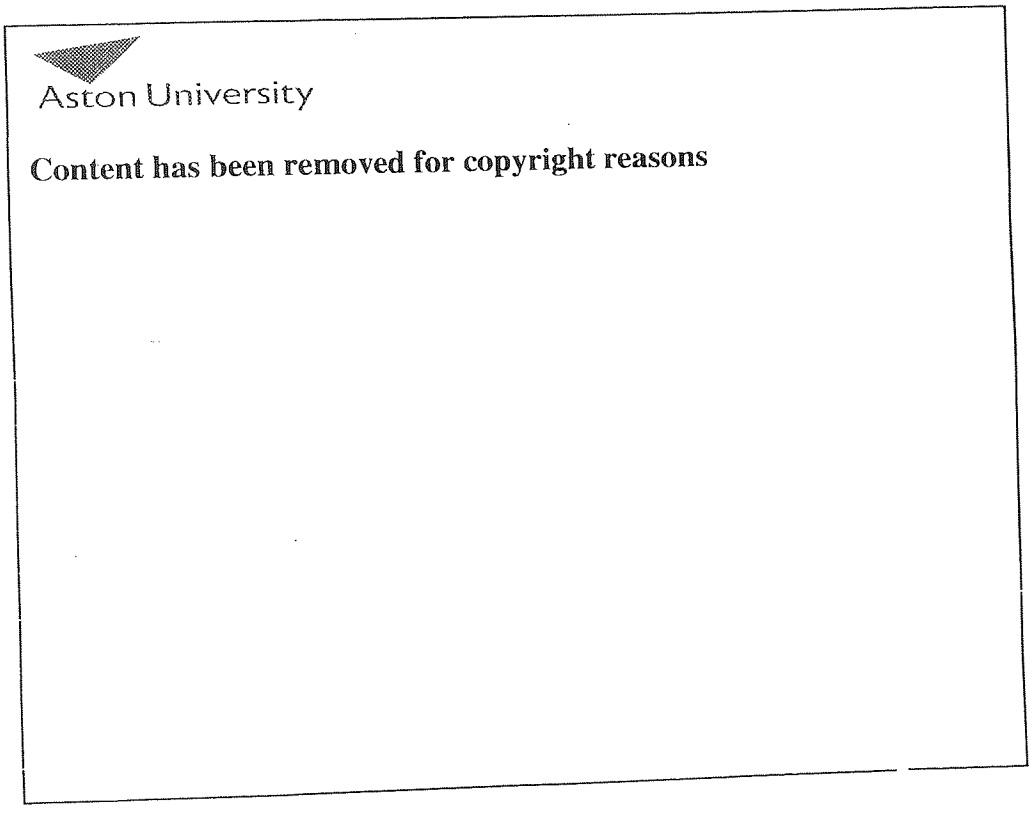


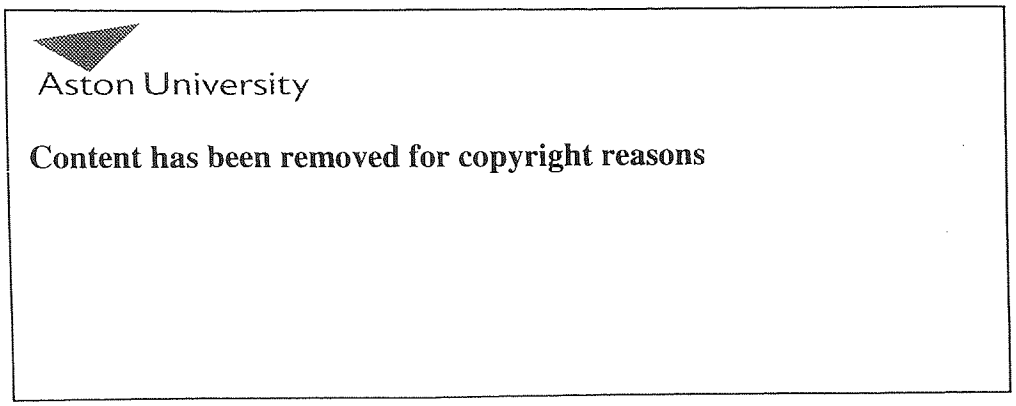
A.2 Secondary (large) test set

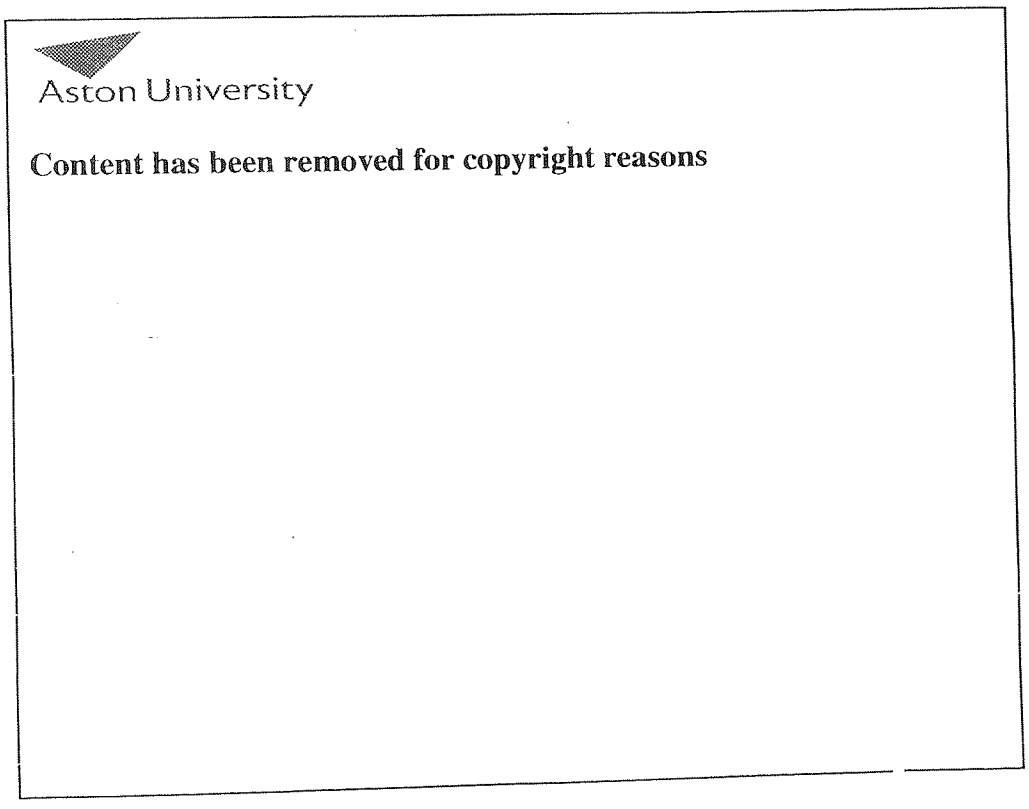
Image Name	Description
announcer:	Photo, Television announcer
beachgirl	Photo, Girl on beach
bluegirl	Photo, Girl with plain background
cablecar	Photo, Cable car with Mountain background
cornfield	Photo, Tractor in cornfield
flower	Photo, Close up of flower with bee on
fruits	Photo, Cut fruits
goldhill	Photo, English Village scene (Very detailed)
jewels	Photo, Model with jewellery
kids	Photo, 2 children on jetty, lake and mountains
koala	Photo, Koala bear in Eucalyptus tree
leopard	Photo, Leopard in tree
model	Photo, Fashion model
narcisus	Dali, "Metamorphosis of Narcissus"
nicki	Photo, Girl (Nicki) in bikini
or-002	Photo, Oregon National Park, Hurricane Ridge
orca7	Photo, orca balancing ball in park
pens	Photo, Pens on desktop
plumehat	Picasso, "Girl with Plumed Hat"
rabbit3	Photo, Rabbit on snow background,
reba08	Photo, Country singer "Reba"
soccer	Photo, Action from soccer game
vi-002	Photo, Beach with palm trees on St. Croix, Virgin Islands
yacht	Photo, 3 Yachts on lake
zebras	Photo, Zebra drinking at pool
zelda	Photo, Girl (Very common test image)

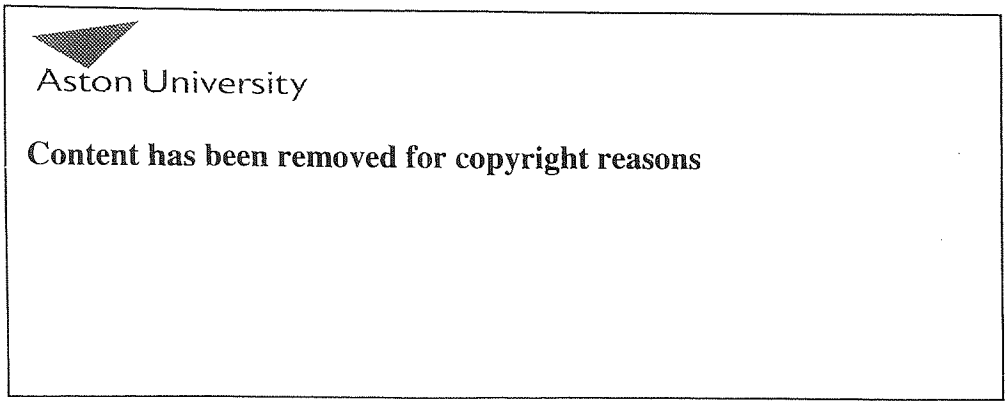
It should be noted that limitations in the printing of these images prevent a true representation of fine details and subjective testing of the images is always performed on high resolution computer displays. However, the printed versions in this Appendix give an indication of the relative complexity of the images.

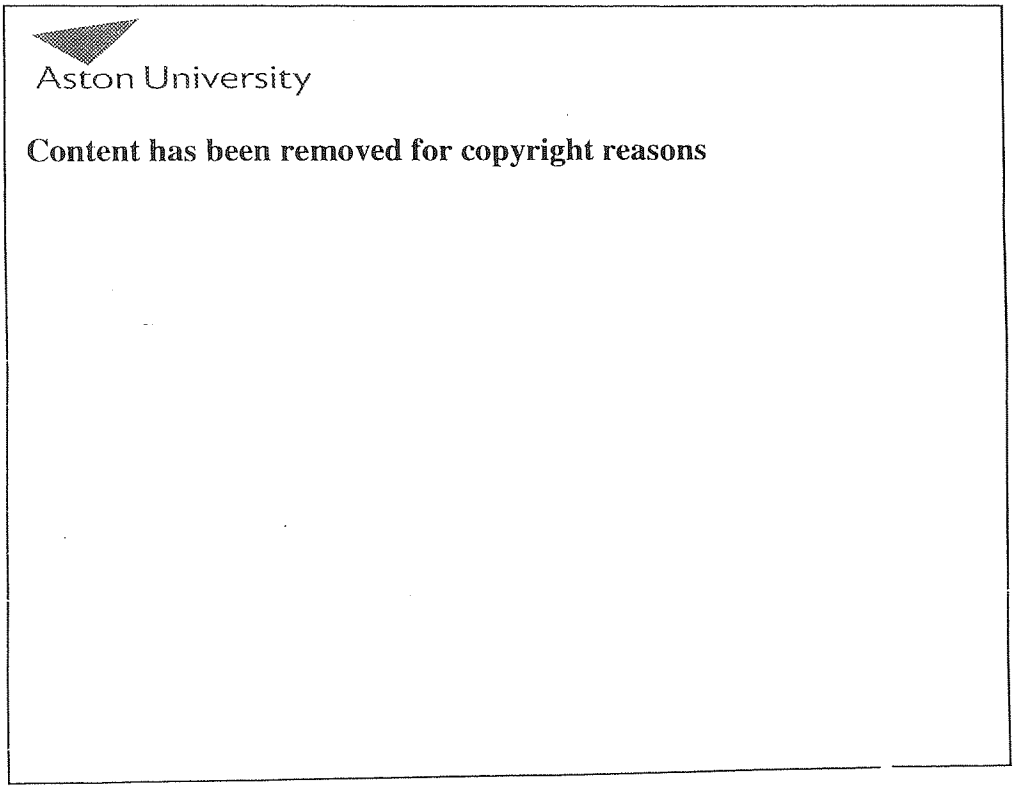

Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons

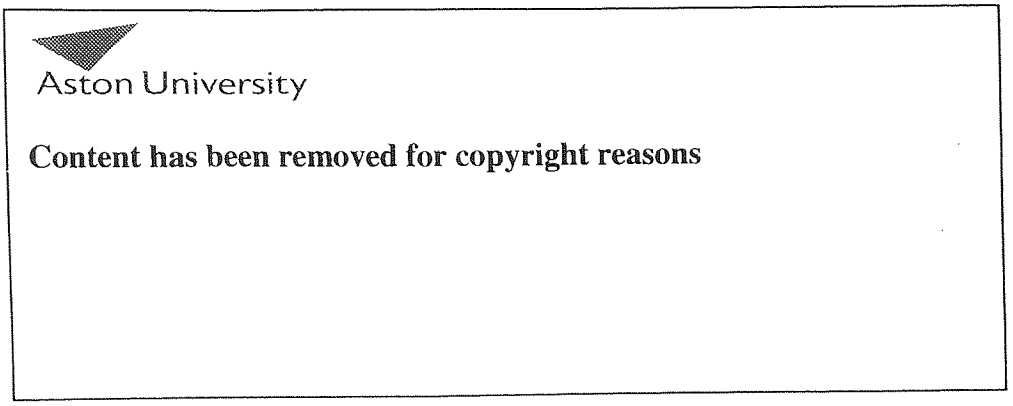

Aston University
Content has been removed for copyright reasons

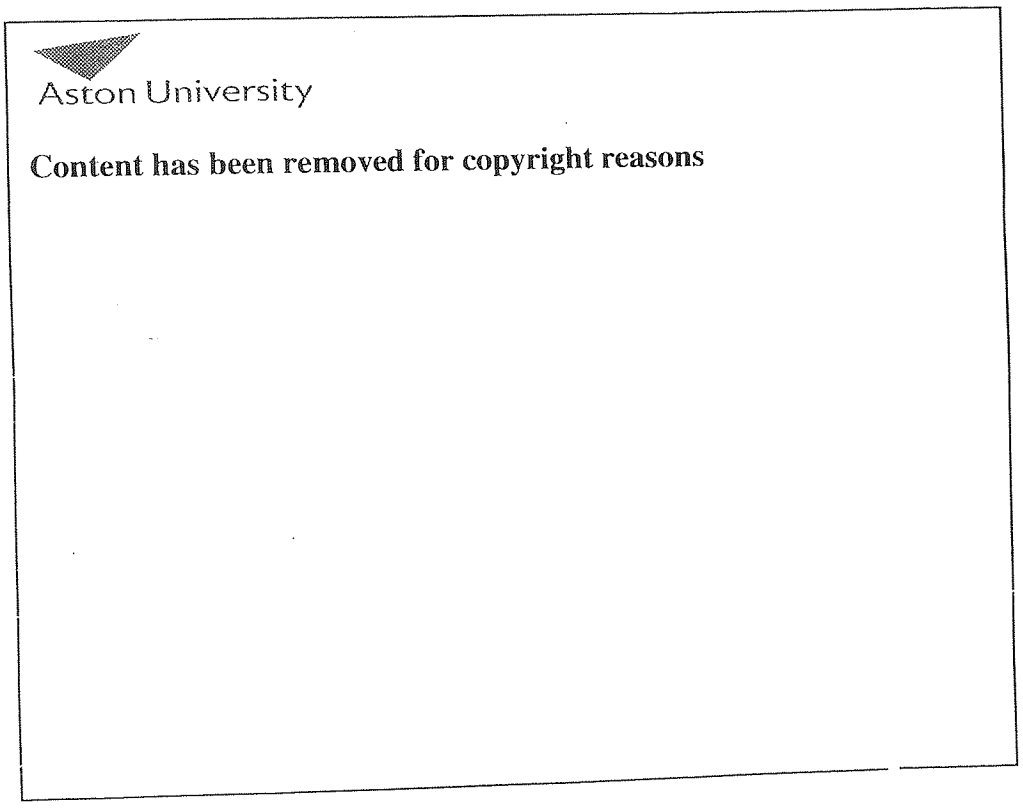

Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons

Aston University

Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons