The Contour Tree Image Encoding Technique and File Format

Martin John Turner

St John's College
University of Cambridge

A Dissertation Submitted for the Degree of
Doctor of Philosophy

April 1994

The Contour Tree Image Encoding Technique and File Format

Martin John Turner

St John's College
University of Cambridge

A Dissertation Submitted for the Degree of
Doctor of Philosophy

April 1994

The Contour Tree Image Encoding Technique and File Format

Martin J. Turner, mjt@uk.ac.cam.cl
University of Cambridge Computer Laboratory
New Museums Site, Pembroke Street
Cambridge. CB2 3QG. England

April, 1994

Abstract

The process of *contourization* is presented which converts a raster image into a discrete set of plateaux or contours. These contours can be grouped into a hierarchical structure, defining total spatial inclusion, called a *contour tree*. A *contour coder* has been developed which fully describes these contours in a compact and efficient manner and is the basis for an image compression method.

Simplification of the contour tree has been undertaken by merging contour tree nodes thus lowering the contour tree's entropy. This can be exploited by the contour coder to increase the image compression ratio. By applying general and simple rules derived from physiological experiments on the human vision system, lossy image compression can be achieved which minimises noticeable artifacts in the simplified image.

The *contour merging* technique offers a complementary lossy compression system to the QDCT (Quantised Discrete Cosine Transform). The artifacts introduced by the two methods are very different; QDCT produces a general blurring and adds extra highlights in the form of overshoots, whereas contour merging sharpens edges, reduces highlights and introduces a degree of false contouring.

A format based on the contourization technique which caters for most image types is defined, called the *contour tree image format*. Image operations directly on this compressed format have been studied which for certain manipulations can offer significant operational speed increases over using a standard raster image format. A couple of examples of operations specific to the contour tree format are presented showing some of the features of the new format.

Preface

Lesser artists borrow,
Great artists steal.
– Igor Stravinsky

Except where stated, this dissertation describes the author's own work carried out at the University of Cambridge Computer Laboratory by kind permission of Professor Roger Needham, Head of Department, and includes nothing which is the outcome of work done in collaboration.

This dissertation is not substantially the same as any that has been submitted for a degree or diploma or any other qualification at any other University.

Overview

In a way, staring into a computer screen
is like staring into an eclipse.
It's brilliant and you don't realize the
damage until its too late.
– Bruce Sterling

Reasons for the Dissertation

In this dissertation I explain and demonstrate a novel image encoding technique and file format, called the *contour tree file format*, and present its advantages and disadvantages.

Much of this dissertation is concerned with data compression techniques. New methods and analytical techniques that I have derived are explained in detail. Proven compression methods which are applied in the work are described separately in Appendix A.

For further information on previously proven techniques the reader may wish to consult some of the currently available compression text-books [Storer, 1988, Bell et al., 1990, Nelson, 1991, Gersho and Gray, 1991, Rabbani and Jones, 1991, Rabbani, 1992].

Structure of the Dissertation

The dissertation is laid out into nine chapters:

- Chapter 1, *Introduction* – A summary of currently used file formats and techniques as well as a discussion of the main aims that I believe need to be fulfilled by any file format.
- Chapter 2, *The Contour Tree Image Format* – A full description of the new image representation.
- Chapter 3, *Image Analysis* – Explanation of how suitable a given image is for being converted to the contour tree format.
- Chapter 4, *Conservative Compression* – A lossless and efficient coding scheme is defined which allows compact storage and description.
- Chapter 5, *Non-conservative Compression* – A physiologically based lossy compression system is defined. This system exploits some of the characteristics of the human visual system.
- Chapter 6, *Colour Maps* – Black-and-white, pseudo colour and true colour images are analysed and converted to the contour tree format.
- Chapter 7, *Image Manipulations* – A toolbox of operators is discussed and presented. These work directly on the image whilst it is in the contour tree format. Two test cases are described; the first deals with intensity quantisation, and the second with image texture visualisation.

- Chapter 8, *Extensions and Specialisations* – Two different variations to the standard file format are considered:
  - Hexagonal Pixel Shapes.
  - Contour Shells: 3D and Time Sequences.
- Chapter 9, *Analysis and Conclusions* – The ideas presented are evaluated, and areas for further research highlighted.
- Appendix A *Popular Stream Coders* are described. These include: Shannon-Fano Coding, Huffman Coding, Arithmetic Coding, Dynamic Markov Models and LZ77 and LZ78 coding.
- Appendix B *Image Gallery* which gives statistics for 16 different test images.

There is a glossary of terms at the end of the dissertation just before the bibliography, which also includes a list of some of the current popular image, printer and archive file formats.

Acknowledgements

To everyone I know and love I send you a message deep from my heart,
boom-boom, boom-boom, boom-boom...
– Mel Brookes

I would like to take this opportunity to thank my supervisor, Dr Neil Wiseman for all his efforts and all the members of the Rainbow Graphics Group in the Computer Laboratory whilst I was there for their eternal support, help, company and criticism: Alex (Siu Chi) Hsu, Danny Hall, Heng Wang, Jane Hunter, Jeremy Ball, John Moore, Jon Sewell, Kwai Chan, Neil Dodgson, Nicko van Someran, Oliver Castle, Rynson (Wing Hung) Lau, Tim Wiegand and Uwe Nimscheck. Also a thanks to the rest of the inmates in Austin floors 4 and 5 for their varied interests which continually reminded me of the world outside my project. My deepest thanks go to Margaret Levitt and Eileen Murray for making sure everyone in the laboratory kept sane and providing a continuous supply of coffee and hot-chocolate.

Appreciation goes to the Science and Engineering Research Council for providing funding which made this possible and my College, St John's for allowing me to continue on even after all my hassling as an undergraduate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is a foolish thing to make a long prologue,
and to be short in the story itself.
– Maccabees

The first big practical uses of continuous tone digital images were on early telegraph lines, at the beginning of this century. The most common initial technique was to use specially created character sets that simulate a half-tone pattern. This allowed an image to be transmitted using regular telegraph personnel. With the interest in news pictures, whose value is only high whilst the news is hot, cables were installed connecting the main capitals of the western world. Early systems could take ages to transmit a single picture, with the first images crossing the Atlantic taking the better part of a week or more.

In 1921 publication of the first pictures transmitted between The Daily Mirror in London and The Daily News in New York established the Bartlane system. This was one of the first image coding schemes to be patented [McFarlane, 1972]. The system reduced the transmission time across the Atlantic to a matter of two or three hours per picture. It remained in successful operation, dealing with nearly 500 pictures of important news events, prior to the outbreak of the Second World War in 1939. This was the start in the use of practical digital image coding techniques.

As images and digital image processing became more popular, a unique technique and format was developed specifically for each problem. This was acceptable in the early days when the number of images and the machines to use them were limited. Now that anyone with a small computer and a moderate amount of memory can capture, display and manipulate image data, there is a need to standardise the methods of image storage and transmission. Since the late seventies a series of image formats has been developed to allow easy transfer of images between sites and image manipulation software. These have often been biased towards the current technology of the day in terms of capture, display and available transmission bandwidth or storage.

1.1 Current Formats

At present there are well over 100 currently used image, archive and printer formats[1]. Described simply are a few examples of the more interesting formats which are currently in vogue:

PPM The most universal format is a simple raster list of pixel values which is provided in Jef Poskanzer's Portable Pixmap (PPM). This defines consecutive raster pixels to be stored as byte or ASCII intensities, preceeded by a very small and simple header. The storage of black-and-white, eight bit grey scale and 24-bit true colour images is catered for. It is very understandable by the user and storage and transmission costs are totally deterministic, but there is no encoding scheme specified at all.

[1] Some of the more popular ones are listed in the second part of the glossary at the end of the dissertation.

RLE In the late 70's a technique of Run Length Encoding (RLE) was developed, used in the University of Utah's RLE format [Peterson et al., 1986]. This codes continuous sequences of similar intensities as a pair of values, one representing the length of the sequence and the other representing the intensity. The coding is very simple and has become a central part of many image formats. Unfortunately as image quality improved, giving higher resolutions and more bits per pixel, the actual occurrence of long sequences of similar intensities became rarer and rarer, sometimes causing RLE files to be larger than the original. Except for specialist fields, including some computer generated imagery, the format has become unpopular.

DPCM Differential Pulse Code Modulation is a lossy coding scheme using the ideas behind RLE. The technique uses a prediction scheme to guess the next pixel. Only the difference between the predicted value and the actual value is stored. These differences are then quantised to gain a level of compression. DPCM is straight forward to encode in software and hardware, and offers a guaranteed but small level of compression in the order of two or four to one. Artifacts can be very noticeably in areas of rapid change. A good description can be found in [Gonzalez and Woods, 1992, Section 6.5.1]. Extensions, which complicated the processing required, offer larger compression ratios in the range of six to ten to one. These deal with groups of two dimensional sub-arrays of the whole image, for example [Hung, 1979].

Group 3 and 4 An RLE scheme is used in facsimile compression of images, which only having black and white pixels often have long sequences. The CCITT[2] introduced Group 3 and Group 4 facsimile schemes specially designed for facsimile resolutions [CCITT, 1985b, CCITT, 1985a]. These schemes, as popular as they are, are to be replaced by the new CCITT JBIG format currently in draft form [CCITT, b]. JBIG uses a neighbourhood probability coding scheme, predicting each pixel's value from a large group of previously defined pixels. A review is given in [Halton, 1994].

Quadtree In the late 70's and early 80's the quadtree format [Samet, 1980, Samet and Webber, 1988, Stewart, 1986] was developed as a two-dimensional coding scheme exploiting the same redundancies as RLE. A square image is divided along its main axes into four equal areas. Each area if it consists of a single colour is represented as that value, else it is further divided in a similar manner. Similarly to RLE the coding scheme works well on bi-level images where the correlation between pixels is high but has the tendency of becoming inefficient as resolutions increase. Quadtrees have the advantage that they are manipulatable as trees as shown in [Oliver and Wiseman, 1983a, Oliver and Wiseman, 1983b, Hunter and Steiglitz, 1979], and there is more of a two dimensional image feel to them. Unfortunately the format has never really become popular outside the research community.

GIF In 1977 and 1978, Zif and Lempel published two coding schemes [Lempel and Ziv, 1977, Lempel and Ziv, 1978] which represented sequences of bytes as offsets to previous parts of the byte stream[3]. After development of these coders the Graphics Interchange Format (GIF) was introduced by CompuServe [CompuServe, 1987] in the mid 80's which has become increasingly popular. This represents an image with a maximum of 256 colours as a byte sequence of pointers into a colour map table. The pointers are coded using Welch's version of the '78 compressor [Welch, 1984]. Compression ratios achieved are moderately good. With very few 24-bit full-colour monitors the restricted number of colours has not been a limiting factor. For high quality images, without using careful colour selection techniques, this lack of colour can be a severe handicap.

JPEG In the mid 80's the CCITT JPEG Committee [CCITT, a] was set up to design a 24-bit colour controlled lossy compression system which reduces the information of the high frequency components in the image that are not so noticeable to the human eye. The JPEG format has in the last few years become popular with graphics designers even when the

[2]CCITT has now become ITU/TSS – the Telecommunications Standardisation Sector of the ITU.

[3]Full details of these Dictionary Coding Schemes are given in Section A.2.

standard was in draft form and is steadily replacing the GIF format. A detailed description, by two of the authors, can be found in [Pennebaker and Mitchell, 1993].

VQ In the research community the technique of Vector Quantisation (VQ) for image compression has become increasingly popular. VQ involves splitting the image into regions and representing each region as an entry in a constructed codebook. The number of entries in the codebook should be substantially less than the total number of unique regions. Optimal creation of the codebook is computationally hard, but compression ratios can be correspondingly large. A full summing up of current VQ techniques is given in [Gersho and Gray, 1991].

New ideas for image representation are always being developed. Two popular methods which have been demonstrated to work, and are likely to make an impact, are:

Subband or Wavelet Coding is designed to give a more accurate representation of the image than the Cosine Transform used in JPEG and others. It is being pushed as a pseudo resolution independent transmission format for future broadcast use, by the Media Lab at MIT [Bove Jr. and Lippman, 1992] and others [Kim and Modestino, 1993, Woods, 1991]. Subband coding broadly consists of applying one, two or three dimensional high-pass and low-pass filters recursively, whose coefficients are then suitably quantised. Lossy compression ratios for a given level of quality have been shown to exceeded those for JPEG by about a factor of two.

Fractal and Affine Transformation Coding as defined by Iterated Systems Ltd. [Sloan, 1992] and others [Ali et al., 1992], offer very high levels of compression by exploiting similarity of different regions in an image. An affine transformation is a combination of rotation, scaling and translation. The current methods split the image into a set of rectangular domain blocks transmitting one or more affine transformation to represent each block related to another part of the image. Finding optimal transforms can be computationally high and quoted compression ratios range from 76:1 to 10,000:1[4]. A good review is given in [Jacquin, 1993].

1.2 Document Integration

Some of these formats have become either official standards or de-facto standards. A larger problem occurs when all these formats along with page description languages have to be integrated to form a complete publishable document. This is not addressed here except to say that solutions are being proposed at the Fraunhofer-Institut für Graphische Datenverarbeitung by Jürgen Schönhut described in [Schönhut, 1993]. This involves either converting all styles to a general universal format, like encapsulated PostScript, or integrate them seamlessly as designed in the TigerRip/TigerScript ESPRIT Project 5167 [Wiedling and Daun, 1991]. Alternatives are being offered with the ISO 8613 Open Document Architecture mixed format standard [ISO/IEC, 1993c], described clearly in [Appelt, 1991], which mixes a set of standards together.

1.3 Aims

Listed here are the seven main aims which I propose all image formats must try and address. It is not necessary for any specific format to satisfy all of these aims in order to be considered useful. A format which neglects too many of these aims runs the risk of becoming unpopular and thus unused.

1. **Compressible** The resulting image size is proportional to some definition of image complexity without losing any information in the original image.

[4]The 10,000:1 compression ratio has been shown only for fractal oriented images like fern leaves [Barnsley and Sloan, 1988].

2. **Lossy Compression** The ability to simplify an image, reducing its information content in a controlled way such that the distortions and artifacts introduced are related to the increase in compression ratio achieved.

3. **Manipulatable** Applying digital image processing techniques not just to the original raster image but also directly to the compressed or semi-uncompressed form.

4. **Rasterisable** Easy translation from and to a standard raster format so it is compatible with the majority of hardware and software support.

5. **Globally Usable** The ability to define as many different image types as possible, from real-world images captured by scanners or cameras to rendered and computer animated images.

6. **Resolution Independence** Allowing receivers of different qualities to take the same image signal and display it as well as possible.

7. **Human Understandable** The format should be comprehensible at a reasonable level so that its operations are to a degree predictable by a user.

1.4 Current Problems

None of the previously described formats satisfies all of the aims to a high degree and very few consider images as more than a one-dimensional stream rather than as a two-dimensional array. I propose to add to the confusion and present a new format, the *contour tree image format* which is designed as a successor to the quadtree format. Not only is the contour tree format aimed directly at representing computer rendered and animation images in as efficient manner as possible, but it has also been designed with a simplification technique for representing captured real world images as well.

Chapter 2

Contour Tree Image Format

An expert is one who knows more and more about less and less.
– Nicholas Murray Butler

The image format proposed considers the one assumption that there are many areas on virtually all images which can be considered homogeneous. That is in the sense that pixels within these areas are related to each other in some constant way. It is proposed to describe a format based on defining these local similarities as separate areas.

2.1 Regions and Contours

Pixels in an image which have some similar feature, whether this is a specific intensity or a texturing pattern, can be grouped together. A *region* is an area whose contents have this constant feature. A *contour* is an outside boundary of a region. A contour must have a boundary with this constant feature which is at least one-pixel thick. Figure 2.1 demonstrates some of the differences between contours and regions.

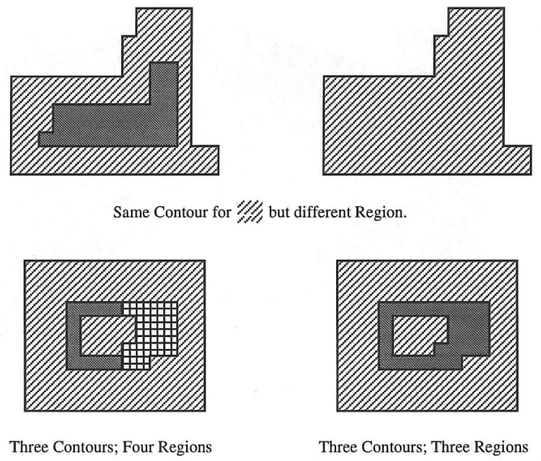


Figure 2.1: Contours and Regions.

Any region or contour is said to be *four–connected* if each pixel is connected to another which is one of the four nearest pixels. Similarly it is *eight–connected* if each pixel is connected to another

which is one of the eight nearest pixels. This is demonstrated in Figure 2.2.

Figure 2.2: Four or Eight Connected Pixels.

Eight–connected contours have the extra ability to *overlap*. This is shown below on a chessboard which can be represented, instead of as 64 separate square contours, as two eight–connected contours[1].

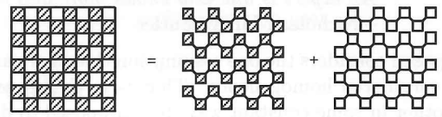


Figure 2.3: Two Eight-Connected Overlapping Contours Representing a Chess Board.

2.2 Contourization – The Basic Principles

Contourization is the creation of an image as a contoured landscape with a one-to-one mapping from pixel values to height values. An image can be fully described and represented as a set of contours. This process is shown graphically in Figure 2.4 for the Bertha image. The three dimensional viewing technique shown is a useful visualisation aid.

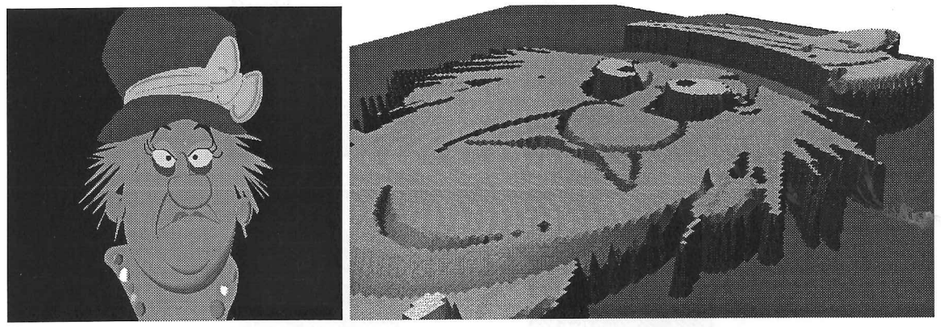


Figure 2.4: Bertha: 2D and 3D View of the Contours.

2.3 Contour Trees – A Structured Approach

The list of contours can be very large, with 8247 eight-connected contours in the simple Bertha image example above, and at present no ordering system has been imposed. For image coding, to

[1]It is worth noting that there are a large number of possible pairs, of overlapping eight-connected contours, which could have been used.

aid compression, a simple raster scan order is very useful, that is to list contours in the order that they are first encountered when scanning the original raster image from top-left to bottom-right. For image manipulation this can be restrictive, as a random pixel search operation is as costly as an ordered list search.

A hierarchical structure was designed called a *contour tree* which stores those contours totally enclosed by another contour as its children. The structure commences with an imaginary contour defining the image boundary and having all image contours as its children. Siblings in the tree can be arranged in some order. A raster scan order is a simple choice, but for some contour manipulations having them sorted in order of size is more useful. This makes random pixel access times equivalent to a hierarchical ordered list search.

A graphical representation for the contour tree of the Bertha image is shown in Figure 2.5, which shows the number of contours at each level. The contours for this contour tree are non-overlapping and eight-connected. The number of contours for each size up to 100 pixels is shown in Figure 2.6 which gives a feel of the type and number of nodes that exist and possibly suggests a metric to define the complexity of an image.

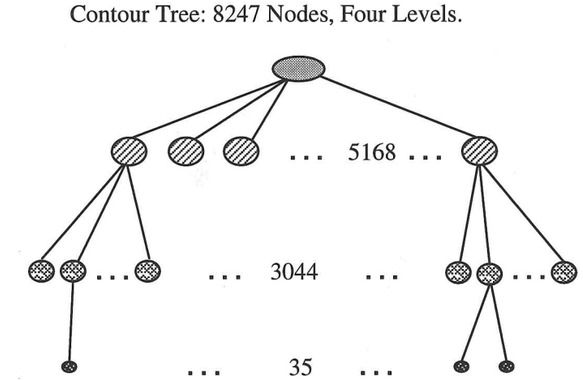


Figure 2.5: Bertha: Contour Tree.

The structure is similar to a quadtree in the sense that nodes subdivide the image space but all of its structure is dictated by the original image rather than by arbitrary division. This means that every single piece of information in the contour tree is related to the original image.

Each node of the contour tree represents one contour in the image. The minimum amount of information each node contains must enable the full reconstruction of the original contour for the contour tree to be a full description of the original image. For image manipulation operations each node can be allocated certain other properties, for example; size, bounding boxes and previous operations carried out, have proved useful. These extra properties can be permanently held by the contour tree for future use, or created and destroyed dynamically as image processing operations are carried out.

The following chapters attempt to justify that this format satisfies the aims defined in Chapter 1. To help with the analysis a selection of common images are referred to which cover a variety of image types. These are shown in Appendix B. All the coding and analysis presented deals with eight bits per pixel grey scale images with extensions for colour dealt with separately in Chapter 6.

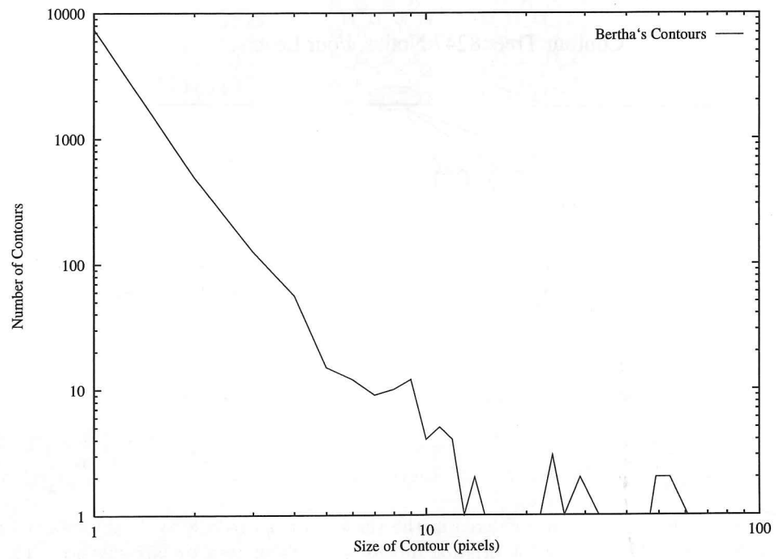


Figure 2.6: Bertha: Number of Different Sized Contours.

Chapter 3

Image Analysis

pixel, n.: A mischievous, magical spirit
associated with screen displays. The computer
industry has frequently borrowed from mythology:
Witness the sprites in computer graphics,
the demons in artificial intelligence
and the trolls in the marketing department.
– Moriarty, aka Jeff Meyer

The previous chapter defined an outline for a two-dimensional image format. It is the purpose of this chapter to confirm, in a number of ways, that the ideas and initial assumptions presented are valid. I will be applying some image analysis techniques, and then defining two simple region coding schemes. This will find out whether a more complex scheme, designed specifically for the contour tree structure, is worth the effort and what compression ratios it should achieve.

3.1 Image Analysis

3.1.1 Histograms

If the image is considered homogeneous[1] then the intensity values can be visualised with a histogram. A first-order histogram of an image is a vector containing the number of occurrences of each intensity quantisation level. Histograms are one of the first and simplest analytical tools available and often show the distortions in the acquisition hardware.

Figure 3.1 shows the first-order histograms for a couple of images, including calculations of mean and standard deviation. The histogram can be simply extended to its second-order, which is a measure of the joint occurrence of pairs of pixels separated by a specific distance. This is shown for the same images in Figure 3.2 for adjacent pixels in a horizontal sense[2].

3.1.2 Shannon Entropy

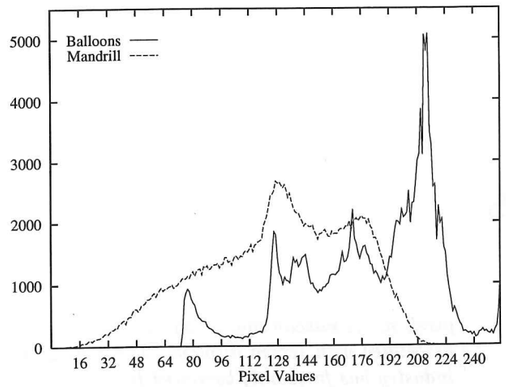
The most commonly used indicator to describe the information content of any set of data is Shannon's formula for entropy [Shannon, 1949]:[3]

$$H_1 = -\sum_{n=0}^{N-1} p_n \log_2 \frac{1}{p_n} \tag{3.1}$$

[1] That is processes on the pixels do not depend on the position of the pixel in the image.

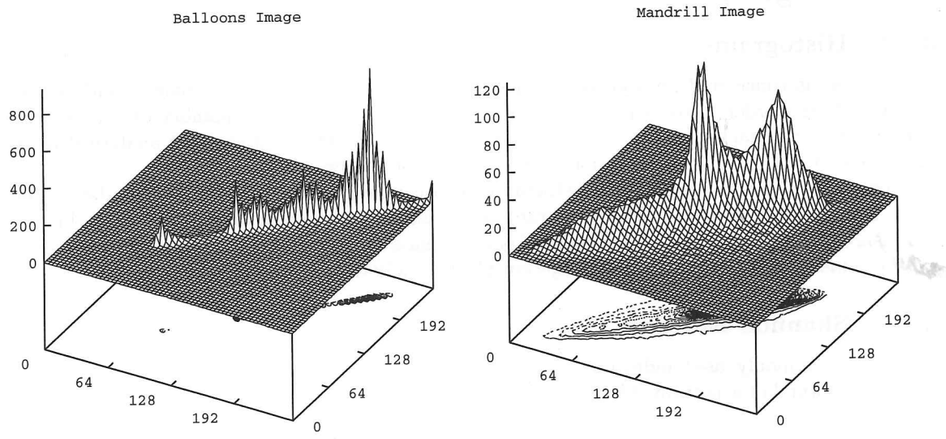
[2] first- and second-order histograms for image data are dealt with in more detail in [Pratt, 1991, pages 142–145]

[3] It is worth pointing out that Shannon's formula for entropy was originally defined for all information streams. It is considered here in terms of image data. An alternative description can be found in [Storer, 1988, pages 3–9].



	Mandrill	Balloons
Mean	128.85	178.26
Std Dev.	41.72	41.12
Median	132	188
Pixels	245760	198656
1-Entropy	7.338	7.0374
2-Entropy	6.737	5.160

Figure 3.1: First-Order Histograms and Entropy Information for Mandrill and Balloons Images.



Pixel-Pair Histograms. A two-dimensional contour map is drawn on the base plane for ease of viewing.

Figure 3.2: Second-order Histograms for Mandrill and Balloons Images.

This gives order-1 entropy[4] where p_n is the probability of pixel value n, in the range $0 \ldots N-1$, of occurring. This formula results in a single number which gives the minimum average code length if every pixel is encoded independently of the other pixels. Entropy is the information content of the histogram vector described earlier.

Order-m entropy can be calculated by extending the formula. By defining $P_m(x_1, x_2, \ldots, x_m)$ as the probability of seeing the sequence of m pixels $x_1, x_2, \ldots, x_m$ we have:

$$H_m = - \sum_{x_1, x_2, \ldots, x_m = 0}^{N-1} P_m(x_1, x_2, \ldots, x_m) \log_2 \frac{1}{P_m(x_1, x_2, \ldots, x_m)} \tag{3.2}$$

And defining $E()$ as the expected value, the full entropy of data is given by:

$$H_{\max} = - \lim_{m \to \infty} \frac{1}{m} E \left(\log_2 \frac{1}{P_m(x_1, x_2, \ldots, x_m)} \right) \tag{3.3}$$

This limit always exists if the data stream is stationary and ergodic. Table 3.1 gives the results of order-1, order-2 and order-3 entropy and also order-1 entropy on pixel differences in both a horizontal and vertical sense. This is shown in terms of compression ratio, which for eight bits per pixel source is simply calculated by dividing eight by the entropy value.

		Order-1		*Order-2*	*Order-3*
Image	*Entropy*	*Vert Diffs*	*Horiz Diffs*	*Entropy*	*Entropy*
Bertha	3.589	10.235	11.952	6.184	8.591
Clown	1.059	1.608	1.850	1.409	1.751
Balloons	1.137	2.380	2.273	1.550	1.898
Mandrill	1.090	1.192	1.262	1.187	1.385

Table 3.1: Shannon's Entropy Compression Ratios for Test Images.

These values now give concrete numbers which are theoretically achievable as compression ratios for any image. This is the first direct numerical indication of the image complexity[5].

3.1.3 Correlations

First-order histograms and order-1 entropy do not tell us about the positional relationship between pixels. A different metric is to calculate the expected value for the product of distinct pixels. Given $f_{i,j}$ is the pixel value at position i, j in the image. For homogeneous images which are thus shift-invariant the Normalised Discrete two dimensional Autocorrelation Function R' (NACF) is defined as:

$$\begin{aligned} R_{i,j} &= f * f_{i,j} \\ &= \sum_{x,y=-\infty}^{\infty} f_{x,y} f_{x+i,y+j} \end{aligned} \tag{3.4}$$

$$R'_{i,j} = \frac{R_{i,j}}{R_{0,0}} \tag{3.5}$$

It is noted that the NACF can be derived from differences in the form of a Normalised Difference Correlation Function D' (NDCF), for slight ease of computation:

[4] There is a lot of disagreement in terminology and although it is called first order entropy it is often also written as order-0 entropy.

[5] An alternative description of entropy is given in [Gonzalez and Woods, 1992, pages 324–343], and a more complete list of quantitative shape descriptions of a first-order histogram is given in [Pratt, 1991, page 561].

$$
\begin{aligned}
D_{i,j} &= \sum_{x,y=-\infty}^{\infty} (f_{x,y} - f_{x+i,y+j})^2 \\
&= \sum_{x,y=-\infty}^{\infty} (f_{x,y}^2 - 2f_{x+i,y+j}f_{x,y} + f_{x+i,y+j}^2) \\
&= 2 \sum_{x,y=-\infty}^{\infty} (f_{x,y}^2 - f_{x+i,y+j}f_{x,y}) \\
&= 2(R_{0,0} - R_{i,j}) \qquad (3.6) \\
D'_{i,j} &= \frac{D_{i,j}}{2R_{0,0}} \\
&= 1 - R'_{i,j} \qquad (3.7)
\end{aligned}
$$

As most images are a mixture of a deterministic and a zero-mean random process it can be useful to subtract the mean and then calculate the correlation. This is termed the autocovariance function C (ACF):

$$C_{i,j} = \sum_{x,y=-\infty}^{\infty} (f_{x,y} - \langle f \rangle)(f_{x+i,y+j} - \langle f \rangle) \qquad (3.8)$$

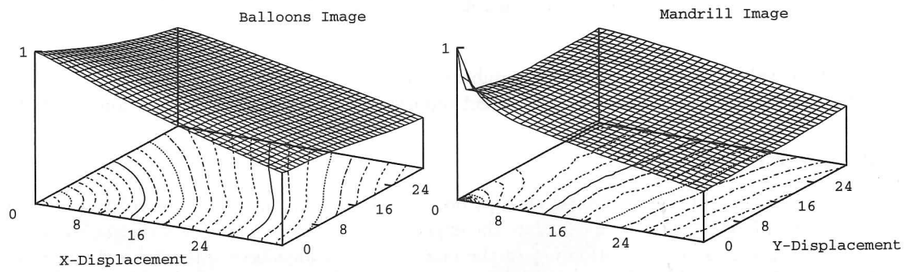


Figure 3.3: Autocorrelations for Balloons and Mandrill Image.

The resulting sharpness of slope in the autocorrelation gives an indication of how strongly related the pixels are. As the correlations are not flat there is a strong indication that a spatial correspondence exists. These are demonstrated on the autocorrelations for the Balloons and Mandrill images drawn graphically in Figure 3.3. A contour map is shown on the base plane for easy of viewing.

The difference in slope between the two images, especially around the peak, shows that the Mandrill image is far less correlated than the Balloons image which was only slightly predicted in the first-order histogram. It is noteworthy that both are slightly more correlated in the y-displacement than the x-displacement. This could be a characteristic of the image capture device used. The normalised correlation graphs above display relative displacement coefficients up to 32 pixels away. The normalisation correlation, ρ for adjacent pixels is defined as:

$$\rho = \frac{E(f_{i+1}f_i)}{E(f_i^2)} \qquad (3.9)$$

According to [Gonzalez and Wintz, 1977], for properly sampled images ρ typically lies above 0.85. For the Mandrill image $\rho = 0.987$ and for the Balloons image $\rho = 0.996$[6].

3.2 Two Simple Region Coding Schemes

The previous section treated an image as a homogeneous whole. In this section we will try and analyse the existence of locally homogeneous regions. Presented below are two different methods which were investigated to analyse and code an array of pixels. Each method has been shown with references to both four-connected and eight-connected regions. The methods considered are:

- Array Covering.
- Region Numbering.

3.2.1 Array Covering

Given a two dimensional array of regions an obvious way to define the edges is to represent them on a similarly sized array. This new array is encoded using standard image coding techniques. The solution to the four colour problem tells us that any four-connected map can be coloured using just four colours (equivalent to two bits) for each pixel. A simple pixel coding scheme has been developed which codes each pixel as one of four states:

1. Pixel is a top and left edge of a region.
2. Pixel is a left edge and not a top edge of a region.
3. Pixel is a top edge and not a left edge of a region.
4. Pixel is neither a left nor a top edge of a region.

This only works for four–connected contours as an infinite number of colours may be required for eight connected overlapping contours. So for an $m \times n$ array this requires a maximum of $2 \times (m \times n - (m + n) + 1)$ bits. For most regioned images the entropy of this representation will be low, and a suitably modified bi-level image coder can be applied.

This gives us a simple coding scheme which unfortunately brings us back to a two dimensional array. It does offer a worst case compression ratio which should be an aim for any other method. The bi-level compression standards CCITT Group 3 and Group 4 facsimile formats offer reasonable bi-level compression systems [CCITT, 1985a, CCITT, 1985b]. Table 3.2 shows the compression ratios achieved by applying these techniques. Included in the size of the files is enough information to define the intensity values for each region. Each image has two lines in the table, the first line represents the number of bytes in the respective files and the second line gives the corresponding compression ratio. Italicised numbers indicate the technique which gives the highest compression ratio[7].

Image	Original	Group 3	Group 4
Bertha	245768	30451	*19385*
Bertha	1	8.071	*12.678*
Clown	109358	84311	*80216*
Clown	1	1.297	*1.363*
Balloons	198664	*134191*	138136
Balloons	1	*1.480*	1.438
Mandrill	245768	*215965*	218869
Mandrill	1	*1.138*	1.123

Table 3.2: Array Covering.

[6] An alternative description of image correlation functions is given in [Russ, 1992, pages 218–224], and a more complete list of quantitative shape descriptions of a second-order histogram is given in [Pratt, 1991, pages 562 563].

[7] This format for comparing compression systems in tabular form is used throughout the text.

3.2.2 Region Numbering

Regions can be systematically counted forming an infinite ordered set. Given an encoder and a decoder, which both know the ordering, each region can be represented and stored or transmitted as a single number. The number of potential regions containing n pixels grow exponentially as shown in Figure 3.4. There are $O(3^n)$ different four-connected regions and $O(7^n)$ different eight-connected regions by size. Two lines are shown for each type of connectedness, the top line representing the total number of regions and the bottom line includes only unique regions excluding all rotations and reflections in the main axes.

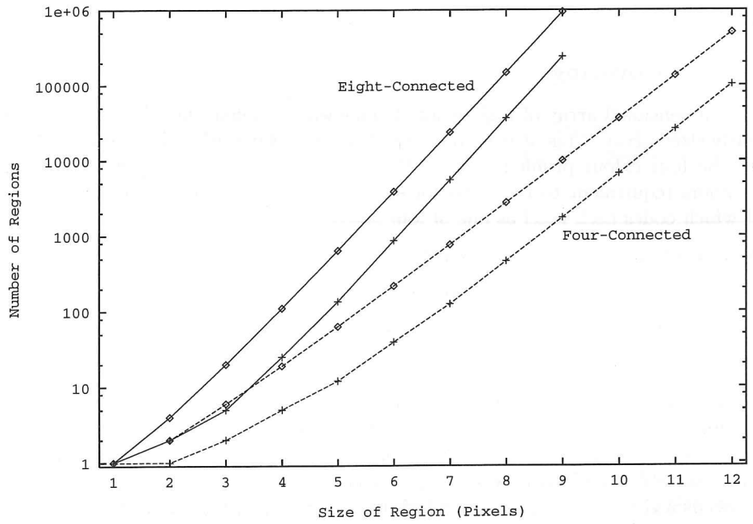


Figure 3.4: Number of Possible Regions of Size n.

A region is represented as an intensity value and a region number with possibly a few bits to specify if rotation or reflection transformations are required. This offers, after an amount of pre-processing at both encoder and decoder ends, a simple and very fast system for coding a set of regions whose size is restricted. The internal representation of a region is a list of sorted pixels which define the region space. A hash function, with an open hash table for each size of region, provides a fast mechanism for encoding regions to intermediary codes and subsequent decoding[8].

As the graph above demonstrates efficient and practical coding is achieved for small sized regions (< 11) and becomes increasingly less efficient as the size increases. As the size of regions becomes very large the pre-processing becomes computationally excessive and thus impractical. For large regions two obvious options become apparent:

- Splitting the region into smaller regions with a system for recombining the split regions back into the original.
- Coding large regions separately using an alternative coding scheme and providing an escape code to switch between the schemes.

The former has the great advantage of being very simple with no overheads whilst the latter gives a potential performance increase for the small cases when there are large contours. Tables 3.3

[8]The technique of separate chaining hashing was used, as described in [Sedgewick, 1988, pages 234–236].

and 3.4 shows us that for most real images, and a lot of animation images, the vast majority of regions are small. This is a reason to use a simple splitting mechanism.

Image	1	2	3	4	5	6	7	8	9	10	> 10
Bertha	8491	321	82	40	7	13	2	7	10	4	51
Clown	46911	8097	3261	1577	933	512	391	262	166	119	585
Balloons	61737	10192	4140	2157	1297	895	644	473	363	282	2138
Mandrill	217872	11256	1370	218	59	13	3	0	0	0	0

Table 3.3: Number of Four-connected Regions by Size.

Image	1	2	3	4	5	6	7	8	9	10	> 10
Bertha	7575	449	122	58	15	13	9	9	12	5	69
Clown	40545	8107	3234	1581	987	545	393	304	208	166	779
Balloons	50335	9912	3882	2011	1210	816	626	472	358	261	2309
Mandrill	207746	14801	2004	388	97	39	10	4	3	0	0

Table 3.4: Number of Eight-connected Regions by Size.

For four-connected regions there are 50148 different regions whose size < 11, which requires a maximum of $\log_2 50148 \approx 15.614$ bits per region. For eight-connected regions there are 28197 different regions whose size < 8, which requires a maximum of $\log_2 28197 \approx 14.783$ bits per region. To code an image requires transmitting a stream of region numbers and region intensity values. Given a maximum size of 10 for four-connected regions and 7 for eight-connected regions, the region identifier numbers are coded using a 16-bit adapted Arithmetic Coder as described in Section A.1.3. The intensity values are encoded using an Arithmetic Coder with a single memory model. The results are given in Table 3.5.

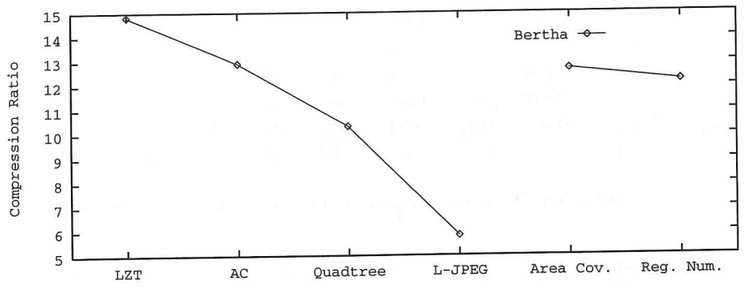
Image	Original	Four–connected	Eight–connected
Bertha	245768	27540	*20173*
Bertha	1	8.924	*12.183*
Clown	109358	*71910*	73196
Clown	1	*1.521*	1.494
Balloons	198664	*106520*	107464
Balloons	1	*1.865*	1.849
Mandrill	245768	209473	*211328*
Mandrill	1	1.173	*1.630*

Table 3.5: Region Numbering.

3.3 Simple Compression Results

A full list of compression results are given in Table 3.6. The general purpose compressors used for comparison purposes are:

- LZT – An LZ78 version compressor [Tischer, 1987], described in some detail in Section A.2.2.
- Bell, Cleary and Witten's Arithmetic Coder (AC) – both memory and memoryless models [Bell et al., 1990]. See Appendix A.1.3 for details of arithmetic coders and the parameters used.
- Bitplane Quadtree Compressor developed in [Turner, 1990] which is a modified version of the quadtree developed in [Hawkins, 1986].



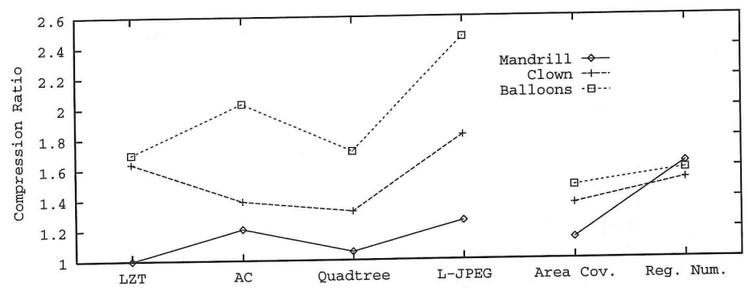


Image	Original	LZT	AC	Quadtree	L-JPEG	Area Cov.	Reg. Num.
Bertha	245768	*16583*	19036	23767	42218	19385	20173
Bertha	1	*14.820*	12.911	10.341	5.821	12.678	12.183
Clown	109358	66837	78756	82847	*60100*	80216	71910
Clown	1	1.636	1.387	1.320	*1.819*	1.363	1.521
Balloons	198664	117009	*97823*	115931	80597	134191	106520
Balloons	1	1.698	*2.031*	1.714	2.465	1.480	1.865
Mandrill	245768	245768	204028	232955	196051	215965	*211328*
Mandrill	1	1.000	1.205	1.055	1.254	1.138	*1.630*

Table 3.6: Final Compression Results.

- Lossless JPEG Compressor [CCITT, a].

3.4 Conclusions

A few simple schemes to highlight the two-dimensional correlations which exist in images have been demonstrated. Usable algorithms have been applied to give simple but practical region coding schemes. These have been shown to give a compression performance comparable with standard techniques commonly available. The new schemes were fairly naïve and potentially not very efficient in terms of achievable compression ratio, and the next chapter defines a contour coding scheme that is designed to gain as high a compression ratio as practically possible.

Chapter 4

Coding the Tree

It is in life as it is in ways,
the shortest way is commonly the foulest,
and surely the fairer way is not much about.
– Francis Bacon

As stated in Chapter 1, one of the main aims of an image format is a good coding scheme to reduce storage and bandwidth. It was shown in the previous chapter that there are redundancies in images which can be exploited. Simple coding schemes based on regions and contours can be used to gain a reasonable compression ratio.

A more sophisticated contour coding scheme designed to improve on the simpler schemes is described in detail. First, algorithms are presented which convert raster images from and to the contour tree representation. Then the coding strategy for a contour tree is described in detail.

4.1 Overview of Algorithmic Implementation of Contourization

The principle is to split an image into contours as it is scanned from top-to-bottom and left-to-right, in the usual raster scan manner. The contours are progressively extracted as they are first encountered. There are two main algorithms involved: the first, to find the location of the next contour to be coded, and the second, to find the route around the outside of the current contour. The sequence of coding is to alternate between the two algorithms, first extracting a contour setting the indicator flags as required and then finding the start location of the next contour. The principle has been developed by others including [Graham, 1967, Wilkins and Wintz, 1970, Cederberg, 1979].

During the process of contourization, information is stored in the form of an *Indicator Flag* for each pixel. This defines four different states, giving a two bit overhead:

- **I** *Initial* state, this pixel has not been seen yet.
- **L** This is a pixel on the *left* hand edge of a contour, and not on the right hand edge.
- **R** This is a pixel on the *right* hand edge of a contour, and not on the left hand edge.
- **B** This is a pixel on the *boundary* of a contour that is neither **L** nor **R**.

Figure 4.1 shows the indicator flags for a complete image. This image is of size 8×6 and has three contours in its contour tree. The second contour is totally enclosed by the first one.

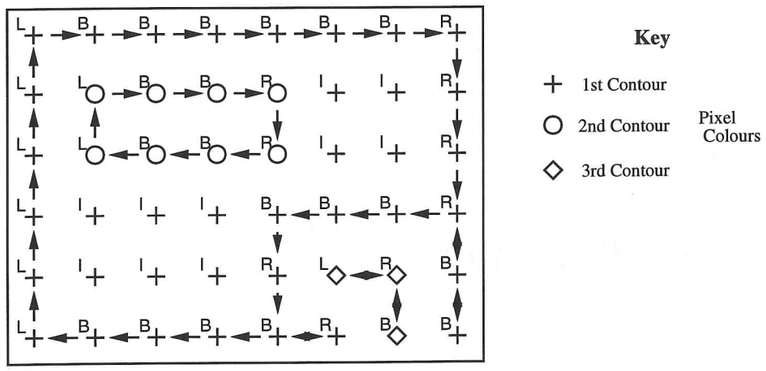


Figure 4.1: Example of Indicator Assignment.

4.1.1 Finding the Start of the Next Contour

It is assumed that we have extracted all the previous contours. The algorithm to find the next contour is:

Construct a *stack* with one value on it which is an illegal height

loc := Start Location of last contour
current := Top element of *stack*
while (height value at *loc* **not equal** *current*) **or**
(*Indicator Flag* at *loc* **not equal I**)
switch *Indicator Flag* at *loc*
L: Add height element to *stack*
R: Remove top element from *stack*
B: Do nothing
I: Do nothing
end switch
current := Top element of *stack*
move *loc* to point to next pixel in image
end while

loc is now pointing to the start of the next contour. The stack is only initialised before the first contour is found and is updated during the contourization process. At the end of each scanline the stack contains the single illegal value.

4.1.2 Finding the Boundary of a Contour

To describe a contour boundary a clockwise route is taken starting from the top left hand pixel of the contour. The algorithm has the characteristic of always wanting to turn left. Because of the similarity with many simple maze following algorithms it has been named in [Pratt, 1991, pages 623–625] as the BBF (Backtracking Bug Follower).

Figure 4.2 shows a simple example. The solid line follows the route around the contour edge whilst the dotted lines indicate where failed searches occurred.

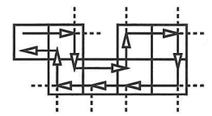


Figure 4.2: The Backtracking Bug Follower Algorithm.

A simple algorithm to perform this operation is:[1]

Start := *loc* := Top left hand corner of new contour
Direction := **East**

repeat
if (pixel to the left is in the contour)
loc := pixel to the left
Direction := turn_left(*Direction*)
else if (pixel in front is in the contour)
loc := pixel in front
else if (pixel to the right is in the contour)
loc := pixel to the right
Direction := turn_right(*Direction*)
else
loc := pixel behind
Direction := turn_around(*Direction*)
until (*loc* = *Start*)

This algorithm works fine for four-connected contours. Unfortunately the algorithm only gives a boundary description so that certain overlapping eight-connected contours, for example the chess board, can not be described in one path. They can be described with multiple contours. The black and white chess board example now requires six contours as shown in Figure 4.3. The solution of using multiple contours has been employed throughout.

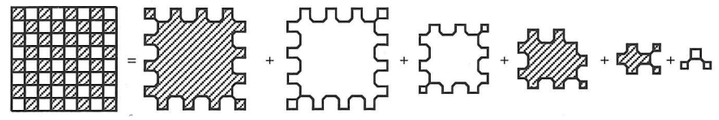


Figure 4.3: BBF Algorithm on an 8×8 Chessboard.

Whilst the contour is being extracted the states of each boundary pixel are updated as required. This is achieved by using two look-up tables on the direction of travel into and out of the pixel and the pixel's previous *Indicator Flag* state.

The first table (Table 4.1 or 4.3) decides what the new state change is given the direction of travel into and out of the pixel. The second table (Table 4.2) takes the new state change and the old state and decides what the final state should be. Table 4.3 is a slight extension of Table 4.1 to

[1] This code works for non single pixel contours and defines the set of contours that are four-connected with movement through the centres of pixels. This is described in detail in Section 4.5.3 (4MTC4). The four-connected neighbours are aligned with the axes of the compass. The functions turn_left(), turn_right() and turn_around() can then be suitable defined.

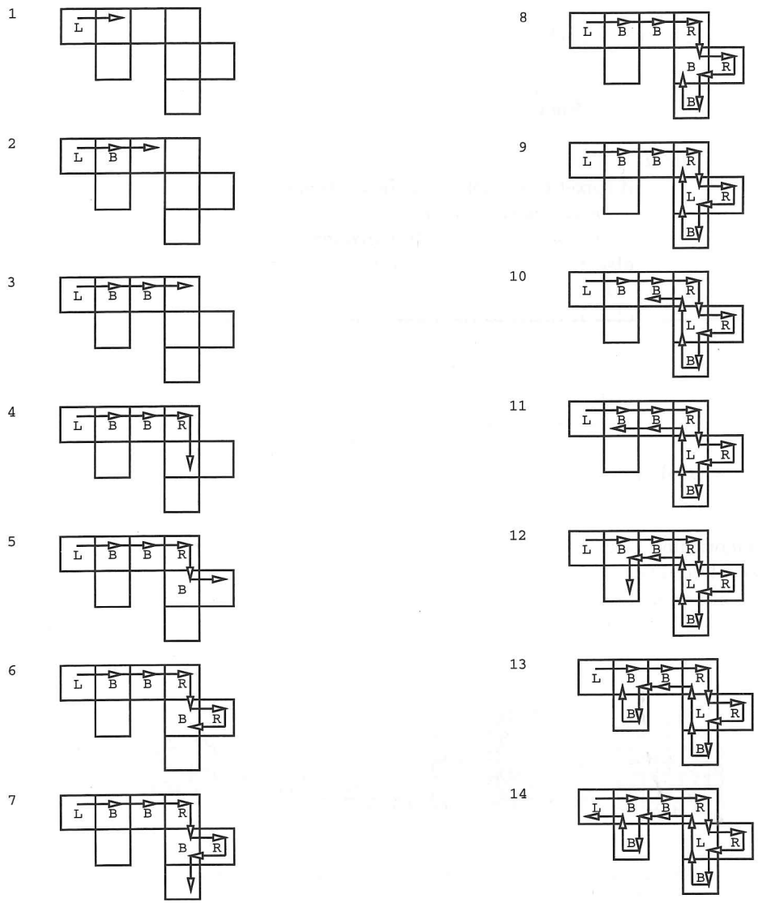


Figure 4.4: Labelling for Four-Connected Contours.

		Travel Direction Out	
		↑ or →	↓ or ←
Travel Dir. In	↑ or ←	**L**	**B**
	↓ or →	**B**	**R**

Table 4.1: Indicator States Decided by Four-Connected Directions.

		Old Flag			
		I	**L**	**R**	**B**
New Flag	**L**	**L**		**B**	**L**
	R	**R**	**B**		**R**
	B	**B**	**L**	**R**	**B**

Table 4.2: Final Assignment of Indicators.

		Travel Direction Out	
		↖ or ↑ or ↗ or →	↘ or ↓ or ↙ or ←
Travel Dir. In	← or ↖ or ↑ or ↗	**L**	**B**
	→ or ↘ or ↓ or ↙	**B**	**R**

Table 4.3: Indicator States Decided by Eight-Connected Directions.

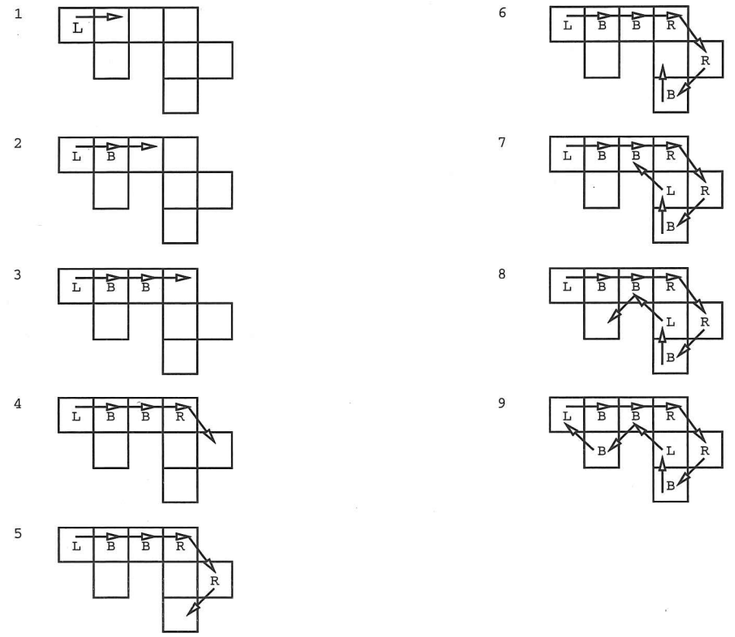


Figure 4.5: Labelling for Eight-Connected Contours.

cater for eight-connected as opposed to four-connected movement steps. It is worth pointing out that during any BBF path, pixels can only be travelled through at most twice.

An example of a four-connected contour is shown in Figure 4.4. An interesting point is movement *Step 9*. The pixel previously incorrectly allocated to state **B**, during movement *Step 5*, is automatically corrected to state **L**. Similarly, Figure 4.5 describes the same contour being described using an eight-connected version of the BBF algorithm.

As the BBF algorithm follows the boundary there are two entries in Table 4.2 which are never encountered. These entries have been left blank.

4.1.3 Recreating the Raster Image

Given a contour tree, it is a simple process of doing the contour extraction in reverse in order to recreate the original raster image. The **L** and **R** states indicate the left and right edges of scan lines for each contour. The following algorithm clears a raster array and after laying out all the boundary pixels with correct indicator states it then fills in, using a raster scan order, all the remaining internal pixels.

```
Construct an empty stack
Clear an output array which includes all indicator flags set to I

for each contour in the contour tree
    Create Indicator assignments for boundary
    Set height elements for boundary pixels
end for

loc := Top left pixel in array

while ( loc is inside the bounds of the array )
    switch Indicator Flag at loc
        L: Add height element at loc to stack
        R: Remove top element from stack
        B: Do nothing
        I: height element at loc := Top element of stack
    end switch
    move loc to point to next pixel in image
end while
```

As stated previously, similar algorithms describing the conversion of raster images from and to a set of contours have been presented. Two alternative descriptions are given in [Kirk, 1992, pages 29–33] and [Gonzalez and Wintz, 1977, pages 253–265].

4.2 Splitting Up The Tree

The contour tree has none of the contour edge pixels in any more than one boundary. To describe the whole tree compactly simply involves describing all the nodes. This in turn gives a complete description of the whole image.

4.2.1 Contour Description

To describe a contour fully three distinct pieces of information are sufficient:

- **Start Location** of the contour in the x–y space of the raster image. This is the top left hand corner of the contour, as in a raster scan of the original image.

- **Height Value** of the boundary.
- **Boundary Description** of the contour. This is a representation of the shape of its perimeter.

4.2.2 Contour Coding

Once the contours have been isolated, using contourization, they can then be separated into three streams of information as stated above. Preceding the streams of information is a header containing at least these three items:

- **Magic Number** specifying which version coded this image. This is purely for development and identification purposes so that versions can be compared and recognised.
- **Horizontal Size** of the original raster image in pixels.
- **Vertical Size** of the original raster image in pixels.

Also included in the header is the ability to insert textual comments which allows for the storage of extra information. This information can flexibly deal with a variety of topics including:

- **Title and Artist** of the image file.
- **Gammas** used in image capture and for image display.
- **Date and Time** of image file creation or modification.
- **History or Notes** on image file.

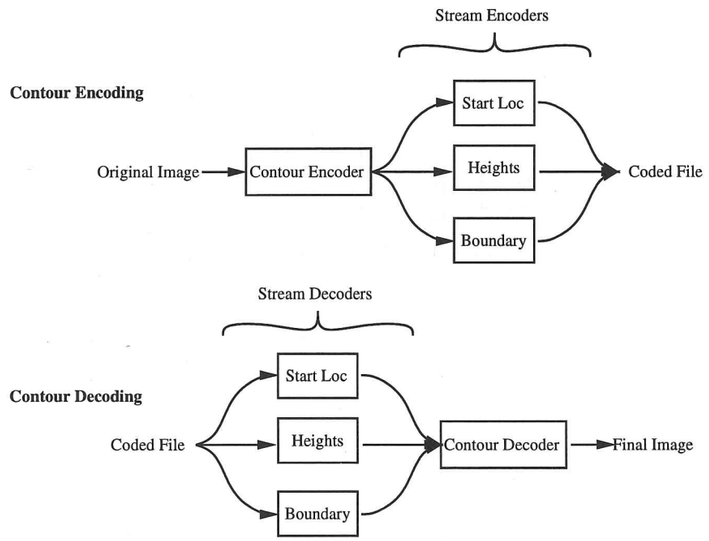


Figure 4.6: Schematic Chart for Contour Coder.

After the header, the three streams of contour information as described above, are separated and encoded using three stream encoders. The decoding procedure is the reverse of the encoding procedure as shown in Figure 4.6.

The following three sections describe in detail the potential choices available in coding each of the three streams. It is observed that a specific value in one stream reduces the choice of values in the other streams. This means that in reality the coding of the three pieces of information for each contour is actually skewed to allow for this reduction in information to occur. The coding of the streams is in the following order:

1. Start Location
2. Contour Height
3. Boundary Description

4.3 Coding Stream 1: Start Locations

Consider that the first n contours have been completely coded and the next step is to code the *start location* for the $(n+1)$th contour. The start location of this contour is defined to be the top left pixel in the contour and is represented as a single value integer defining the offset from the start location of the nth contour.

4.3.1 Restrictions

The decoder knows at this stage many locations to be impossible start locations. This consists of the set of boundary points of the previous n contours. The offset is reduced by the number of impossible start locations which lie between the nth and $(n+1)$th start locations.

Figure 4.7 shows a case where contours W and X have been coded and the next contour to be encoded is Y. To the decoder, any pixel after the start location of contour X that is not on a known boundary of a previous contour is a potential start location for contour Y.

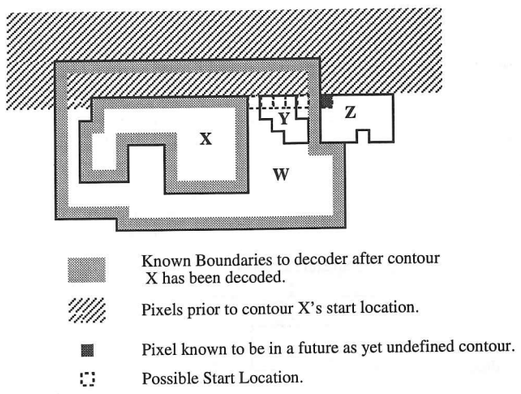


Figure 4.7: Contour Decoding just before Contour Y is to be Decoded.

Also known by the decoder is a maximum offset that the start location can be. This is the first possible start location pixel which can not belong to any of the previous contours. This pixel must then belong to an as yet undefined contour. In the example of Figure 4.7 this is the start location pixel for contour Z. So in this case, the set of possible start locations for contour Y consists of any of the six pixels highlighted with dotted lines. This means that the encoder has to code an offset within the range 0–5.

Start location offsets are defined to be deterministic when there is only one possible start location. These do not have to be coded and are removed from the stream.

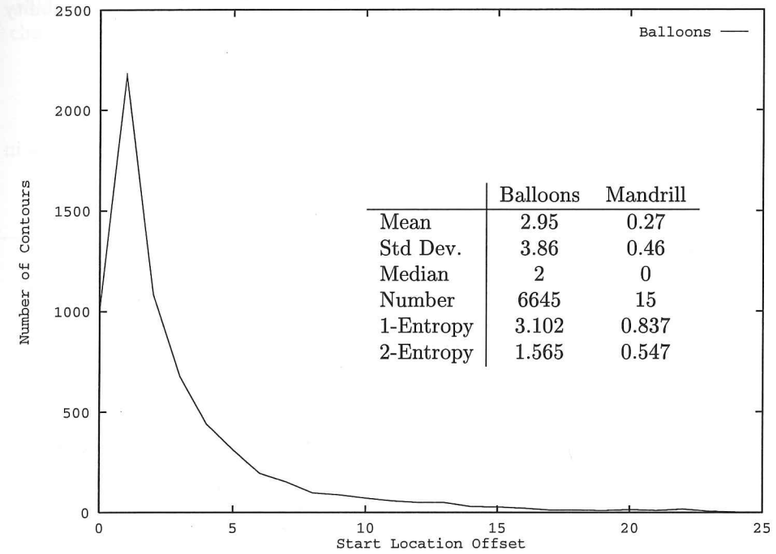
4.3.2 Probability Coding

As shown in Figure 4.8, in the test images there is very little entropy even once the deterministic cases are removed. To exploit this an Arithmetic Coder model was written which allows values above the maximum offset to be given a probability of zero.

The coder is designed to deal with offsets smaller than 254 which may be highly correlated. Large offsets, as expected, are very uncorrelated and little compression if any is achieved. For larger offsets two escape codes are defined:

254 Codes two bytes representing the offsets in the range 254–65789.

255 Codes four bytes representing the offset 65790 and larger.



	Balloons	Mandrill
Mean	2.95	0.27
Std Dev.	3.86	0.46
Median	2	0
Number	6645	15
1-Entropy	3.102	0.837
2-Entropy	1.565	0.547

Figure 4.8: Histograms of Start Locations.

Due to the information known to the decoder it is worth pointing out that for these images the majority of start locations are fully deterministic. Only 6645 of the 65790 (10:1) start locations for the Balloons image and 15 of the 219859 (14657:1) start locations for the Mandrill image have to be coded. Due to these restrictions the remaining codes have very low entropy. This yields the following file sizes for the start location streams:[2]

Image	Original	AC
Balloons	6645	*2106*
Balloons	1	*3.155*
Mandrill	15	*4*
Mandrill	1	*3.750*

Table 4.4: File Sizes of Start Locations.

This means that we are representing the 219859 start locations for the Mandrill image into 4 bytes, and 65790 start locations for the Balloons image into 2106 bytes.

[2] Arithmetic Coders are discussed in Section A.1.3. A reminder about compression performance comparisons in tables as defined for Table 3.2. The first line lists the file size in bytes whilst the second line gives compression ratios.

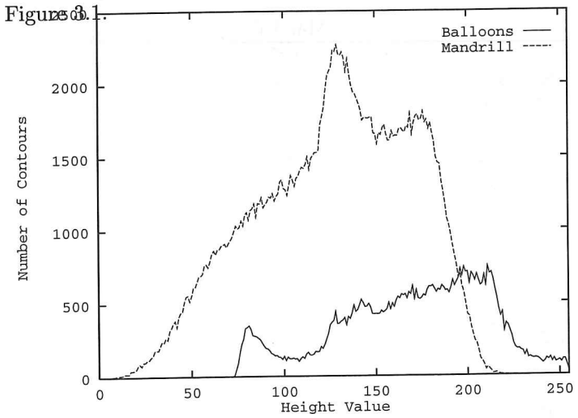
4.4 Coding Stream 2: Contour Heights

Intensity values in a raster image often have an inherent skew exploited by many prediction based image compressors. The most common of these being the Differential Pulse Code Modulation (DPCM) family of coders described in [Huang, 1965, Essman and Wintz, 1973, Zeitterberg et al., 1982]. This correlation is drastically reduced when only one height value per contour needs to be coded but it is still possible to take a slight advantage of it. Similarly, encoding the difference from previous contour height values usually reduces the information content. A memory model adaptive Arithmetic Coder was constructed and used to exploit this.

At this stage the decoder knows the start location for this contour and a few height values can not be equal to the height value of this contour. These are the height values of the contours that are already known and are surrounding this start location. These values have a probability of occurring of zero.

4.4.1 Probability Coding

The histogram in Figure 4.9 should be compared with the original histogram of all pixel values in Figure 3.1.



	Balloons	Mandrill
Mean	169.63	127.97
Std Dev.	41.04	41.72
Median	175	131
Number	65790	219859
1-Entropy	7.254	7.345
2-Entropy	6.118	6.780

Figure 4.9: Histograms of Height Values.

It is noticed that the Mandrill image has remained virtually the same due to it containing so many contours whilst the Balloons image has flattened considerably. If we compare the number of pixels with the number of contours, the ratios are 1.118:1 for the Mandrill image and 3.020:1 for the Balloons image. The main point to notice is that the entropy values of the two streams are almost identical with very little computational advantage of using predictive coders. This can be seen in Table 4.5 which gives final file sizes for the height values[3].

Image	Original	AC-1	AC-2
Balloons	65790	59234	*51060*
Balloons	1	1.111	*1.289*
Mandrill	219859	199634	*185558*
Mandrill	1	1.101	*1.185*

Table 4.5: File Sizes of Height Values.

[3] AC-1 and AC-2 are memoryless and one-level memory versions of Arithmetic Coders discussed in Section A.1.3.

4.5 Coding Stream 3: Boundary Descriptions

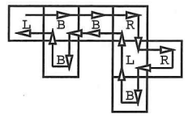
As shown in Chapter 3, to describe a region as one optimal number becomes computationally unreasonable as its size increases. Also describing the image as a couple of bit planes results in poor compression ratios. A compromise is required.

4.5.1 Edge Following

The technique of following the edge of a contour or a line has been used since the early 60's starting with Freeman's paper and others [Freeman, 1961, Graham, 1967, Freeman, 1974]. Freeman used a simple eight-connected chain code to describe movement in any of the eight directions along a line, whilst [Gonzalez and Wintz, 1977, Wilkins and Wintz, 1970] and others have used four connected chain codes.

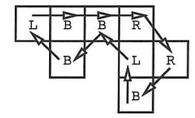
Figure 4.10: Four–Connected and Eight–Connected Freeman Codes.

An example of the contour chain code for the same contour as previously used in Figure 4.4 and 4.5 is shown in Figure 4.11. The code length is 28 bits for the four-connected case and 27 bits for the eight-connected case.



Four-Connected Chain Code:

01 01 01 10 01 11 10 00 00 11 11 10 00 11



Eight-Connected Chain Code:

010 010 010 011 101 000 111 101 111

Figure 4.11: Example of Chain Coding.

4.5.2 Movement Restrictions

Using the edge following method a stream of movement chain codes that fully describe the contour boundary can be produced. Unfortunately, it is intrinsically very inefficient as there is a large number of impossible chains. This is due to restrictions imposed by the definition of a contour. There are six main restriction rules. These fall into two main categories.

Firstly, there are three restrictions from the environment:

1. Pixels prior to the start location in a raster sense are known not to be included in this contour as we are starting from the top left pixel.

2. Horizontal and vertical edges of the full raster image are known.

3. All the contours previously coded are known by the decoder at this stage, and any whose boundary abuts the boundary for this contour imposes movement restrictions.

Secondly, there are three restrictions from the part line chain already described so far for this contour:

4. Whilst describing the boundary description this perimeter line can not cross itself so the previous part boundary so far described also acts as a movement restriction.

5. As a clockwise route is taken to define the boundary description, the pixels on the immediate outside of the part boundary so far described can not subsequently form part of the remaining section of the boundary.

6. If during the clockwise route a neighbouring pixel is known to be on the contour any pixel that requires a further rotation to the right can not thus be on the next step.

4.5.3 Overview of Movement Chain Codes

There is a total of five different contour coding methods. There are three possible coding methods for four-connected contours, and two methods for eight-connected contours. These define the options of moving from centres of pixels or around the edges of pixels.

- **Four-Connected Movement Through Centres 4MTC4.** Codes four-connected contours using a four-connected chain code which travels through the centres of pixels.
- **Eight-Connected Movement Through Centres 4MTC8.** Codes four-connected contours using an eight-connected chain code which travels through the centres of pixels.
- **Movement Along Edges 4MAE.** Codes four-connected contours using a four-connected chain code which travels around the edges of pixels.
- **Eight-Connected Movement Through Centres 8MTC.** Codes eight-connected contours using an eight-connected chain code which travels through the centres of pixels.
- **Movement Along Edges 8MAE.** Codes eight-connected contours using a four-connected chain code which travels around the edges of pixels.

The next five sections give further details of these movement codes and Figure 4.12 shows the differences graphically.

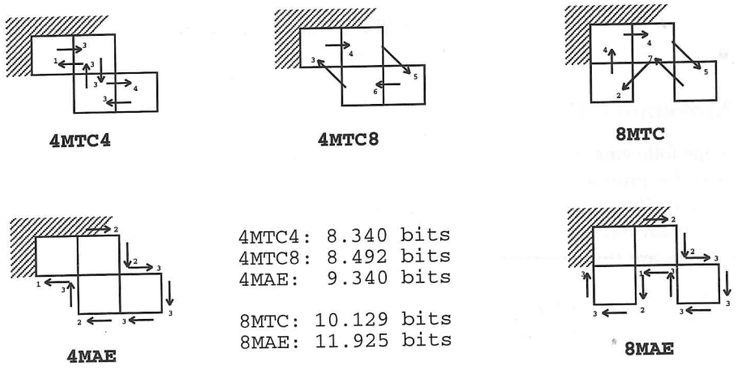


Figure 4.12: Example Boundary Descriptions.

4.5.4 Four-Connected Movements Through Centres (4MTC4)

Starting from the top left pixel of the boundary the movement steps can travel in any of the four directions. This means that for each step a maximum of two bits is required. There are two special cases to be taken into consideration:

- **The single pixel contour.** This has to be treated separately as there is as yet no code for defining "go nowhere" as is required in this case. This is solved by redefining one of the illegal movement directions to represent a single pixel. In implementation, the move up to the pixel in the previous scanline which is impossible, represents the single pixel contour.
- **Contours which are not comma- or prefix-free.** This occurs when the movement path returns to the start location but has not yet fully described the contour boundary. Figure 4.13 shows two examples of these contours.

  Two techniques were employed to resolve this problem:

  1. Ignore the problem and do not allow these contours as single contours. This means that they are coded as two separate contours.
  2. Have a simple two way code choosing between a finished code from one which should be extended. This code is inserted at times of indecision.

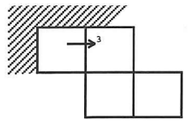
Figure 4.13: Special Case 4MTC4 Contours.

A detailed step-by-step example of encoding is shown in Figure 4.14. This also lists the restrictions imposed, and gives the theoretical total number of bits required to encode the chain code if it were sent through an optimal entropy coder. Descriptions for each movement step:[4]

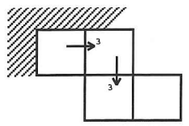
- *Step 1* Restricted from going **North** or **West** because prior pixels to the start location can not be in this contour. (Restriction Rule **1**)
- *Step 2* Restricted from going **North** for the same reasons as in *Step 1*.
- *Step 3* No restrictions; all four movement steps are possible.
- *Step 4* Restricted from going **North**. The **South** move of *Step 2* tells us that the **North** pixel is not on this boundary. (Restriction Rule **5**)
- *Step 5* Restricted from going **East**. The **North** pixel is already known to be on the contour and any further rotation to the right can not be on the next step. (Restriction Rule **6**)
- *Step 6* The pixel immediately to the left and to the **West** is known to be on the contour and has to be taken. (Restriction Rule **6**)

This results in six movement steps to encode. The total entropy is calculated as 8.340 bits which is a vast improvement over the 12 bits required by the original chain code.

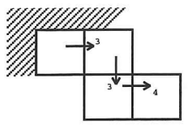
[4] Step-by-step examples are given for all five different coding methods. Detailed descriptions of movement restrictions are only given for 4MTC4.

1 

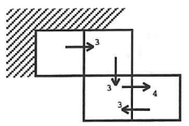
```
Move East.
Three possible movements:
 Single pixel, East and South.
Hence log(3) bits is required to
encode it.
```

2 

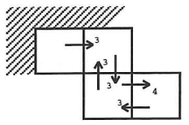
```
Move South.
Three possible movements:
 East, South and West.
Hence log(3) bits is required to
encode it.
```

3 

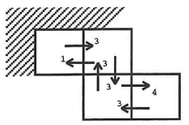
```
Move East.
Four possible movements:
 North, East, South and West.
Hence log(4) bits is required to
encode it.
```

4 

```
Move West.
Three possible movements:
 East, South and West.
Hence log(3) bits is required to
encode it.
```

5 

```
Move North.
Three possible movements:
 North, South and West.
Hence log(3) bits is required to
encode it.
```

6 

```
Move West.
One possible movement:
 West.
Hence log(1) bits is required to
encode it.
```

```
Chain Code:
 01 10 01 11 00 11

Total Number of bits required:
 log(3) + log(3) + log(4) + log(3) + log(3) + log(1) = 8.340 bits
```

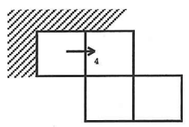
Figure 4.14: Chain Coding Example of 4MTC4.

4.5.5 Eight-Connected Movements Through Centres (4MTC8)

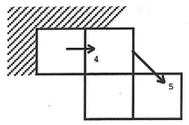
This is very similar to 4MTC4 except that due to the extra flexibility of moving direction, fewer movement steps are likely to be required. This also makes the code comma-free except single-pixel contours still need to have a little extra code designed for them.

The main difference is that at each step there are six possible directions that can be taken if the previous move was a four-connected move, and seven possible directions if the previous move was a diagonal move. This means a maximum of $\log_2 6 \approx 2.585$ or $\log_2 7 \approx 2.807$ bits per movement step is required.

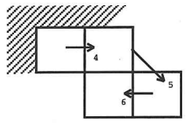
The following is an example for the reasons behind this characteristic. After travelling **North** if the next move is either **SouthWest** or **West** then these pixels are also accessible from the previous pixel. The move from the previous pixel should have gone there directly. This means that the moves **SouthWest** or **West** can never occur after a **North** move. This is directly the application of restriction rule **6**.

1 

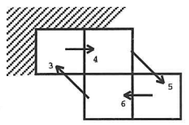
```
Move East.
Four possible movements:
 Single Pixel, East, SouthEast and South.
```

2 

```
Move SouthEast.
Five possible movements:
 East, SouthEast, South, SouthWest and West.
```

3 

```
Move West.
Six possible movements:
 NorthEast, East, SouthEast, South,
 SouthWest and West.
```

4 

```
Move NorthWest.
Three possible movements:
 SouthWest, West or NorthWest.
```

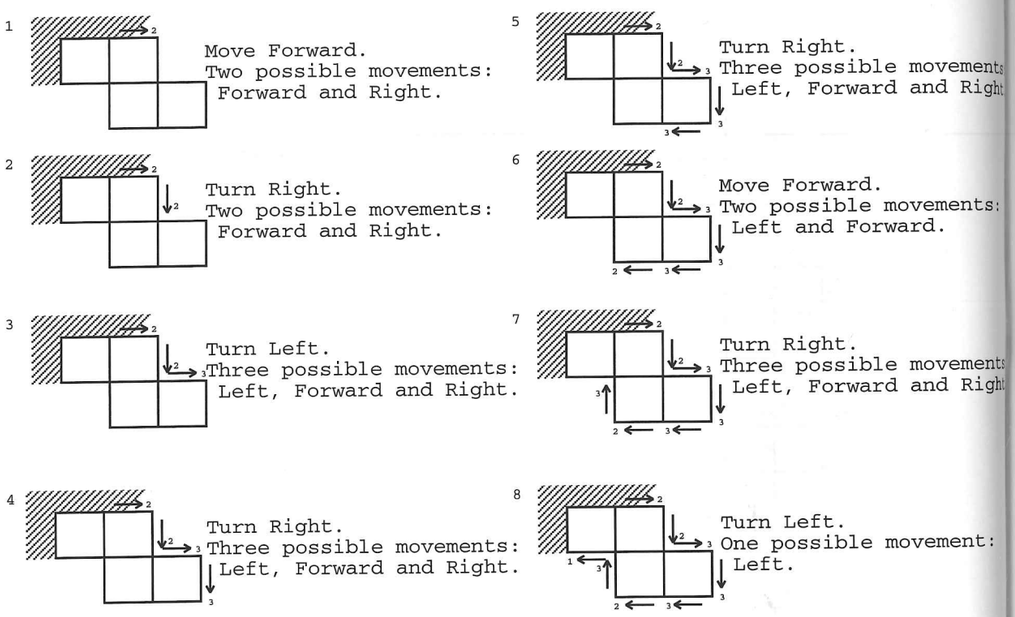
```
Chain Code:
 010 011 110 111

Total Number of bits required:
 log(4) + log(5) + log(6) + log(3) = 8.492 bits
```

Figure 4.15: Chain Coding Example of 4MTC8.

Figure 4.15 shows the step-by-step encoding. The total entropy is calculated as 8.492 bits which is an improvement over the 12 bits required by the original chain code. It is slightly worse than that for 4MTC4.

4.5.6 Movements Along Edges (4MAE)

Going round the edges of pixels has the advantage that the path never meets itself except at the start location. This means that there are no special cases, either single pixel contours or those cases shown in Figure 4.13. Also as the path never returns along the edge it has just come from only three of the possible directions are valid at any point so a maximum of $\log_2 3 \approx 1.585$ bits per movement step are required. The disadvantage with an MAE method is that extra movement steps are required over an MTC method which is discussed in Section 4.5.9.




```
Chain Code:
 01 10 01 10 11 11 00 11

Total Number of bits required:
 log(2) + log(2) + log(3) + log(3) + log(3) +
  log(2) + log(3) + log(1) = 9.340 bits
```

Figure 4.16: Chain Coding Example of 4MAE.

Figure 4.16 shows the step-by-step encoding. Instead of using compass directions the labels are given in relative movement directions of **Right**, **Forward** and **Left**. The total entropy is calculated as 9.340 bits which is an improvement over the 16 bits required by the original chain code. It is slightly worse than that for both 4MTC4 and 4MTC8.

4.5.7 Eight-Connected Movements Through Centres (8MTC8)

8MTC8 is very similar to 4MTC4 and 4MTC8. Movement steps are as costly as 4MTC8 ($\log_2 6$ or $\log_2 7$ bits per movement step). The single pixel contour has to be coded separately and there is a slightly different sort of contour that causes a non-comma-free code to be introduced. Figure 4.17 shows two non-comma-free eight-connected contours.

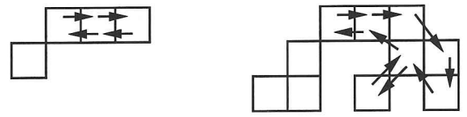


Figure 4.17: Special Case 8MTC Contours.

Either of the same two techniques that were proposed for 4MTC4 can be applied:

1. Ignore the problem and do not allow these contours as single contours. This means that they have to be coded as two separate contours.

2. Have a simple two way code choosing between a finished code from one which should be extended. This code is inserted at times of indecision.

In both 4MTC4 and 8MTC the first technique was chosen which this simplified the implementation. If a strictly correct representation of all contours is required then the 4MAE or 8MAE schemes must be used.

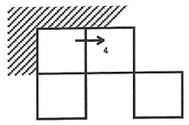
Figure 4.18 shows the step-by-step encoding. The total entropy is calculated as 10.129 bits for the five movement steps. This is an improvement over the 15 bits required by the original chain code.

4.5.8 Movements Along Edges (8MAE)

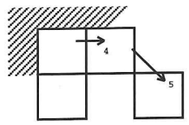
8MAE is almost exactly the same as 4MAE except that there are fewer movement restrictions imposed. The code has no special case and requires a maximum of $\log_2 3 \approx 1.585$ bits per movement step.

Figure 4.19 shows the step-by-step encoding. The total entropy is calculated as 11.925 bits for the 10 movement steps. This is an improvement over the 20 bits required by the original chain code. It is slightly worse than that for 8MTC.

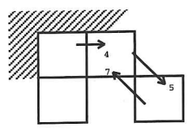
Chapter 2 described how eight-connected contours have the ability to overlap. This does not affect these movement algorithms except by introducing further movement restrictions. The decision to allow overlapping contours or not is discussed in Section 4.5.9.

1 

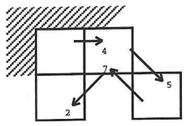
Move East.
Four possible movements:
Single Pixel, East, SouthEast and South.

2 

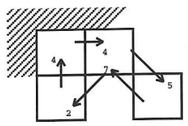
Move SouthEast.
Five possible movements:
East, SouthEast, South, SouthWest and West.

3 

Move NorthWest.
Seven possible movements:
NorthEast, East, SouthEast, South,
SouthWest, West and NorthWest.

4 

Move SouthWest.
Two possible movements:
SouthWest and West.

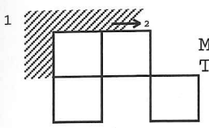
5 

Move North.
Four possible movements:
South, SouthWest, West and North.

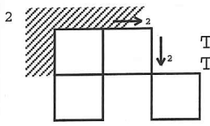
Chain Code:
010 011 111 101 000

Total Number of bits required:
log(4) + log(5) + log(7) + log(2) + log(4) = 10.129 bits

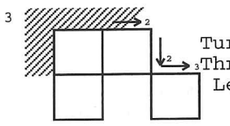
Figure 4.18: Chain Coding Example of 8MTC.

1 

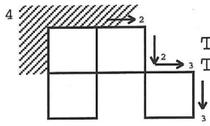
Move Forward.
Two possible movements:
Forward and Right.

2 

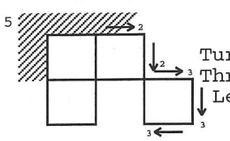
Turn Right.
Two possible movements:
Forward and Right.

3 

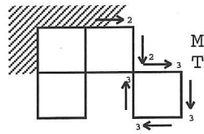
Turn Left.
Three possible movements:
Left, Forward and Right.

4 

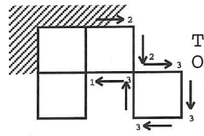
Turn Right.
Three possible movements:
Left, Forward and Right.

5 

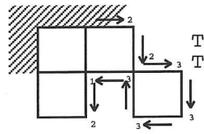
Turn Right.
Three possible movements:
Left, Forward and Right.

6 

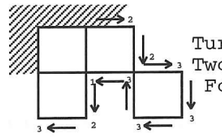
Move Right.
Three possible movements:
Left, Forward and Right.

7 

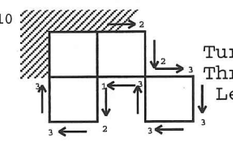
Turn Right.
One possible movement:
Right.

8 

Turn Right.
Two possible movements:
Left and Forward.

9 

Turn Right.
Two possible movements:
Forward and Right.

10 

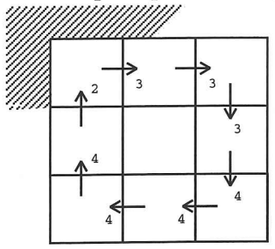
Turn Right.
Three possible movements:
Left, Forward and Right.

Chain Code:
01 10 01 10 11 00 11 10 11 00

Total Number of bits required:
log(2) + log(2) + log(3) + log(3) + log(3) + log(3) +
log(1) + log(2) + log(2) + log(3) = 11.925 bits

Figure 4.19: Chain Coding Example of 8MAE.

Movement Through Centres
Start at top left pixel heading North.

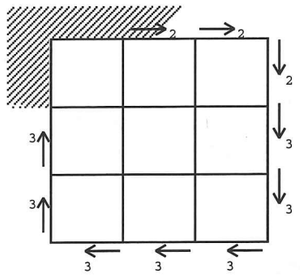


Path : E, E, S, S, W, W, N, N.

Choices: 3, 3, 3, 4, 4, 4, 4, 2.

bits: log(3)+log(3)+log(3)+log(4)+
log(4)+log(4)+log(4)+log(2) = 13.755

Movement Along Edges
Start at top right of top left pixel heading East.



Path : E, E, E, S, S, S, W, W, W, N, N.

Choices: 2, 2, 2, 3, 3, 3, 3, 3, 3, 3.

bits: log(2)+log(2)+log(2)+log(3)+
log(3)+log(3)+log(3)+log(3)+
log(3)+log(3) = 14.095

Path gives a list of relative movement steps from "North", "East", "South" and "West".
Choices gives the total number of different possible movement directions known to the decoder.
bits calculates the total number of bits required to code this path.

Figure 4.20: Example Coding of a Simple Eight Pixel Contour Using Two Alternative Methods.

4.5.9 Comparisons of Design Choices

The next three sections address some of the choices that are available when implementing a contour chain coder. It is possible to code the image using all five methods, and then select the most appropriate with a simple switch being coded so that the decoder knows which method has been applied. As permutations increase this very soon becomes computationally excessive. To try and make the decision early it is worth studying the possible results on theoretical data before applying the techniques to real data.

Movement Through Centres vs Movement Along Edges

The main advantage MTC methods have over MAE methods is that the number of coding steps needed to describe a contour is reduced with a subsequent increase in the entropy content. The maximum number of steps for 4MAE is up to four more than 4MTC4. This means as boundaries become longer the maximum size to code 4MAE approaches $\left(\frac{\log_2 3}{2} \approx 0.7925\right) \times$ maximum size to code 4MTC4. On the other hand for small contours less than 16 pixels, 4MAE can be less efficient.

The following graph in Figure 4.21 compares the number of bits required to represent the 49 rectangular contours whose sides are less than eight pixels long, with movement restrictions (as described above). The identity line $y = x$ graphically displays when one method gives a higher compression ratio then the other, and the line $y = x\frac{\log_2 3}{2}$ is the asymptotic approached as contour sizes increase to infinity. Table 4.6 shows the actual numbers for all 49 contours, and Figure 4.20 shows how to calculate the number of bits required to describe the route round a 3×3 contour using the two movement methods.

The graph seems to indicate that except for those rare images which contain only very small contours an advantage is gained by using an MAE method.

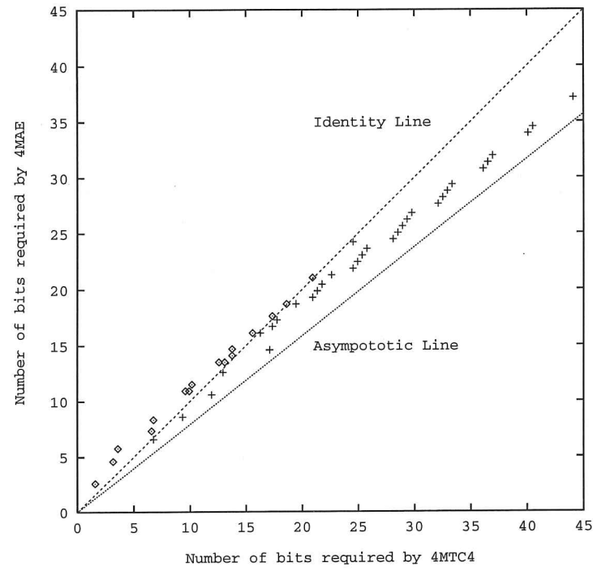


Figure 4.21: Plot Comparing Compactness of 4MTC4 and 4MAE for Rectangular Contours.

Movement Through Centres
Horizontal Size

	1	2	3	4	5	6	7
1	*1.585*	*4.170*	6.755	9.340	11.925	13.925	17.095
2	*3.585*	*6.755*	*9.925*	*13.095*	16.265	19.435	22.605
3	*6.585*	*10.170*	*13.755*	17.340	20.925	24.510	28.095
4	*9.585*	*13.755*	17.755	21.340	24.925	28.510	32.095
5	*12.585*	*17.340*	21.755	25.340	28.925	32.510	36.095
6	*15.585*	*20.925*	25.755	29.340	32.925	36.510	40.095
7	*18.585*	24.510	29.755	33.340	36.925	40.510	44.095

Movement Along Edges
Horizontal Size

	1	2	3	4	5	6	7
1	2.585	4.585	*6.585*	*8.585*	*10.585*	*12.585*	*14.585*
2	5.755	8.340	10.925	13.510	*16.095*	*18.680*	*21.265*
3	7.340	11.510	14.095	*16.680*	*19.265*	*21.850*	*24.435*
4	10.925	14.680	*17.265*	*19.850*	*22.435*	*25.020*	*27.605*
5	13.510	17.585	*20.435*	*23.020*	*25.605*	*28.189*	*30.774*
6	16.095	21.020	*23.605*	*26.190*	*28.774*	*31.360*	*33.945*
7	18.680	*24.190*	*26.774*	*29.360*	*31.944*	*34.529*	*37.114*

Table 4.6: Data Points of 4MTC4 and 4MAE for Rectangular Contours.

Eight–Connected vs Four–Connected

Similarly to the previous debate, when dealing with eight-connected contours as opposed to four-connected the resulting contour tree has the same number or fewer leaves, but results in a more complicated chain code.

To find out when eight-connected should be used over four-connected a very simple toy contouring system can be developed. Consider a scenario in which contours of size one and two only can be coded. This means that there are three differently shaped contours for four-connected (□, □□, ⊟) and five differently shaped contours for eight-connected (□, □□, ⊡, ⊟, ◱). Let p represent the probability that a neighbouring pixel is the same intensity as itself. Also let z_4 and z_8 represents the probability that a pixel is not the second half of a contour which is respectfully four-connected or eight-connected. If there are $2^8 = 256$ different values for any pixel then it takes eight bits to encode each pixel value. The costs per pixel for this restricted four-connected and eight-connected contour codings (C_4, C_8), are approximately:

$$\begin{aligned} C_4 &= z_4(8 - p\log_2 p - \\ &\qquad (1-p)p\log_2(1-p)p - \\ &\qquad (1-p)^2\log_2(1-p)^2\ ) \end{aligned} \tag{4.1}$$

$$\begin{aligned} z_4 &= (1-p)^2 + 2(1-p)p(1-z_4) + p^2(1-z_4)^2 + O(p^3) \\ 0 &\approx p^2 z_4^2 - (1+2p)z_4 + 1 \end{aligned} \tag{4.2}$$

This gives an approximate value for z_4:

$$z_4 \approx \frac{(2p+1) - \sqrt{4p+1}}{2p^2} \tag{4.3}$$

$$\begin{aligned} C_8 &= z_8(8 - p\log_2 p - \\ &\qquad (1-p)p\log_2(1-p)p - \\ &\qquad (1-p)^2 p\log_2(1-p)^2 p - \\ &\qquad (1-p)^3 p\log_2(1-p)^3 p - \\ &\qquad (1-p)^4\log_2(1-p)^4\ ) \end{aligned} \tag{4.4}$$

$$\begin{aligned} z_8 &= (1-p)^4 + 4(1-p)^3 p(1-z_8) + 6(1-p)^2 p^2 (1-z_8)^2 \\ &\qquad +4(1-p)p^3(1-z_8)^3 + p^4(1-z_8)^4 + O(p^5) \\ 0 &\approx p^4 z_8^4 + 4p^3 z_8^3 - 6p^2 z_8^2 + (1+4p)z_8 - 1 \end{aligned} \tag{4.5}$$

This gives an approximate value for z_8:

$$z_8 \approx \frac{(4p+1) - \sqrt{1+8p-8p^2}}{12p^2} \tag{4.6}$$

It should be noted that the calculations of z_4 and z_8 are only accurate for small values of p. Figures 4.22 and 4.23 show how these two functions compare for small values of p. As expected eight-connected is more expensive than four-connected when p is very small reaching a maximum near to the completely random probability of $\frac{1}{2^8} \approx 0.0039$. Then as p increases the cost of eight-connectedness becomes less then that of four-connectedness. It is interesting that the cross-over point for this particular case is when $p \approx \frac{2.690}{2^8} \approx 0.01051$. This means that if there is an image where a correct guess of the neighbouring pixel can be made with an accuracy of greater than 1% rather than the random chance of 0.4% then it is preferable to use eight-connected contours.

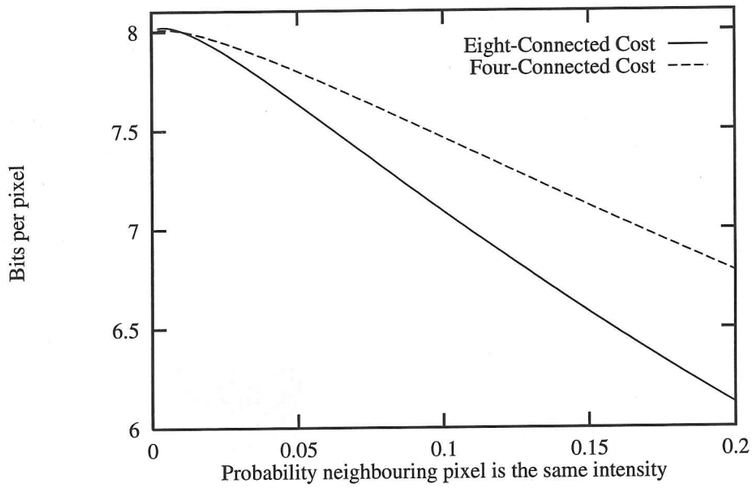


Figure 4.22: Plot Comparing Four-Connected Versus Eight-Connected. No. 1.

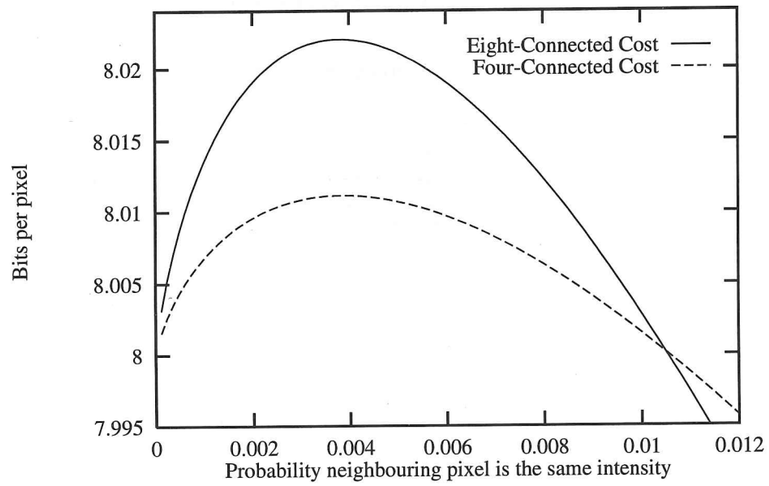


Figure 4.23: Plot Comparing Four-Connected Versus Eight-Connected. No. 2.

This is a nice simple toy example and the conditional probabilities for real four-connected and eight-connected contours are much more complicated. The factors to note are that there is likely to be a value of p for which the extra coding complexity involved in eight-connected coding is made up for in the savings in having to code a reduced number of contours. Also, according to the graph the gain in using eight-connected is possibly far greater in percentage terms.

A big difference in the actual coding scheme is that the entropy coders are adaptive, changing over time, whilst this toy example deals with a static set of probabilities. With adaptive coders the smaller the set of choices implies the faster they can change. This means that for smaller images the four-connected entropy coders with fewer codes may tailor themselves more quickly then the eight-connected coders. So as long as p is not too small and given large images, then eight-connected contours are likely to give a higher compression ratio.

Overlapping vs Non-Overlapping

When using eight-connected contours there is the possibility of contours overlapping without disobeying the rules that pixels are associated with at most one contour. Figure 4.24 demonstrates this with two very simple contours.

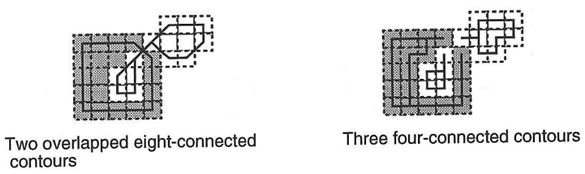


Figure 4.24: Overlapping and Non-overlapping Versions.

This can be very useful for certain images as shown in the chess board in Figure 4.3, but can lead to some nonintuitive behaviour. A few problems become apparent when dealing with overlapping contours:

- As shown in Figure 4.25 because of this overlapping principle, occasionally regions can overrun into already defined spaces.

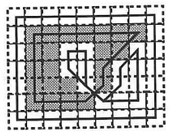


Figure 4.25: Overspill due to Overlapping Contours.

- With overlapping contours there is now an arbitrary choice of filling-in the contour, as shown in Figure 4.26. The choice is arbitrary and in the current implementation it is fixed at not changing the filling-in intensity value.
- Also the structure of the contour tree can be severely "disturbed". Contours can be either wholly or partially within another contour. This means they can have multiple parent contours at any number of different levels in the tree.
- A final aesthetic reason is that people when viewing contour creation on a graphical terminal find the principle of overlapping confusing. This clashes with the aim for the design of the contour tree to be human understandable.

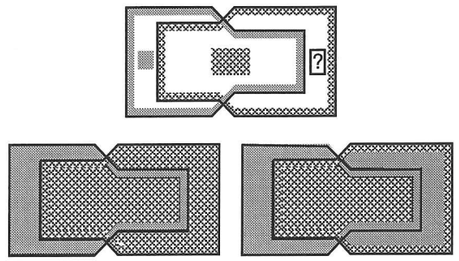


Figure 4.26: Two Choices for Contour Filling.

An intermediary step is to use non-overlapping eight-connected contours. This leads to a choice of coding as shown in Figure 4.27 as there are two possible ways to contourize the image.

Figure 4.27: Choice of Contourization with Non-overlapping Contours.

As both possibilities are available a first-come first-served system was developed. It is necessary to check if an an overlapping contour is about to be formed to prevent its creation. When coding eight-connectedness without overlapping there are more contours to encode, and there are some extra movement restrictions to the boundary chain codes.

4.5.10 Efficient Chain Code Coding

With the stream of movement steps a second stream is created dependent on the above restrictions. Each element of this stream consists of a set of flags indicating impossible moves. This drastically reduces the amount of possibilities as shown by the numbers next to the arrow heads in Figure 4.20[5].

To reduce the entropy of the movement steps relative directions are coded rather than absolute directions, as shown in Figure 4.28. It is worth noting that there is a large skew due to the representation of single pixel contours; *North* by 4MTC4 Fix., *Forward* by 4MTC4 Rel., *NorthEast* by 8MTC8 Fix. and *Forward* by 8MTC8 Rel. To take 4MTC4 as an example, a simple observation shows that in any closed contour there is an equal number of *North* movement steps as *South* movement steps, and an equal number of *East* movement steps as *West* movement steps. Alternatively for relative movements the total number of degrees rotated must equal 360 degrees – *Right* is +90 degrees, *Left* is -90 degrees, *Forwards* is 0 degrees and *Backwards* is 180 degrees.

Using relative moves gives an overall probability skew which can be exploited by entropy coders as shown visually in Figure 4.28, and empirically in Table 4.7. This shows a consistent reduction in entropy from absolute movement steps to relative movement steps.

[5] Used to calculate the number of bits required to code each example contour.

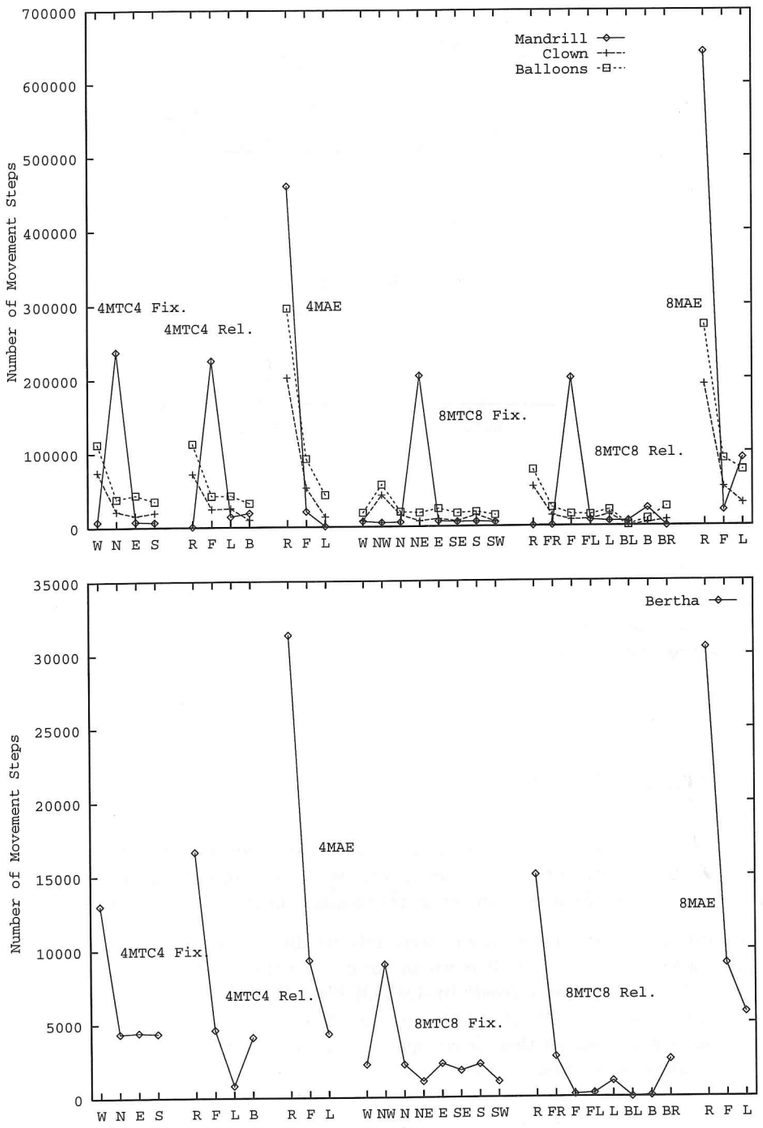


Figure 4.28: Comparison of Fixed and Relative Movement Steps.

Mmt Step	4MTC4 Fix.	4MTC4 Rel.	4MAE	8MTC8 Fix.	8MTC8 Rel.	8MAE
Clown	1.673	1.664	1.766	2.637	2.304	1.171
Bertha	1.796	1.431	1.156	2.590	1.468	1.225
Mandrill	0.556	0.710	0.274	1.156	1.056	0.710
Balloons	1.804	1.802	1.896	2.842	2.509	1.333

Table 4.7: Entropy of the Movement Steps.

Entropy Coding

The last stage of compression is to take the stream of movement steps with known restrictions and encode them without loss as efficiently as possible. Two alternative schemes were investigated:

- The first used a DMM (Dynamic Markov Model) on a reduced character set Arithmetic Coder[6]. The starting model used includes an initially large fan-out of five or more movement steps deep, as shown in Figure 4.29. When a new contour is to be coded the DMM changes state to the initial starting state allowing the model to resynchronise.

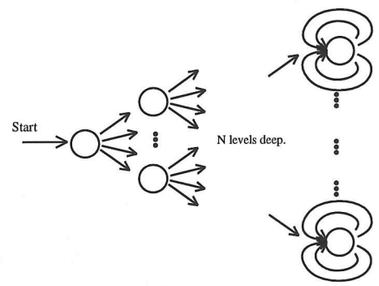


Figure 4.29: Initial Starting State for DMM.

- An alternative scheme used was a reduced character set LZ78 modification (LZT)[7]. Although this coding very rarely improved on the performance of an Arithmetic Coder the scheme is very simple and similar hardware schemes have been built.

4.6 Final Compression Size

Listed are only non-overlapping contours which encode the cases of non-comma-free contours for 4MTC4 and 8MTC8 as two separate contours. The differences between the number of contours to be described are shown in Table 4.8. There is a substantial reduction when eight-connectedness is used.

Image	4MTC4	4MTC8	4MAE	8MTC8	8MAE
Clown	63851	62839	62839	54244	53720
Bertha	9037	9019	9019	8249	8247
Balloons	85930	84311	84311	66335	65790
Mandrill	231086	230791	230791	220190	219859

Table 4.8: Number of Contours for different Contour Coding Movement Codes.

[6] A description of Dynamic Markov Models is given in section A.1.4.

[7] A description of the LZT algorithm is given in Section A.2.2.

The resulting compression ratios for the full images, including start locations and height values are given in Table 4.9[8].

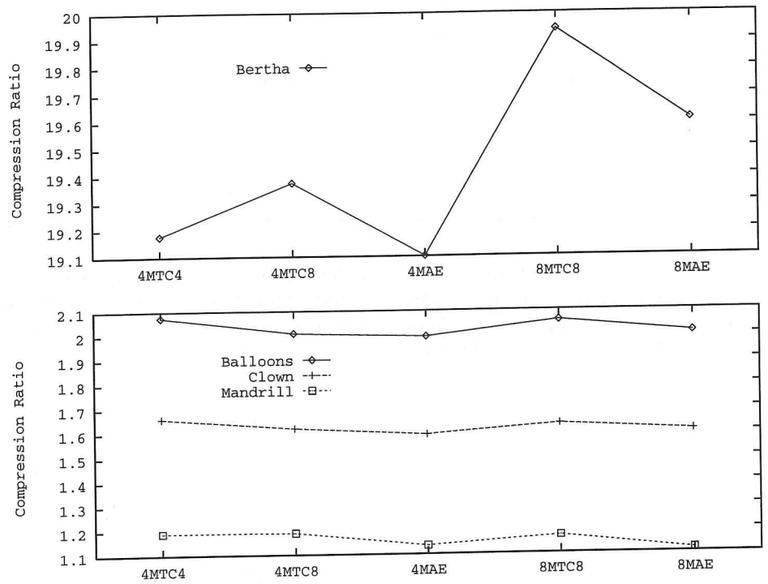


Image	Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE
Clown	109358	*65857*	67531	68685	67001	68183
Clown	1	*1.661*	1.619	1.592	1.632	1.604
Bertha	245768	12816	12686	12867	*12326*	12536
Bertha	1	19.177	19.373	19.101	*19.939*	19.605
Balloons	198664	*95792*	98915	99674	96587	98968
Balloons	1	*2.074*	2.008	1.993	2.057	2.007
Mandrill	245768	*206458*	206641	216577	209817	220603
Mandrill	1	*1.190*	1.189	1.135	1.171	1.114

Table 4.9: Contour Coding Compression Sizes.

For this set of results it has been shown that when there is a significant difference between compression ratios for the five methods, the techniques of 8MAE and 8MTC8 achieve the best results. For those images which are less connected, which includes most captured or scanned images, the method 4MTC4 has been shown to give a slight performance edge. This is to some degree as predicted in Section 4.5.4. Further results comparing the five methods for a variety of different image types are given in Appendix B.

It is worth calculating the values of p, the probability that a neighbouring pixel has the same colour value, as defined in Section 4.5.4. For the images these are listed in Table 4.10. The calculated values and the relative compression performances for the different methods seem to have a slight agreement with the theoretical expectations.

[8] A reminder about compression performance comparisons in tables as defined for Table 3.2. The first line lists the file size in bytes whilst the second line shows the compression ratios achieved. Italicised numbers indicate the technique which gives the highest compression ratio. In this table the compression ratios are also plotted on a graph for easy comparison.

Image	p
Clown	26.8%
Bertha	94.5%
Balloons	33.6%
Mandrill	3.0%

Table 4.10: Probability Values of p.

4.7 Conclusions

During this chapter, some theoretical justification for the preference of eight–connectedness without overlapping and movement around the edges of pixels has been undertaken. Practical demonstration has indicated that this is the correct strategy for a general purpose coding system that gives the best average compression ratio. The compression results achieved using the contour coder compare very favourably with the previously presented results in Table 3.10. The results also out-perform most commercial compressors.

This has demonstrated in a practical sense that there is a two-dimensional correlation in images which is worth the effort to exploit.

Chapter 5

Pruning the Contour Tree

So naturalists observe, a flea
Hath smaller fleas that on him prey;
And these have smaller fleas to bite 'em.
And so proceed ad infinitum.
Thus every poet, in his kind,
Is bit by him that comes behind.
– Jonathan Swift

5.1 Lossy Image Compression

For continuous-tone images the resulting contour tree, due to inherent noise in the image acquisition hardware, can be very flat, consisting of many nodes. This is shown for the Mandrill image, whose contour tree is shown in Figure 5.2, and consists of over 91% single pixel contours. Presented here is a process to simplify the contour tree increasing the compression ratio achieved from a contour coder. The initial ideas were first presented publically in [Turner, 1992], and described briefly in [Purcell and Wiseman, 1992, page 3].

5.2 Contour Merging to Aid Lossy Compression

Given a contour tree, for each contour we can assign a rating of noticeability according to its size, actual intensity and neighbourhood. It can then be decided if two neighbouring contours when merged, creating a joint contour of equal mean intensity, are noticed as different by a human observer up to some fidelity criteria. This gives us a way of calculating a merging threshold value. By repeated merging until all contours can not merge any further without going over this fidelity criteria, the resulting contour tree is pruned. Each node is usually more complicated, but contour coding should result in a higher compression ratio.

This is akin to work carried out by [Samet, 1985] on quadtrees, where under certain conditions the quadtree is repeatedly pruned giving a simplified tree. The concept is very similar except that the choice of pruning is gained from the actual structure of the image rather than just from the local pixel neighbourhood. The problem is deciding what criteria constitutes the level of noticeability for the merging of two contours. The criteria uses the following rules:[1]

1. The size of the larger contour is irrelevant as the eye concerns itself with local changes. It has been observed that details attract attention [Yarbus, 1967].

[1] General rules regarding the physiological and psychological aspects of the eye were extracted mainly from [Boff and Lincoln, 1988], with countless exceptions from other sources mainly the Physiology Dept, Cambridge, UK.

2. The size of the smaller contour is critical and its area is proportional with the possibility of being recognised as a separate region. Experiments have shown that noticeability of an object in the human eye is proportional to its area.

3. The intensity ratio over background intensity is treated proportionally with the possibility of merger. It was discovered that average light intensity seems to follow Weber's Rule [Boff and Lincoln, 1988, Vol 1:632]:

$$\frac{\delta I}{I} = n \; a \; constant$$

This means that there exists a value n under certain conditions such that a change in intensity δI is only noticed when it is greater than $n\%$ of the background intensity I. Experimentally this has been shown to be accurate only for average light intensities and short exposure times.

4. The Mach factor is considered for neighbouring intensities, which states that light objects appear lighter at the boundary with darker objects and vice-versa. This is illustrated in Figure 5.1. This phenomenon was first described and named in [Mach, 1865] and later described in more detail in [Ratliff, 1965] but had been clearly and accurately described by Leonardo da Vinci in 1508.

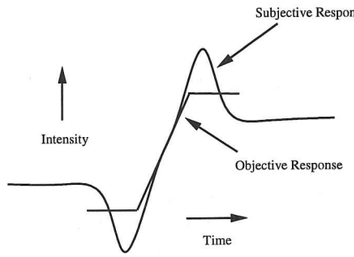


Figure 5.1: Actual and Subjective Intensities Due to the Mach Effect.

The values for the overshoot of the subjective response and their position is very subject dependent. Overshoot values of 50% the difference in intensity for bright edges, and 25% the difference in intensity for dark edges is considered. This means that intensity values brighter than the brightest and darker than the darkest have to be stored and dealt with within the contour structure.

5. The maximum "historic" difference of contour intensity values is considered, not just the current value for each contour. This means as a contour expands the intensities of its constituent contours are still considered.

A single threshold value is specified and two contours are merged when the deciding algorithm returns a smaller number. The resulting image can then be passed to the previously described contour coder. Figure 5.2 shows a tree structure representation for the Mandrill image before and after merging. The ideal way for the contour merger to operate is to repeatedly take the two contours which have the smallest merging cost and merge them until the threshold value is exceeded. This continual sorting would be excessive, so a simple queue is constructed which lists all contours in order of size. If the smallest contour on the top can be merged it is and the resulting larger contour is sent to the back of the queue, else it is removed from the queue. Once the queue is empty the image is fully contour merged to the defined threshold.

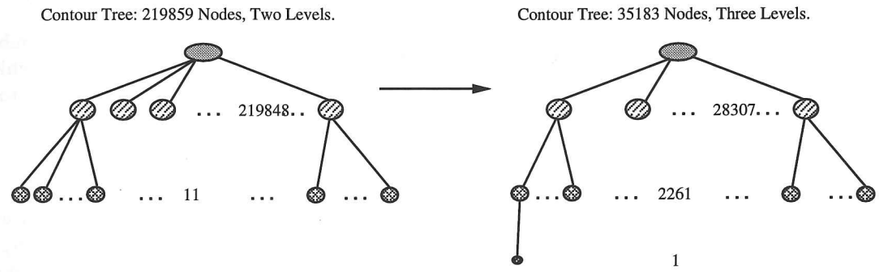


Figure 5.2: Contour Tree for Mandrill Image Before and After Contour Merging.

5.3 Operational Costs

The two operations of contour merging and contour coding are quite separate. Contour merging operations purely pre-filter the image before being transmitted or stored, as shown in Figure 5.3. The critical time operations for transmission and storage then are the contour encoding and decoding times.

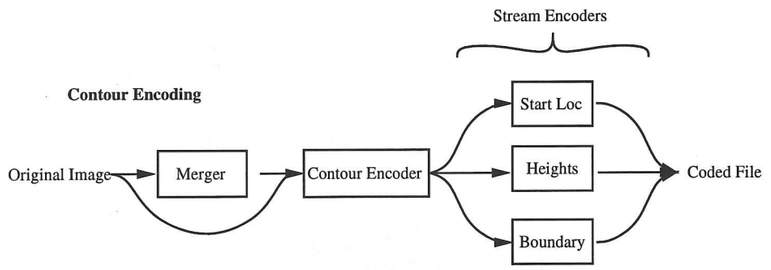


Figure 5.3: Schematic Diagram of the Contour Encoding with and without Application of the Pre-manipulating Contour Merging Algorithm.

As shown in Chapter 4 the algorithms for contour coding deal with images pixel by pixel which makes the computational time proportional to the number of pixels in the image. The three streams can be coded and transmitted separately splitting the work required to three linear probability coders. The coding process as it presently works requires frame store for the entire picture to be held both at the encoder and decoder, which with present systems is not an unacceptable amount of storage.

The costs for the contour merging algorithm are more severe being $O(n\bar{l})$, where n is the number of contours and $\bar{l}$ is the average length of the resulting contour boundary descriptions. As the contour merging process needs only to be carried out once in the lifetime of the image, the effort involved is not as critical as it might be.

5.4 Results

For many computer generated images, even rendered and ray-traced images, the resulting compression ratio of the original is, as already shown, reasonably high. Real images, which have been

captured with cameras or scanners, have their own biases and noise that lead to very large and flat contour trees.

As the threshold or quality values are decreased there is a consistent reduction in number of contours, and a corresponding increase in contour size. This is shown in Figure 5.4 which demonstrates that the progression from high quality to low quality is smooth, and relates to a smaller and smaller sized tree.

5.4.1 Analysis

To define quality of an image simplified by the contour merging algorithm the term *excellent* was devised. This has the simple criteria that an observer can not tell the original from the merged version when they are shown separately. These images are shown in Figures 5.5 and 5.6 and Colour Plates 1 and 2[2]. Table 5.1 quantifies the number of contours and compression ratios achieved, in the original (first line) and "excellent" images (second line). The hierarchical depth shows the number of contours at each level of the contour tree.

Image	*Size in bytes*	*Compression Ratio*	*Total Number of Contours*	*Hierarchical Depth* 1	2	3	4
Balloons	198664	2.286	65790	62616	3147	27	
		4.729	6689	5707	978	4	
Escher	138632	1.488	86592	1	85797	794	
		4.479	6658	1	5968	688	1
Garden	49160	1.402	38050	37938	121		
		4.253	2289	2145	144		
Mandrill	245768	1.254	219859	219848	11		
		3.807	13546	11936	1609	1	

Table 5.1: Compression Ratios: Non-Conservative.

5.4.2 Visual Artifacts

Those images which are "excellent" can still have up to one fifteenth the number of contours which with the contour coder gives compression ratios between 4:1 and 5:1. Further compression ratios have been achieved using dithering techniques on the simplified contour tree as described in Section 7.3.2. When larger threshold values are given to the contour merging algorithm, artifacts become noticeable. The main artifacts are:

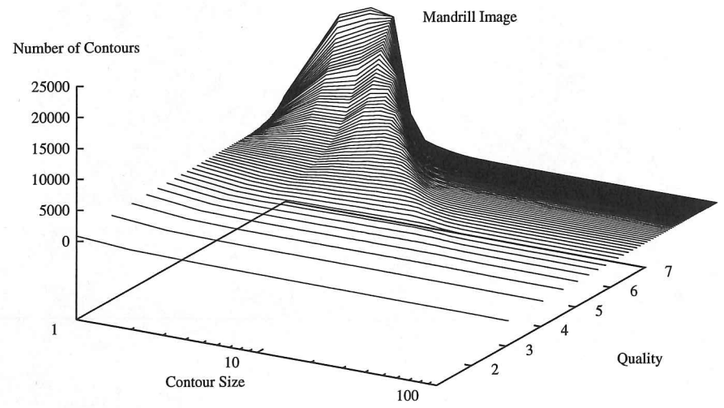
Loss of Highlights as in the flattening of the pupil of the Mandrill's eye, shown in Figure 5.7. With the removal of the Mach band parameters the resulting loss of highlights is greatly increased.

Loss of Connectivity between regions which the mind psychologically fills in. This can be seen in the top railing on the Escher image in Figure 5.8. Merging operations on the left and right of the already partially disconnected railing separates it even further.

Blotching occurs due to a high level of merging which is noticeable in areas of randomness. An example is in the flower bed of the Garden image, as shown in Figure 5.9.

False Contouring occurs when smoothly shaded areas group into regions. Noticeable on low quality printers which add to the effect, due to intensity quantisation. This effect is reduced drastically by using eight-connected contours as the jagged edges diffuse the false contour.

[2]Both four-connected and eight-connected contour mergers have been produced and the results shown are using the eight-connected contour merger. Optimal coding is achieved by using a Movement Around Edges coding scheme as due to the merging process there are very few small contours.



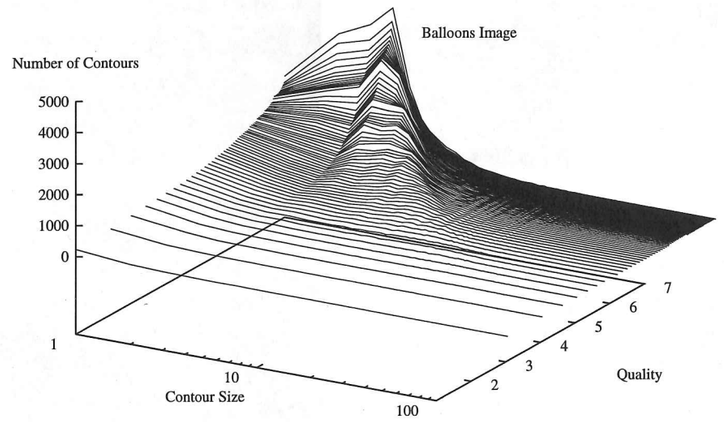


Figure 5.4: Contour Size vs Threshold Value for Mandrill and Balloons Image.

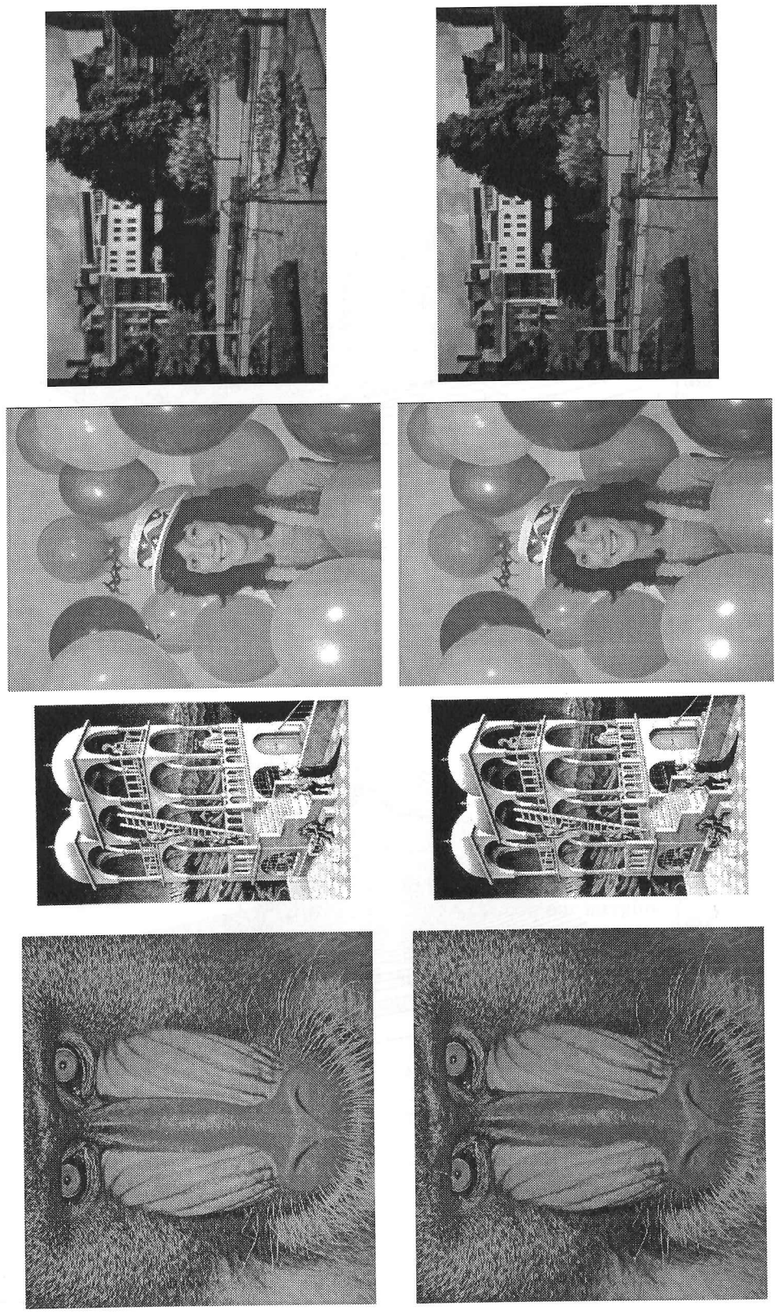


Figure 5.5: Comparison of Originals with *Excellent* Versions: Normal Colour Map. Garden Image Enlarged by a Factor of Two.

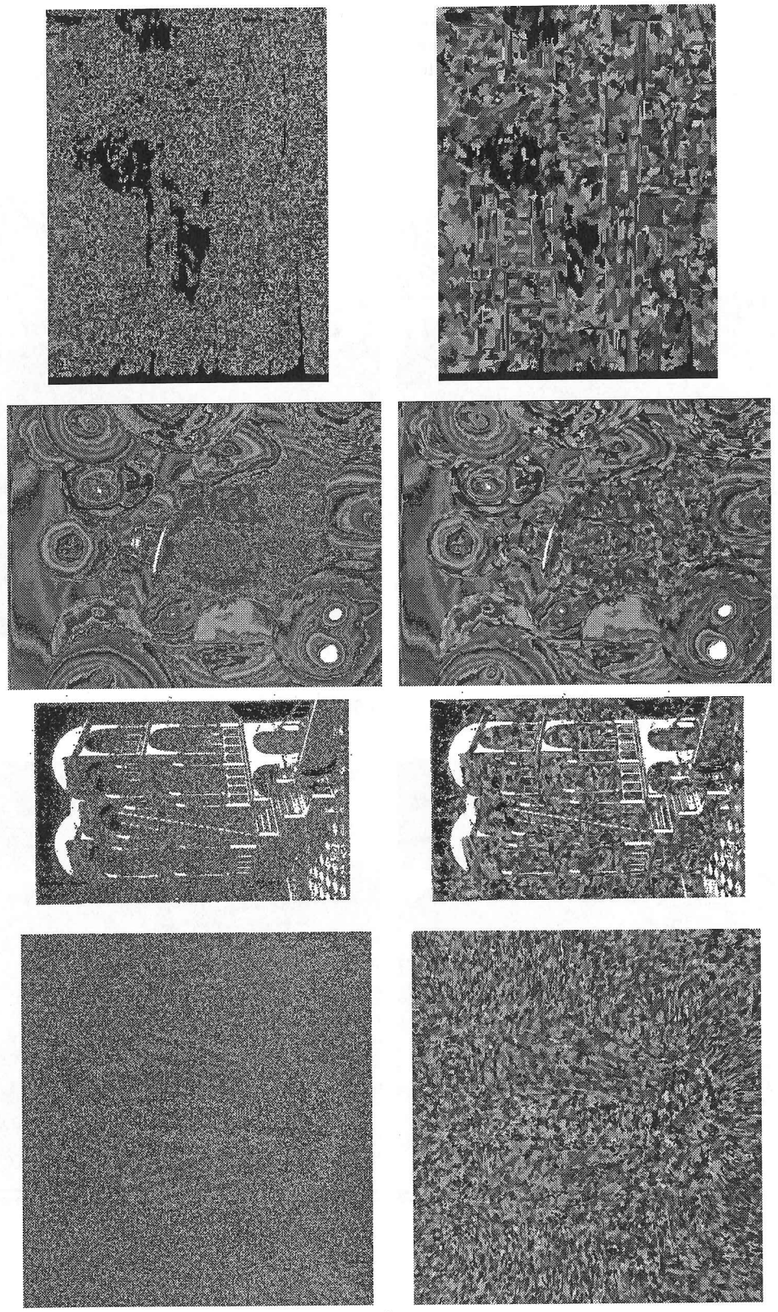
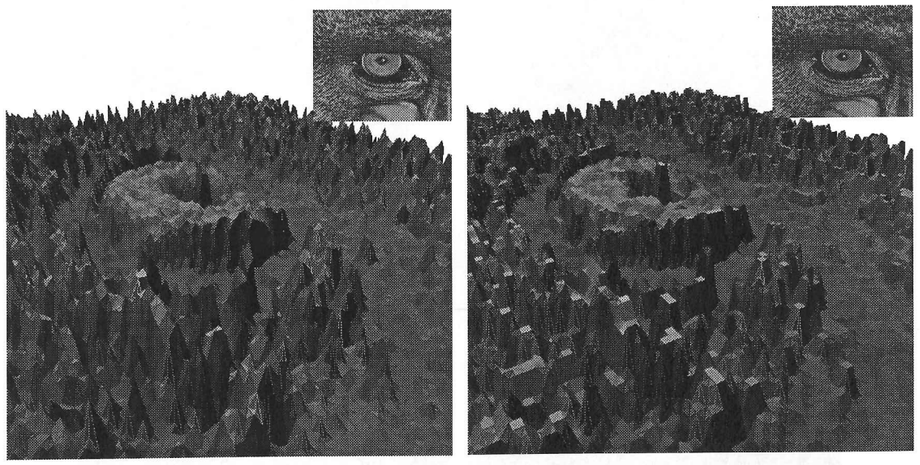


Figure 5.6: Comparison of Originals with *Excellent* Versions: Random Colour Map. Garden Image Enlarged by a Factor of Two.

The 3D View Shows the Contours around the Mandrill's Right Eye as seen when Standing above the Bridge of the Nose.

Original 1.115:1 2.778:1



"Excellent" 3.807:1 5.689:1

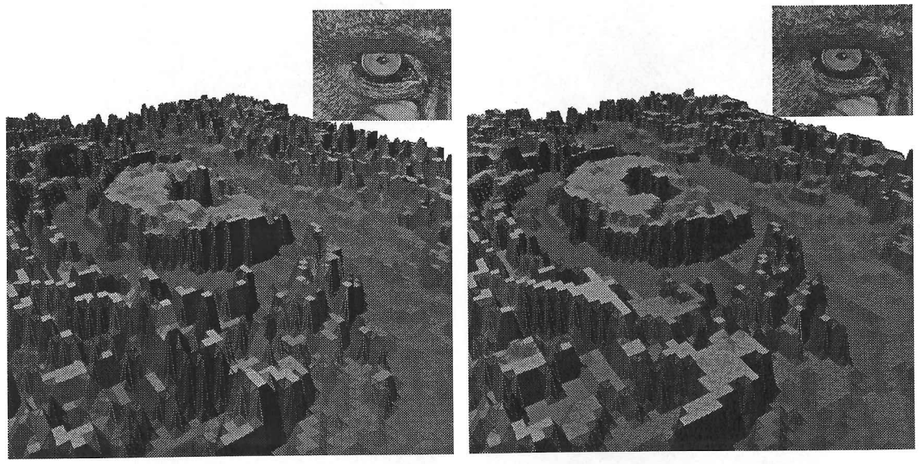


Figure 5.7: A Sequence of 3D Representations of an Image which Demonstrates how Increased Threshold Values Affect the Contour Merging Process.

Two features which the merging algorithm exploits with its simple set of rules:

Edges are Sharpened up which according to [Marr, 1982, Marr and Hildreth, 1980] and others is particularly important in the recognition action of the human brain, and thus aid perception and image understanding.

Texture Shape via the contour merging algorithm is detected at the low level, and the merging process flows with it.

The extraction of texture information is a very interesting side effect. If one false colours the contours the texture is displayed as long thin contours that describe the flow of texture granularity. This is shown in great effect in the grassy parts of the Garden image and the hairy parts in the Mandrill image. Further details of texture highlighting are described in Section 7.4

5.5 Comparison with JPEG

A common method for lossy compression is to employ the CCITT standard lossy continuous compression method T.81 devised by JPEG (Joint Photographic Expert Group) [CCITT, a] and explained accurately and clearly in [Pennebaker and Mitchell, 1993]. This method uses a blocking structure on the image, and translates each 8×8 block into its Discrete Cosine Transform and quantises the coefficients before entropy coding the final stream. Artifacts introduced by JPEG include:

- Blocking structure of the original image appearing due to discontinuities at the 8×8 block edges.
- Blurring of the image as the high frequency components are reduced.
- Overshoots in the inverse transform due to changes in the components coefficients.

These artifact can be seen in Figure 5.10 around the Mandrill's eye in both two dimensional and three dimensional visualisation[3]. The artifacts introduced by the two methods are very different and can be considered complimentary. Images that are unsuitable due to their content for lossy compression using the JPEG method, may be better suited to being contour merged and coded.

5.6 Quantitatively Analysis

At present a quality rating is taken directly from the threshold values and "excellent" status is achieved when viewers believe they are the same image with no noticeable artifacts when shown separately.

This criteria is very prone to many external factors, but does bring to light some interesting non-linear anomalies. For example, the level of acceptable distortion on the Mandrill's face can be higher then that on the model's face in the Balloons image. This is possibly because a human face is much more common and discernable then that of a mandrill's. Ignoring these problems it is worth attempting to find an objective means to quantify the amount of artifacts. This would enable a system to automatically contour merge images to a specified visual quality rating.

5.6.1 Univariant Quality Rating

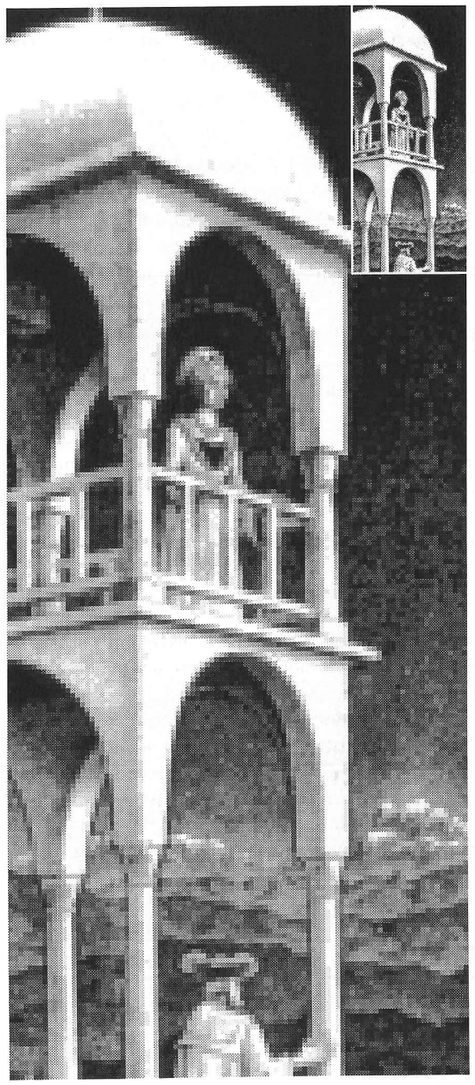
To achieve a univariant quality rating, for the degraded image, we need to devise a formula $\odot()$, such that given $f_{i,j}$ is the original and $\overline{f}_{i,j}$ is the degraded image, the quality rating Q is defined as:

$$Q = \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \odot(f_{i,j}, \overline{f}_{i,j}) \tag{5.1}$$

[3]The JPEG quantisation tables used where those suggested in the guidelines. Image dependent tables have been extensively studied at NASA Ames, which give a slight improvement in compression ratios [Watson, 1993, Solomon et al., 1994, Watson et al., 1994].

The image on the right has been coded using the contour merging algorithm. Visual artifacts include loss of connectivity, slight false contouring and loss of highlight.

1.367:1



6.685:1

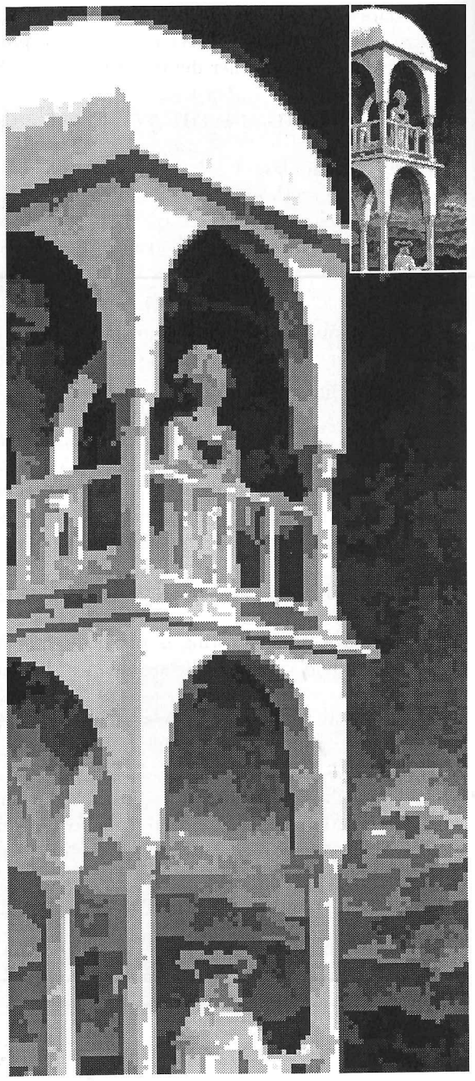
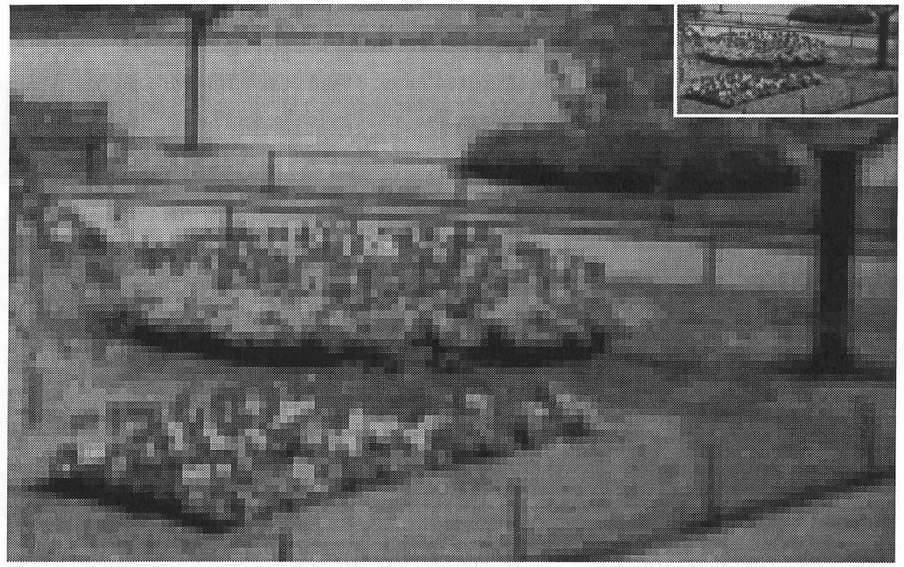


Figure 5.8: A Blown up Section of the Escher Image.

The flower bed becomes blotchy with a large threshold value.

1.160:1



5.873:1

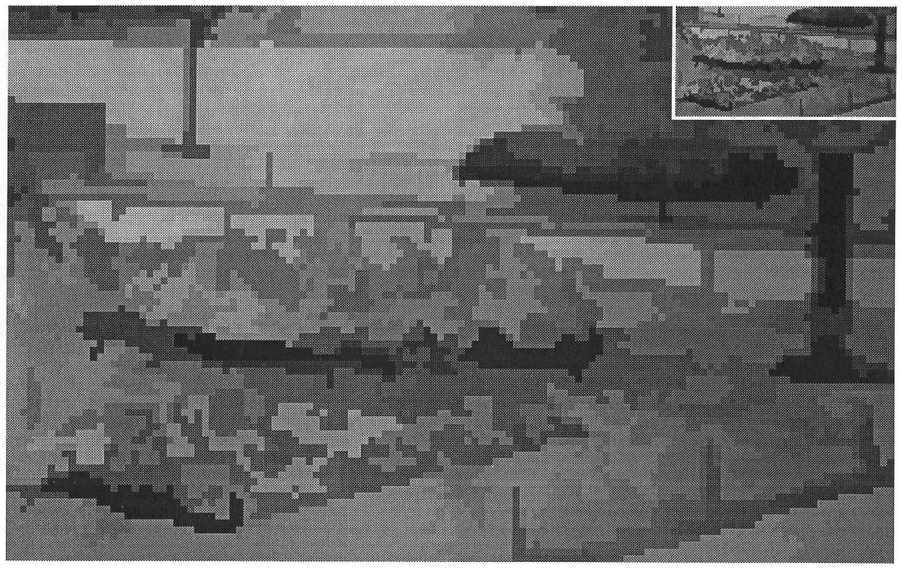
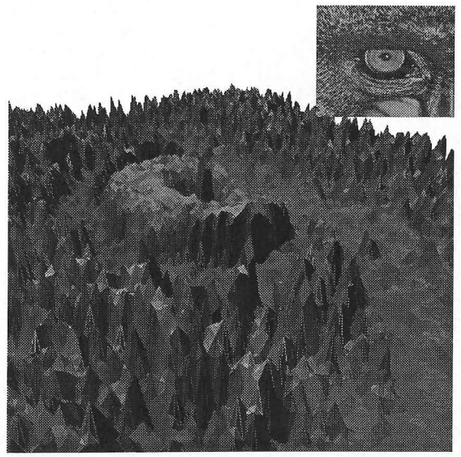
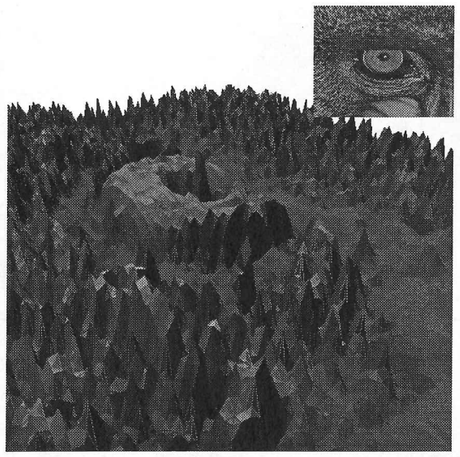


Figure 5.9: A Blown up Section of the Garden Image.

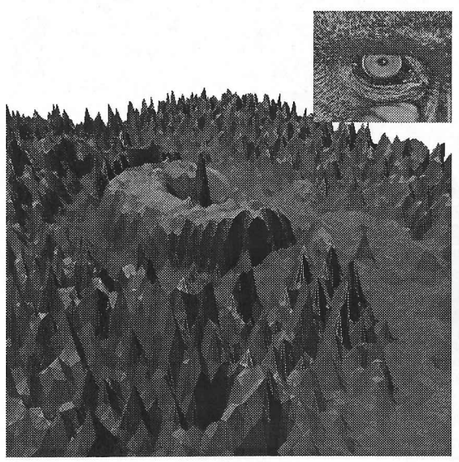
3.310:1



4.989:1



6.621:1



10.586:1

Figure 5.10: JPEG on the Mandrill Eye.

Some of the possible operators are listed and studied in [Eskicioglu and Fisher, 1993]. A common set of error-measures is one based on L_p-norm. L_1 is absolute error, and L_2 is root-mean-square error.

$$L_p = \left(\frac{1}{M \times N} \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \left| f_{i,j} - \overline{f}_{i,j} \right|^p \right)^{\frac{1}{p}} \tag{5.2}$$

Other suggestions for this metric include the popular signal-to-noise ratio. This is defined by normalising the signal power by the error power and taking a scaled logarithm measured in dB. It is assumed that the signal power is the same as the variance σ_f^2 of the signal samples, so:

$$\begin{aligned} \text{SNR} &= 10 \log_{10} \frac{E(f^2)}{D} && (5.3) \\ &= 10 \log_{10} \frac{E(f^2)}{\frac{1}{N \times M} \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \left| f_{i,j} - \overline{f_{i,j}} \right|^2} && (5.4) \\ &= 10 \log_{10} \frac{\sigma_f^2}{\sigma_{f-\overline{f}}^2} && (5.5) \end{aligned}$$

These quality metrics, although popular, have been shown in the literature to be very inadequate quality performance measures. L_p-norms and SNR for the "excellent" versions of the test images are listed in Table 5.2.

Image	L_1	L_2	L_3	L_4	L_5	SNR
Balloons	1.451	3.365	5.776	8.614	8.614	21.076
Escher	7.478	12.762	17.592	22.229	26.678	16.089
Garden	7.262	10.638	13.815	16.861	19.767	14.619
Mandrill	8.660	12.176	15.472	18.573	21.511	9.029

Table 5.2: L_p-norm and SNR Error Measurements.

5.6.2 Knee Points

An idea to automatically find an acceptable level of quality without having to resort to subjective experiments was presented during a NASA workshop [NASA Workshop, 1993]. It was suggested to plot a numerical level of quality verses compression ratio achieved. This should have a smooth almost linear curve with a knee point, before which the compression ratio steeply rises and after which it shallowly reduces. This is meant to represent the point at which all the redundant noise in the image has been removed or replaced. The idea is that at this point the compression method has reached its limit before the actual image signal is corrupted, and thus the quality should never be lowered beyond this point.

Plotting compression ratios against the contour merging quality values and L_2 values are shown in Figures 5.11 and 5.12. The arrows point to the threshold values which were considered to give "excellent" images. The graphs are far from conclusive as the change in curvature is very slight around these points. Some very slight correlations exist with the contour merging curves but nothing that offers an automated system. The graphs do show that L_2 values are definitely a poor and badly over used measure of quality. None of these metrics can be said at present to be automatable and using the threshold value as a new metric of quality is not a conclusive solution.

A preliminary report to address this problem [Eskicioglu et al., 1994] has been published from the University of North Texas. Their aim is to devise and test quality functions which are

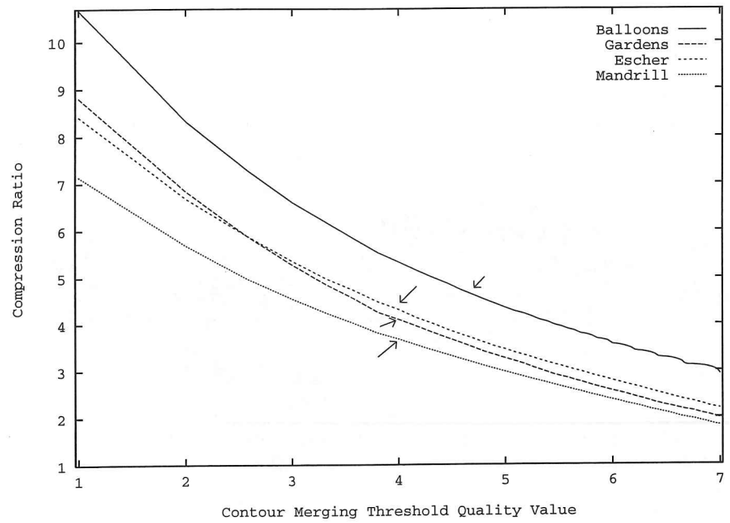


Figure 5.11: Knee Point for Contour Merging Threshold Values.

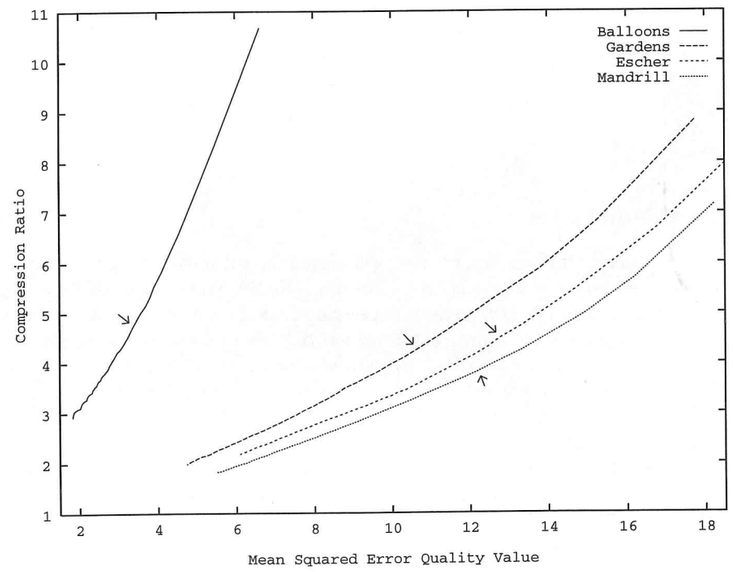


Figure 5.12: Knee Point for L_2.

related to a more complex HVS (Human Visual System) model, rather than a mean-square-error model. Initial work with HVS models have been proposed using modified mean square errors [Saghri et al., 1989], cosine transform models [Ngan et al., 1989] or power spectrum filters [Nill and Bouzas, 1992]. A good review of current techniques is given in [Jayant et al., 1993].

I now present two asides to partially justify the choices taken by the contour merging algorithm, and also highlight some of the inherent disadvantages. These include a physiological and psychological view of the human vision system.

5.7 Physiological Considerations

It has been an aim of lossy image compression that if we understood the information which made a complete representation of an image, as seen by the human visual system, then we would be able to create a set of images which were perceived as exactly the same. Current physiological knowledge is nowhere near that goal yet, and at present we are just touching the surface; in a physical as well as meta-physical sense. This does not mean that inferences and hints can not be taken from the knowledge already gathered. Described here are features which have been discovered that emphasise the importance of edges and contours.

The transfer of information from an eye starts in about 125 million rods and cones in the retina. These are compressed into one million ganglion cells which travel up the optic nerve to the Lateral Geniculate body in the centre of the brain before continuing into the Primary Visual Cortex structure at the back of the brain. The Visual Cortex has been divided into, at present, 32 physically different regions[4] [van Essen, 1993]. Information is thought to flow through these regions creating, for the mind, a good "picture" of the view presented. The next section describes the journey of a quantum of light through the visual system.[5]

5.7.1 The Eye

As the quantum of light passes through the pupil of the eye into the lens, avoiding being absorbed in the cornea or lens, it is focussed onto the retina and onto the rods and cones. If it was able to pause for a while the quantum may marvel at the precision engineering it has entered. At its most receptive the human eye is at the diffraction limit for its pupil's size[6] and the rods are so sensitive that a single quantum of light can be magnified and trigger a perceptible response. The next stage is to negotiate the 125:1 reduction in lines from the rods and cones to the optic nerve, which actually requires about 250:1 compression due to differences in signal rate. This is achieved by three layers of intermediate cells, situated between the receptors in the retina and the ganglion cells in the optic nerve. The following techniques are employed:

- Not only are the photoreceptors more numerous and tightly packed around the fovea or centre of vision, but the cells are sampled and averaged at coarser levels further out. In the monkey there are two distinct types of cells one small P, and the other large M, which are found to give emphasis to high and low spatial frequencies respectfully.
- The photoreceptors trigger an excited response when stimulated, whilst the ganglion cells have a receptive field which is surrounded by an inhibiting neighbourhood. The centres as shown in Figure 5.13 can be either excited by a stimulus or inhibited. These have been termed on- or off-centre cells. This means that many photo-receptors contribute to a single complex signal[7].

[4] The exact number of regions is hotly disputed and estimates change all the time.

[5] Most research that has been undertaken concerns itself with the macaque monkey brain instead of with the human brain, which is moderately similar.

[6] In the fovea, the centre of vision, cones have centre-to-centre spacing of about 2.5 micrometers giving us the ability to separate two points as close as 0.5 minutes of arc.

[7] On average the full receptive field area of a ganglion cell is 50 times larger then that of the photo-receptors.

Figure 5.13: Centre-Surround, Ganglion Cells.

- To solve sampling problems that cause serious aliasing effects, there is a rich amount of overlapping involved. Each receptor contributes a little to possibly many ganglion cells.

This pre-processing could be considered as applying a slightly disjoint hierarchy of linear filters to the image, with a strong bias around the fovea which is the eye's focus of attention.

5.7.2 Optic Nerve and Lateral Genicular Nucleus

The quantum has been transformed into a part of a set of signals, each which carries information on the local retina area where the quantum hit. Following one of the signals, it travels first through the optic chasm where the two optic nerves for the two eyes cross over. Signals from the right hand side of the viewable image are channelled to the left hand side of the brain and those from the left hand side of the viewable image are channelled to the right hand side of the brain. When the bundles reach the Lateral Genicular Nucleus (LGN), a large area in the lower back of the brain, they fan out topologically and enter the first part of the Visual Cortex, V1[8].

5.7.3 Visual Cortex – V1

The area V1 can be shown to map out the topological structure of the eye, but the input connections are about 1000 times more numerous than the ganglion cells entering it. There is no firm idea as to what the expansion criteria is, but John G. Robson[9] and colleagues suggest that a full set of transforms are expanded out in a similar manner to laying out a full Vector Quantisation dictionary. This would allow further processing levels to take information directly from V1, invariant to a large degree on the exact number of photo-receptors and the details of compression that has just occurred. In V1, cell complexity has changed, as was first discovered by Hubel and Wiesel in 1956 [Hubel and Wiesel, 1959, Hubel, 1990], with the existence of predominantly two different types of cell:

Simple cells are often termed as "bar" recognisers as they are most receptive to a bar of specific thickness with inhibitors flanking the bar. Dark/light edges also are specified in this category. The exact orientation is required for maximum response and they operate over limited regions of the receptive field. Examples of the three main types are shown in Figure 5.14.

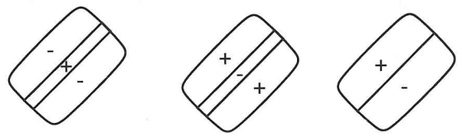


Figure 5.14: Simple Cells.

[8]It is believed that there is a high degree of information feedback in the regions of the Visual Cortex and LGN whose purpose is at present unknown.

[9]Professor in the Physiology Department, University of Cambridge, UK.

Complex cells are similar to simple cells but require the bar to be swept across the receptive field. Position is not as critical as for simple cells, removing the rigid placements of on and off areas but, correct orientation is still essential. The receptive fields tend to be only slightly larger, and optimum bar sizes are comparable. This is shown in Figure 5.15.

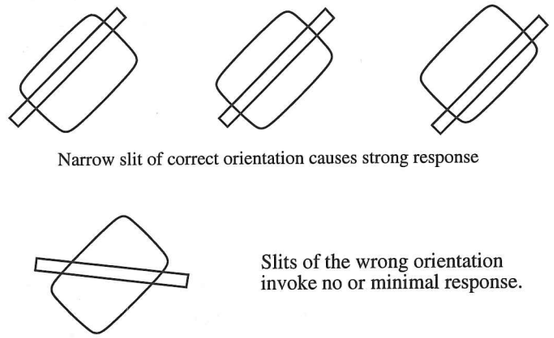


Figure 5.15: Complex Cells.

Variants of complex cells that have been discovered which can differentiate the movement direction of the bar, and some called **End-stopped** which prefer certain lengths. Normally increasing the bar increases the response until the bar is no longer within the receptive field. For an end-stopped cell once a certain length has been reached, increasing the length further results in an inhibition of the signal sometimes reducing the signal to nothing. Inhibiting regions can be at one or more commonly at both ends of the receptive field.

V1 accounts for about 10% of the visual cortex, and due to the folds of the brain it rests almost adjacent to the next region, V2, which is similar in size.

5.7.4 Visual Cortex – V2

Area V2 appears to continue the low level processing of vision cells, before more specialised regions extract relevant features for image understanding.

The cells in V2 appear to have an extra degree of complexity. Recently there has been the discovery and classification of a new set called **Contour cells** by Friedrick Heitger and others [Heitger et al., 1992]. This cell is similar to a complex cell except that it does not just signal when a rod is present but also when only bits of the rod exist. Simulating these cells and applying them to images they can be used to detect and fill in occluded edges. This possibly explains how illusions such as the kanitza triangle may work as shown in Figure 5.16. The dotted lines simulate the imaginary stimuli created by the contour cells to aid the extraction of the triangle and three complete circles.

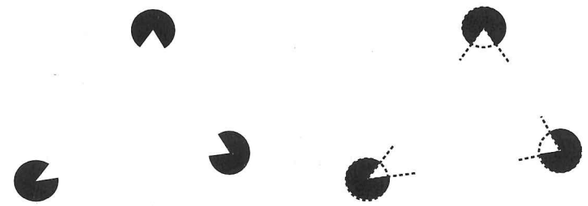


Figure 5.16: Triangle Illusion. Dotted Lines Show Inferences from Contour Cells.

5.7.5 Higher Visual Cortex

As the signals progress out of V2 they appear to travel down two main different styles of routes. One tries to find out "What" it is whilst the other tries to find out "Where" it is. These routes process features including motion, depth, colour and form. A very simplified schematic adapted from [DeYoe and van Essen, 1988] and [Kandel et al., 1991] is presented in Figure 5.17. This schematic does not show either feedback communication or cross talk between the routes which definitely exists.

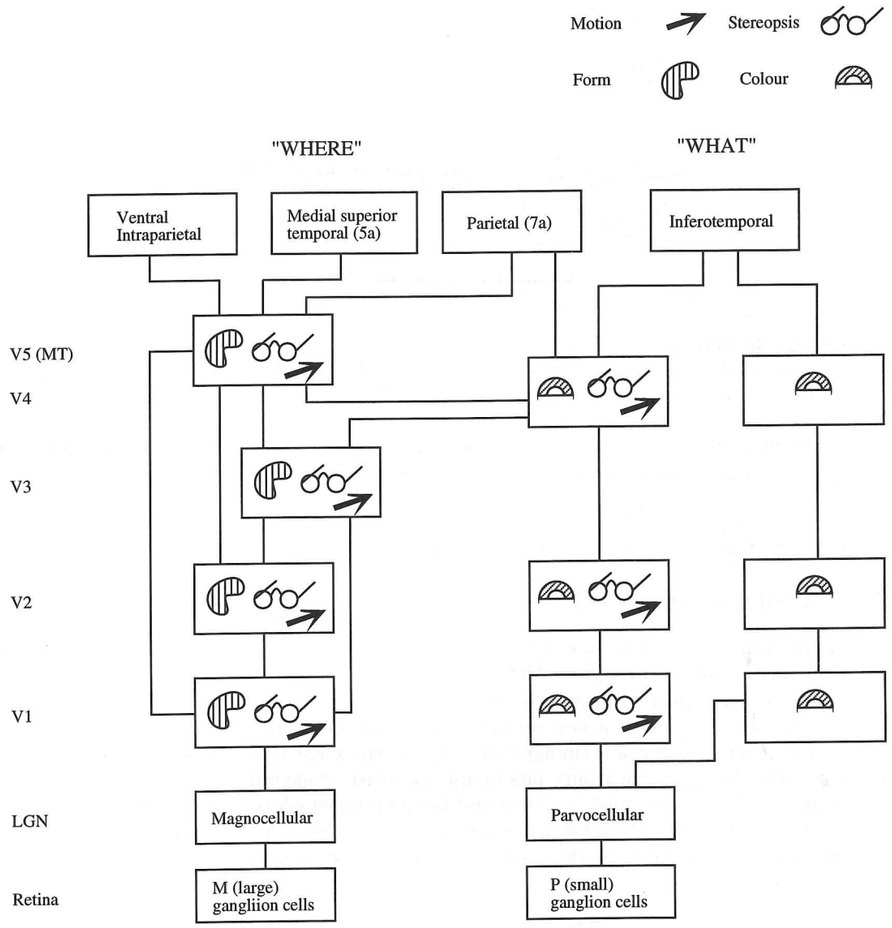


Figure 5.17: Parallel Visual Pathways and their Suggested Functions.

Finally once all the features have been extracted, actions resulting from this information can be made by the brain. This could involve moving attention to another part of the image. The eye has a standard update rate of about 100ms during normal vision.

As a final point it is known that by training and concentrating the human or monkey brain can become hyper-sensitive to certain stimulus. This means that due to a life time of training, both in the short term and long term, everybody sees objects differently relating to age, sex, extra

chemical stimulus, living conditions and all previous experiences. These differences are described in [Davidoff, 1975].

5.7.6 Edges and Contours in the Vision System

In the first stages of the visual pathway up to the V2 level there has been an emphasis for edge, contour and obscured contour detectors. It can be speculated that end-stopped cells and standard complex cells can be employed to detect curvature and edges in shapes, as shown in Figure 5.18.

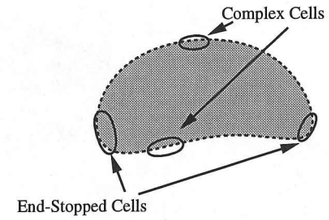


Figure 5.18: Shape Detecting.

The contour cells allow for gaps in lines or areas to be missing or partially incomplete. This infers that merging together an edge which was made up of slices, as carried out in the contour merging algorithm, may actually promote a very similar neural response.

It is also noted that that cells are often modelled using a set of Gabor functions. These are chosen to match the response signal as closely as possible. This indicates that some form of local transform is taking place, and shows the possible advantages to using the quantised Discrete Cosine Transform or Subband coding methods.

5.8 Psychological Considerations

Before physiologists developed techniques to explore the workings of the visual pathway psychological phenomena had been characterised. The parameters, for the contour merging algorithm, are defined from psychological knowledge and supported by physiological evidence. I present a few illusions that demonstrate some of the reasons which support this idea, and describe mechanisms for the visual artifacts which occur during the contour merging process.

5.8.1 Loss of Highlights

We have already seen the Mach banding phenomena which allows us to perceive intensities that are brighter then the maximum brightness, or darker than the minimum brightness. The illusions are very well demonstrated by Cornsweet [Cornsweet, 1970], who showed how to create areas which looked the same when they were not and vice-versa. A simple example, adapted from [Stockholm Jr., 1972, pages 833–834], is shown in Figure 5.19. This shows two small square boxes which have the same intensity value, but appear to have different values depending on the background intensity. This is explained by the fact that differences in intensity rather than constant intensity play a much more important role in the visual system. as has been shown in the neurological cell response fields described earlier. This has been shown to be the case very early on in the human visual system starting at the centre surround ganglion cells. Increasing the values of the Mach parameters, to the current settings of brighter overshoots being 50% the difference in

intensity for bright edges and 25% the difference in intensity for dark edges, has been shown to retain most if not all of the important highlights.

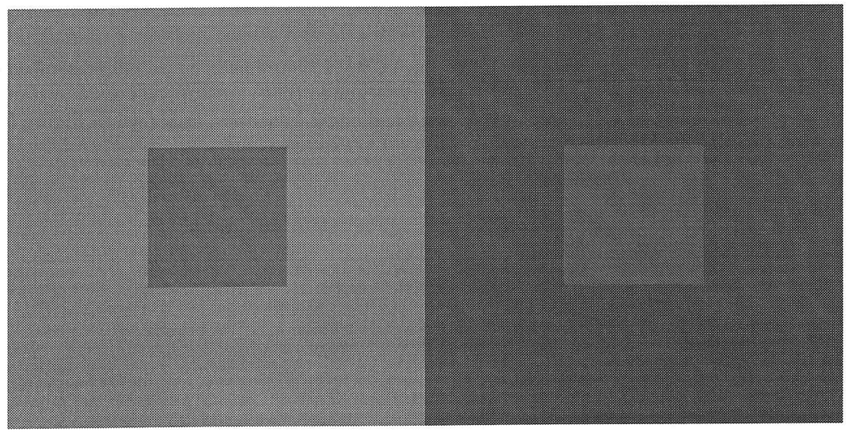


Figure 5.19: Mach Illusion.

5.8.2 Loss of Connectivity

The kanitza triangle illusion was shown in Figure 5.16 which demonstrated how we perceive unconnected regions as connected, and is an example of *Amodal Completion.* This deals with the way one object influences another in our perception,[10] and relates to Gestalt theory[11] which interprets reality in terms of wholes, shapes and structures, maintaining that elements should not be considered in isolation but rather as parts of a wider and more global context. This leads to illusions which join actually unconnected objects as in Figure 5.20, where the four segments seem to join forming a cross but in reality they are disconnected.

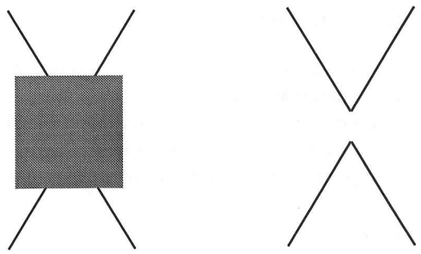


Figure 5.20: Connected or Unconnected Crosses?

As the actual discontinuity increases in size the psychological completion process fails. It may be noted that the concepts of amodal completion are applicable in the time domain as long as the

[10]Demonstrations from Albert Michotte founder of *Institut de Psychologie Appliquée at de Pédagogie* of the University of Louvain and the *Société Belge de Psychologie* and president of the *International Union for Scientific Psychology.*

[11]Gestalt theory is a little dated as a philosophical technique but its ideas are still relevant. An English introduction is found in [Ellis, 1938, Selection 1] and ideas relating to cognitive processes are described in [Henle, 1961, Part III].

time delay is not too large. The artifacts of "loss of connectivity" as described in Section 5.4.2 are likely to result directly from the break down in amodal completion due to the merging process.

5.8.3 Blotching Artifacts

Blotching is the only artifact that can not be slightly appeased by careful application of parameters. It gives a feel of *Painting by Numbers* where subtle local changes which influenced the image as a whole have been removed. For most objects this is perceived only at a high level of merging, but on very familiar images including human faces the affects can be very noticeable.

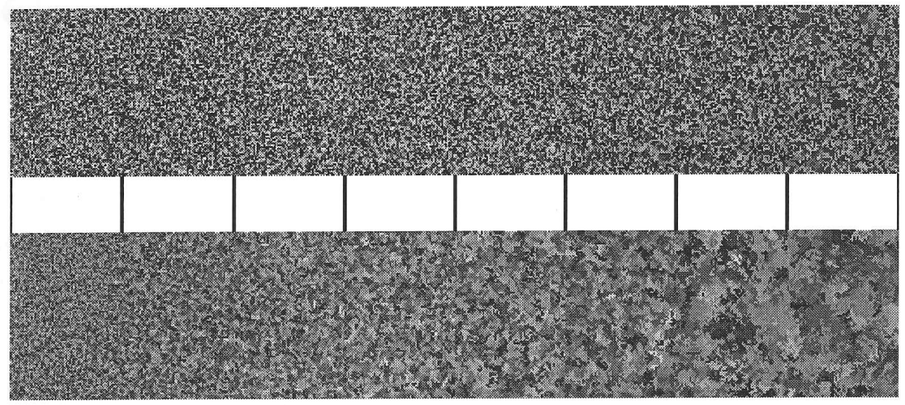


Figure 5.21: Blotching on a Random image.

An interesting demonstration is shown in Figure 5.21. Displayed are eight slices of the Random image; one slice of the original and seven slices from images that have been contour merged with increasingly higher threshold levels. The bottom image consists of exactly the same slices except that a random gray-scale colour map has been applied to highlight the individual contours. As the merging threshold value is increased blotching affects become evident, shown on the slices on the far right[12]. Blotching has been found to be the effective limiting factor of the contour merger as a visual simplification process.

5.9 Conclusions

I have demonstrated a new technique in lossy image compression that offers an alternative to other currently used methods. The results show that when combined with the contour coder it is a viable technique. This simplification technique may be used to allow a faster image transmission, and allows a progressive image transformation built up from a simplified and small contour tree to a full contour tree. The results were compared with the popular Quantised Discrete Cosine Transform method, as used in the JPEG compression standard, and artifacts from both were displayed.

Some justification has been given that edge and contour information, physiologically speaking, with the classification of **complex**, **end-stopped** and **contour** cells, is important and thus should be preserved as far as possible. It has been postulated that it is edges and connectivity that are the key elements to image understanding. Psychological tests have been presented, which show possibly why, and when, the artifacts introduced by contour merging become noticeable and objectionable. This gives a better footing and justification for the use of the contour merging algorithm. Guide

[12]The compression ratios, for the entire image, which is shown in Appendix B, range from 0.997:1 for the original to 3.734:1 for the far right hand slice.

lines have been made up and used, based on the physiological and psychological evidence, to parametrically adjust the noticeability criteria of two neighbouring contours.

Chapter 6

Colour Map Considerations

All colours agree in the dark.
– Francis Bacon

At present only gray scale images have been analysed. This has eased the task of describing how to represent, code and simplify the contour tree. The following chapter considers colour images in detail. It is split into two parts:

Black-and-White or facsimile images, which are often huge, turn out to be ideal candidates for the contour tree format.

Colour in images and displays are considered describing the current use of colour and showing how the contour merger can exploit the increased redundancy in colour.

6.1 Black and White Images

Black-and-white or facsimile images are an interesting extreme that can be exploited by the contourization process. Obviously only two heights need to be allocated to represent black and white which leads to a set of further restrictions.

6.1.1 Further Restrictions

Given the task of coding the three streams which describe the contours in the image, the aim is to reduce the number of choices. An observation is made that when a contour is outlined not only are all the boundary pixels to the contour defined but also all the outside neighbouring pixels to the boundary pixels are defined to be of the opposite colour. By extending the *Indicator Flag* array this extra information about these pixels can be represented. A second operational decision is to enlarge the image by one white pixel around the outside, which requires no encoding as its movement is 100% deterministic. This defines a background colour of white, a common background colour for facsimile images.

Start Locations

Start Location offsets can now effectively skip over all of these new known extra pixel values. This means the total sum of the offset values equals the number of pixels that are not just a boundary pixel, but also not on the neighbourhood outside a boundary pixel.

Height Values

After the height value of the first contour has been transmitted all future height values are deterministic. This is because there must be a defined neighbouring pixel to the current *start location* which has a different height value. As the first contour is known to be white this stream is not required at all, and it does not need to be stored or transmitted.

Boundary Description

By referencing this extra pixel information further movement directions become impossible. This can be signalled using the same *Movement Restriction* flags already available.

6.1.2 Improvements

The changes to the contour coder are very simple. A comparison with CCITT Group 3 and Group 4, facsimile codes is given for a set of test images in Table 6.1.

Image	Original	Contour Coder	Group 3	Group 4
Dog	21773	*1974*	7171	2730
Dog	1	*11.030*	3.036	7.975
Man	15040	*1586*	5723	2415
Man	1	*9.483*	2.628	6.228

Table 6.1: Black-and-White Compression Results.

It is worth noting that the general contour coding compression size is proportional in some sense to the number of bits per pixel used. When a low number of intensities are available there is a corresponding reduction in number of contours, and thus an increase in compression ratio.

6.2 Colour Considerations

For lossless representation chrominance colour information can be converted to height values as simply as monochrome values. In modern images there are two different common methods of specifying colour information:

- True-colour images which store for each pixel full colour information. This is often as a triple of RGB (Red, Green, Blue) values.
- Pseudo-colour images which store all the used colours in a look-up table. Each pixel is then described as a reference to this look-up table.

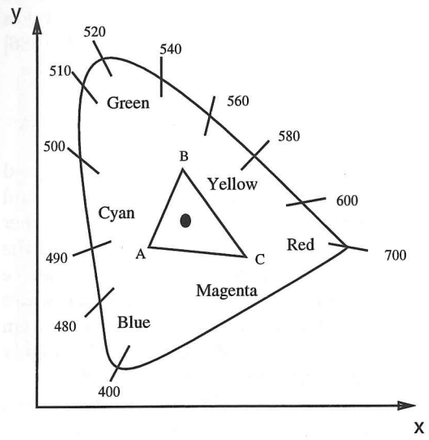
These two methods are considered separately after a short description on the use of colour in computer imagery.

6.2.1 Colour in Images

Since Newton's experiments, it has been well known that a wide range of colours can be generated from a fairly free choice of three primary colours. Modern displays generate their images by mixing light from three primaries; red, green and blue. In 1931 the CIE (Commission Internationale de L'Eclairage) developed the colour chart used in early television specifications. To define the viewable colours for a display device consists of, drawing a triangle on the chart with each of its primary colours at the endpoints. The colour space inside the triangle represents the possible colours available to the display and is called the gamut. This is shown in Figure 6.1.

The CIE chart was partially used by the NTSC (National Television System Committee) to define a transmission format in terms of luminance and chrominance which was called YIQ; representing luminance, in-phase chrominance and quadrature chrominance coordinates [Pritchard, 1977, U.S. Senate, 1950].

Later in Europe the PAL (Phase-Alternation-Line) format, for the UK [Townsend, 1970], and the SECAM (Séquentiel Couleurà Mémoire) format, for France, were developed.These used a colour space called YUV or YC_rC_b. The difference between YIQ and YUV is a 33 degree rotation in UV space.



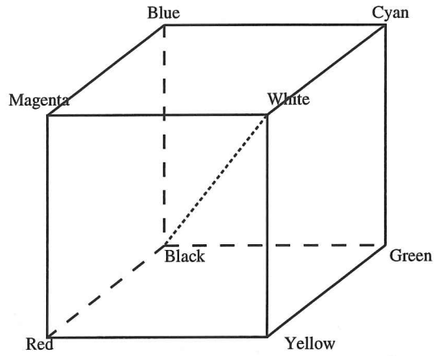


Figure 6.1: Typical CIE Colour Map and RGB Colour Cube.

This system proved fine for analogue transmission, but slight modifications were made to accommodate digital storage and transmission. Most digital systems store the three channels as eight bit quantities allowing 256 values or quantisation levels. 10 bit, 12 bit or 16 bit quantities are also used in situations where higher resolution is required. Digital conversions between RGB and YIQ or YC_rC_b commonly use the following matrices:

$$\begin{bmatrix} \mathrm{Y} \\ \mathrm{I} \\ \mathrm{Q} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.60 & -0.28 & -0.32 \\ 0.21 & -0.52 & 0.31 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} \tag{6.1}$$

$$\begin{bmatrix} \mathrm{Y} \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} \tag{6.2}$$

Due to the ease of calculation of the conversion in the digital world the YC_rC_b format is favoured over YIQ. It is worth noting that whilst the Y values range from 0—255, the C_r and C_b values range from 0—±128, which are often stored by adding 128 and forcing to the range 0–255. This defines a few values which do not map one-to-one and thus there is a very slight loss of information[1]. The YC_rC_b format is designed to concentrate as much of the image information into the luminance and less in the chrominance. This means that the three streams are less correlated and thus can be coded separately. The RGB and YC_rC_b entropy values for the Lenna image are shown in Figure 6.2. The overall entropy after converting from RGB has slightly increased, but there is a marked difference between the Y plane and the two C_rC_b planes. YC_rC_b has become the standard for the JPEG JFIF format representation as defined in [Hamilton, 1992]. Slight differences are carried out for H.261 and MPEG standards which perform the CCIR Recommendations 601 specifications. The following shifts and scales have to be carried out:

$$Y' = 219/255 \times Y + 16 \tag{6.3}$$

$$C'_r = 224/255 \times C_r + 128 \tag{6.4}$$

$$C'_b = 224/255 \times C_b + 128 \tag{6.5}$$

[1]This loss of information affects only a few values and at most changes the value of the least significant bit.

From physiological testing it is observed that the chrominance channels do not need to be specified as frequently as the luminance channel. Thus the CCIR-601 recommendations also state that only every other C_r and C_b value is required to be stored. This means a 3:2 conversion takes place from RGB to YC_rC_b. A full description of the CCIR Recommendations 601-1 [CCIR, 1986] is given in [Slater, 1991, Chapter 6].

6.2.2 Colour in Printing

It is worth noting that other colour models exist, and the most popular alternative is that used by the printing industry. This uses the subtractive primary colours; Cyan, Magenta, Yellow and Black ($CMYK$). Black is used as a separate colour to achieve the best visual quality rather than combining equal quantities of C, M and Y, although some recent printers have removed the black. This makes the conversion from RGB to $CMYK$ a one-to-many mapping, and very device dependent. An image processed and printed in exactly the same way in Europe and the States gives different results as printing inks and paper are different. Also the simple calibration system described above for visual displays fails for printing devices as mapping a specific colour is a highly non-linear process, and it is not sufficient to measure a set of points and interpolate.

These problems are due to the lack of a model describing the printing process in sufficient detail. Commercially Colour Management Tools (CMT) have started to appear which calibrate a set of devices together[2]. At present the defined set is normally small and often restricted to one manufacturer – normally the same as the one producing the Colour Management Tool

6.2.3 True–Colour

True-colour images are commonly represented as 24-bit RGB values, often captured images, containing a degree of noise. This means that they result in very low lossless compression ratios, but are ideal candidates for the lossy contour merging process. As 24-bit values are bulky to handle it is sensible to convert the values to YC_rC_b as recommended, and reduce the resolution of the C_r and C_b components by half in both vertical and horizontal dimensions. The reduction in resolution was carried out using a cubic resampler which provided good reconstruction properties, as recommended in [Dodgson, 1992]. This gives an automatic 2:1 conversion in image size. For lossy compression each stream is sent through the contour merging process, with varying threshold values. Quality of "excellent", as previously defined, gives fairly high levels of compression ratio, up to 10:1. Resulting images for six different levels of compression are shown in Figure 6.3 and Colour Plates 3, 4 and 5.

Artifacts

Similar artifacts and distortions as described in Section 5.4.2 are evident, but it was found that the colour information hides distortion levels previously considered noticeable. The chrominance difference channels C_r and C_b could be far more severely compressed then the intensity channel. Additional artifacts included:

Colour Bleeding between channels occurs. This happens when edges of contours in different channels do not match up precisely. This is more noticeable when using the three RGB channels rather than the YC_rC_b channels.

6.2.4 Pseudo–Colour

For lossless coding pseudo-colour images can be treated very simply by using the same reference number as a height value. This means that the colour look-up table needs to be stored with the

[2]Two examples are the Scanmatch colour calibration software [Savitar, 1991] which uses the TekColor Colour Management System, and is designed for matching and correcting a set of Pantone colours for scanners and the Advanced Digital Control System, ColorControl feature [NEC, 1991] used in NEC MultiSync 5FG/6FG monitors to adjust monitor colours to printer colours.

Red Entropy 4.810

Y Entropy 6.711

Green Entropy 5.698

C_r Entropy 5.400

Blue Entropy 5.002

C_b Entropy 5.384

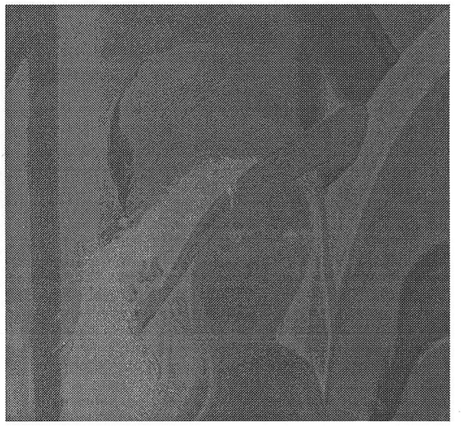


Figure 6.2: RGB and YC_rC_b planes for the Lenna Image.

1.834:1 Original YC_rC_b

8.136:1 High Quality all planes

8.405:1 High Quality Y Medium Quality C_rC_b

9.462:1 Medium Quality Y Low Quality C_rC_b

10.564:1 Medium Y Very Low Quality C_rC_b

17.486:1 Very Low Quality YC_rC_b

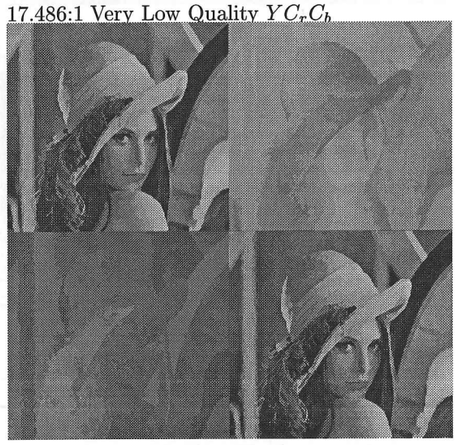


Figure 6.3: Colour Compression via YC_rC_b.

image. It is likely to be small and takes up very little space. Pseudo-colour images are used for two main reasons:

1. A space saving technique, as often there is a maximum of 256 entries in the look-up table which contain RGB values each stored as eight bit quantities. This gives instantly at least a 3:1 compression ratio.

2. Designed for use with displays with limited colour look-up tables. Due to hardware realisation these displays can only display a certain number of colours.

Often digital image processing has been carried out on a pseudo-colour image to reduce its colours to 256 or below. This involves dithering [Holliday, 1980, Bayer, 1973, Judice et al., 1975, Jarvis et al., 1976, Jarvis and Roberts, 1976] or diffusion techniques [Floyd and Steinberg, 1975, Knuth, 1987, Ulichney, 1987] for speed, or complex colour selection techniques [Coltelli et al., 1993, Voloboj, 1993] for higher quality. There is often very little reason to apply a lossy compression system. This is because the lack of colours already leads directly to high levels of lossless compression. Applying the contour merging algorithm would involve first converting to true-colour and loosing the compression ratio already achieved.

Chapter 7

Topiary: Tree Manipulation

If a digital image is something one can see
(by experiencing it with one's eyes),
one cannot compute it;
but if one can apply mathematical operations to it,
then it has no intrinsic visual manifestation.
– T. Binkley

One of the stated aims, in Chapter 1, was to be able to apply typical digital image processing operations on the compressed image as easily as on the original raster image. This may, and does mean, that for certain image operations the cost is proportional to the number of contours rather than the number of pixels, and could easily lead to a substantial speed-up. It is worth remembering that an emphasis on the contour merging process was a controlled reduction in the number of contours.

This is very similar to extensions of work applied to the quadtree image format by [Samet, 1982, Samet and Tamminen, 1985] and on JPEG compressed images by [Smith and Rowe, 1993]. Unlike quadtree operations no emphasis is placed on the advantage of having a reduced storage capacity in main memory, which is a possibility. The main objective is to attain the fastest method of applying the operation itself.

The coded form of the contour tree is, due to entropy stream coders, inherently in an awkward state. The three streams are uncompressed into a pointer based contour tree in main memory allowing fast random access throughout the tree. Also, for some operations, an array equal in size to the resolution of the original raster image, containing just the information about edges, is used. Contours at the same level of the tree with the same parent are stored in a linked list structure in raster scan order. Once an operation has been carried out, the contour tree node can store this result in the structure so that it does not need to be calculated again.

For many universal image formats a set of conversion utilities and toolkit operations are provided. These allow general purpose operations to be carried out without having to write specific pieces of code. These operations include:

- Colourmap and colour space modifications.
- Splicing, cutting and merging images together.
- Rotations, reflections and rescaling images.
- Compositing two images together.

As the encoding and decoding processes are not slow, compared with some of the operations, the advantages of creating unique pieces of code has to be weighed against using already written routines which work on raster data. Unfortunately, this is not a cheap option in all cases and the contour merging algorithm is an ideal example of where specialisation is nearly essential.

The next section deals with general image based operations that can be applied to the contour tree. These present the minimum operators required in a general purpose toolkit. The final two sections present two examples of contour manipulation which are specific to the contour tree:

1. A general introduction to intensity value selection and quantisation is given, with a full description of a novel form of contour tree intensity quantisation.

2. A simple texture analysis operation is described which is inherently stored in the contour tree structure. In addition an impressionistic painting scheme using this texture information is defined.

7.1 Quick Operations on Contours

Given a contour description the following things can be discovered whilst following the path of the contour boundary:

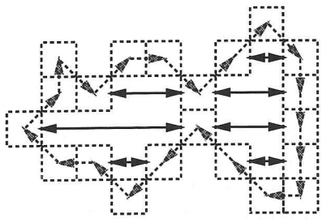
- Length of the boundary is simply the number of four-connected movement steps plus the number of eight-connected movement steps $\times\sqrt{2}$.

- The number of pixels on the boundary is simply a count of unique locations encountered on the boundary description.

- The inside area of the contour is just as easily calculated by temporarily storing the right and left sides to the contour boundary, as in the contour decoding stage, for each scanline. The sum of the differences between these pairs is the full inside area of the contour. Reducing this value by the sizes of its children gives the number of pixels in that region.

- $X-Y$ Bounding boxes for the contour can be calculated simply as the minima and maxima encountered during the route.

- In principle any shape feature can be computed from the contour boundary.



Boundary Length	26.385
Boundary Area	21
Inside Area	13

Figure 7.1: Contour Area Calculation.

A feature of a boundary description is that it is a *translation invariant* representation that allows for feature extraction and comparison with other contour shapes. Contour matching has been used in [Liu and Srinath, 1990], as a technique for feature recognition, whose method was designed to account for an unknown transformation from the actual image and a degree of noise. Also a chain pyramid structure which keeps contour information in a hierarchical format has been used in [Meer et al., 1990] as an improved method to aid the grouping of features.

7.2 Contour Operations

The common toolkit style operations which can be specialised to deal with contour trees are presented. Colour map operations are dealt with separately in Section 7.3.

7.2.1 Pixel Finding

The fundamental operation in all raster image manipulation is to extract the colour value at a specific location. This operation on contour trees requires searching through an ordered list of contours at the top level until one of the contours encloses the required pixel. If this is a boundary pixel then its contour is found and its colour value can be directly extracted, otherwise the operation has to be recursively carried out on the children of this contour. Termination also occurs when there are no child contours, or the raster position of the next child contour's start location is further away from the pixel's location in a raster scan sense. For efficient operations with a large number of child nodes a binary chop algorithm is employed[1].

In the worst case this could involve looking at all the contours in an image. If there is for example, a series of concentric contours whose regions are the same area as the boundary and the request is for the colour value of the innermost pixel, then all the pixels in the image will have to be referenced. The worst case never has to look at more than all the boundary pixels in all the contours, but this could easily be considered excessive. Even in the average case which deals with a large number of contours in a large number of levels, the operation is a hierarchical ordered list search. If the tree has N contours in m levels and each node is a leaf node or has k children, then by using a binary chop algorithm at each stage, the number of contours to be examined for a balanced tree ($N = k^m$), is $\mathrm{O}(m \log_2 k) = \mathrm{O}(\log_2 k^m) = \mathrm{O}(\log_2 N)$.

It is shown that the cost of this pixel search operation is very dependent on the structure of the contour tree. For this reason an edge map is used, which consists of a single array equal in size to the original raster image holding the boundary information for each contour. Pixel value inquiries are then a simple matter of travelling along the horizontal axis until it reaches an unmatched contour boundary flag. This cost is no more than looking at half the pixels in a scan line.

7.2.2 Joining and Splitting of Contours

We have already encountered the ability to join or merge two adjacent contours. This requires going around the two contours joint boundary describing the new outermost boundary, and merging the children of both contours to form a single list of children for the new contour. There is the chance that during the joining process one or more contours may be caught and become the children of the new larger contour. To detect this case the internal area for the new contour is calculated on the fly. If there is a discrepancy between the full size of the new contour and the two original contours, then the new captured children can be isolated and become new children of this joint contour. Similarly contour splitting can be undertaken which may result in the release of children.[2]

In the contour merging algorithm all contours keep a record of their region size and contour size, and an edge map the size of the original raster image is created for speed. Children are stored in order of internal contour size potentially requiring a linear cost merge sort each time two contours are joined. Contour merging also involves calculating a resulting mean intensity value for the new contour from the sizes and intensity values of the two original contours.

7.2.3 Chopping, Splicing and Compositing Images

Joining together or chopping out a section of an image is a common operation, and involves recreation of specific contours on the borders which are effected. This results in potentially many contour merging or splitting operations as defined above.

Two or more contour trees can be merged together making a whole by associating an alpha blending value to define the level of transparency, or a z-value to define a depth for each contour or tree. This allows single scenes to be modelled as a set of contour trees each one defining an actor or background to the final image. This image composition operation involves the sub-division of one contour tree with the contours of another.

---

[1] Using a standard $\mathrm{O}(\log N)$ binary search as described in [Aho et al., 1983, pages 365–366].

[2] Joining and splitting have been known as marriage and divorce which may cause "sibling kidnapping" or "throwing the children out of the home".

7.2.4 Rotations, Reflections and Rescaling

Rotations of 90, 180 and 270 degrees, reflection about an axis and integer scaling can be achieved by very simple rewrites of the boundary codes and start locations. For arbitrary operations which are not axis aligned this is often an imperfect operation. For example, rescaling requires a resampling operation, and rotations and reflections requires an anti-aliasing operation. For these operations it is recommended that the raster version equivalent is applied.

A case where contour trees could be used is with precise translation operations in black-and-white images, or where anti-aliasing is either not required or not available. Then by translating just the pixels on the contour boundary a decision based on pixels being mostly covered by the enclosed area of the translated points can be used to describe the new boundary. This technique is used to transform a contour by a 30 degree anti-clockwise rotation as shown in Figure 7.2. The dotted line represents the true contour translated, and the solid line shows the final pixel boundary line.

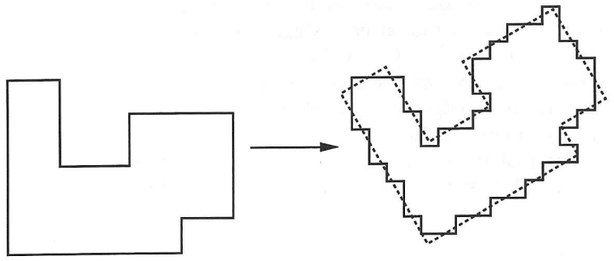


Figure 7.2: Contour Rotated by 30 Degrees.

Resolution Independent Rescaling

The use of edge lines and contours to achieve a "pseudo" resolution independent format for binary character images has been proposed amongst others by [Namane and Sid-Ahmed, 1990] and [Cabrelli and Molter, 1990]. These are alternatives to defining the characters in terms of cubic splines, line segments etc. The contour filling algorithm proves to be a good candidate for creating exact area coverage, allowing even automatic antialiased fonts. By applying smooth contour scaling interpolation functions, fonts of virtually any size can be created purely by scanning in a binary image example of the characters[3].

7.2.5 Converting Back to Raster Format

It must be remembered that converting to and from raster form is not a costly operation. So for linear filter operations and others which work directly, and very efficiently at the pixel level, unless the contour tree is substantially smaller than the raster format, converting to the raster format is a wise move.

7.3 Colourmap Operations

All modifications to the colourmap inherently have cost proportional to the number of contours, as any changes do not affect the internal regions of any contour. Two neighbouring contours may

[3]Both references Cabrielle and Molter and Namane and Sid-Ahmed used cubic spline interpolants.

end up with the same colour value which requires them to be merged. These operations include gamma correction as well as applying a whole new colour map. One operation is the conversion of a pseudo colour image to three contour trees of RGB or YC_rC_b format. This requires triplication of the contour tree and possible contour merging[4].

We deal with the problem of designing a technique for reducing the number of intensity values to a specified minimum, with as little disruption to the resulting image quality as possible. Given a monochrome image and a medium on which it is to be displayed, such as a CRT, photographic slides or coated paper, it is wise to calculate how many intensity values are required to reproduce a continuous-tone image.

7.3.1 Gamma Correction

As stated in Section 5.2 the human eye follows a power law in intensity values. This means that given an intensity scale from 0–1 the perceived difference in brightness between 0.20 and 0.22 is the same as that between 0.80 and 0.88. If we have 256 different levels of intensity they should relate to each other with the following relations:

$$I_1 = rI_0, \ldots I_j = rI_{j+1}, \ldots, I_{255} = rI_{254} = 1$$

$$r = (1/I_0)^{1/255}, I_j = r^j I_0 = (1/I_0)^{j/255} I_0 = I_0^{(255-j)/255} \tag{7.1}$$

The minimum intensity I_0 from a CRT is in the range 0.005 to 0.025. Now the light output by phosphor is proportional to the number of electrons N, and the number of electrons is proportion to the control-grid voltage, V. So for constants K, k and y;

$$I = kN^y, I = KV^y \tag{7.2}$$

If we wish to display intensity I, we search through the table of possibilities by choosing $j = \text{round}\left|\log_r \frac{I}{I_0}\right|$ then calculating I_j from (7.1). The required voltage for that pixel is $V_j = \text{round}\left|\frac{I_j}{K}^{\frac{1}{y}}\right|$. Measuring a screen or other device allows for it to be calibrated and the results stored in the required lookup table. It is important to repeat this process occasionally during the equipment's life and essential in devices like scanners which can have noticeable changes over weeks as the lighting bulb characteristics change.

For the human eye, continuity of intensities is only achieved when $r \leq 1.01$ according to [Wyszecki and Stiles, 1982], or $r \leq 1.025$ according to [Crow, 1978]. So given a dynamic range of $\frac{1}{I_0}$ for a medium the required number of intensities is $n = \log_r \frac{1}{I_0}$. Typical values of n, with $r = 1.01$, are shown in Table 7.1, which is adapted from [Foley et al., 1990, page 566][5].

Media	Dynamic Range	Intensities Required
Photo. Slides	1000	700
Photo. Prints	100	465
Coated B/W Paper	100	465
CRT	50 – 200	400 – 530
Coated Colour Paper	50	400
Newsprint B/W	10	235

Table 7.1: Number of Intensities Required by Display Medium.

[4]If a sub-optimal tree was permitted, which allowed adjacent contours to have the same intensities, then this would avoid all the merging operations.

[5]Using $r = 1.025$ the number of required intensity values is reduced by a factor of $\log_{1.01}(1.025) \approx 2.482$.

7.2.4 Rotations, Reflections and Rescaling

Rotations of 90, 180 and 270 degrees, reflection about an axis and integer scaling can be achieved by very simple rewrites of the boundary codes and start locations. For arbitrary operations which are not axis aligned this is often an imperfect operation. For example, rescaling requires a resampling operation, and rotations and reflections requires an anti-aliasing operation. For these operations it is recommended that the raster version equivalent is applied.

A case where contour trees could be used is with precise translation operations in black-and-white images, or where anti-aliasing is either not required or not available. Then by translating just the pixels on the contour boundary a decision based on pixels being mostly covered by the enclosed area of the translated points can be used to describe the new boundary. This technique is used to transform a contour by a 30 degree anti-clockwise rotation as shown in Figure 7.2. The dotted line represents the true contour translated, and the solid line shows the final pixel boundary line.

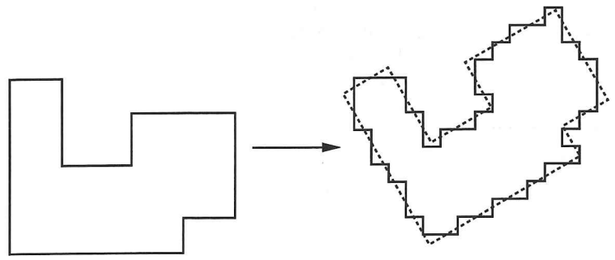


Figure 7.2: Contour Rotated by 30 Degrees.

Resolution Independent Rescaling

The use of edge lines and contours to achieve a "pseudo" resolution independent format for binary character images has been proposed amongst others by [Namane and Sid-Ahmed, 1990] and [Cabrelli and Molter, 1990]. These are alternatives to defining the characters in terms of cubic splines, line segments etc. The contour filling algorithm proves to be a good candidate for creating exact area coverage, allowing even automatic antialiased fonts. By applying smooth contour scaling interpolation functions, fonts of virtually any size can be created purely by scanning in a binary image example of the characters[3].

7.2.5 Converting Back to Raster Format

It must be remembered that converting to and from raster form is not a costly operation. So for linear filter operations and others which work directly, and very efficiently at the pixel level, unless the contour tree is substantially smaller than the raster format, converting to the raster format is a wise move.

7.3 Colourmap Operations

All modifications to the colourmap inherently have cost proportional to the number of contours, as any changes do not affect the internal regions of any contour. Two neighbouring contours may

[3]Both references Cabrielle and Molter and Namane and Sid-Ahmed used cubic spline interpolants.

end up with the same colour value which requires them to be merged. These operations include gamma correction as well as applying a whole new colour map. One operation is the conversion of a pseudo colour image to three contour trees of RGB or YC_rC_b format. This requires triplication of the contour tree and possible contour merging[4].

We deal with the problem of designing a technique for reducing the number of intensity values to a specified minimum, with as little disruption to the resulting image quality as possible. Given a monochrome image and a medium on which it is to be displayed, such as a CRT, photographic slides or coated paper, it is wise to calculate how many intensity values are required to reproduce a continuous-tone image.

7.3.1 Gamma Correction

As stated in Section 5.2 the human eye follows a power law in intensity values. This means that given an intensity scale from 0–1 the perceived difference in brightness between 0.20 and 0.22 is the same as that between 0.80 and 0.88. If we have 256 different levels of intensity they should relate to each other with the following relations:

$$I_1 = rI_0, \ldots I_j = rI_{j+1}, \ldots, I_{255} = rI_{254} = 1$$

$$r = (1/I_0)^{1/255}, I_j = r^j I_0 = (1/I_0)^{j/255} I_0 = I_0^{(255-j)/255} \tag{7.1}$$

The minimum intensity I_0 from a CRT is in the range 0.005 to 0.025. Now the light output by phosphor is proportional to the number of electrons N, and the number of electrons is proportion to the control-grid voltage, V. So for constants K, k and y;

$$I = kN^y, I = KV^y \tag{7.2}$$

If we wish to display intensity I, we search through the table of possibilities by choosing $j = \text{round}\left|\log_r \frac{I}{I_0}\right|$ then calculating I_j from (7.1). The required voltage for that pixel is $V_j = \text{round}\left|\frac{I_j}{K}^{\frac{1}{y}}\right|$. Measuring a screen or other device allows for it to be calibrated and the results stored in the required lookup table. It is important to repeat this process occasionally during the equipment's life and essential in devices like scanners which can have noticeable changes over weeks as the lighting bulb characteristics change.

For the human eye, continuity of intensities is only achieved when $r \leq 1.01$ according to [Wyszecki and Stiles, 1982], or $r \leq 1.025$ according to [Crow, 1978]. So given a dynamic range of $\frac{1}{I_0}$ for a medium the required number of intensities is $n = \log_r \frac{1}{I_0}$. Typical values of n, with $r = 1.01$, are shown in Table 7.1, which is adapted from [Foley et al., 1990, page 566][5].

Media	Dynamic Range	Intensities Required
Photo. Slides	1000	700
Photo. Prints	100	465
Coated B/W Paper	100	465
CRT	50 – 200	400 – 530
Coated Colour Paper	50	400
Newsprint B/W	10	235

Table 7.1: Number of Intensities Required by Display Medium.

[4]If a sub-optimal tree was permitted, which allowed adjacent contours to have the same intensities, then this would avoid all the merging operations.

[5]Using $r = 1.025$ the number of required intensity values is reduced by a factor of $\log_{1.01}(1.025) \approx 2.482$.

64 Levels

32 Levels

16 levels

8 Levels

Figure 7.3: Balloons Image Quantised to Different Intensity Levels.

7.3.2 Intensity Quantisation

The values in Table 7.1 are theoretical, requiring perfect processing and viewing conditions, and in reality blurring and noise substantially reduces the value of n. Figure 7.3 shows the Balloons image quantised to a variety of different intensity levels.[6] This demonstrates that the limit for this paper medium is about 50 to 60 levels. For a good CRT in a dark room many more levels are required. Numerous experimental and analytical results for CRTs range from 64 values as a minimum, up to a maximum of 256 values for virtually all images [Crow, 1978, Gonzalez and Wintz, 1977, Rosenfeld and Kak, 1982, Foley et al., 1990, Pavlidis, 1982].

Experiments[7] on a few CRTs found actual dynamic ranges from 20 to 60. These values are again lower than those in Table 7.1. In a normal "office" setting with good lighting the effective dynamic range is drastically reduced, meaning a reduction in the required number of intensities. If VDUs become brighter and sharper it may then be necessary to increase the standard gray-scale image to include more than the current 256 levels of intensity, but this is a sort of Catch-22 scenario as "good" screens have a degree of de-focussing built in to enable images to "look good" on them.

Floyd-Steinberg Error Diffusion

When forced to deal with fewer levels, but start with the image at higher levels one of the most popular technique is to employ an error diffusion algorithm. The most commonly used version was invented by Floyd and Steinberg [Floyd and Steinberg, 1975]. This involves, during the raster scan, the process of adding proportions of the error of a quantised pixel to the intensity values of the four neighbours, to the right-of and below the current pixel, as shown in Figure 7.4[8].

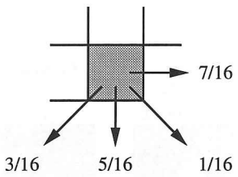


Figure 7.4: Floyd Steinberg Error Diffusion.

Improvements have been made by alternatively scanning from right to left as well as left to right, as described in [Ulichney, 1987]. The resulting diffused images for the Balloons image are shown in Figure 7.5. The operation works well as local neighbourhoods retain their mean intensity values and distort evenly rather than sharply at specific areas.

Contour Error Diffusion

A similar system using a contour tree description was undertaken, and when combined with the contour merging algorithm it further reduces the information of the contour tree. The contour structure remains the same but as contour intensity values are quantised the total intensity error for the contour is spread over all neighbouring contours proportional to the size of the edge in contact between the contours. The order of diffusing is dictated by the contour tree, starting at the lowest leaves and working up to the top. This has the advantage of spreading the error in all directions.

---

[6] The resulting n levels of quantisation are evenly spaced out but always include both black and white.

[7] On a very dark winter night with a Minolta LS-110 Light Intensity Meter and a cup of hot chocolate.

[8] It should be noted that errors can be negative as well as positive.

64 Levels

32 Levels

16 Levels

8 Levels

Figure 7.5: Error Diffusion on Balloons Image.

The results are shown in Figure 7.6 which present a compromise, giving better results than simply quantising but not as smooth as diffusing at the pixel level. It does have the advantage that edges remain sharp with all regions well defined, and the contour tree representation is maintained. As can be seen this method is still poor when there are very few levels of intensity, and is not a suitable technique to create black-and-white or even low intensity resolution images as the original diffusion technique was first designed to do.

By applying this contour error diffusion technique to a contour merged tree, further simplification can take place allowing, with the loss of quality on high resolution outputs, an increase in probability skew for the contour coder. Assuming that error diffusion to six bits giving 64 different gray values is acceptable, the increased performances on the "excellent" contour merged images are listed in Table 7.2. The third line in the table now lists error diffused contour trees[9]. The resulting compression ratios are considerably larger than the simple reduction in size of intensity values would predict which is due to the corresponding reduction in number of contours. The contour error diffused "excellent" versions are compared with the originals in Figure 7.7 and Colour Plates 6 and 7, in a similar style to Figure 5.5.

Image	*Size in bytes*	*Compression Ratio*	*Total Number of Contours*	*Hierarchical Depth* 1	2	3	4	5
Balloons	198664	2.286	65790	62616	3147	27		
		4.729	6689	5707	978	4		
		7.999	3589	2865	667	35	21	1
Escher	138632	1.488	86592	1	85797	794		
		4.479	6658	1	5968	688	1	
		5.010	5952	1	5330	616	5	
Garden	49160	1.402	38050	37938	121			
		4.253	2289	2145	144			
		4.721	2064	1901	160	3		
Mandrill	245768	1.254	219859	219848	11			
		3.807	13546	11936	1609	1		
		4.168	12569	10843	1724	3		

Table 7.2: Compression Ratios for Original, Contour Merged and Contour Error Diffused Images.

If representing images at very low bits per pixel is required where contour dithering breaks down, then an alternative is to use Floyd-Steinberg error diffuse on the image first and then contour code the result. If eight-connected contours are used there are still many large contours due to the small number of intensity values, and the corresponding compression rate reduction is acceptable due to the gain in quality.

7.4 Texture Visualisation

As stated previously in Section 5.4.2, one of the features of the contour merging algorithm is the highlighting of "texture flow".

[9] As a reminder Table 7.2 quantifies the number of contours and compression ratios achieved, in the original images (first line), "excellent" images (second line) and contour dithered images (third line). The hierarchical depth shows the number of contours at each level of the contour tree.

64 Levels

32 Levels

16 Levels

8 Levels

Figure 7.6: Contour Error Diffusion on Balloons Image.

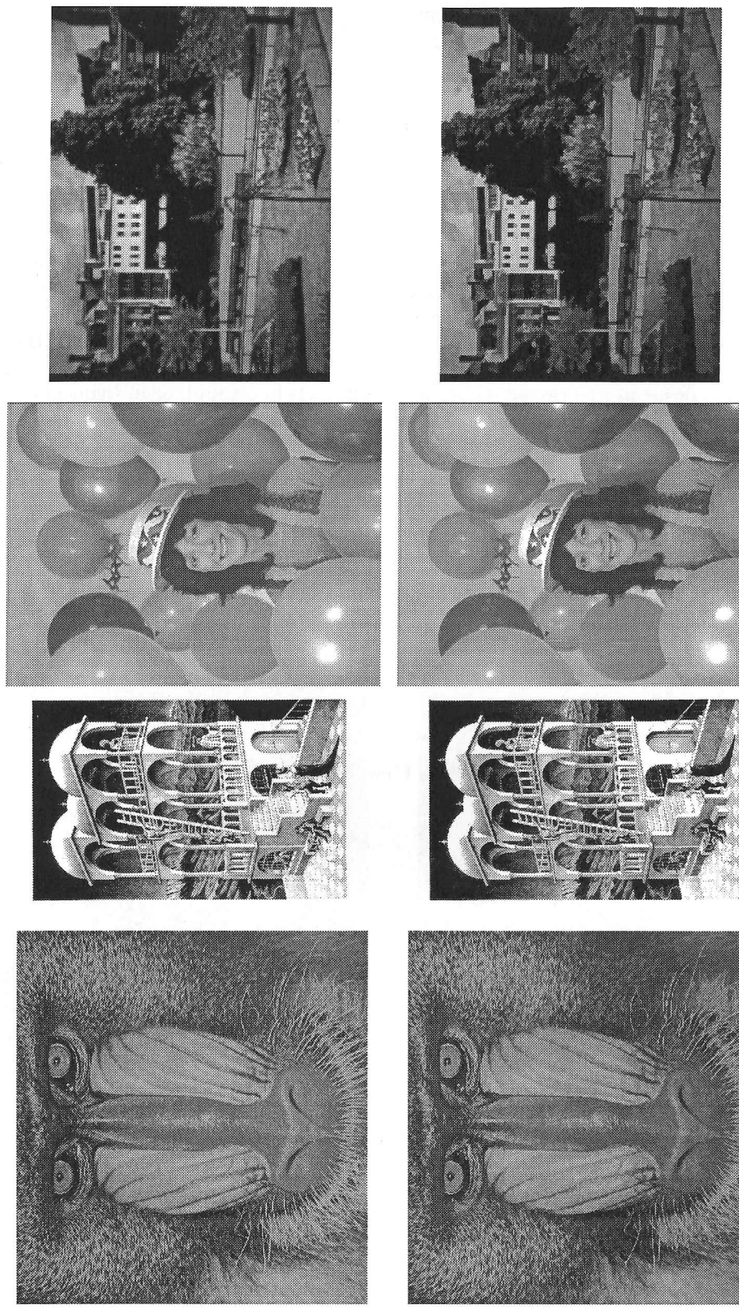


Figure 7.7: Comparison of Originals with Dithered *Excellent* Versions: Normal Colour Map. Garden Image Enlarged by a Factor of Two.

7.4.1 Texture Description Parameters

Texture in images is a fairly loose term defining the variation in intensity between pixels in a region. Four parameters that have been suggested in [Jähne, 1991] and used to describe textured areas in gray scale images are:

1. Mean intensity values.
2. Mean variance of intensity values.
3. Orientation of intensity structure.
4. Characteristic scale of intensity values.

The first two parameters are simple pixel value calculations and only apply to a small subset of texture types. The second two parameters have been used by many current texture differentiation techniques. These define each area as having a specific frequency component in a specific orientation. Techniques have used hierarchical Laplace structure, as presented in [Burt, 1981, Burt, 1984], or a Gabor wavelet pyramid structure, as presented in [Bovik et al., 1990]. These fail when multiple frequencies constitute the texture either in direction, as in a spiral, or in frequency and there is also a lack of detection in the boundary between two separate regions.

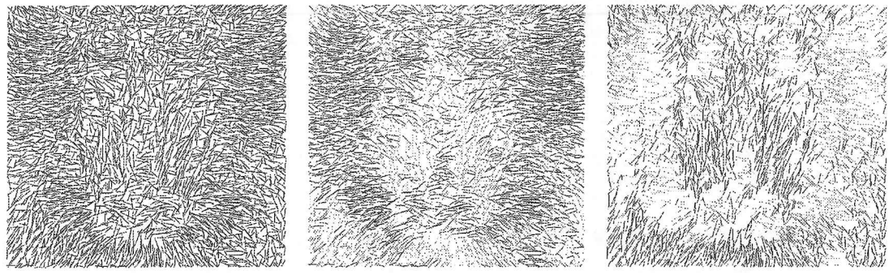


Figure 7.8: Texture Flow for Mandrill Image.

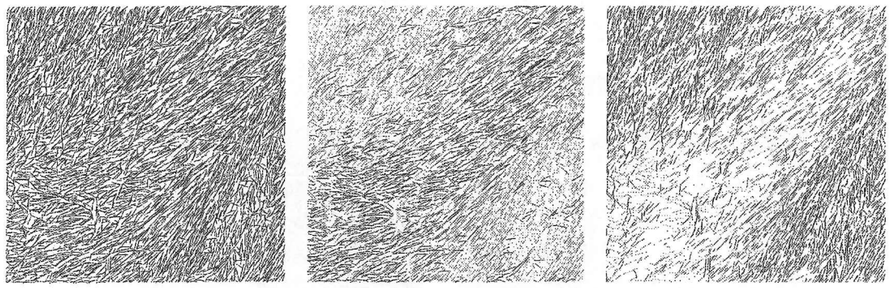


Figure 7.9: Texture Flow for Straw Image.

7.4.2 Contour Texture Visualisation

To visualise the flow from contours, the major and minor axis of each contour is extracted. If the contour is non-circular with more than a 3% difference between the two axes, then the contour is represented as a curved line travelling along the major axis curved in proportion to the size of the minor axis towards the centre of the minor axis. With high levels of contour merging on the Mandrill and Straw images, "texture flow" is shown in the first of the three images in

Figures 7.8 and 7.9. A highlighting technique is used which emphasizes a direction of orientation by changing intensity according to the tan of the difference between the requested orientation and the actual orientation. The second two images in Figures 7.8 and 7.9 highlight horizontal and vertical orientations.

Orientation can be extracted directly from the flow lines, but frequency components have to be calculated separately. The problem of interpretation and grouping regions is still there, but if there are clear borders between areas then the divide is potentially well defined. A simple first approximation for regionalising the image is to apply the orientation value to the height value of the contour. Then, applying the contour merging algorithm again, regions of similar orientation group together. This is shown in Figure 7.10 and 7.11 with a stylised version highlighting vertical orientations alongside the original images.

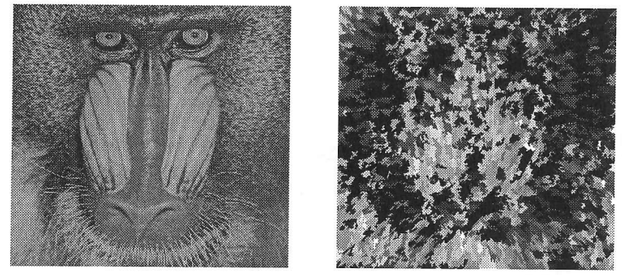


Figure 7.10: Vertical Texture Detection for Mandrill Image.

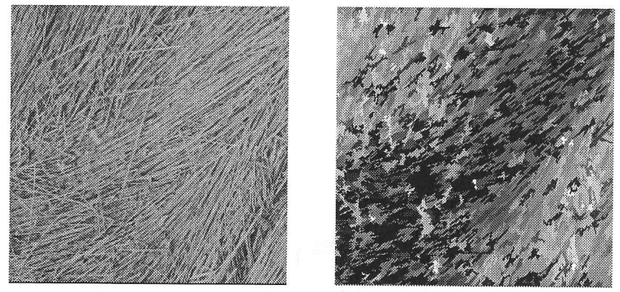


Figure 7.11: Vertical Texture Detection for Straw Image.

By increasing or decreasing the threshold value in the contour merging algorithm, coarse or fine grained texture can be detected. Keeping a record of all merges at many levels of threshold value means that orientations at one level can be compared with other levels. This creates a pyramid structure.

The technique still only detects one orientation at a time but has the main advantage over other methods that it is an inherent edge detector, and therefore neighbouring textures are less likely to interfere.

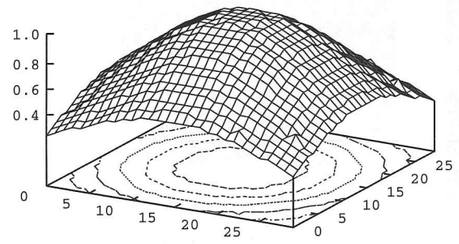
7.4.3 Repainting The Image

Contour Merging at the high level has been described as "Painting by Numbers" when random blotching artifacts appear[10]. Given these flow lines a repainting scheme is envisioned which can

[10]This term was first used by Jon Sewell at a Computer Laboratory Rainbow Graphics Group Meeting [Sewell, 1993].

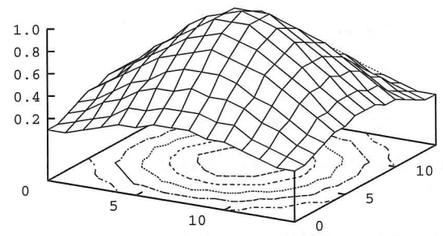
create a painter's abstract version of the original image. Given that an artist can create a painting with around 5,000 to 10,000 strokes, a similar interpretation of the contoured image can be attempted. Taking the line of flow of each contour as defining a paintbrush stroke of the correct colour the image can be repainted onto a blank canvas.

A set of Gaussian shaped brushes was developed. Given a stroke of width n a gaussian brush with standard deviation σ is chosen, such that $n = 5\sigma$. σ is a two-dimensional generalisation of the usual definition of standard deviation. For a one-dimensional Gaussian distribution, 68% of the area under the curve lies within plus or minus one standard deviation[11]. Examples of two brushes are shown in Figure 7.12. These brushes look more like a child's crayon then an artist's brush but have symmetrical properties.



Standard Deviation:

σ	6.6 pixels
5σ	33.0 pixels



Standard Deviation:

σ	2.6 pixels
5σ	13.0 pixels

Figure 7.12: Two Gaussian Brush for Strokes of Size 33 and 13.

The brushes were designed so that there is a degree of overlap between strokes. The amount of intensity applied to a particular pixel on the canvas is the product of the intensity value of the contour and the normalised gaussian value of the brush. Strokes move at the rate of one pixel per application of the brush. This means that strokes have to be merged together. Four techniques are employed to compose the new stroke with the canvas.

- *Max* – Choose the Maximum value. This highlights the bright areas of the image.

[11]This definition and calculation of the gaussian curve was taken from [Russ, 1992]. A good description of the careful use required when dealing with the discrete rather than analogue gaussian is given in [Lindeberg, 1990].

- *Min* – Choose the Minimum value. This highlights the dark areas of the image.
- *Mix* – Calculate both Maximum and Minimum values with the resulting value being the most extreme value from average gray. This is designed to highlight both bright and dark areas of the image.
- *Alpha* – Alpha blend the values into the image using the respective gaussian brush value as the alpha value[12].

The painted images are shown in Figures 7.13 and 7.14 for the Escher image, with 2408 brush strokes and 11749 brush strokes respectfully. The resulting images can be aesthetically pleasing with the *Min* image looking like a charcoal sketch, and the *Alpha* image looking like a very wet watercolour painting.

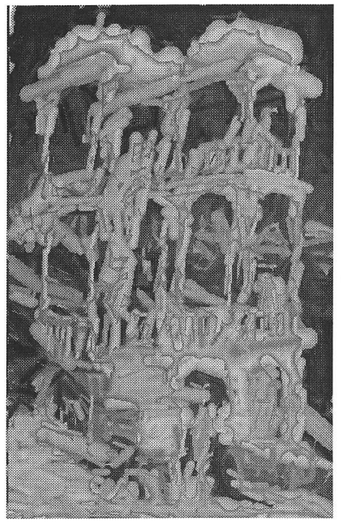
It is not envisaged that this is a suitable universal texture analyser or a perfect image reconstructor. It is simply a tool for segmenting an image and the resulting *painted* images should be considered as pure interpretations of images. They may even be pleasing to the eye.

[12]The use of α-channel composition is a common image painting technique, and is described in [Foley et al., 1990, pages 835–840].

Min

Max

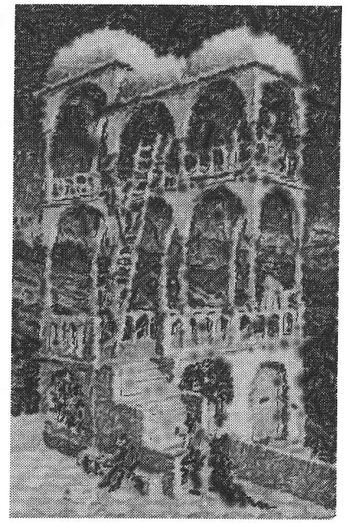
Mix



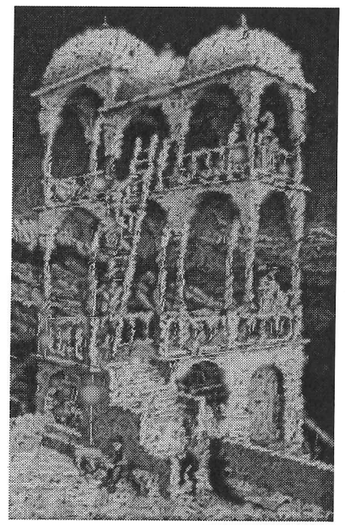
Alpha

Figure 7.13: Painted Images of Escher using 2408 strokes.

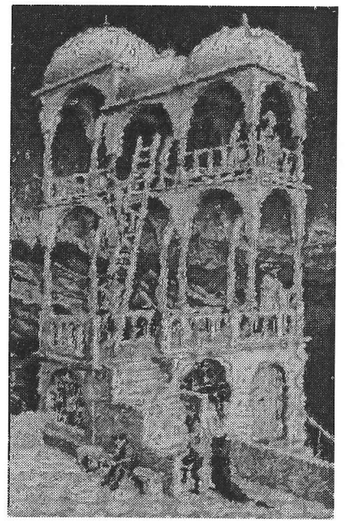
Min



Max



Mix



Alpha

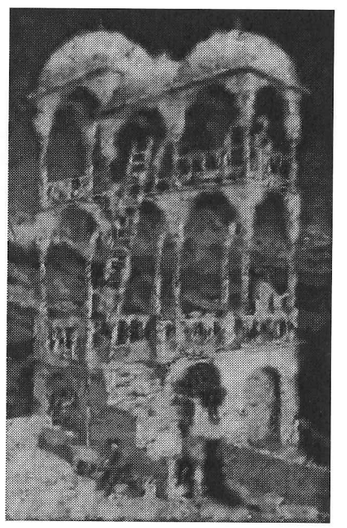


Figure 7.14: Painted Images of Escher using 11749 strokes.

Chapter 8

Specialisations and Extensions

A fool ... is a man who has never tried an experiment in his life.
– Erasmus Darwin

This chapter deals with expanding the use and adaptability of the contour tree format to cover features of different images. Two different ideas are presented:

Hexagonality. The use of rectangular pixel based displays has become universally widespread. In an elegant sense the application of hexagonal based pixels offers added advantages.

The Next Dimension. Image sequences, whether as motion sequences or stereo sets, are becoming increasingly popular with the use of multimedia and stereo viewers. Expanding the contour tree into the next dimension creating a shell tree is considered.

I have implemented these extensions to a moderate degree to test that they work, but each still has problems that need to be solved.

8.1 Hexagonality

As stated previously this format is designed for raster pixels each being represented as a square area on the image. The use of hexagonal pixels would be a sensible alternative, and both hardware and software techniques have been developed [Wüthrich and Stucki, 1991, Wüthrich et al., 1989]. It's main advantage for a contour tree is that all six neighbours of a hexagonal pixel are the same distance away from the centre, unlike the eight neighbours surrounding a square pixel.

Hexagonal pixels have not become popular for a few justifiable reasons, and although all the principles of contourization can be applied to hexagons as well as squares the main emphasis of research has avoided the topic.

8.1.1 Using Hexagons Instead of Squares

If we were to introduce hexagons to replace the standard square raster displays the layout is fairly straightforward, with effectively each line being displaced by half a pixel width and stretched to take into account that hexagons are taller then they are wide[1].

Assuming the area of a hexagon pixel H is the same as the area of a square pixel S, and let N be the distance between the centres of four-connected square pixels then the distances shown in Figure 8.1 are:

$$\triangle H = \frac{3}{2} D^2 \tan\left(\frac{\pi}{6}\right) = \frac{3}{2\sqrt{3}} D^2 = N^2 = \triangle S$$

[1] Or wider then they are tall depending on how they are orientated, see Figure 8.1. The ratio is $\frac{T}{D} \approx 1.233$.

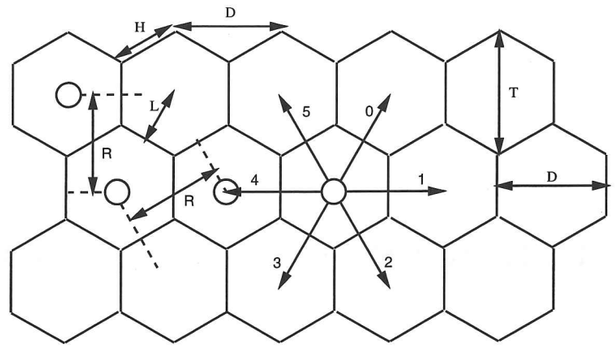


Figure 8.1: Hexagonal Pixel Arrangements.

$$D = N\sqrt{\left(\frac{2\sqrt{3}}{3}\right)} \approx 1.075N \tag{8.1}$$

$$L = \frac{D}{2} \approx 0.537N \tag{8.2}$$

$$H = D\tan\left(\frac{\pi}{6}\right) = \frac{D}{\sqrt{3}} \approx 0.620N \tag{8.3}$$

$$R = D\frac{\sqrt{3}}{2} \approx 0.9309N \tag{8.4}$$

$$T = 2\left(R - \frac{L}{2}\right) \approx 1.3249N \tag{8.5}$$

This tells us that the centre-to-centre packing density is reduced leading to possibly closer correlation between neighbouring pixels. If there exists a function $q()$ which when given a distance returned a probability that the value at this point is the same as the current point then we can construct P_n which is the chance of a pixel being connected to at least one of its neighbours when it is n-connected:

$$P_4 = 4 \times q(1) \tag{8.6}$$
$$P_6 = 6 \times q(1.075) \tag{8.7}$$
$$P_8 = 4 \times q(1) + 4 \times q(\sqrt{2}) \tag{8.8}$$

The auto-correlation graphs in Figure 3.3 showed that the correlation between pixels, rapidly reduces as distance between them increased. We have the property that when the relationship $P_6 > P_8$ is true, hexagonal images are likely to yield more connected regions than eight-connected images. It is also very likely that $P_6 > P_4$ which gives us an improved cleaner coding scheme, as it has none of the problems associated with overlapping contours.

8.1.2 Creating and Coding the Tree

The algorithms for creating a contour tree from an hexagonal image are exactly the same with slight modifications to Table 4.1 as shown in Table 8.1. Start locations and height values can be coded as before. Boundary descriptions have an ideal coding scheme of 6MAE (Six-connected Movement Along Edges code). As each vertex joins three edges a Movement Along Edge scheme requires at most one bit to code the two possible movement choices. Therefore if there is a movement restriction then the movement step becomes 100% deterministic. This results in a simpler coding scheme than is currently being applied to travels around the edges of square pixels.

		Travel Direction Out	
		↖ or → or ↗	↘ or ← or ↙
Travel Dir. In	↖ or ← or ↗	**L**	**B**
	↘ or → or ↙	**B**	**R**

Table 8.1: Indicator States Decided by Six-Connected Directions.

The Movement Through Centre method 6MTC6 is also feasible with a maximum of $\log_2(5) \approx 2.322$ bits per movement step required. 6MTC6 is similar to 4MTC4 and 8MTC8 as the code is not comma-free and thus requires, as in the previous cases, either an extra bit to be coded as required or certain contours to be coded as two separate contours. The single pixel contour also has to be coded as a special case.

8.1.3 Contour Tree Manipulation and Analysis

Similarly to square pixel contour trees, the hexagonal variety can undertake transformation and other modifications. Rotations of 60, 120, 180 and 240 degrees are, for obvious reasons, preferred over those for 90 and 270 degrees.

8.1.4 Testing

Given a very high resolution image it is possible to resample and create both hexagonal pixel and square pixel images that have the same spatial sampling. Applying this to a 3236×4001 test image, we result with a 119×172 hexagonal image, and a 129×160 square image, when using an exact area sampling reconstructor. The number of regions in the three cases is shown in Table 8.2.

Connection	Total	1	2	3	4	5	6	7	8	9	10	> 10
Four	17510	16225	1005	144	56	20	13	8	3	7	6	23
Six	17239	15978	996	136	46	24	16	6	7	5	4	21
Eight	17038	15490	1227	169	69	20	12	6	8	7	8	23

Table 8.2: Number of Connected Regions by Size.

Calculating the normalised correlation ρ in Formula 3.9, we have that a neighbouring four connected pixel has $\rho(1) = 0.965$, a six connected pixel has $\rho(1.075) = 0.961$ and a diagonally connected pixel has $\rho(1.414) = 0.950$.

8.1.5 Conclusions

I have shown some of the coding benefits of using hexagonal pixels over square pixels, and an outline for a coder has been given. As soon as, or more likely if, the graphics community changes to using hexagonal representation of images, the contour tree format can be directly and easily applied.

8.2 The Next Dimension

So far everything has been dealing with still-pictures, but the vast majority of images taken are to be viewed as a sequence. There are two main different uses of sequential image data:

Image Movies A sequence of images is shown giving the impression of motion. The two main areas are in animation sequences and video sequences.

Stereoscopic Sets The new autostereo displays and older stereo pairs take a set of two dimensional views. Each view is displaced slightly in viewing position parameters so that when projected into the eyes of an observer a stereoscopic effect is created. Autostereo displays have been designed to use up to 64 or 128 different views, for example the Cambridge Autostereo display [Lang et al., 1992] and wide-angle Lenticular screens [Börner, 1987].

The initial assumption stated in Chapter 2 can easily be extended as the number of dimensions increases. It is hoped that the amount of correlation increases as the extra dimension is added which results in a higher compression ratio even with the extra coding complexity that has been introduced.

8.2.1 Shell Trees

The main idea is to extend the concept of a contour into a *shell* which encloses pixels bounded in three dimensions. A *shell tree* can be created giving structure to the picture by allowing shells to be defined inside others. There is now the choice of how connected the shells outer case should be, either six, eighteen or twenty-six connected. This leads onto similar debates regarding overlapping shells.

The choice of how the extra dimension is defined can be left up to the user. If it represents time then moving contours are defined as tubular shells passing through x-y space, or if it represents multiple stereo views from multiple cameras then contours close to the viewing plane are represented by short shells and those far away as long shells.

Once the tree has been formed the simplifying merging algorithm and any other tree manipulation routines, once suitably adapted, can be applied. Rules for the psychological noticeability values have to be redefined to take into account the differences between spatial recognition and temporal recognition of objects. There is the possible advantage that with the increase in information a corresponding increase in redundancy exists so lower quality images may be acceptable.

8.2.2 Defining a Shell

A scan fill algorithm similar to the one which defines a two-dimensional contour is used. This means that only right and left edges are required by the filling scheme. The four *Indicator Flags* can still be used, but a similar algorithm to the Backtracking Bug Follower needs to be developed.

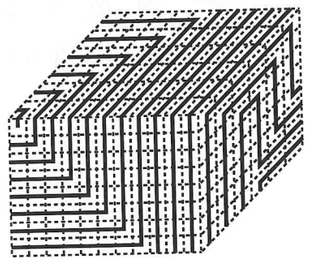


Figure 8.2: An Example of Spiral Covering a Simple ($11 \times 8 \times 6$) Block.

Spiral Covering of a Shell

It is proposed to cover the faces of the shell by using an anti-clockwise spiral route around the shell. Defining each pixel to be a cube with six faces, without loss of generality, the route can start at the front-top-left pixel and take a spiral path over all faces of the shell. This route has the property that it always keeps a previously defined face on the left. Dealing with the six-connected case whilst moving from one face to the next there are three different modes of movement as shown in Figure 8.3.

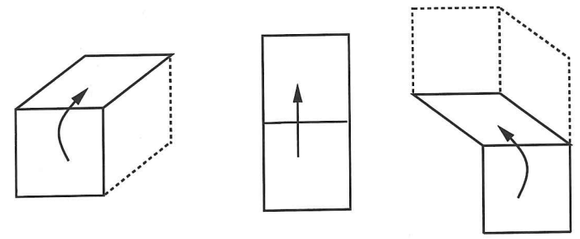


Figure 8.3: Three Modes of Shell Movement.

A list of possible faces to move is shown by the nine faces in Figure 8.4, which could be on the shell and have not yet been visited. The next movement step is a possible movement step to the face with the least number. An example spiral covering for a simple block is shown in Figure 8.2 with the bold line describing the route.

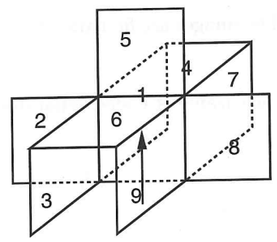


Figure 8.4: Nine Choices of Movement in Order of Preference.

Using a spiral route around the outside of the shell is useful as it results in no special cases. This is similar to MAE methods for contours. Probability coders are often able to reduce the coding choice for the decoder, and thus increase the compression ratio. At each stage there is a maximum of nine possible movement directions from one face to another which requires a maximum of $\log_2 9 \approx 3.170$ bits. The end of a spiral wrapped shell is noticed when all the faces of the shell have been travelled across. This is detected when there is no possible movement direction from the current position, and there have been no empty faces not yet seen. During the spiral route, it travels over faces and there are times on certain shells when areas of the shell can be effectively cutoff. Maintaining a fringe of faces which must be seen before completing the Spiral Covering algorithm, allows a restarting position to be specified. Synchronising both encoder and decoder also means that there is no need to code any extra information to signal the new starting position.

The disadvantage of using a spiral is that the elegance of using simple *Indicator Flag* tables, as defined in Section 4.1.2, is no longer possible. Whilst keeping a record of all the faces which are

seen a separate list is maintained for each of the three types of face; a left face, a front face and a top face[2]. To create the edge *Indicator Flags* all front and top faces result in a **B** flag whilst left faces alternately define a scan-line fill for the **L** and **R** flags. A pixel which is deemed to be both **L** and **R** is converted to **B** in the style of Table 4.2. All the six movement restrictions described in Section 4.5.2 can be applied to reduce the choices available at each coding stage.

8.2.3 The Problem with Movies

Movie sequences can be very long with about 216000 35mm frames in a 2.5 hour film. If the shell tree was to contain all the images in the sequence then its size would get seriously out of hand, with approximately 432 billion pixels for a 2.5 hour HDTV movie[3]. Fortunately the pixels in the first frame are very unlikely to be correlated to the pixels in the last frame, and the coding efficiency involved are thus likely to be very small. This implies that splitting the movie into small sections would be a sensible option. This is fine for conservative coding but may cause unwanted artifacts with non-conservative shell merged coding. Similar techniques to extend the Block Discrete Cosine Transform coding scheme into the next dimension have resulted in noticeable temporal "jumps" as time aliasing occurs. This has been demonstrated in [Martineau, 1991].

8.2.4 Application

An 11 view autostereoscopic image, of a porsche car in a showroom, is used as three dimensional test data. Figure 8.5 demonstrates the difference in number of shells for multiple single frames compared with a full shell tree representation. This gives an indication of the large savings in image manipulation operations if shells are used rather than contours. Figure 8.6 compares compression ratios for contours and shells using the following five schemes[4]:

- **Shell Tree** One single shell tree representation.
- **Single Frame Shell Tree** The images are first tiled together and then sent as a single frame to be coded as a shell tree.
- n **Separate Shell Trees** Each frame is coded separately as a shell tree and the resulting files concatenated together.
- **Single Frame Contour Tree** The images are first tiled together and then sent as a single frame to be coded as a contour tree.
- n **Separate Contour Trees** Each frame is coded separately as a contour tree and the resulting files concatenated together.

As each image in the sequence is relatively small, 320×240, tiling the images together achieves significant improvements for the single frame shell coding and the single frame contour coding. This is because the models for the entropy coder do not need to be recreated at the start of each frame, and thus there is not a poor compression ratio at the start whilst the model "learns" dynamically about the new frame.

In this example the coding compression ratio achieved by the shell coder is larger than the that for the contour coder when the number of frames is greater than or equal to three. As more frames are added the increased improvements in compression ratio reduce.

[2]Right, back and bottom faces can be automatically deduced from the other three lists.

[3]HDTV resolution of 1920×1152 (European standard) approaches that of standard 35mm quality film resolution which is quoted as being approximately 3000×2000.

[4]The contour coder used was a 4MAE coder to be as comparable with the shell covering algorithm as possible.

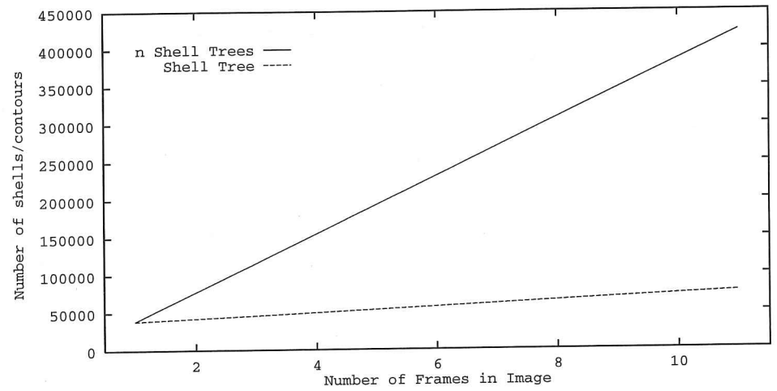


Figure 8.5: Number of Shells in Porsche Image Sequences.

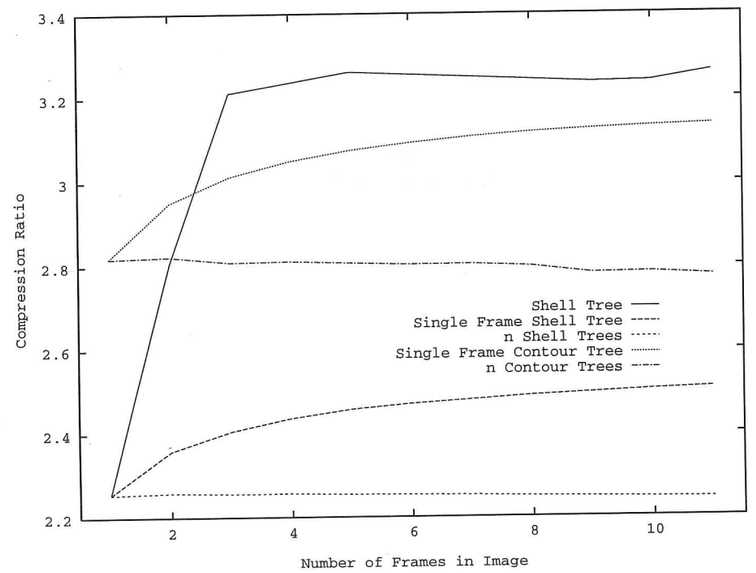


Image	Original	LZT	AC	L-JPEG	Contour Tree	Shell Tree
Porsche	844811	171855	255971	522283	259173	258966
Porsche	1	4.916	3.300	1.612	3.134	3.262

Figure 8.6: Shell Tree Compression Results for Porsche Image.

8.2.5 Conclusions

The extension into the third dimension has been undertaken with mixed results. Describing a full shell has increased the coding costs as there are more faces than edges to describe. A slight coding compression ratio improvement over contour coder can be achieved when enough frames in the sequence are considered. The main advantage is that the number of shells in the sequence can be drastically less than the number of contours. A lossy compression system, similar to the one described in Chapter 5, has also been developed and shown to reduce the number of shells by a similar factor to the reduction in number of contours required before visual artifacts are perceived [Turner, 1994].

The disadvantage of a shell tree structure is the increase in complexity and space required. Splitting the image sequences into small short sequences does make the process computationally achievable.

Chapter 9

Conclusions and Further Research

Thou shalt not live within thy means,
Nor on plain water and raw greens,
If thou must choose
Between the chances, choose the odd:
Read The New Yorker, trust in God;
And take short views.
– W.H. Auden

9.1 Format Performance

The main part of this dissertation has been spent describing the details and features of the contour tree image format. Initially I presented seven aims that any new image format has to address. I now summarise these aims as they apply to the contour tree image format:

1. **Compressible** The lossless contour coder developed in Chapter 4 has on all test example images shown to be comparable, and often superior, with other general purpose compressors and file formats around.

2. **Lossy Compression** The contour merging algorithm, developed in Chapter 5, demonstrated a physiologically based technique in lossy image compression. When combined with the contour coder increased levels of compression ratio are obtained. Compression ratios are not as great for similar levels of subjective artifact noticeability as compared with the popular JPEG method, but they have the feature that the artifacts are almost complimentary.

3. **Manipulatable** Chapter 7 presented a possible toolkit of operators for contour tree manipulation. It also gave a description of two operations very suitable to the contour tree format.

4. **Rasterisable** Algorithms described in Chapter 4 show how simple it is to translate a raster image representation to and from the contour tree format. All the operations have been designed to be computationally linear with the size of the pixel image.

5. **Globally Usable** Appendix 2 lists a range of the variety of images that have been shown to be suitable for conversion to and from the contour tree format.

6. **Resolution Independence** At present due to the pixel based boundary description of a contour the highest resolution extractable is the same as the original. Reductions in resolution are highly suited to the contour definition, but increases in resolution require suitable interpolations to be devised, as described in Section 7.2.4. To solve this problem properly a true resolution independent contour description has to be developed.

7. **Human Understandable** I believe that the principles of the contour tree format are intrinsically easy to describe. Also the resulting increase in size of the contour tree is proportional to the number of objects in the scene and roughly as expected by the user.

The format was designed to achieve as best a balance as possible of the original seven aims. It offers one of the best lossless compression system for a complete range of raster images and incorporates an almost unique lossy technique. The main failing is the non-resolution independence which although possible, even in the current format, it is never likely to be an ideal structure. As a conclusion the contour tree description of an image offers two advantageous features over a standard raster image format:

- Image operations can be carried out on contours in one step rather than at the pixel level, giving certain image manipulations potential speedups.
- This representation potentially allows a more compact description format for reduced storage and transmission time.

9.2 Further Research

The fact that there are so many de-facto and official standards for image data implies that there is still a lot more research required. The contour tree image format like all other methods is not perfect, and improvements are achievable. Four areas which offer new avenues for research and likely interesting results are:

- **Shell Trees** Chapter 8 proved that the next dimension can be successfully incorporated into the contour tree structure giving potential benefits. A more detailed study into shell tree coders similar to that studied on contour coders needs to be carried out. Possibly slightly different techniques are required for dealing with huge shell trees which could be gigabytes in size. Extensions of the contour merging algorithm to deal with shells, with a rethink on the psychological criteria, would also be a requirement.
- **Progressive Transmission** In talks, that I have given, the hierarchical structure of the contour tree has been proposed as an automatic method to allow progressive transmission. This would entail transmitting the contour nodes in a depth first traverse of the contour tree. At present the number of levels in a contour tree is not sufficient to enable a sensible system to be developed. Increasing the depth of the tree can be achieved by keeping all nodes in the contour merging algorithm. This would at most double the size of the tree, but allow a psychologically developed progressive image to appear.
- **True/Pseudo Resolution Independence** Animation image formats are often represented as pseudo-resolution independent formats. These are often defined in terms of curves and connected objects. This allows the final animation sequence to be created in a range of resolutions from video to 35mm quality. Converting a contour tree to describe a single animated character with contours being represented as connected areas with common boundaries to other contours would be a useful extension. Animation operations could be carried out on a set of trees which describe the full scene and allow for automatic merging of the characters.
- **Subjective vs Objective** The problems of creating an automated image quality function are not likely to be solved until much more is understood about the human visual system. Many attempts have been proposed, but at present all have images which act as counter examples. This is a separate problem, and is being tackled at many different locations, whose solution would be of fundamental interest to all lossy image compression systems.

Appendix A

Stream Coders

I only took the regular course ...
the different brands of Arithmetic –
Ambition, Distraction, Uglification and Derision.
– Lewis Carroll

Given a set of possible input symbols, there are many ways of coding this stream accurately and exploiting the redundancies in the stream. Two popular methods are described here. Modifications and versions which were used in the main part of this dissertation are described.

Probability Coders which code each input symbol into a variable length code determined by how likely it is to occur.

Substitution Coders which code a sequence of input symbols into a fixed or variable length code.

A.1 Probability Coding

A probability coder is one which when given a probability of occurrence for each possible input symbol, $p(\alpha), \alpha \in \Sigma$, produces a code for each symbol such that its length is inversely proportional to its probability. An optimal probability coder is one which produces an average code length equal to Shannon's entropy formula (see Section 3.1.2). This implies that every symbol is independent of the others which in virtually all streams of data is not the case.

A.1.1 Shannon–Fano Coding

Shannon's coding scheme, which was discovered independently by R.M. Fano [Fano, 1949] and C.E. Shannon [Shannon, 1949], uses the simple algorithm:

- Divide the set into two almost equal halves based on the probabilities.
- Mark one half with a **0** and the other with a **1**.
- Repeat division until all sides contain just one item.

This forms a binary tree which gives an average code length for the symbols which lie between $[H, H+1)$[1]. It has been shown that Shannon–Fano do not necessarily produce optimal codes for any given set of symbols.

[1] H is Shannon's entropy as defined in Section 3.1.2.

A.1.2 Huffman Coding

Shortly afterwards Huffman developed the Huffman Coder [Huffman, 1952]. Huffman codes are described as an optimal probability coder when all the probabilities are integral powers of $\frac{1}{2}$. For any set of symbols there are possibly many Huffman codes and an algorithm for deterministically creating one of these codes is:

- List all symbols in order of probability.
- Successively combine the two symbols of the lowest probability to form a composite symbol with joint probability, marking one side with a **0** and the other with a **1**.
- Once a complete binary tree is formed the code representing each leaf symbol is the path to that leaf

It is also possible to create a *canonical* huffman tree, which has all its leaves at unique depths in the tree, with minimal loss of efficiency. This allowing each leaf to be represented compactly as a single bit length.

A.1.3 Arithmetic Coding

As stated above Huffman and Shannon–Fano codes are only optimal when probabilities are integral powers of $\frac{1}{2}$. When probabilities of actual symbols do not fit this criteria either a poor code is produced of joint probabilities of symbols have to be considered which increases the complexity.

AC (Arithmetic Coding) is a technique which codes the stream of input symbols into a real number in the range $[0, 1)$. Each input symbol is given a sub-range proportional to the size of its probability, and encoding any symbol is simply achieved by returning any number in its sub-range. To make the process iterative, the whole sub-range is returned and this is used as the new current range to encode the next symbol. A simple example giving binary code words for all possible sequences of three characters on a two symbol alphabet (A with probability $\frac{1}{3}$ and B with probability $\frac{2}{3}$) is shown in Figure A.1.

In practice AC is implemented for speed using scaled arithmetic on up to 256 symbols with up to 12 bits of information for probabilities. This slight approximation has been shown not to affect the performance which for long coded sequences approaches Shannon's entropy.

It has been proposed in [Bell et al., 1989] that the *coder* part should be thought of and implemented separately from the *model* part which allows for adaptive models to exist. By adjusting the symbol probabilities as symbols arrive it is possible to have the model adjust as the input stream changes. The most common form of adaptation is to simply keep a frequency count of all symbols and divide this by the total number of frequencies to obtain the probability. When the sum of the frequencies is greater than a pre-defined maximum all the frequencies are halved. Adaptive models have proved to out-perform static models for virtually all data streams.

The two standard AC were used throughout the text as common stream coders:

AC-1 1-Order Adaptive AC for up to 255 symbols using one list of frequencies.

AC-2 2-Order Adaptive AC for up to 255 symbols using a separate list of frequencies for each possible symbol giving conditional probabilities.

Separating the model from the coder means that it is relatively easy to change the probabilities on the fly. This means reducing the number of symbols to code is a simple matter of specifying the redundant symbols probabilities as zero. The arithmetic coder as specified in [Witten et al., 1987] caters for up to 256 symbols with a range from $[0, 1)$ specified using 12 bit fixed arithmetic. Versions which could handle 2^{16} symbols and up to 24-bit precision were used when the extra flexibility is required.

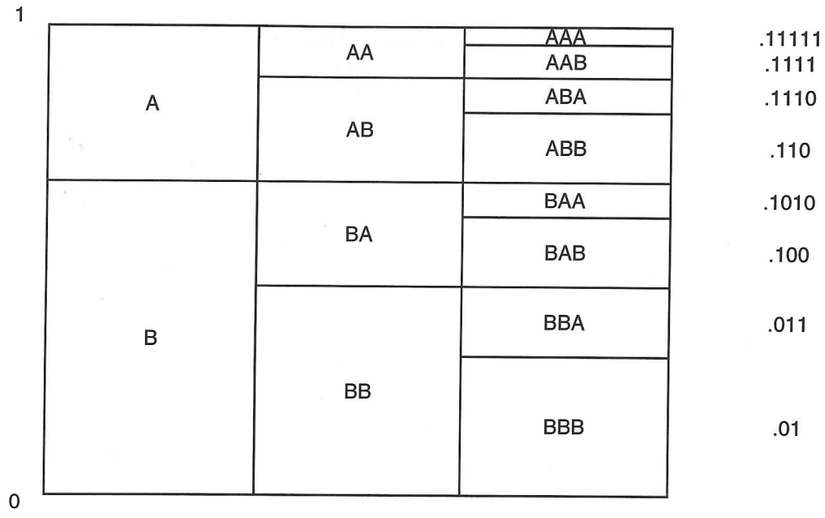


Figure A.1: Example of Arithmetic Coding.

A.1.4 Dynamic Markov Coding *DMC*

It has been shown in Chapter 3 that the symbol probabilities themselves are not a particularly good estimate of the full entropy for the data. It is essential to considering the inter-symbol probabilities. A DMC model takes advantage of these dependencies and starts with a 0-order markov model as shown in Figure A.2 a) and expands it as symbols are coded.

DMC works by maintaining an adaptive list of probabilities for predicting the next relative step in each state, and a count of transitions from each state to another. If transitions from any state to another raises above a certain threshold then the resulting state is cloned duplicating all internal probabilities. This is shown in Figure A.2 b) with state transition $A \rightarrow M$.

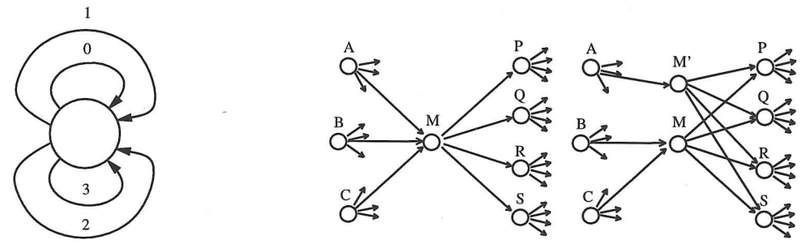


Figure A.2: *a)* Poss. Initial State for DMC model. *b)* State Cloning.

DMCs are often used for binary symbols but with suitable modification the movement codes defined in Chapter 7 were suitable coded. More details and compression results are given in Section 7.5.1. Figure A.2 shows a four symbol state setup. The model used is a modified version of Horspool and Cormack's model described in [Horspool and Cormack, 1986, Horspool and Cormack, 1987].

A.2 Substitution Coding

The basic idea behind substitution coders is to replace an occurrence of a phrase or sequence of bytes by a reference to a previous occurrence of that phrase. There are two main families of coders named after Jakob Ziv and Abraham Lempel, who first proposed them in [Lempel and Ziv, 1977, Lempel and Ziv, 1978].

A.2.1 LZ77: Window Substitution Compression

LZ77 coders have a window of the last n bytes of data encountered. When a new phrase is in this window it is encoded as a displacement and a length. The most common implementation is derived from the LZSS scheme [Storer and Szymanski, 1982]. The LZSS maintains both an n byte *window* and a *lookahead buffer*. The coding algorithm is as follows:

```
loop until lookahead buffer empty
    get a pointer (position, match) to the longest match
    if length > MINIMUM MATCH LENGTH
      output a (position, length) pair
      shift the window length character along
    else
      output the first character in the lookahead buffer
      shift the window 1 character along
    end if
end loop
```

Variants of this method apply extra coding to the output stream, which have included simple variable-length code (LZB) and dynamic Huffman code (LZH) [Brent, 1987]. Most of the popular commercial archivers are variants of the LZ77 algorithm.

A.2.2 LZ78: Dictionary Compression

LZ78 coders work by storing phrases into a dictionary and substituting phrases as dictionary entries. The most popular implementation is Terry Welch's LZW [Welch, 1984]. In LZW a 4k dictionary is used, initially with the 256 individual bytes. The remaining 3840 entries refer to phrases. The coding algorithm is as follows:

```
S := NULL
loop until end of stream
    E := INPUT SYMBOL
    if S*E exists in the dictionary
      S := S*E
    else
      output dictionary code for phrase S
      store S*E in dictionary
      S := E
    end if
end loop
output dictionary code for phrase S
```

As the dictionary is built up during the coding process no entries need to be transmitted. A variant LZC, used in the UNIX *compress* program, have variable length dictionary pointers starting at 9 bits [Thomas et al., 1985]. Alternative versions including LZT [Tischer, 1987] which uses a least-recently-used algorithms to discard and replace uncommon phrases once the dictionary becomes full and LZMW [Miller and Wegman, 1984] which concatenates the previous two phrases rather than just concatenating single characters. The favoured compressor for use as a general stream coder was a modified version of LZT as developed in [Turner, 1990] which seemed to consistently outperform the LZC algorithm.

A.2.3 Hybrid LZ77 and LZ78

Developments in a hybrid algorithm, (LZFG) using a standard sliding window and a trie dictionary data structure has been developed in [Fiala and Greene, 1989]. This tries to gain advantages from both systems.

Appendix B

Image Gallery

On voit que l'histoire est une gallerie de tableaux où il y a peu d'originaux et beaucoup de copies.
– Alexio de Tocqueville

Throughout the dissertation a wide variety of images have been used and presented. The following pages list general statistics for all the images and a selection of others[1]. Descriptive comments are given here:

Ali-Bongo (300 × 219) A computer animated front shot of the Ali-Bongo animated character in front of a scanned painted background. Copyright Cambridge Animation Systems.

Babbage (1152 × 768) A large ray-traced image of a lecture hall. The model consists of about 30,000 polygons rendered using the public domain raytracer *RayShade* Version 4.0 [Kolb, 1991].

Balloons (388 × 512) A famous test image, showing a girl within a cascade of smoothly shaded balloons, in 8-bit gray scale.

Bertha (512 × 480) A computer animated front face shot of the character "Bertha". Copyright Cambridge Animation Systems. To clarify the histogram pixel value 0, black, has been removed which accounts for 60.7% of the pixels.

Clown (405 × 270) A computer animated side shot of the "Animo Clown" in front of a scanned painted background. Copyright Cambridge Animation Systems.

Dog (462 × 377) A black and white image of one frame of an animated walking dog. Copyright Cambridge Animation Systems.

Domain (406 × 512) A computer generated image showing a shaded and textured map, created on a Silicon Graphics Workstation. To clarify the histogram pixel value 255, white, has been removed which accounts for 57.2% (118980) of the pixels.

Escher (304 × 456) A 300dpi scanned on an Apple Scanner, sketch drawing by M.C. Escher in 8-bit gray scale. The image has a white border so the histogram pixel value 255, white, has been removed which accounts for 25.2% (34988) of the pixels. Interesting characteristics of the scanner acquisition software reveals a bug that has resulted in every seventeenth value being zero.

Gardens (256 × 192) An image captured from a video camera showing a public garden with surrounding buildings.

[1]Statistics were calculated using specifically designed pieces of code and directly from the graphics display packages of XV 3.0 [Bradley, 1993, pages 24–25] and Photoshop [Adobe, 1991, pages 154–156].

Lenna (512 × 480) A common test image of a Playboy centre-fold Miss Lenna Sjooblom, Miss November, 1972, in both 8-bit gray scale and true colour versions.

Letter H (2480 × 3600) A scanned in image of a study of the letter H. Copyright Jacqueline A. Davies.

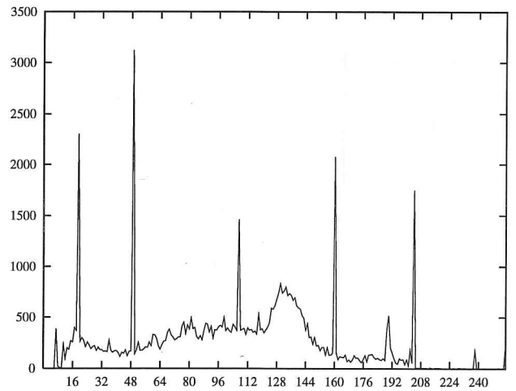
Man (357 × 337) A black and white image of one frame of an animated walking man. Copyright Cambridge Animation Systems.

Mandrill (512 × 480) a famous test image of a full frontal view of the mandrill monkey with lots of nearly random hair, in 8-bit gray scale.

Porsche (320 × 240 × 11) A ray-traced image sequence of a porsche in a showroom using the public domain raytracer *RayShade* Version 4.0. The sequence of eleven frames is designed for viewing on the Cambridge Autostereo Display [Travis and Lang, 1990].

Random (512 × 512) A picture created using my favourite random number generator by G.J. Mitchell and D.P. Moore, described in [Knuth, 1969, Section 3.2.2, pp 26–27].

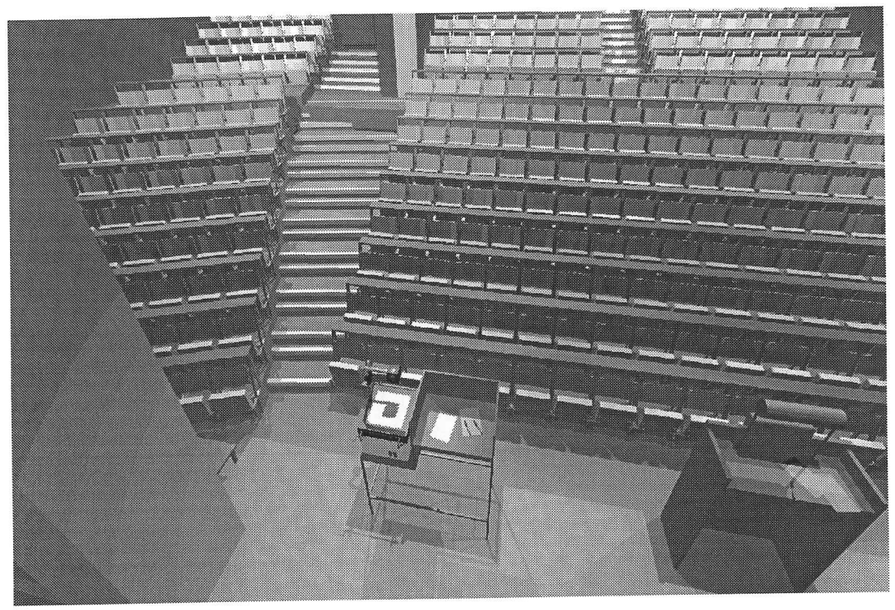
Straw (512 × 512) One of a set of texture images. Statistics show the image both in its original state and once it had been heavily contour merged as defined in the texture analysis in Section 7.4.

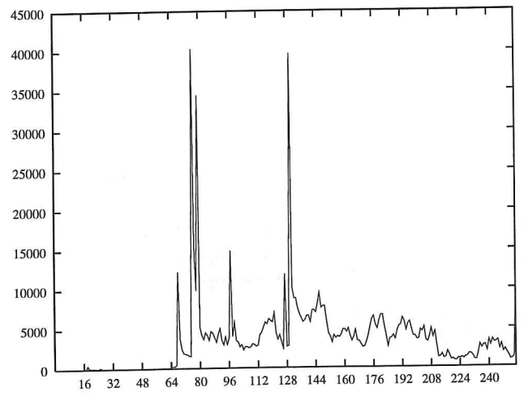


Pixels	65700
X-Size	300
Y-Size	219
Mean	103.62
Std Dev.	51.54
Median	107
1-Entropy	7.165
2-Entropy	5.473

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
65708	*40191*	40685	41962	40608	42002	56096	47236
1	*1.635*	1.615	1.566	1.618	1.564	1.171	1.391

Figure B.1: Test Image: Ali-Bongo.

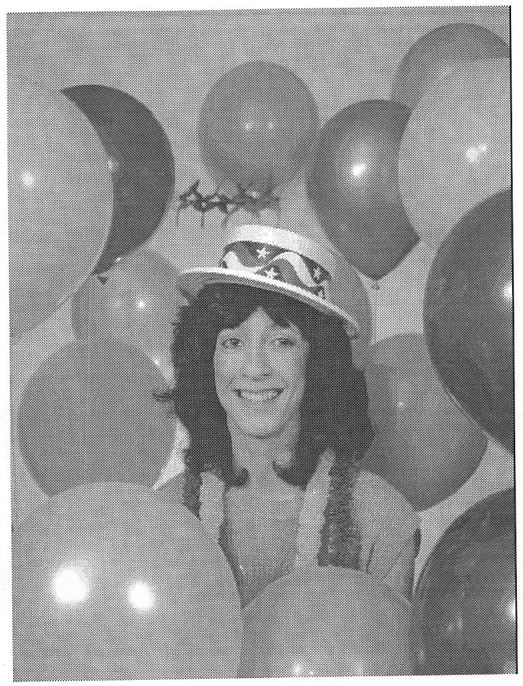


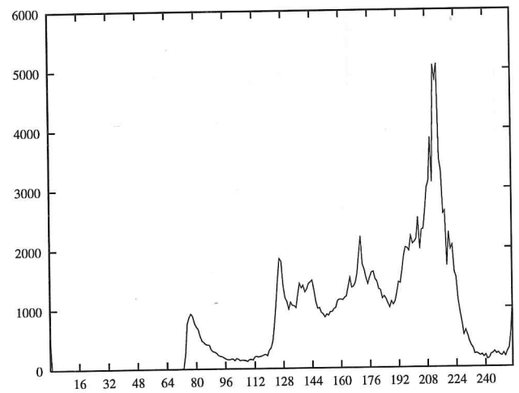


Pixels	884736
X-Size	1152
Y-Size	768
Mean	143.56
Std Dev.	50.05
Median	138
1-Entropy	7.138
2-Entropy	4.669

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
884745	*210133*	215229	218106	215082	223146	701784	322606
1	*4.210*	4.111	4.056	4.114	3.965	1.261	2.742

Figure B.2: Test Image: Babbage.

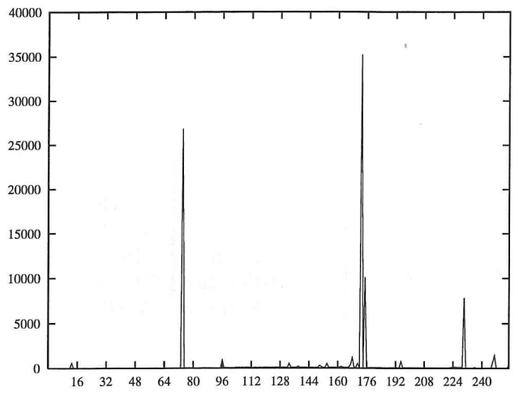




Pixels	198656
X-Size	512
Y-Size	388
Mean	178.26
Std Dev.	41.12
Median	188
1-Entropy	7.0374
2-Entropy	5.160

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
198664	*95792*	98915	99674	96587	98968	165468	117009
1	*2.074*	2.008	1.993	2.057	2.007	1.201	1.698

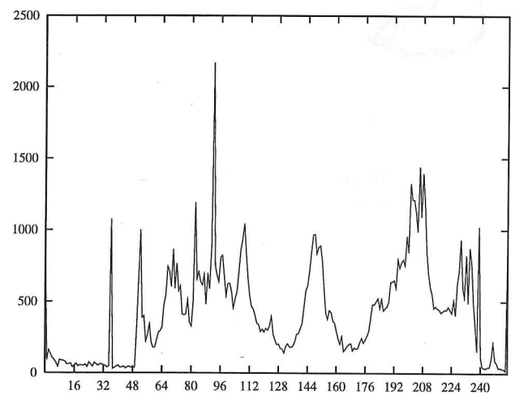
Figure B.3: Test Image: Balloons.



Pixels	245760
X-Size	512
Y-Size	480
Mean	58.37
Std Dev.	80.02
Median	0
1-Entropy	2.229
2-Entropy	1.294

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
245768	12816	12686	12867	*12326*	12536	55162	19036
1	19.177	19.373	19.100	*19.939*	19.605	4.455	12.911

Figure B.4: Test Image: Bertha.



Pixels	109350
X-Size	405
Y-Size	270
Mean	144.26
Std Dev.	62.69
Median	147
1-Entropy	7.551
2-Entropy	5.678

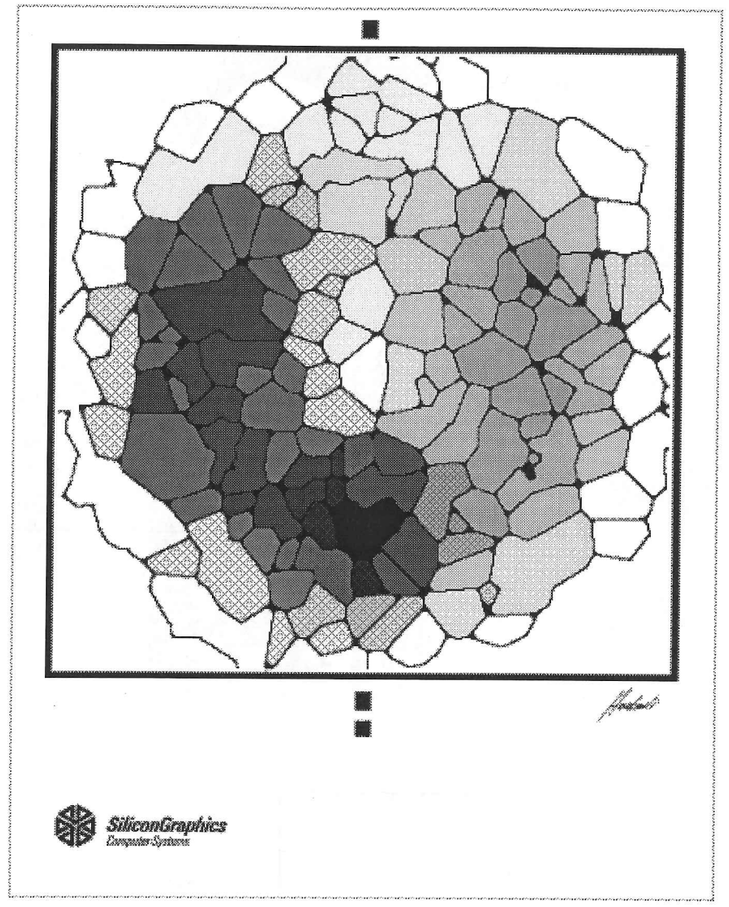
Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
109358	*65857*	67531	68685	67001	68183	95676	78754
1	*1.661*	1.619	1.592	1.632	1.603	1.143	1.389

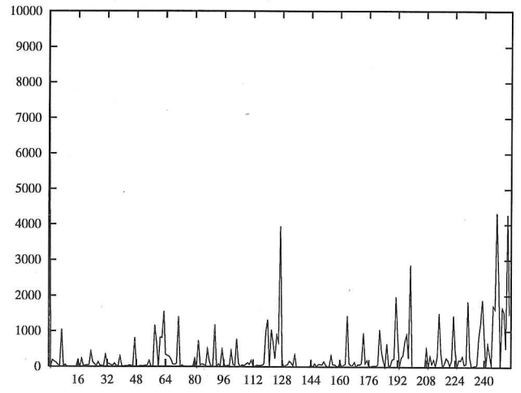
Figure B.5: Test Image: Clown.

Pixels	174174	Mean	240.95
X-Size	462	Std Dev.	58.18
Y-Size	377	Median	255
No. Black	9597	1-Entropy	0.308
No. White	164577	2-Entropy	0.230

Original	Contour Coder	Group 3	Group 4	AC1	LZT
21773	*1974*	7171	2730	7242	4743
1	*11.030*	3.036	7.975	3.006	4.591

Figure B.6: Test Image: Dog.

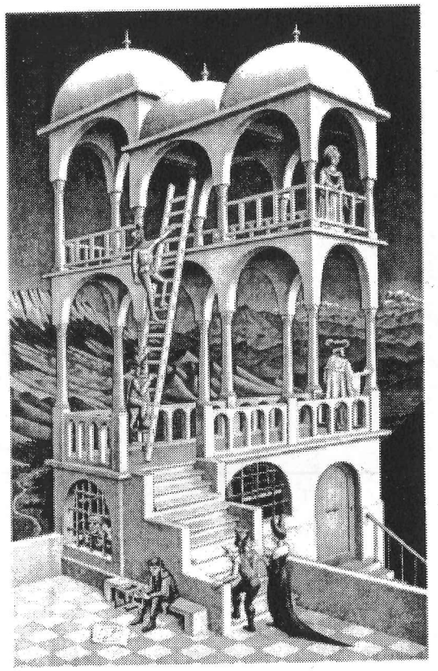


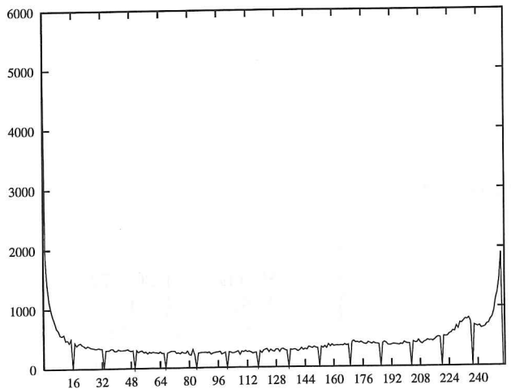


Pixels	207872
X-Size	406
Y-Size	512
Mean	211.33
Std Dev.	76.71
Median	255
1-Entropy	3.590
2-Entropy	2.280

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC	LZT
207880	31839	30870	31832	28357	*27690*	74878	34227
1	6.529	6.734	6.531	7.331	*7.507*	2.776	6.074

Figure B.7: Test Image: Domain.

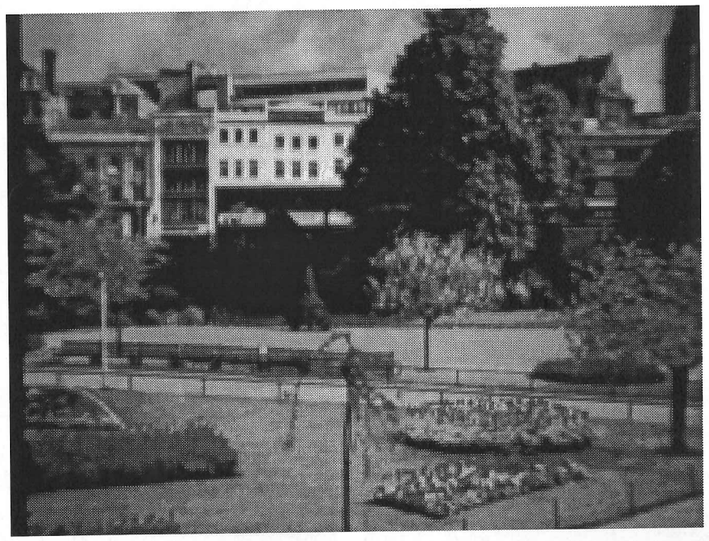


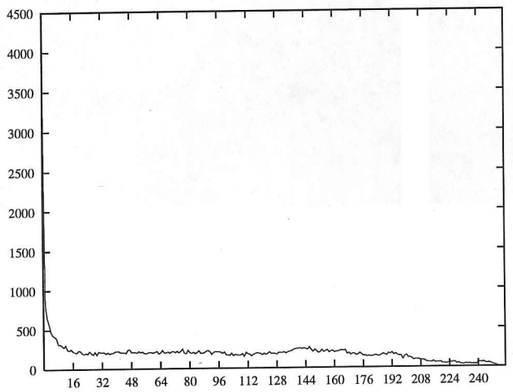


Pixels	138624
X-Size	304
Y-Size	456
Mean	162.62
Std Dev.	93.66
Median	195
1-Entropy	6.502
2-Entropy	5.518

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC	LZT
138632	*95445*	95799	100044	96710	101402	109039	102121
1	*1.452*	1.447	1.386	1.433	1.367	1.271	1.358

Figure B.8: Test Image: Escher.





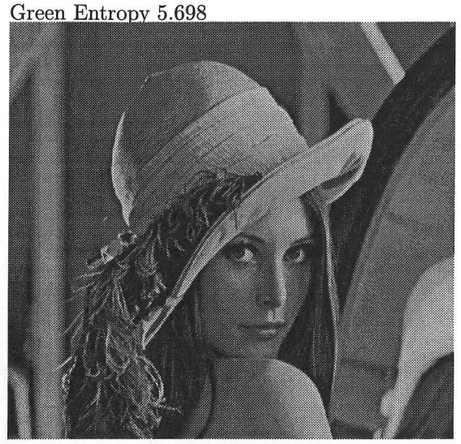
Pixels	49152
X-Size	256
Y-Size	192
Mean	90.47
Std Dev.	68.58
Median	85
1-Entropy	7.553
2-Entropy	6.226

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
49160	*40168*	40462	41901	40699	42394	45928	44036
1	*1.224*	1.215	1.173	1.208	1.160	1.070	1.116

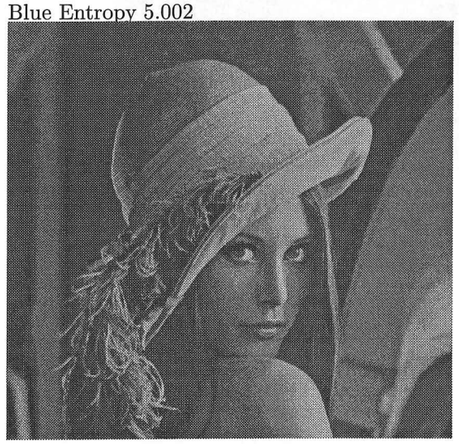
Figure B.9: Test Image: Gardens.

Red Entropy 4.810

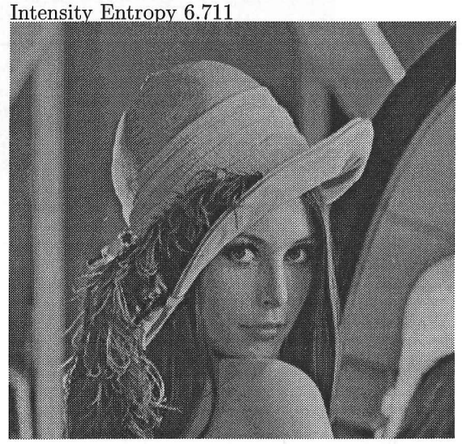
Green Entropy 5.698



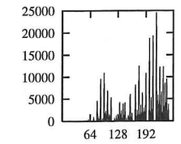
Blue Entropy 5.002



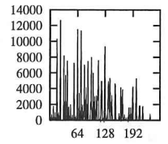
Intensity Entropy 6.711



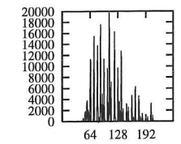
Red



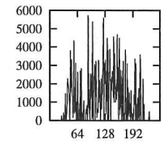
Green



Blue



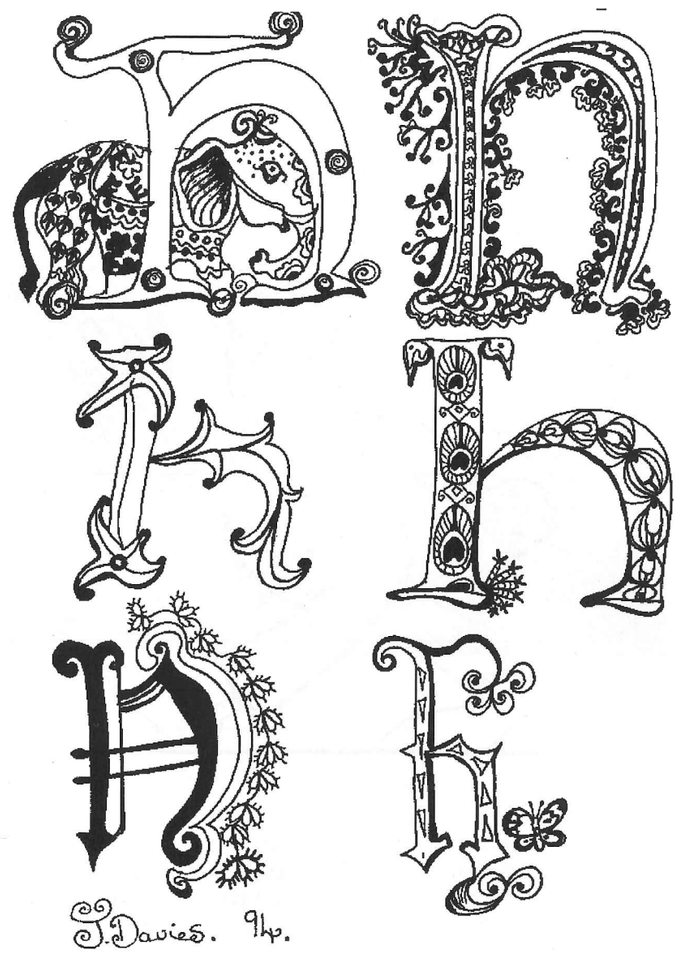
Intensity



	Red	Green	Blue
Pixels	245760	245760	245760
X-Size	512	512	512
Y-Size	480	480	480
Mean	180.70	99.32	106.16
Std Dev.	48.84	53.06	34.82
Median	200	100	104

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
737289	339280	346238	352258	326371	*326163*	460698	379302
1	2.173	2.129	2.093	2.259	*2.260*	1.600	1.944

Figure B.10: Test Image: Lenna.



Pixels	8928000	Mean	213.52
X-Size	2480	Std Dev.	94.11
Y-Size	3600	Median	255
No. Black	1452431	1-Entropy	0.641
No. White	7475569	2-Entropy	0.379

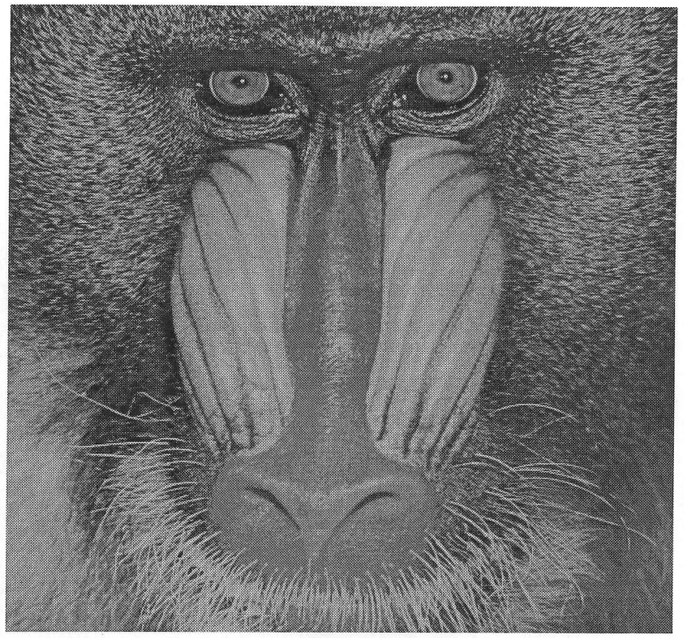
Original	Contour Coder	Group 3	Group 4	AC1	LZT
1116001	*48717*	250287	86876	689734	157993
1	*22.908*	4.459	12.846	1.618	7.064

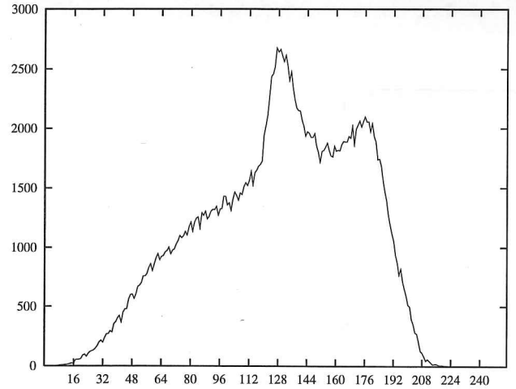
Figure B.11: Test Image: Letter H.

Pixels	120309	Mean	243.41
X-Size	357	Std Dev.	53.12
Y-Size	337	Median	255
No. Black	5470	1-Entropy	0.267
No. White	114839	2-Entropy	0.211

Original	Contour Coder	Group 3	Group 4	AC1	LZT
15040	*1586*	5723	2415	4550	3538
1	*9.483*	2.628	6.228	3.305	4.251

Figure B.12: Test Image: Man.

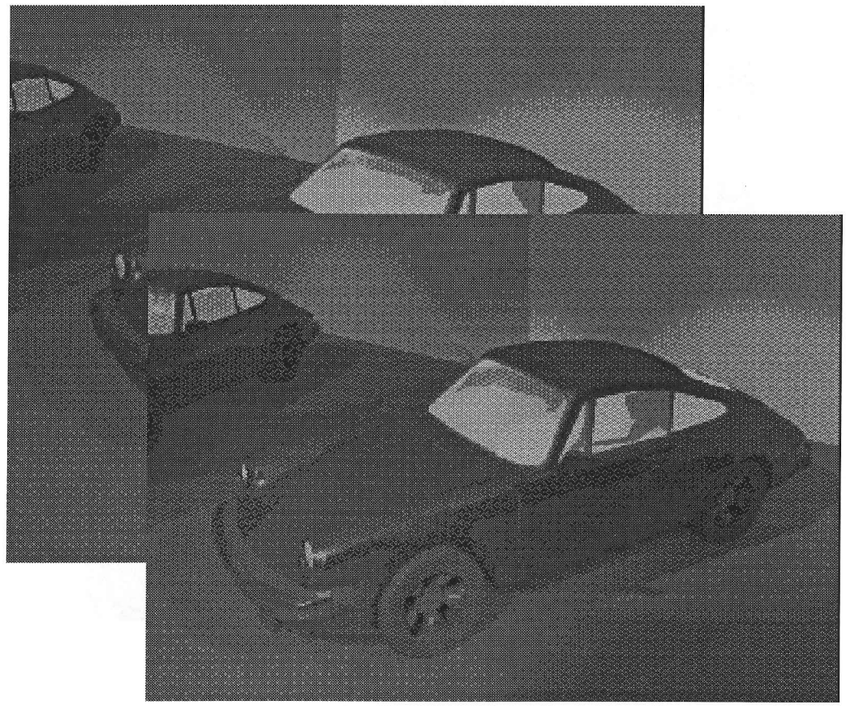


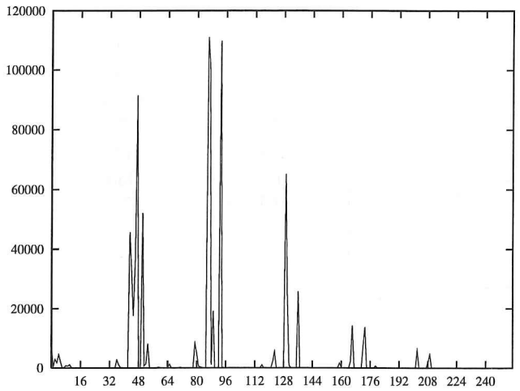


Pixels	245760
X-Size	512
Y-Size	480
Mean	128.85
Std Dev.	41.72
Median	132
1-Entropy	7.338
2-Entropy	6.737

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
245768	*206458*	206641	216577	209817	220603	222461	233575
1	*1.190*	1.189	1.135	1.171	1.114	1.105	1.052

Figure B.13: Test Image: Mandrill.

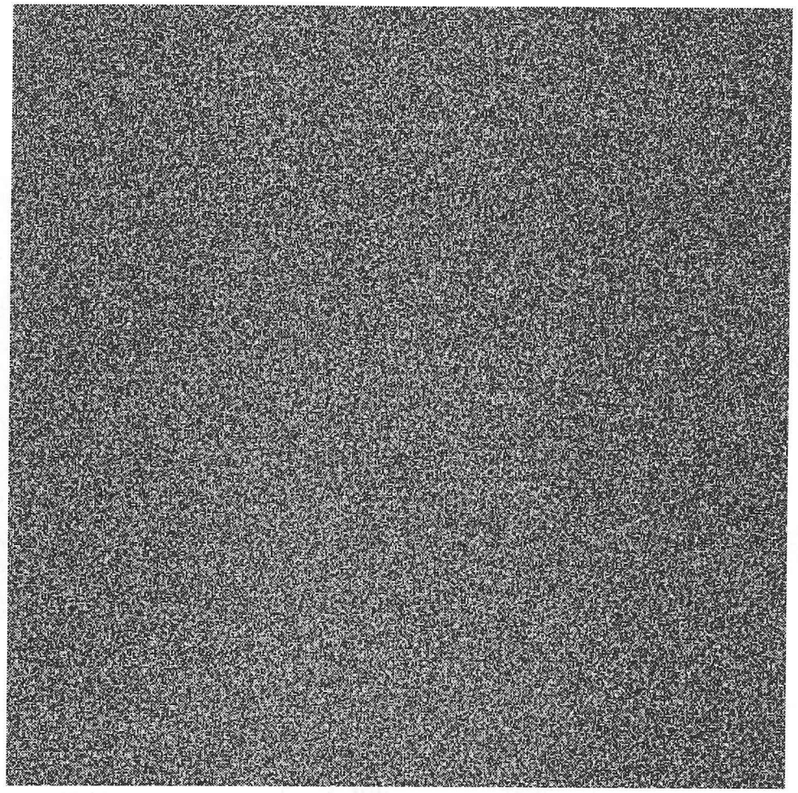


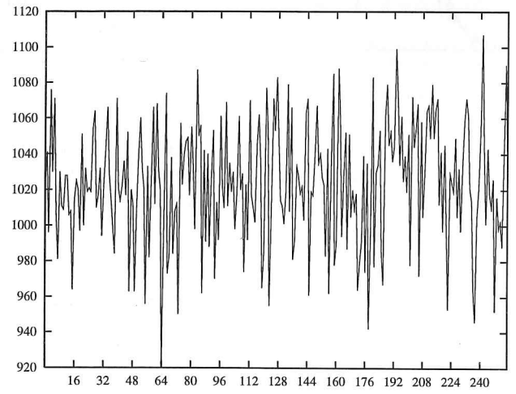


Pixels	884736
X-Size	320
Y-Size	240
No. Frames	11
Mean	83.42
Std Dev.	39.25
Median	86
1-Entropy	4.157
2-Entropy	3.244

Original	LZT	AC	L-JPEG	Contour Tree	Shell Tree
844811	171855	255971	522283	259173	258966
1	4.916	3.300	1.612	3.260	3.262

Figure B.14: Test Image: Porsche.

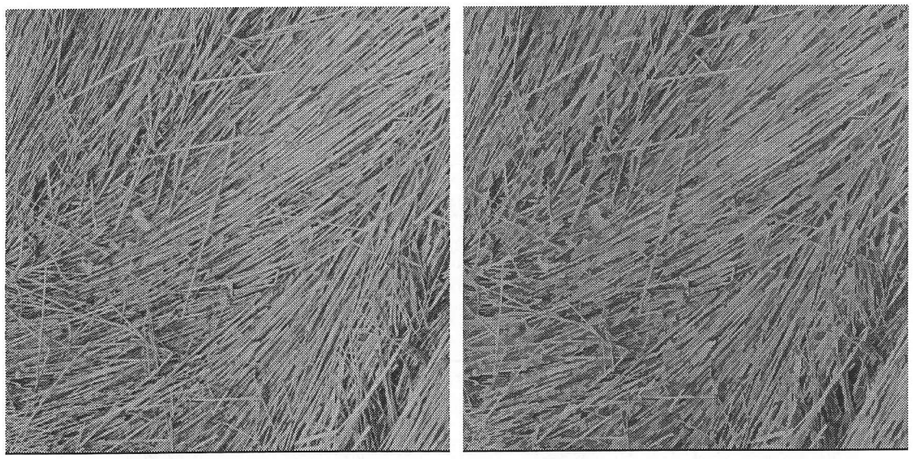


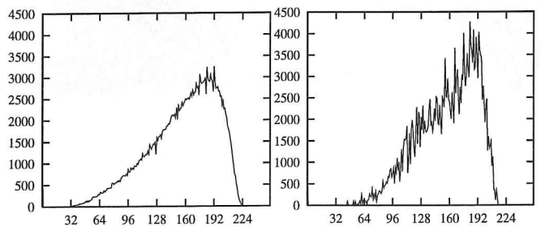


Pixels	262144
X-Size	512
Y-Size	512
Mean	127.69
Std Dev.	73.89
Median	128
1-Entropy	7.999
2-Entropy	7.904

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
262152	262892	263112	266430	263560	269336	262485	281229
1	0.997	0.996	0.984	0.995	0.973	0.999	0.932

Figure B.15: Test Image: Random.





	Original	Merged
Pixels	262144	262144
X-Size	512	512
Y-Size	512	512
Mean	156.77	157.12
Std Dev.	38.83	34.35
Median	164	163
1-Entropy	7.152	6.930
2-Entropy	6.782	4.899

Original	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
262152	*226280*	226332	236480	229172	242374	233679	249914
1	*1.159*	1.158	1.109	1.144	1.082	1.122	1.049

Merged	4MTC4	4MTC8	4MAE	8MTC8	8MAE	AC1	LZT
262152	54531	57042	55322	49626	*46831*	219411	130404
1	4.807	4.596	4.739	5.283	*5.598*	1.195	2.010

Figure B.16: Test Image: Straw; Original and Heavily Merged Version.

Glossary

adaptive coding — A type of compression scheme which specifies a model to be used for coding on the data already encoded – also known as dynamic coding.

arithmetic coding — A coding method which assigns codes to symbols which have a known probability distribution.

conservative — Conservative coding refers to any coding method which results in an exact copy of the original once decoded. Also called lossless or error-free coding.

comma-free code — See prefix code.

compression — A reversible process which aims to reduce the size of a set of data without removing any relevant information.

contour — A contour is a description of the outside boundary and characteristics of a region.

contour tree — A hierarchical structure defining a set of contours. It stores those contours totally enclosed by another contour as its children.

decoder — An algorithm which performs decompression.

decompression — The reverse operation to compression.

dynamic markov compression — Compression scheme based on an evolving state model of symbol's frequencies.

encoder — An algorithm which performs compression.

entropy — A measure of information content in a stream of data. Often described as the maximum number of bits per symbol required to completely represent any message defined in the stream.

ergodic — A stream of data is ergodic if any sufficiently long sequence extracted is representative of the entire stream.

image — In this dissertation almost exclusively refers to a discrete image.

lossless — See conservative.

lossy — See non-conservative.

leaf — A node in a tree with no children.

non–conservative	Non-conservative coding refers to any coding method which does not result in an exact copy of the original once decoded. Also called lossy coding.
pixel	(a) a single sample value in an image; (b) the theoretical area represented by such a sample value in an intensity surface which generates, or is generated by, that image. Also called a *pel*.
prefix code	A code which has no codeword as a prefix of another codeword. Also called a comma-free code.
region	A connected area on an image whose pixels have a constant feature.
resample	To take a discrete image and produce another image which appears to be a spatially transformed version of the first.
sampler	A process that produces a discrete image from a continuous intensity surface.
shell	A three dimensional extension of the contour.
shell tree	A three dimensional extension of the contour tree.
trie	An n-way tree which has a path for each possible string inserted into it. Allows for rapid string searches.
transform	A process which takes a continuous intensity surface and produces another surface from it.

Acronyms

Presented is a list of the acronyms used in the dissertation.

AC	Arithmetic Coder
ACF	Auto-Correlation Function
ANSI	American National Standards Institute
ASCII	American Standard Code Information Interchange
BBF	Backtracking Bug Follower Algorithm
CC	Contour Coder
CCD	Charged Couple Device
CCIR	Comitée Consultatif International Radio
CCITT	Comitée Consultatif International Télégraphique et Téléphonique
CD	Compact Disc
CIE	Commission Internationale de L'Eclairage
CIF	Common Intermediate Format. (Spatio-temporal resolution of $360 \times 288 \times 30$)

CODEC	COder-DECoder
CT	Contour Tree
DCT	Discrete Cosine Transform
DMC	Dynamic Markov Coder
DMM	Dynamic Markov Model
DPCM	Differential Pulse Code Modulation
FDCT	Fast Discrete Cosine Transform
FFT	Fast Fourier Transform
FIFO	First In First Out
FT	Fourier Transform
GIF	Graphical Interchange Format
HDTV	High Definition Television (1920×1080 – Japanese, 1920×1152 – European or 1920×1050 – American standard)
ISO	International Standards Organisation
JBIG	Joint Bi-level Image experts Group
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group
LIFO	Last In First Out
LZ77/LZ78	Lempel Zif coders '77 and '78
LZT	Lempel-Zif '78 coder by Tischer
LZW	Lempel-Zif-Welch coder
MPEG	Moving Picture Experts Group
NACF	Normalised Auto-Correlation Function
NCSA	National Centre for Supercomputing Applications
NTSC	National Television System Committee (640×480)
PAL	Phase-Alternation-Line (768×576)
SECAM	Séquentiel Couleurà Mémoire
SMPTE	Society for Motion Picture and Television Engineers
VCR	Video Cassette Recorder
VDU	Visual Display Unit
VQ	Vector Quantisation
4MAE	Four-connected Square Contours, Movement Around the Edge of Pixels, Boundary route.

4MTC4 Four-connected Square Contours, Movement Through the Centres of Pixels in a Four-connected Boundary route.

4MTC8 Four-connected Square Contours, Movement Through the Centres of Pixels in a Eight-connected Boundary route.

6MAE Six-connected Hexagonal Contours, Movement Around the Edge of Pixels Boundary route.

6MTC6 Six-connected Hexagonal Contours, Movement Through the Centres of Pixels in a Six-connected Boundary route.

8MAE Eight-connected Square Contours, Movement Around the Edge of Pixels Boundary route.

8MTC8 Eight-connected Square Contours, Movement Through the Centres of Pixels in a Eight-connected Boundary route.

The information in the next few lists has been gathered over the past few years whilst maintaining the University of Cambridge Computer Laboratory's Rainbow Graphics Group's filespace. Due to the evolving nature of software the information below can not claim to be complete nor fully accurate. Introductions to the popular image file formats currently available are given in [Brankin and Mumford, 1992] or [Carlson, 1991], and there are numerous related news groups within the Internet.

Popular File Formats

A list of some of the popular file endings for image format files, printer files and archived files.

arc ARChive compressed file, for MSDOS, Unix (wuarchive.wustl.-edu:/mirrors/misc/unix/arc521e.tar-z), VMS, Mac and Amiga. Uses an LZW algorithm.

arj ARJ compressed file used on IBM PCs. Uses an LZ77 algorithm with hashing, plus secondary static Huffman encoding on a block basis.

bmp IBM PC Windows and OS/2 BitMaP picture format.

cpt Macintosh CopactPro compressed file (sumex-aim.stanford.-edu:/info-mac/util/compact-pro-133.hqx [36.44.0.6]).

ct Scitex Continuous Tone colour printer format for CMYK colour images and gray-scale images [Scitex, 1988].

dc DigiCipher proprietary broadcast video compression algorithm developed by General Instruments. Based heavily on MPEG-2 with the main difference being in the audio which supports Dolby AC-3 audio (a 5.1 channel system, part of Dolby SR-D). This is partially to avoid the use of MPEG audio which is proprietary with several different organisations holding ownership.

dham Amiga, Dyna-Hold-And-Modify picture file in IFF.

dib IBM PC Windows and OS/2 BitMaP picture format.

dl System independent animated picture file.

dvi DeVice Independent printer file originally designed by David R. Fuchs in 1979 and used in Donald E. Knuth's TEX. A DVI file is a stream of 8-bit bytes, which can be regarded as a sequence of commands. It is fully defined in [Knuth, 1986, Parts 583–591].

dxf An interchange format for the CAD package Autocad. It has become an industry standard [Autodesk, 1990] and a conformance testing suite has been developed by the CAD-CAM Data Exchange Technical Centre in Leeds, UK.

exe Self-extracting MSDOS executable which creates files on disk when run.

F Freeze compressor for Unix, using an LZ77 coder with hashing, plus secondary dynamic Huffman encoding (wuarchive.wustl.-edu:/usenet/comp.sources.misc/volume25/freeze/part0[1-2].Z).

fig Xfig mixed picture file format, Original copyright (c) 1985 by Supoj Sutanthavibul, University of Texas at Austin. Current version 2.1.5 uses protocol 2.1.

fli/flc System independent animated picture file format developed by Jim Kent for Autodesk Animator and used by IBM's Ultimedia Tool Series and Microsoft's Video for Window. Uses a full frame difference run length coding scheme, full details in [Kent, 1993].

fs Usenix FaceSaver picture file. FaceSaver is a registered trademark of Metron Computerware Ltd. of Oakland, CA.

fits Flexible Image Transport System picture file. Defined by the NSDSSO (NASA Science Data Systems Standards Office) which has published draft 100-0.1 December 1990 [NASA, 1990].

g3/g4 CCITT Group 3 and Group 4 facsimile picture file standards. Details in [CCITT, 1985a, CCITT, 1985b].

gf Geometric Features file format. Represents a mixture of geometric features e.g. lines, points, polygons, arcs, circles, etc. combined with real numbers, text and pointers to other features in simple plain ASCII. Designed for viewing under the Xgf display program.

gif The Compuserve Graphics Interchange Format. Highly compressed colour image format using lookup tables. Image resolution varies from 1 bit to 8 bits per pixel. It uses a version of the LZW algorithm. The Two main versions are GIF87a [CompuServe, 1987] and GIF89a [CompuServe, 1990].

gl System independent animated picture file.

gz UNIX gnu LZ77 *gzip* compressed file format, see section A.2.1. Uses the same algorithm as zip 1.9 below.

ha Compressor *ha* 0.98 for MSDOS. Uses an improved PPMC - 4th Order Markov modeling Arithmetic Coder Model.

ham Amiga, Hold-And-Modify picture file in IFF. Encoding scheme which packs the equivalent of 12 bits of image data into 6 bits per pixel. HAM images initially could only be 320×300 pixels or 320×300 pixels but non-standard HAM images are now used. HAM pixels are rectangular requiring a resize of $83\% \times 120\%$ for non-interlaced images or $166\% \times 60\%$ for interlaced images.

hdf Hierarchical Data Format is a multi-object file format for transfer of graphical and floating point data. It was developed by the NCSA [NCSA, 1989].

hips Human Information Processing Laboratory, NYU picture file.

hqx Macintosh BinHex encoded file, for ASCII only storage and transmission.

iae/ipe Image Analysis and Processing Environment picture file formats designed by Neil A Dodgson in the Computer Lab, Cambridge University, UK.

ice Image Coding Environment picture file format designed by Martin J. Turner in the Computer Lab, Cambridge University, UK. Full details in [Turner, 1990].

icr NCSA (National Centre for Supercomputing Applications) Telnet Interactive Colour Raster graphic file format, implemented in telnet for the Macintosh version 2.3. The ICR protocol shares characteristics of the Tektronix graphics terminal emulation protocol.

iff Interchangeable File Format: ILBM/HAM/DHAM/SHAM, developed by Electronic Arts [Morrison, 1985]. Images are specified from 1 bit to 8 bits per pixel. The format can also deal with a variety of different objects which may include text, font, music, sound affects, animation etc.

iif Image Interchange Facility part of the Image Processing and Interchange currently under review by ISO/IEC JTC1/SC24 *Computer Graphics and Image Processing*, designed to represent multi-dimensional data [ISO/IEC, 1993a].

ilbm Amiga, InterLeaved BitMap picture file in IFF.

im8/rast Sun RASTer picture file format designed for use with the pixrect graphics library [Sun, 1988].

imj GEM picture file format.

it8 American National Standards Committee file formats for the graphics art [ANSI, 1988a, ANSI, 1988b, ANSI, 1990]. Media dependent designed for magnetic tapes. Describes line-art, monochrome and colour image types.

jpg/jpeg Joint Photographic Experts Group picture file format described in [Wallace, 1991]. Designed for lossy colour and monochrome image compression. Includes a lossless mode. Details of the JPEG File Interchange Format from C-Cube in [Hamilton, 1992].

lzh/lha Compressed file, output from *lharc* or *lha*. Uses an LZ77 coder with a trie data structure, plus secondary static Huffman coding on a block basis.

mtv MTV ray tracer picture file format by Mark Terrence VandeWettering. Very simple three channel 24-bit RGB raw format.

mac/macp Macintosh MacPaint picture file for bitmapped images no larger than 576×720 pixels.

miff ImageMagick picture file format for use on the ImageMagick image manipulator written by cristy@dupont.com. Copyright 1991 E. I. du Pont de Nemours and Company.

mpg/mpeg Moving Picture Experts Group picture file format. Good described in [Legall, 1991]. Designed for lossy full colour motion image compression. MPEG1 has been standardised for CIF images, and a higher resolution version MPEG2 is currently being designed.

off 3D Object File Format developed by Digital Equipment Corporation [Rost, 1989]. Only polygonal objects can be handled but extensions to cater for bezier surface patches and other primitives should be straightforward.

pac Pak for MSDOS which uses an LZW coder (wuarchive.wustl.-edu:/mirrors/msdos/archivers/pak251.exe).

pcx IBM PC Paintbrush picture file format established by Zssoft. Supports up to 8 bits per pixel indexed colour. Defined in ZSoft's technical reference booklet [ZSoft, 1988]. The encoding used is a simple byte oriented RLE technique.

PGZ PostScript file compressed using the UNIX gnu LZ77 *gzip* compressed file format, see section A.2.1.

pi1/pi3 Atari Degas picture file format.

pict Macintosh Quckdraw PICTure file allowing image resolutions from 1 bit to 32 bits per pixel [Mac, 1988]. Includes an 8 bit per pixel System Palette option which allows automatic selection of the Macintosh's system colour table.

pit Macintosh PackIt archive file (sumex-aim.stanford.edu:/info-mac/util/stuffit-151.hqx [36.44.0.6]).

pj Hewlett Packard's PaintJet picture file format.

pnm/ppm/pgm/pbm Jef Poskanzer's X11 portable file formats. Distributed with X Windows. Latest version for the full pbmplus toolkit was released on 1st March 1994. Contactable at jef@well.sf.ca.us or {apple, ucbvax}!well!jef.

pp Amiga PowerPacker (ftp.funet.fi:pub/amiga/fish/501-600/ff561/-PPLib.lha).

psid PostScript Image Data used to represent 1,2,4 or 8 bits per sample, gray scale raster data for the *image* command. Described in [Adobe, 1985a, page 181].

raw/rgb 24-bit RGB system independent picture file with no header.

rgbn/rgb8 Image file formats used in Impulse Inc's Silver and Turbo Silver 3D rendering packages. They are almost identical to ILBM.

rlb Wavefront rlb image file format.

rlc Wasatch Systems image file format.

rle University of Utah Run Length Encoded picture file format. Full details in [Peterson et al., 1986].

sea Macintosh Self-Extracting Archive file.

shar Shell ARchive which is not a compression program.

sham Amiga, Sliced-Hold-And-Modify picture file in IFF.

shk Macintosh SHrinKit compressed file.

sit Macintosh StuffIT compressed file (sumex-aim.stanford.edu:/info-mac/util/stuffit-lite-30.hqx [36.44.0.6]).

sixel DEC sixel picture file format. Designed for output to a DEC colour printer for example the LJ250 inkjet. Format has two modes one uses an uncompressed raw data format whilst the other uses a simple "repeat pixel" command.

spu Atari Spectrum picture file format. Includes a compressed and non-compressed versions.

sqz MSDOS Squeeze compression file. Uses an LZ77 coder with hashing (wuarchive.wustl.edu:/mirrors/msdos/archivers/sqz10-83e.exe).

tar A multivolume archive format file incorporated with the UNIX operating system.

tga Truevision TarGA TIPS picture file format commonly supported by MS-DOS colour applications. Current version 2.0 is described in [Truevision, 1991].

TGZ Tar file compressed using the UNIX gnu LZ77 *gzip* compressed file format, see section A.2.1.

tiff/tif Tagged Image File Format developed by Aldus [Aldus, 1992]. It is one of the most versatile formats allowing in its baseline form bilevel, grayscale, indexed RGB and has extensions for CMYK and various CIE colour spaces. Various compression schemes can also be specified.

uil Motif UIL icon file format.

uue UNIX UUE encoded file for ASCII storage and transmission.

viff File format developed for the Khoros [Khoros, 1991] image processing package.

xbm/xpm X11 or X10 Bit or Pix Map format.

xim Portable X11 or X10 picture format.

xwd X11 or X10 window dump file format.

Y Yabba compressed file. Uses Y coding an LZ78 variant. Contact Dan Bernstein at <brnstnd@nyu.edu> for details.

Z UNIX LZW *compress* file compression, using a modified LZ78 algorithm as described in Section A.2.2.

z UNIX Adaptive Huffman *pack* file compression, using an adaptive version of huffman coding.

zinc Zinc Interface library bitmap file.

zip MS_DOS PKZIP compressed file format. Uses an LZ77 algorithm with hashing, plus secondary static Huffman coding on a block basis. Current version is 2.04g, contact pkware.inc@mixcom.com for up to date details.

zoo MS_DOS ZOO compressed file archive format. Algorithm copied from the *lha* compressor above.

Popular Document Languages

Acrobat Adobe's new *Super-Format* announced in late 1992 based on PostScript and on a technology called Portable Document Format (PDF) [Adobe, 1990a].

CGM Computer Graphics Metafile designed for exchanging picture information. Included in the Department of Defence's CALS initiative. New edition, CGM 1992, of the standard is superior in some capabilities to PostScript [ISO/IEC, 1992].

EPS Encapsulated PostScript either in ASCII or binary code. Quark developed the *Desktop Color Separations DCS* extension for CMYK documents. EPS is a single image PostScript file with restrictions and special comment conventions allowing integration of images from different sources.

HPGL Hewlett-Packard Graphics Language is a mnemonic graphics language which can be interpreted directly by HP plotters [HP, 1984].

ODA Open Document Architecture and Interchange Format developed to allow open transfer of character, raster graphics and geometric graphics content. Specified in [ISO/IEC, 1993c], and concisely defined in [Appelt, 1991]. Developed in harmony with the CCITT Study Group VIII who produced the T.410 Series of recommendations which is technically identical.

PCL 5 Hewlett-Packard page description language.

PhotoCD Compact Disc high resolution image format introduced by Kodak [Kodak, 1992].

PS PostScript printer-ready file format by Adobe Systems, defined in the PostScript Language Reference Manual [Adobe, 1990c, Adobe, 1985b, Adobe, 1990b]. Latest version is Level 2 which includes improved pattern fills, space management, text handling and most importantly a colour concept that deals with device dependent and device independent colours.

SGML Standard Generalized Markup Language [ISO/IEC, 1986] specifies a document markup syntax defining a Document Type Definition (DTD). A Document Style Semantics and Specification Language (DSSSL) is required to create a final form [ISO/IEC,]. A guide is given in [Bryan, 1988].

SPDL Standard Page Description Language has a very similar semantics to PostScript with different Syntax [ISO/IEC, 1993b].

Image Conversion Tools

ImageMagick Interactive manipulation and X11 viewer. Anonymous FTP from export.lcs.mit.edu:contrib/ImageMagick.tar.Z

Img Software Set Manipulation and X11 viewer. Anonymous FTP from export.-lcs.mit.edu:contrib/img_1.3.tar.Z and venera.isi.edu:pub/img_1.3.-tar.Z

Independent JPEG Group A set of routines for reading and writing JPEG files. Anonymous FTP from uunet.uu.net:graphics/jpeg/jpegsrc.v?.tar.Z

LaboImage A SunView image manipulation and analysis package. Anonymous FTP from ads.com:pub/VISION-LIST-ARCHIVE/SHARE-WARE/LoboImage_3.1.tar.Z

PBMPLUS A comprehensive format conversion and manipulation package. Anonymous FTP from ftp.ee.lbl.gov:pbmplus*.tar.Z, wuarchive.-wustl.edu:graphics/graphics/packages/pbmplus/pbmplus*.tar.-Z and export.lcs.mit.edu:contrib/pbmplus*.tar.Z

TIFF software Portable library for reading and manipulating TIFF files. Included is **xtiff** an X11 TIFF file viewer. Anonymous FTP from ucbvax.berkely.edu:pub/tiff/*.tar.Z or uunet.uu.net:graphics/-tiff.tar.Z

Utah RLE Toolkit Conversion and manipulation package. Anonymous FTP from cs.utah.edu:pub/urt-*, weedeater.math.yale.edu:pub/urt-* and freebie.engin.umich.edu:pub/urt-*

Xim Manipulation and X11 viewer, available in the X11R4 source tree. Anonymous FTP from video.mit.edu

xloadimage X11 image viewer. Anonymous FTP from export.lcs.mit.edu:-contrib/xloadimage*

Referenced FTP sites

ads.com	128.229.30.16
cs.utah.edu	128.110.4.21
export.lcs.mit.edu	18.24.0.12
freebie.engin.umich.edu	141.212.68.23
ftp.ee.lbl.gov	128.3.112.20
nl.cs.cmu.edu	128.2.222.56
pprg.eece.umn.edu	129.24.24.10
sdsr.edu	132.249.20.22
sumex-aim.stanford.edu	36.44.0.6
titan.cs.rice.edu	128.42.1.30
ucbvax.berkeley.edu	128.32.133.1
venera.isi.edu	128.9.0.32
weedeater.math.yale.edu	130.132.23.17
wuarchive.wustl.edu	128.252.135.4

Bibliography

[Adobe, 1985a] Adobe (1985a). *PostScript Language Reference Manual.* Addison-Wesley, first edition. Preface by John Warnock, June 1985. ISBN: 0-201-10174-2.

[Adobe, 1985b] Adobe (1985b). *PostScript Language Tutiorial and Cookbook.* Addison-Wesley. Preface by Charles Geschke, August 1985. ISBN: 0-201-10179-3.

[Adobe, 1990a] Adobe (1990a). *Adobe Acrobat Products and Technology – An Overview.* Adobe Systems Incorporated, Hill Place House, 55a High Street, Wimbledon, London. SW19 5BA.

[Adobe, 1990b] Adobe (1990b). *Adobe Type 1 Format.* Adobe Systems Incorporated, Hill Place House, 55a High Street, Wimbledon, London. SW19 5BA.

[Adobe, 1990c] Adobe (1990c). *PostScript Language Reference Manual.* Addison-Wesley, second edition. Preface by John Warnock and Chuck Geschke, Decenmber 1990. ISBN: 0-201-18127-4.

[Adobe, 1991] Adobe (1991). *Adobe Photoshop: User Guide.* Adobe Systems Incorporated, Hill Place House, 55a High Street, Wimbledon, London. SW19 5BA. Macintosh Version 2.

[Aho et al., 1983] Alfred V. Aho, John E. Hopcroft, and Jeffery D. Ullman (1983). *Data Structures and Algorithms.* Addison-Wesley. ISBN: 0-201-00023-7.

[Aldus, 1991] Aldus (February 1991). *Aldus FreeHand User Manual: Version 3.0 for Apple Macintosh computers.* Aldus Corporation, 411 First Avenue, South Seattle, Washington 98104-2871 USA. +1 206 622 5500.

[Aldus, 1992] Aldus (1992). *TIFF – Tagged Image File Format – Revision 6.0.* Aldus Corporation, 411 First Avenue, South Seattle, Washington 98104-2871 USA. +1 206 622 5500.

[Ali et al., 1992] Maaruf Ali, Costas Popadopoulos, and Trevor Clarkson (March 1992). The use of fractal theory in a video compression system. In James C. Tilton and Martin Cohn, editors, *Data Compression Conference*, pages 259–268. IEEE Computer Society Press Order Number 2717.

[ANSI, 1988a] ANSI (1988a). User exchange format for the exchange of colour picture data between electronic prepress systems via magnetic tape. IT8.1 NPES.

[ANSI, 1988b] ANSI (1988b). User exchange format for the exchange of line art data between electronic prepress systems via magnetic tape. IT8.2 NPES.

[ANSI, 1990] ANSI (1990). User exchange format for the exchange of monochrome image data between electronic prepress systems via magnetic tape. IT8.5 NPES.

[Appelt, 1991] W. Appelt (1991). *Document Architecture in Open Systems: The ODA Standard.* Springer Verlag.

[Autodesk, 1990] Autodesk (1990). *Autocad Reference Manual.* Autodesk Ltd., Version 11.

[Barnsley and Sloan, 1988] Michael F. Barnsley and Alan D. Sloan (January 1988). A better way to compress images. *Byte*, pages 215–223.

[Bayer, 1973] B.E. Bayer (1973). An optimal method for two-level rendition of continuous-tone pictures. *Conference Record of the International Conference on Communication.*

[Bell et al., 1989] T.C. Bell, J.G. Cleary, and I.H. Witten (December 1989). Modelling for text compression. *ACM Computing Surveys*, 21(4):557.

[Bell et al., 1990] T.C. Bell, J.G. Cleary, and I.H. Witten (1990). *Text Compression.* Prentice Hall. ISBN: 0-13-911991-4.

[Boff and Lincoln, 1988] K.R. Boff and J.E. Lincoln, editors (1988). *Engineering Data Compendium, Human Perception and Performance.* Harry G. Armstrong Aerospace Medical Research Lab.

[Börner, 1987] R. Börner (1987). Progress in projection of parallax panoramagrams onto wide-angle lenticular screens. *Proc. SPIE: True 3D imaging techniques and display technologies*, 761:35–43.

[Bove Jr. and Lippman, 1992] V. Michael Bove Jr. and Andrew B. Lippman (January 1992). Scalable open-architecture television. *SMPTE*, pages 2–5.

[Bovik et al., 1990] Alan Conrad Bovik, Marianna Clark, and Wilson S. Geisler (January 1990). Multichannel texture analysis using localized spatial filters. *IEEE Trans. Pat. Anal. and Mac. Int.*, 12(1):55–73.

[Bradley, 1993] John Bradley (1993). *XV Interactive Image Display for the X Window System.* 1053 Floyd Terrace, Bryn Mawr, PA 19010. +1 (215) 898 8813, version 3.00 edition. Email: bradley@cis.upenn.edu.

[Brankin and Mumford, 1992] L.A. Brankin and A.M. Mumford (1992). File formats for computer graphics – unraveling the confusion. In *State of The Art Reports.* Eurographics Technical Report Series EG92 SA ISSN 1017-4656.

[Brent, 1987] R.P. Brent (1987). A linear algorithm for data compression. *Australian Computer J.*, 19(2):64–68.

[Bryan, 1988] M. Bryan (1988). *SGML: An Author's Guide.* Addison Wesley.

[Burt, 1981] Peter J. Burt (1981). Fast filter transforms for image processing. *Computer Graphics and Image Processing*, 16:20–51.

[Burt, 1984] Peter J. Burt (1984). The pyramid as a structure for efficient computation. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis.* Springer-Verlag.

[Cabrelli and Molter, 1990] Carlos A. Cabrelli and Ursula M. Molter (December 1990). Automatic representation of binary images. *IEEE Trans. Pat. Anal. and Mac. Int.*, 12(12):1190–1196.

[Carlson, 1991] W.E. Carlson (April 1991). A survey of computer graphics image encoding and storage formats. *Computer Graphics*, 25(2).

[CCIR, 1986] CCIR (1986). Recommendations 601–1: Encoding parameters of digital television for studios. Recommendations and Reports of the CCIR, Volume XI, Part 1.

[CCITT, a] CCITT. Information technology – digital compression and coding of continuous-tone still images: Requirements and guidelines. Rec. T.81 ISO/IEC 10918-1.

[CCITT, b] CCITT. ISO/IEC draft international standard 11544 coded representation of picture and audio information – progressive bi-level image compression. Draft T.82 WG9–S1R6.1.

[CCITT, 1985a] CCITT (1985a). Standardization of group 3 facsimile apparatus for ducument transmission. Recommendation T.4, Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services.

[CCITT, 1985b] CCITT (1985b). Standardization of group 4 facsimile apparatus for ducument transmission. Recommendation T.6, Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services.

[Cederberg, 1979] R.L.T. Cederberg (1979). Chain-link coding and representation for raster scan devices. *Comput. Graph. Image Proc.*, 10:224–234.

[Coltelli et al., 1993] P. Coltelli, G. Faconti, and F. Marfori (September 1993). On the application of quantization and dithering techniques to history of arts. In R.J. Hubbold and R Juan, editors, *Eurographics '93*, volume 12-3.

[CompuServe, 1987] CompuServe (15 June 1987). *GIF Graphics Interchange Format – a standard defining a mechanism for the storage and transmission of raster-based graphics information.* CompuServe, Incorporated, an H and R Block Company, 5000 Arlington Centre Blvd. Columbus, Ohio 43220 (+614) 457-8600.

[CompuServe, 1990] CompuServe (31 July 1990). *Graphics Interchange Format Programming Reference, Version 89a.* CompuServe, Incorporated, an H and R Block Company, 5000 Arlington Centre Blvd. Columbus, Ohio 43220 (+614) 457-8600.

[Cornsweet, 1970] T. N. Cornsweet (1970). *Visual Perception.* Academic Press.

[Crow, 1978] F.C. Crow (November 1978). The use of grayscale for improving raster display of vectors and characters. *Computer Graphics*, 12(3):1–5.

[Davidoff, 1975] J.B Davidoff (1975). *Differences in visual perception; the individual eye.* Crosby Lockwood Staples, London. ISBN: 0-258-96924-5.

[DeYoe and van Essen, 1988] E.A. DeYoe and D.C. van Essen (1988). Concurrent processing streams in monkey visual cortex. *Trends Neurosci.*, 11:219–226.

[Dodgson, 1992] Neil A. Dodgson (August 1992). Image resampling. Technical report, University of Cambridge, Computer Laboratory, New Museums Site, Pembroke Street, CB2 3QG. United Kingdom. TR-261.

[Ellis, 1938] Willis D. Ellis, editor (1938). *A source book of Gestalt Psychology.* Kegan Paul, Trench, Trubner and Co., Ltd.

[Eskicioglu and Fisher, 1993] Ahmet M. Eskicioglu and Paul S. Fisher (April 1993). A survey of quality measures for gray scale image compression. In James C. Tilton, editor, *Space and Earth Science Data Compression Workshop*, pages 49–61. NASA Conference Publication 3191.

[Eskicioglu et al., 1994] Ahmet M. Eskicioglu, Paul S. Fisher, and Siyuan Chen (April 1994). Image quality measures and their performance. In James C. Tilton, editor, *Space and Earth Science Data Compression Workshop*, pages 55–67. NASA Conference Publication 3255.

[Essman and Wintz, 1973] J. Essman and P.A. Wintz (1973). The effects of channel errors in DPCM systems and comparisons with PCM systems. *IEEE Trans. on Comm.*, COM-21(8):867–877.

[Fano, 1949] R.M. Fano (1949). The transmission of information. Technical Report 65, Research Laboratory of Electronics, MIT, Cambridge, MA.

[Fiala and Greene, 1989] E.R. Fiala and D.H. Greene (April 1989). Data compression with finite windows. *Com. ACM*, 32(4):490–505. LZFG algorithm C2 has U.S. patent applied for.

[Floyd and Steinberg, 1975] R. Floyd and L. Steinberg (1975). An adaptive algorithm for spatial gray scale. *Society for Information Display.* Symposium Digest of Technical Papers.

[Foley et al., 1990] James Foley, Andries van Dam, Steven Feiner, and John Hughes (1990). *Computer Graphics: Principles and Practice.* Addison Wesley, second edition. The Systems Programming Series. ISBN: 0-201-12110-7.

[Freeman, 1961] H. Freeman (June 1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, EC-10:260–268.

[Freeman, 1974] H. Freeman (1974). Computer processing of line-drawing images. *Comp. Surveys*, 6:57–97.

[Gersho and Gray, 1991] Allen Gersho and Robert M. Gray (1991). *Vector Quantization and Signal Compression.* Series in Communications and Information Theory. Kluwer Academic Press.

[Gonzalez and Wintz, 1977] R.C. Gonzalez and P. Wintz (1977). *Digital Image Processing.* Addison-Wesley, first edition. Advanced Book Program. ISBN: 0-201-02597-3.

[Gonzalez and Woods, 1992] R.C. Gonzalez and R.E. Woods (1992). *Digital Image Processing.* Addison-Wesley, third edition. ISBN: 0-201-50803-6.

[Graham, 1967] D.N. Graham (March 1967). Image transmission by two–dimensional contour coding. *Proceedings of the IEEE*, 55(3).

[Halton, 1994] Keith. C. Halton (January 1994). The evolution and future of group 3 facsimile standards. *BT Technol Journal*, 12(1).

[Hamilton, 1992] Eric Hamilton (September 1 1992). *JPEG File Interchange Format, Version 1.02.* C-Cube Microsystems, 1778 McCarthy Blvd., Milpitas, CA 95035. +1 408 944 6300. E-mail: eric@c3.pla.ca.us.

[Hawkins, 1986] Stuart Hawkins (1986). Compressing pictures for an animation server. Rainbow Graphics Group Research note, Computer Laboratory, University of Cambridge, New Museums Site, Pembroke Street, CB2 3QG. United Kingdom. 14th August.

[Heitger et al., 1992] Friedrich Heitger, Lukas Rosenthaler, Rüdiger von der Heydt, Esther Peterhans, and Olaf Kübler (1992). Simulation of neural contour mechanisms: from simple to end-stopped cells. *Vision Res.*, 32(5):963–981.

[Henle, 1961] Mary Henle, editor (1961). *Documents of Gestalt Psychology.* University of California Press and Cambridge University Press.

[Holliday, 1980] T.M. Holliday (1980). An optimum algorithm for halftone generation for displays and hard copies. *Proc. Soc. Inf. Disp.*, 21(2):185–192.

[Horspool and Cormack, 1986] R.N. Horspool and G.V. Cormack (January 1986). Dynamic markov modelling – a prediction technique. In *Proc. International Conference on the System Sciences.*

[Horspool and Cormack, 1987] R.N. Horspool and G.V. Cormack (1987). Data compression using dynamic markov modelling. *The Computer Journal*, 30(6):541–550.

[HP, 1984] HP (1984). *HP7475A Graphics Plotter, Interface and Programming Manual.* Hewlett Packard, Coin Road, Bracknell, Berkshire. RG12 1HN. UK.

[HP, 1992] HP (October 1992). *HP LaserJet 4 and 4M Printers User's Manual.* Hewlett Packard, Coin Road, Bracknell, Berkshire. RG12 1HN. UK. Part No. C2001-90950.

[Huang, 1965] T.S. Huang (1965). PCM picture transmission. *IEEE Spectrum*, 2(12):57–63.

[Hubel, 1990] D.H. Hubel (1990). *Eye, Brain and Vision.* Scientific American Library. ISBN: 0-7167-5020-1.

[Hubel and Wiesel, 1959] D.H. Hubel and T.N. Wiesel (1959). Receptive fields of single neurones in the cat's striate cortex. *J. Physiol*, 148:574–591.

[Huffman, 1952] D.A. Huffman (September 1952). A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(10):1098–1101.

[Hung, 1979] Stephen H.Y. Hung (January 1979). A generalization of DPCM for digital image compression. *IEEE Trans. Anal. and Mac. Int.*, PAMI-1(1):100–109.

[Hunter and Steiglitz, 1979] G.M. Hunter and K. Steiglitz (1979). Operations on images using quadtrees. *IEEE Trans. Pat. Anal. and Mac. Int.*, PAMI-1(2):145–153.

[ISO/IEC,] ISO/IEC. Information technology – text and office systems – document style semantics and specification language DSSSL. DIS 10179.

[ISO/IEC, 1986] ISO/IEC (1986). Information technology – text and office systems – standard generalized markup language SGML. IS 8879.

[ISO/IEC, 1992] ISO/IEC (1992). Information technology – computer graphics – metafile for the storage and transfer of picture description information. IS 8632.

[ISO/IEC, 1993a] ISO/IEC (1993a). Information technology – computer graphics – image processing and interchange – functional specification – part 3: Image interchange facility. DIS 12087.

[ISO/IEC, 1993b] ISO/IEC (1993b). Information technology – text and office systems – standard page description language SPDL. IS 10180.

[ISO/IEC, 1993c] ISO/IEC (1993c). Open document architecture (ODA) and interchange format international standard. ISO 8613, Originally titled Office Document Architecture (ODA) and Interchange Format International Standard.

[Jacquin, 1993] Arnaud E. Jacquin (October 1993). Fractal image coding: A review. *Proc. of the IEEE*, 81(10):1451–1465.

[Jähne, 1991] B. Jähne (1991). *Digital Image Processing – Concepts algorithms and Scientific Applications.* Springer–Verlag. ISBN: 3-540-53782-1.

[Jarvis et al., 1976] J.F. Jarvis, J.N. Judice, and W. Ninke (March 1976). A survey of techniques for the image display of continuous tone pictures on bilevel displays. *CGIP*, 5(1).

[Jarvis and Roberts, 1976] J.F. Jarvis and C.S. Roberts (August 1976). A new technique for displaying continuous tone images on a bilevel display. *IEEE Trans.*, COMM-24(8):891–898.

[Jayant et al., 1993] Nikil Jayant, James Johnson, and Robert Safranek (October 1993). Signal compression based on models of human perception. *Proc of the IEEE*, 81(10):1385–1421.

[Judice et al., 1975] J.N. Judice, J.F. Jarvis, and W.H. Ninke (1975). Using ordered dither to display continuous tone pictures on an AC plasma panel. *Proc. Soc. Inf. Disp.*

[Kandel et al., 1991] Eric K. Kandel, James H. Schwartz, and Thomas M. Jessell, editors (1991). *Principles of Neural Science.* Elsevier, third edition. ISBN: 0-444-01562-0.

[Kent, 1993] Jim Kent (March 1993). The flic file format - a fast and simple file format for graphics and animation. *Dr Dobb's Journal.*

[Khoros, 1991] Khoros (1991). *Khoros – an overview.* The Khoros Group, University of New Mexico.

[Kim and Modestino, 1993] Yong Han Kim and James W. Modestino (June 1993). Adaptive entropy-coded subband coding of image sequences. *IEEE Trans Com.*, 41(6):975–987.

[Kirk, 1992] David Kirk, editor (1992). *Graphics Gems III.* Accademic Press. ISBN: 0-12-409670-0.

[Knuth, 1969] D.E. Knuth (1969). *The Art of Computer Programming Volume 2: Seminumerical Algorithms.* Addison-Wesley. ISBN: 0-201-03822-6.

[Knuth, 1986] D.E. Knuth (1986). *TEX: The Program.* Addison-Wesley. ISBN: 0-201-13437-3.

[Knuth, 1987] D.E. Knuth (October 1987). Digital halftones by dot diffusion. *ACM Trans. on Graphics*, 6(4):245–273.

[Kodak, 1992] Kodak (1992). *Kodak Photo CD Products – A Planning Guide for Developers.* Eastman Kodak Company.

[Kolb, 1991] Craig E. Kolb (15 January 1991). *Rayshade User's Guide and Reference Manual,* draft 0.1 edition.

[Lang et al., 1992] S.R. Lang, A.R.L. Travis, O.M. Castle, and J.R. Moore (5–6 September 1992). A 2nd generation autostereoscopic 3-D display. In P.F.Lister, editor, *Seventh Workshop on Graphics Hardware*, pages 53–63. Eurographics Technical Report Series EG92 HW, ISSN 1017-4656.

[Legall, 1991] Didier Legall (April 1991). MPEG – a video compression standard for multimedia applications. *Comms. ACM*, 34(4):47–58.

[Lempel and Ziv, 1977] A. Lempel and J. Ziv (1977). A universal algorithm for sequential data compression. *IEEE Trans. Info. Theory*, IT-23(3):337–343.

[Lempel and Ziv, 1978] A. Lempel and J. Ziv (1978). Compression of individual sequences via variable-rate coding. *IEEE Trans. Info. Theory*, IT-24(5):530–536. LZ78 has been patented in the U.S. 4,464,650.

[Lindeberg, 1990] Tony Lindeberg (March 1990). Scale-space for discrete signals. *IEEE Trans. Pat. Anal. and Mac. Int.*, 12(3):234–254.

[Liu and Srinath, 1990] Hong-Chih Liu and Mandyam D. Srinath (November 1990). Partial shape classification using contour matching in distance transformation. *IEEE Trans. Pat. Anal. and Mac. Int.*, 12(11):1072–1079.

[Mac, 1988] Mac (1988). *Inside Macintosh.* Volume 5. Addison-Wesley.

[Mach, 1865] E. Mach (1865). On the visual sensations produced by intermittent excitations of the retina. *Philosophical Magazine*, 30(2):319–320. Fourth Series.

[Marr, 1982] D. Marr (1982). *Vision, a computational investigation into the human representation and processing of visual information.* W.H. Freeman and Company, San Fransisco. ISBN: 0-7167-1284-9.

[Marr and Hildreth, 1980] D. Marr and E. Hildreth (1980). Theory of edge detection. *Proc. R. Soc. London*, pages 187–217. B207.

[Martineau, 1991] Didier Martineau (Mai à Juillet 1991). Camparison de deux méthodes de compression de seéquence d'images. Technical report, Digital Equipment Corporation Au Paris Research Laboratory à Rueil-Malmaison.

[McFarlane, 1972] M.D. McFarlane (1972). Digital pictures of fifty years ago. *Proc IEEE*, 60(7):768–770.

[Meer et al., 1990] P. Meer, C.A. Sher, and A. Rosenfeld (April 1990). The chain pyramid: Hierarchical contour processing. *IEEE Trans. Pat. Anal. and Mac. Int.*, pages 363–376.

[Miller and Wegman, 1984] V.S. Miller and M.N. Wegman (1984). Variations on a theme by ziv and lempel. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume F12, pages 131–140. Springer-Vellag. NATA ASI Series.

[Mitsubishi, 1991] Mitsubishi (1991). *S3600-30D/B Sublimation Full Color Printer.* Mitsubishi Electronic Corporation.

[Morrison, 1985] J. Morrison (1985). EA IFF 85 – standard for interchange format files. *Electronic Arts.*

[Namane and Sid-Ahmed, 1990] A. Namane and M.A. Sid-Ahmed (June 1990). Character scaling by contour method. *IEEE Trans. Pat. Anal. and Mac. Int.*, 12(6):600–606.

[NASA, 1990] NASA (December 1990). *Flexible Image Transport System (FITS).* NASA Science Data Systems Standards Office, Code 933, NASA Goddard Space Flight Center, Greenbelt MD 20771. Internet: NSDSSO@nssdca.gsfc.nasa.gov, SPAN: nssdca::NSDSSO, Phone +1 301 286-3575, draft standard edition. NSDSSO 100-0.1.

[NASA Workshop, 1993] NASA Workshop (2 April 1993). NASA – Space and Earth Science Data Compression Workshop. Discssion topic during morning session.

[NCSA, 1989] NCSA (March 1989). *HDF Calling Interface and Utilities.* National Center for Supercomputer Applications, University of Illinois. NCSA HDF Version 3.1.

[NEC, 1991] NEC (1991). *MultiSync 5FG/6FG User's Manual.* NEC Corporation.

[Nelson, 1991] Mark Nelson (1991). *The Data Compression Book: Featuring Fast, Efficient Data Compression Techniques in C.* M&T Books. ISBN: 1-55851-216-0.

[Ngan et al., 1989] K.N. Ngan, K.S. Leong, and H. Singh (December 1989). Adaptive cosine transform coding of images in perceptual domain. *IEEE Trans. Acoustic, Speech and Signal Processing*, 37(11):1743–1750.

[Nill and Bouzas, 1992] N.B. Nill and B.H. Bouzas (April 1992). Objective image quality measures derived from digital image power spectrum. *Optical Engineering*, 31(4):813–825.

[Oliver and Wiseman, 1983a] Martin A. Oliver and Neil E. Wiseman (1983a). Operations on quadtree encoded images. *The Computer Journal*, 26(1):83–91.

[Oliver and Wiseman, 1983b] Martin A. Oliver and Neil E. Wiseman (1983b). Operations on quadtree leaves and related image areas. *The Computer Journal*, 26(4):375–380.

[Pavlidis, 1982] T. Pavlidis (1982). *Algorithms for Graphics and Image Processing.* Springer-Verlag.

[Pennebaker and Mitchell, 1993] William B. Pennebaker and Joan L. Mitchell (1993). *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold.

[Peterson et al., 1986] J.W. Peterson, R.G. Bogart, and S.W. Thomas (1986). *The Utah Raster Toolkit.* University of Utah, Department of Computer Science, Salt Lake City, UT.

[Pratt, 1991] William K. Pratt (1991). *Digital Image Processing. Second Edition.* Wiley-Interscience. ISBN: 0-471-85766-1.

[Pritchard, 1977] D.H. Pritchard (November 1977). U.S. color television fundamentals – a review. *IEEE Trans. Consumer Electronics*, CE-23(4):467–478.

[Purcell and Wiseman, 1992] Patrick Purcell and Neil Wiseman (1992). Broadcast graphics: A venue of new applications. In *State of The Art Reports.* Eurographics Technical Report Series EG92 SA ISSN 1017-4656.

[Rabbani, 1992] M. Rabbani (1992). *Selected Papers on Image Coding and Compression.* Milestone Series Vol. MS 48. SPIE Optical Engineering Press. ISBN: 0-8194-0889-1.

[Rabbani and Jones, 1991] Majid Rabbani and Paul W. Jones (1991). *Digital Image Compression Techniques.* SPIE Optical Engineering Press. ISBN: 0-8194-0648-1.

[Ratliff, 1965] F. Ratliff (1965). *Mach Bands: Quantitative studies on neural networks in the retina.* Holden-Day Inc.

[Rosenfeld and Kak, 1982] A. Rosenfeld and A.C. Kak (1982). *Digital Picture Processing.* Academic Press, second edition. ISBN: 0-12-597301-2.

[Rost, 1989] R.J. Rost (1989). *OFF a 3D Object File Format.* Digital Equipment Corporation (WSE), Palo Alto, CA.

[Russ, 1992] J.C. Russ (1992). *The Image Processing Handbook.* CRC Press. ISBN: 0-8493-4233-3.

[Saghri et al., 1989] J.A. Saghri, P.S. Cheatham, and A. Habibi (July 1989). Image quality measure based on a human visual system model. *Optical Enginerring*, 28(7):813–818.

[Samet, 1980] Hanan Samet (1980). Region representation: Quadtrees from binary arrays. *Computer Graphics and Image Processing*, 13(1):88–93.

[Samet, 1982] Hanan Samet (1982). Neighbour finding techniques for images represented by quadtrees. *Computer Graphics and Image Processing*, 18(1):37–57.

[Samet, 1985] Hanan Samet (September 1985). Data structures for quadtree approximation and compression. *Comms. of the ACM*, 28(9).

[Samet and Tamminen, 1985] Hanan Samet and M. Tamminen (1985). Computing geometric properties of images represented by linear quadrees. *IEEE Trans. Pat. Anal. and Mac. Int.*, PAMI-7(2):229–239.

[Samet and Webber, 1988] Hanan Samet and R.E. Webber (1988). Hierarchical data structures and algorithms for computer graphics, part i: Fundamentals. *IEEE Computer Graphics and Applications*, 8(3):48–68.

[Savitar, 1991] Savitar (1991). *Scanmatch: Color Calibration Software.* Savitar Inc., 139 Townsend St., Suite 203, San Francisco, CA 94107.

[Schönhut, 1993] Jürgen Schönhut (September 1993). Electronic document printing and publishing – which format for what? In *State of the Art Reports*, pages 1–20. Eurographics '93 Technical Report Series.

[Scitex, 1988] Scitex (1988). *HandShake – Foreign File Transfer Protocol.* Scitex Corporation Ltd.

[Sedgewick, 1988] Robert Sedgewick (1988). *Algorithms.* Addison-Wesley, second edition. ISBN: 0-201-06673-4.

[Sewell, 1993] Jon M. Sewell (23 August 1993). Discussion during rainbow graphics group meeting. Number 356. Chaired by Dr Neil E. Wiseman at the University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, CB2 3QG. United Kingdom.

[Shannon, 1949] C.E. Shannon (January 1949). Communication in the presence of noise. *Proc. of the IRE*, 37(1):10–21.

[Slater, 1991] Jim Slater (1991). *Modern Television Systems to HDTV and Beyond.* UCL Press Limited. ISBN: 0-273-03122-8.

[Sloan, 1992] Alan D. Sloan (March 1992). Fractal image compression: A resolution independent representation for imagery. In James C. Tilton, editor, *Space and Earth Science Data Compression Workshop*, pages 73–79. NASA Conference Publication 3183.

[Smith and Rowe, 1993] Brian C. Smith and Lawrence A. Rowe (September 1993). Algorithms for manipulating compressed images. *IEEE Computer Graphics and Applications*, pages 34–42.

[Solomon et al., 1994] Joshua A. Solomon, Andrew B. Watson, and Albert Ahumada Jr. (March 1994). Visability of DCT basis functions: Effects of contrast masking. In James A. Storer and Martin Cohn, editors, *Data Compression Conference*, pages 361–370. IEEE Computer Society Press.

[Stewart, 1986] I.P. Stewart (1986). Quadtrees: Storage and scan conversion. *The Computer Journal*, 29(1):60–75.

[Stockholm Jr., 1972] T.G.S Stockholm Jr. (July 1972). Image processing in the context of a visual model. *Proc. IEEE*, 60(7):828–842.

[Storer, 1988] J.A. Storer (1988). *Data Compression: Methods and Theory.* Computer Science Press. ISBN: 0-88175-161-8.

[Storer and Szymanski, 1982] J.A. Storer and T.G. Szymanski (1982). Data compression via textual substitution. *Journal of the ACM*, 29(4):928–951.

[Sun, 1988] Sun (1988). *SunOS Pixrect Reference Manual.* Sun Microsystems Inc.

[Synergy, 1991] Synergy (1991). *PhotoScript: The colour PostScript printing solution.* Synergy (UK) and MetaForce Software Design.

[Thomas et al., 1985] S.W. Thomas, J. McKie, S. Davies, K. Turkowski, J.A. Woods, and J.W. Orost (1985). Compress (version 4.0) program and documentation. Available from joe@petsd.UUCP.

[Tischer, 1987] P. Tischer (1987). A modified lempel–ziv-welch data compression scheme. *Australian Computer Science Communications*, 9(1):262–272.

[Townsend, 1970] Boris Townsend (1970). *PAL Colour Television.* Cambridge University Press.

[Travis and Lang, 1990] A.R.L. Travis and S.R. Lang (1990). A CRT based autostereoscopic display. In *EuroDisplay 90, Proc. SID.*

[Truevision, 1991] Truevision (January 1991). *Truevision TGA File Format Specification Version 2.0.* Truevision, Inc., 7340 Shadeland Station, Indianapolis, IN 46256-3925, +1 317 841 0332.

[Turner, 1990] Martin J. Turner (1990). ICEBERG - gray scale image compression scheme. Part II, Computer Science Tripos. Dissertation, Cambridge University, Computer Laboratory, New Museums Site, Pembroke Street, CB2 3QG. United Kingdom.

[Turner, 1992] Martin J. Turner (March 1992). Entropy reduction via simplified image contourization. In James C. Tilton, editor, *Space and Earth Science Data Compression Workshop*, pages 27–42. NASA Conference Publication 3183.

[Turner, 1994] Martin J. Turner (March 1994). Shell coding of 3D image sets. In James A. Storer and Martin Cohn, editors, *Data Compression Conference*, page 518. IEEE Computer Society Press. Extended Abstract. ISBN: 0-8186-5637-9.

[Ulichney, 1987] R. Ulichney (1987). *Digital Halftoning.* MIT Press.

[U.S. Senate, 1950] U.S. Senate (1950). The present status of color television – report of the advisory committee on color television to the committee on interstate and foreign commerce. 2nd Session Document No. 197, Washington D.C.

[van Essen, 1993] David C. van Essen (16th August 1993). CVI lecture series: Biology of vision, principles of neural architecture and function. Isaac Newton Institute for Mathematical Sciences, University of Cambridge, United Kingdom.

[Voloboj, 1993] Aleksej G. Voloboj (1993). The method of dynamic palette construction in realistic visualization systems. *Computer Graphics Forum*, 12(5):289–296.

[Wallace, 1991] G.K. Wallace (April 1991). JPEG still picture compression standard. *Comms ACM*, 34(4):30–44.

[Watson, 1993] Andrew B. Watson (March 1993). Visually optimal DCT quantization matrices for individual images. In James A. Storer and Martin Cohn, editors, *Data Compression Conference*, pages 178–187. IEEE Computer Society Press.

[Watson et al., 1994] Andrew B. Watson, Joshua A. Solomon, and Albert Ahumada Jr. (March 1994). Visability of DCT basis functions: Effects of display resolution. In James A. Storer and Martin Cohn, editors, *Data Compression Conference*, pages 371–379. IEEE Computer Society Press.

[Welch, 1984] T.A. Welch (June 1984). A technique for high performance data compression. *IEEE Computing Society, Computing*, 17(6).

[Wiedling and Daun, 1991] H.P. Wiedling and S. Daun (1991). EuroRip – a european development of a raster image processor for page description languages with common font resources. *Eurographics '91 Graphics Research and Development in European Community Programmes*, pages 115–127. Eurographics Association.

[Wilkins and Wintz, 1970] L.C. Wilkins and P.A. Wintz (1970). Studies on data compression, part i: Picture coding by contours, part ii: Error analysis of run-length codes. Technical report, School of Electrical Engineering, Purdue University, Lafayette, Indiana. TR-EE 70-17.

[Witten et al., 1987] I.H. Witten, R. Neil, and J.G. Cleary (June 1987). Arithmetic coding for data compression. *Comms. of the Assoc. for Comp. Mac.*, 30(6):520–540.

[Woods, 1991] J.W. Woods, editor (1991). *Subband Image Coding.* Kluwer.

[Wüthrich and Stucki, 1991] Charles A. Wüthrich and Peter Stucki (July 1991). An algorithmic comparison between square- and hexagonal-based grids. *CVGIP: Graphical Models and Image Processing*, 53(4):324–339.

[Wüthrich et al., 1989] Charles A. Wüthrich, Peter Stucki, and Abdelhakim Ghezel (October 1989). A frequency domain analysis of square- and hexagonal-grid based images. In J. Andreé and R.D. Hersch, editors, *Raster Imaging and Digital Typography, Proceedings of the International Conference, Ecole Polytechnique Fédérale, Lausanne*, pages 261–270. Cambridge University Press. ISBN: 0-521-37490-1.

[Wyszecki and Stiles, 1982] G. Wyszecki and W. Stiles (1982). *Color Science: Concepts and Methods, Quantitative Data and Formulae.* Wiley, second edition.

[Yarbus, 1967] A.L. Yarbus (1967). *Eye Movements and Vision.* Plenum Press.

[Zeitterberg et al., 1982] Lars H. Zeitterberg, Staffan Ericsson, and Harold Brusewitz (August 1982). Interframe DPCM with adaptive quantization and entropy coding. *IEEE Trans on Com.*, COM-30(8):1888–1899.

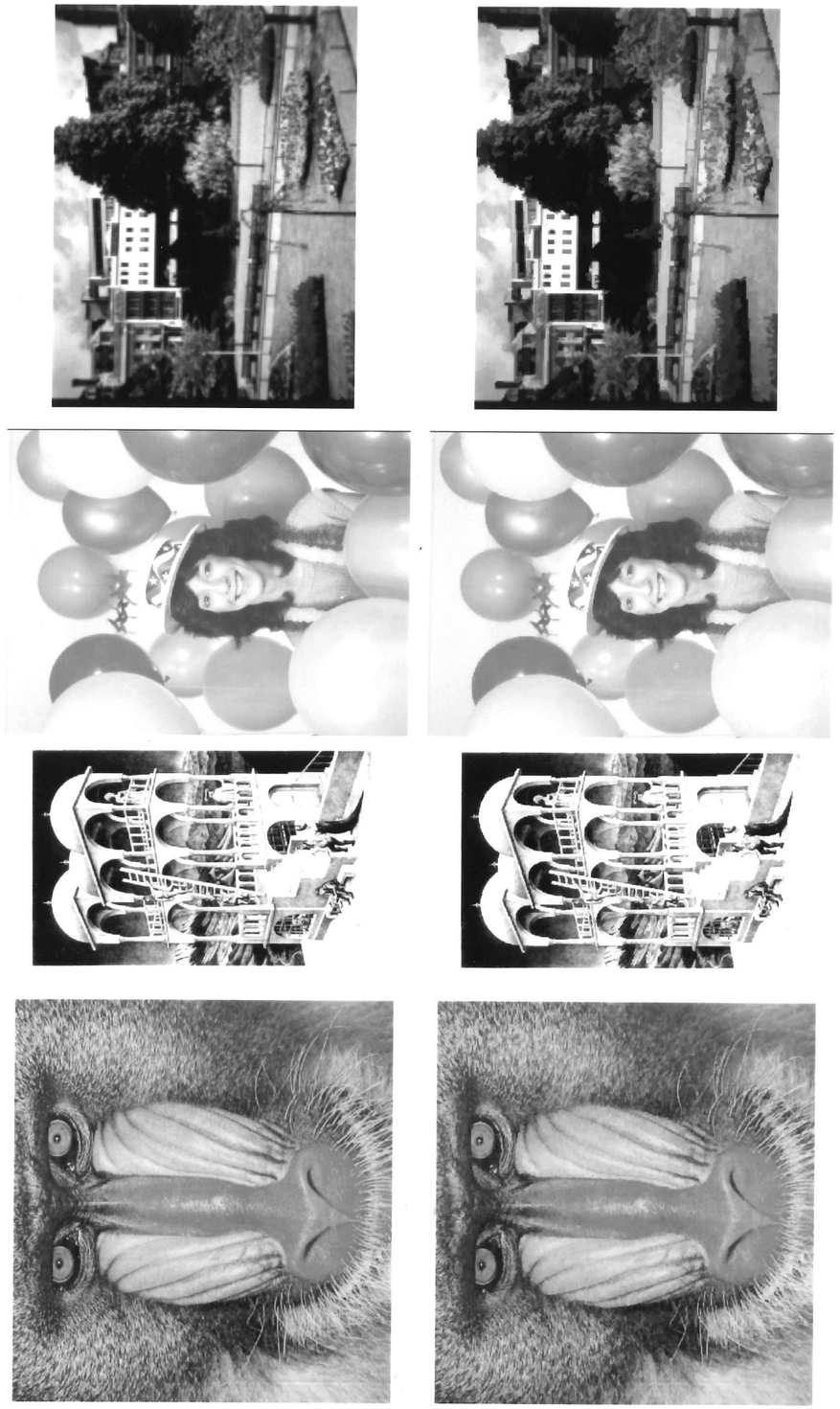
[ZSoft, 1988] ZSoft (1988). *ATTN: Technical Reference Manual.* ZSoft Corporation, 450 Franklin Rd. Suite 100, Marietta, GA 30067. +1 (404) 428-0008. Aspects of .PCX file format and the use of FRIEZE.

Colour Plates

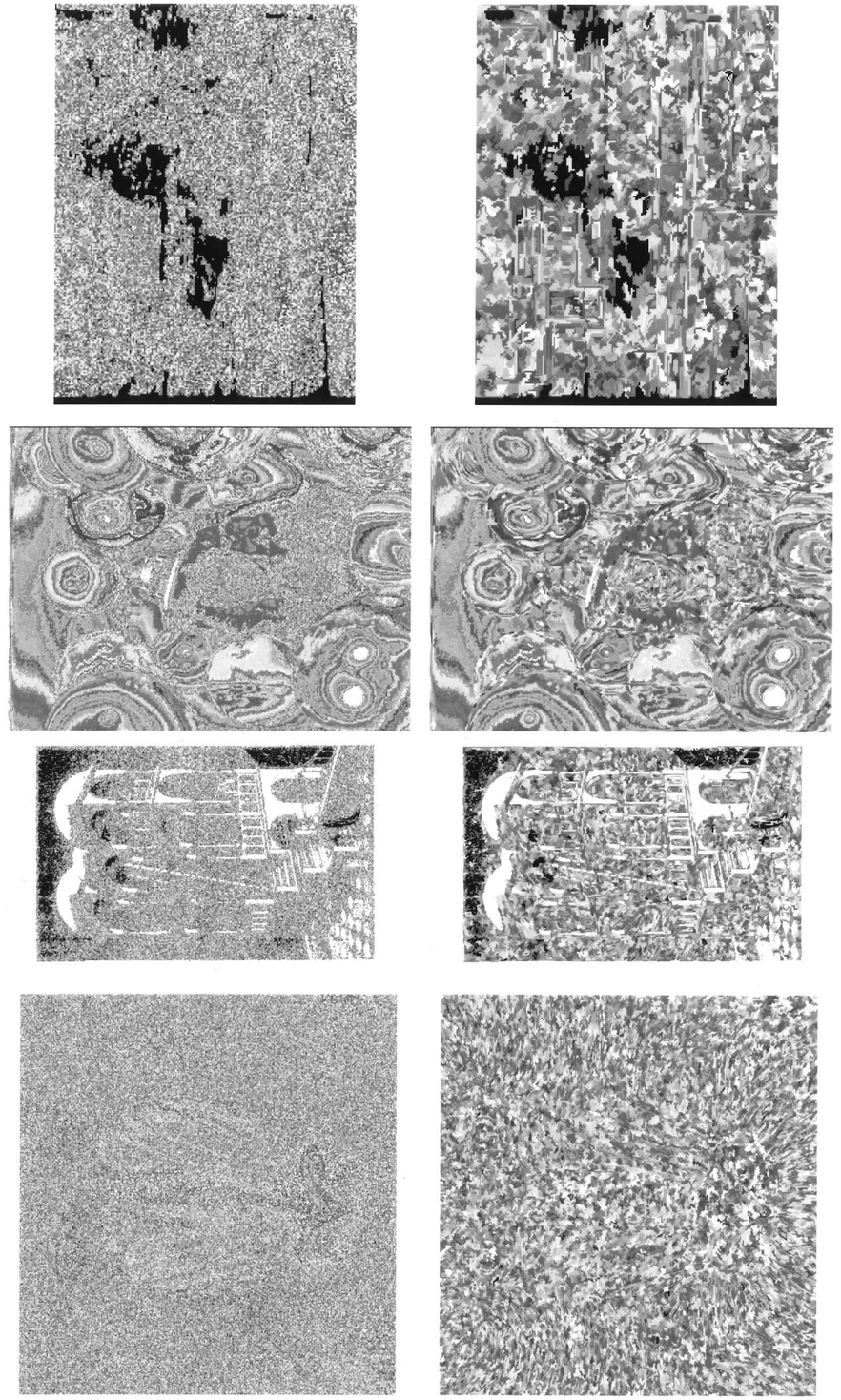
Throughout this dissertation the text and images have been printed using a black and white 600 dpi (dots per inch) HP LaserJet 4 printer. To achieve gray scale the printer uses a halftoning technique to mimic levels of gray, and for large full page images the Resolution Enhancement Technology was switched off [HP, 1992, Section 2 pages 20–23]. All the images were created for printing, as 8-bit PostScript Image data, from a variety of different software sources. They were combined with the LaTeX text using the *psfig* library and converted to PostScript with the *dvips* software converter, available from TeXware.

Some of the Figures are also replicated using a Mitsubishi S3600-30 Sublimation Full Color Printer. This uses a thermal-sublimation method, sequentially printing three colours; yellow, magenta and cyan. The resulting resolution of the image is 300 dpi using any of 16.7 million colours [Mitsubishi, 1991, Appendix A.1]. Translation of the images to colour PostScript was carried out using Aldus FreeHand [Aldus, 1991, pages 270–271], on a Quadra 950, through the PhotoScript [Synergy, 1991, pages 35–36] driver card. The four Figures 5.5, 5.6, 6.3 and 7.7 are printed with this method on the next seven pages.

The main text and images consist of about 45,000 words and 172 Encapsulated PostScript files which occupy as PostScript 58.8Mb, and the colour plates take up another 43.9Mb of PostScript.



Colour Plate 1: Figure 5.5



Colour Plate 2: Figure 5.6

1.834:1 Original YCrCb

8.136:1 High Quality all planes

Colour Plate 3: Figure 6.3 A

8.405:1 High Quality Y Medium Quality CrCb

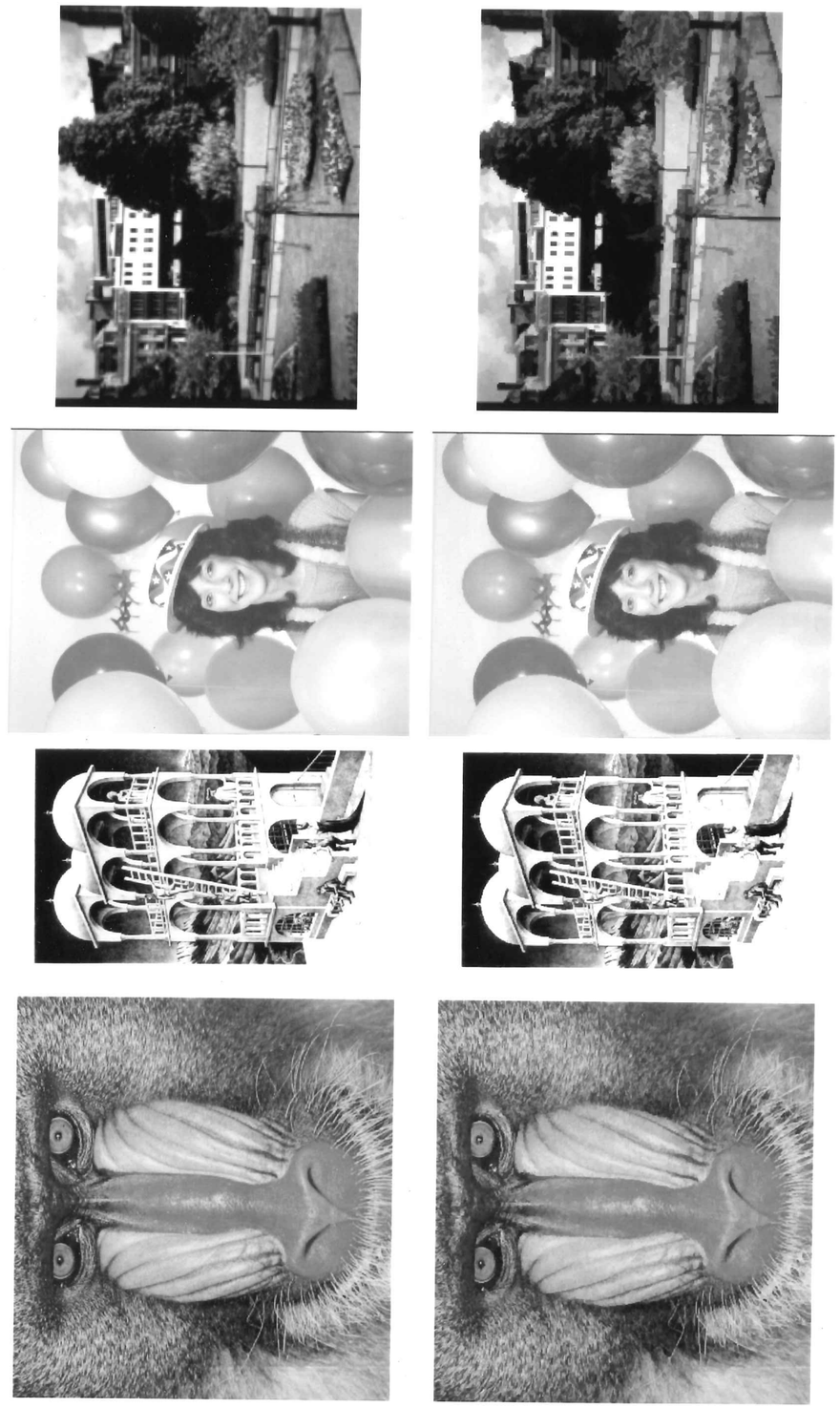
9.462:1 Medium Quality Y Low Quality CrCb

Colour Plate 4: Figure 6.3 B

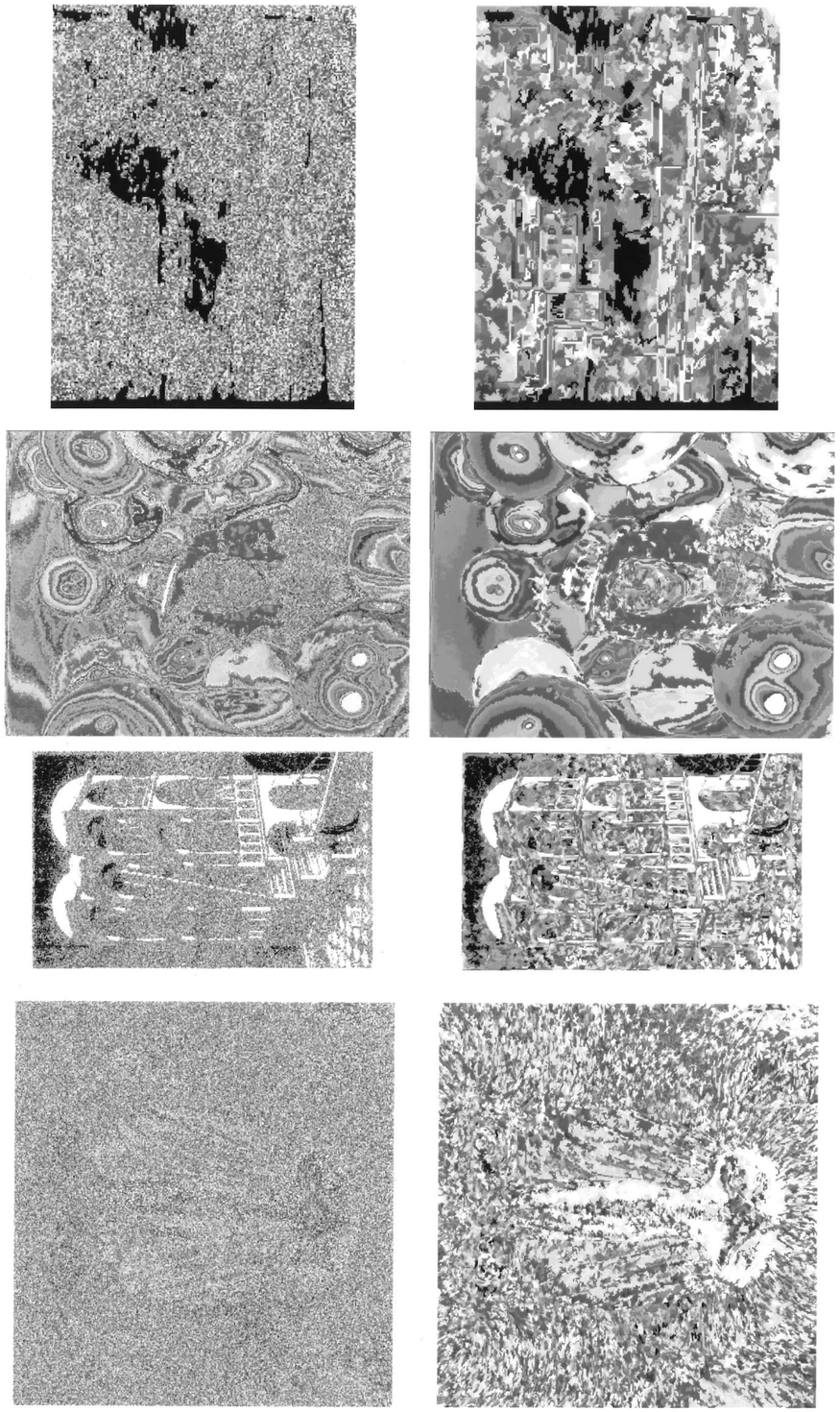
10.564:1 Medium Quality Y Very Low Quality CrCb

17.486:1 Very Low Quality YCrCb

Colour Plate 5: Figure 6.3 C



Colour Plate 6: Figure 7.7



Colour Plate 7: Figure 7.7 B