# The Application of Murray Polygons to the Compression of Digital Image Data

Ian M. Buntin

University of St. Andrews
St. Andrews

December 1988

I Ian M. Buntin hereby certify that this thesis has been composed by myself, that it is a record of my own work, and that it has not been accepted in partial or complete fulfilment of any other degree or professional qualification.

Signed Date 21/12/88

Copyright

In submitting this thesis to the University of St. Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker.

I hereby declare that the conditions of the Ordinance and Regulations for the degree of Master of Science (M.Sc.) in the University of St. Andrews have been fulfilled by the candidate Ian M. Buntin.

Signed　　　　　　　　　　　　　　　　Date 9/6/89

# Acknowledgements

Firstly and most importantly I would like to thank Professor A. J. Cole, my supervisor, for his thoughtful advice and encouragement. His patient guidance and understanding were invaluable throughout my work.

I am also indebted to Brian McAndie, technician at St. Andrews, for advice given and work carried out on photographic illustrations.

My sincere thanks to Mr Richard Biggs of English Electric Valve Co.(EEV) for arranging the initial loan of the CCD camera and for extending it until the work was completed.

Lastly I thank my wife, Sheila, and children, Gordon and Linsay, for their uplifting good humour.

# Abstract

This thesis reports on the application of murray polygons, which are a generalisation of space filling curves and of Peano polygons in particular, to the compression of digital image data.

Initially, a brief review of the fundamental concepts and techniques of data compression as applied to digital images is outlined. Several common but contrasting data compression techniques are described in more detail. Space filling curves are introduced and the stages of the development from Peano polygons via multiple radix arithmetic to murray polygons is clearly described. The various associated methods along with outlines of software implementation of the basic and fast algorithms are also given.

The application of murray polygons to the scanning of digital images is explained and using run length coding the ensuing data compression is evaluated. Techniques for exact and approximate coding are discussed, and the extension to the coding of different areas of the same image in either an exact or approximate coding is described.

Details of the investigation and compression results achieved are given for bilevel images. This is then extended via bit plane encoding to cover monochrome images and results reported for monochrome images captured from the real world by a frame grabber.

The work on monochrome images is extended to a preliminary investigation of the use of bit plane frame differences for the transmission of motion sequences for 'head and shoulder' type images.

# Contents

# INTRODUCTION

## 1930 Sub-editors office, New York Times

"I want the picture from London within an hour and good enough quality that the Editor, in his smoke filled room, recognises General Foch." the sub-editor barked at the young reporter.

The reporter left the office hoping there would be none of the usual transmission errors and that the 45 min preparation of the punched tape in London, the 1 hour transmission time across the Atlantic and the suspense filled 2 minutes in New York reproducing the image from tape would go smoothly. Often errors would force retransmission. At least it was better than 1920 when they had to wait over a week to get pictures of 'hot' news from Europe, but what standard of photographers have they got over there always taking photos on foggy days.

~~~~~~~~~~~~

## 1978 NASA, Control Centre ( 10.00 a.m. )

"Listen we have invested billions of dollars in this system. I don't care how long it takes but by noon I want a full analysis of the exact satellite image information for the
~~~~~~~~~~~~

**week before the last launch " the controller growled at the straight faced technician.**

**The technician thought what a week's image data meant. LANDSAT alone with 30 frames per day has about 42,000,000,000 bits per week archived on magnetic tape.**

~~~~~~~~~~~~~~

## 1988 St Andrews University, Scotland

**" I think there are possibilities of my scanning technique helping in the search for a means of compressing image data sufficiently to allow transmission of still and moving pictures over narrow bandwidths. In particular the introduction of the 64 kbit/s digital network may offer a real chance of sending these at a quality more than acceptable to the user." the professor enthusiastically suggested to his research student.**

**The student smiled hoping to hide his confusion. He knew digital images by their nature require a large bandwidth and moving pictures require about 25 frames per second. It looked like a lot of reading ahead.**

~~~~~~~~~~~~~~~~~~~~~~

These extracts are fictitious but emphasise some of the aspects of the subject matter of this thesis. However, such requests for compression of image data have been around for over 50 years.

The criteria, standards and technologies change but many of the varied demands and expectations for image transmission remain and often look impossible. The more these demands are met, with what at the time seem relatively fast and accurate solutions, the greater the number of applications that are spawned in what appears a limitless process.

The work reported in this thesis is part of that process and will look at possible applications of a particular class of space filling curves to the compression of data for the storage and transmission of images. The motivation was roughly that suggested in the last of the three introductory situations.

The introductory conversations serve to highlight many of the conflicting demands that exist. For example speed of transmission versus quality of image, namely the compression of bandwidth while at the same time preserving adequate image quality. Image quality will be dealt with later but at present it leads to the clear separation into **irreversible** and **reversible** coding of images. In irreversible coding information is lost but the remaining result meets the requirements of the user, who is usually a human viewer. Reversible coding is error free such that reconstruction of the original is possible.

The first item is an example of an irreversible coding where speed of transmission was so exciting that fidelity criteria were almost non-existent. In 1920 the Bartlane cable system [1,2] allowed picture information to be transmitted across the Atlantic making the transfer of news pictures 80 times faster ( previous times of around one week were reduced to 2-3 hours ). This was a spur to create great interest in image processing which has rarely waned since. Initially the increase in speed was so dramatic that quality of image was of minor importance. However within 10 years the quality of the image had increased significantly, the original 5 brightness levels achieved by typefaces simulating half-tone patterns (fig 1) had become 15 and used code on a punched tape to control light beams exposing a photographic negative ( fig 2 ).

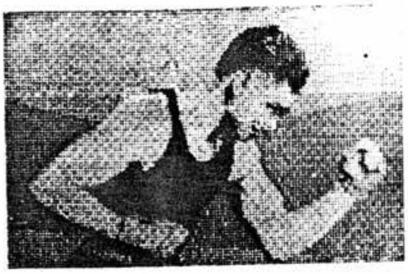


Fig 1 Digital image printed using typefaces to simulate half-toning. ( 1921 )[1]

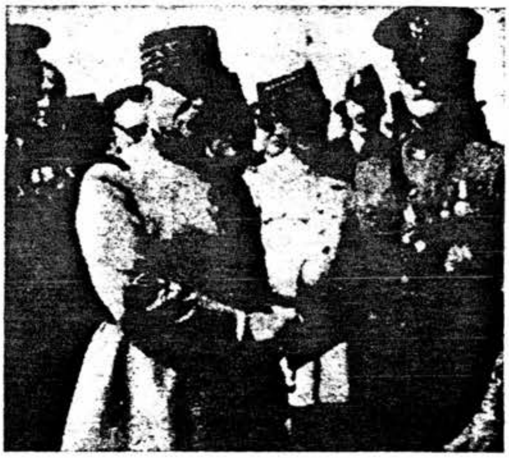


Fig 2 Bartlane cable image transmitted from London to New York ( 1930 )[1].

It was soon realised that data compression opportunities were to be found in areas of the picture with uniform tone. In these areas a tone code along with the number of times it was to be repeated seemed considerably more efficient than continuously sending the same tone code.This was done by comparing the codes with the previous one and summing until a change was found, similar to the run-length encoding still used today for grey level images. Unfortunately,attempts to use this method floundered. The additional time for the more complex system and the fact that small errors in the compressed data corrupted large areas of the image demanded retransmission. This meant that the gains due to compression were totally overwhelmed. Complexity of implementation and transmission errors are still of significant concern today.

In contrast to the cable transmitted images a reversible information preserving coding was required for LANDSAT, Land Satellite, data. LANDSAT captured 30 frames a day. Each frame covering an area of 100 x 100 nautical miles produced an image of 2340 x 3234 pixels, each with an integer value 0 - 127. Other than the initial quantisation this information had to be saved *exactly* for future analysis. This was a rate of 6,000,000,000 bits per day which were stored on magnetic tape, obviously a ripe case for data compression techniques.

The tremendous growth in image processing is closely tied to the widespread availability of modern computers at a relatively low cost and in a physical form that can be transported to anywhere on Earth, and on the back of a rocket to space or the surface of another planet. In addition the low costs of image acquisition systems such as low-cost frame grabbers means image processing technology is finding its way into small businesses and becoming available to home users.

'Every picture tells a story' and thus is such an effective means of communicating information that a list of applications would fill several pages. The main push seems to have been the demand from space exploration and satellite operation which flooded over into major applications in medical research, video conferencing, flight simulation, commercial graphics, computer art, entertainment, education, industrial design, weather forecasting, archiving of pictorial data, etc.. The advances in communication technology and the ensuing demands from applications such as those just mentioned for the storage and transmission of digital image data, fuels the search for more effective data compression.

# Chapter 1
# Data Compression
# A brief review of some concepts and techniques.

Before considering compression techniques it is worthwhile looking at some of the underlying fundamentals.

## 1.1 The Communication Process

Most communication systems convey information at a rate well below the capacity of the channels provided for them. The excess capacity is required to accommodate the redundancy, or repeated information, which signals contain in addition to the actual information.

It is of interest to consider first Shannon's [3] definition as outlined in Fig 1.1. The information to be communicated is first represented by a suitable code, transmitted via a channel subject to noise and finally received and decoded .

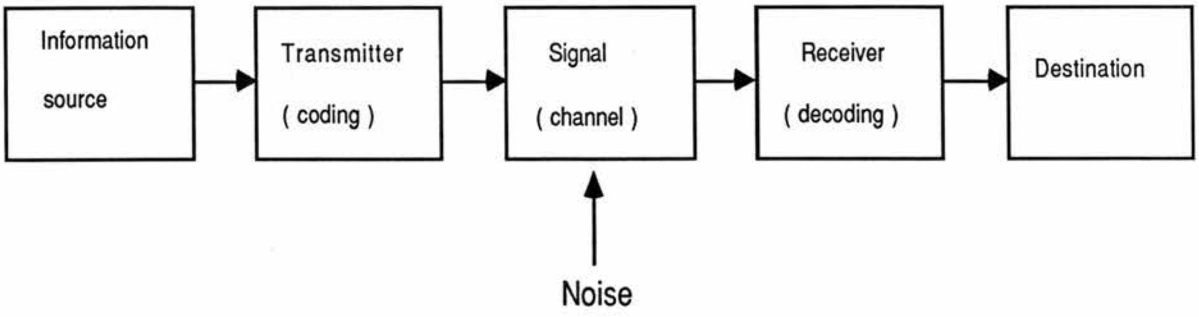


Fig. 1.1. Shannnon's model of a communication system

Shannon assumed the information source had a calculatable entropy(H) given by, $H = \Sigma\, p_i \log_2( 1/p_i )$ .

Where $p_i$ is the probability of an input symbol.

Entropy is therefore a measure of the randomness of a set of random variables and in the context of coding represents the amount of information associated with a set of coder input values and gives a lower bound on the average number of bits required.
This requires a knowledge of the underlying probability distribution.
In the case of visual communication systems dealing with digital images and vast amounts of possible data this raises the following questions.

What is the underlying probability distribution?
What are the basic elements or source symbols to be considered?

Various source elements have been used such as pixels, scanlines and defined objects. Using pixels which may take one of 256 values in an image of size 512 x 512 gives the number of possible images as

$$256^{512 \times 512} \approx 2^{2000000} .$$

An intuitive understanding of the probability distribution seems impossible as the amount of data is of a huge order and the probabilities thus exceedingly small. If the underlying distribution of images is uniform then no compression is possible. Jain [4] suggests that for monochrome images the entropy is believed to be very low as only a small proportion of the possible images are likely to occur.

The problem is therefore more one of visual communication engineering and in the experimental stage the model that appears most useful is that given in Fig 1.2, which owes something both to Fano [5] and Pearson [6]. The source encoder finds a representation of the visual world in binary digits. The channel coder takes the binary data from the source encoder and codes them in the most efficient manner for a given application. The introduction of a fidelity criterion which may consist of subjective and/or objective elements and of a feedback loop from the receiver, in most cases a human viewer, allows the adjustment of various parameters in the system to obtain an optimum position.

Optimisation demands bandwidth compression in two cases;

1. Error-free coding to minimise channel capacity while holding the fidelity constant

2. non-exact coding to maximise fidelity while holding the channel capacity constant.

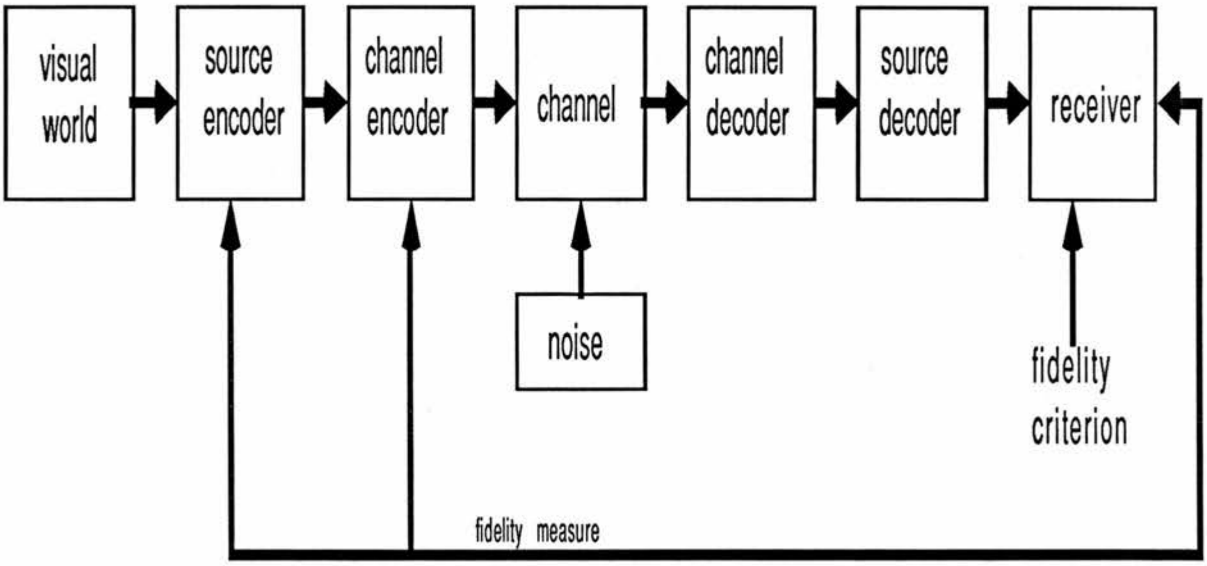


Fig. 1.2. A visual communication system model.

Schreiber [7] reflects the views of several workers in the field of data compression who feel the approach to communication of image data should be from the point of view given by the following question.

For a given channel, what relationship between the scene in the visual world and the transmitted signal produces the "best" pictures? What is meant by "best" is subject to the limitations imposed by a particular application.

The investigations that will be reported later are concerned mainly with the transmission of digital data on digital communication links and as such the main noise that the signal will suffer from will be major interruptions such as power faults. The source encoder will consist of a camera, low-cost frame grabber and associated circuitry.

## 1.2 An Image Processing System

The processing of an image involves the transformation of that image from one form into another. The block diagram in Fig. 1.3 outlines the main elements of the image processing system which takes information from the visual world and captures a representation which can be digitised then stored, processed or displayed on screen.

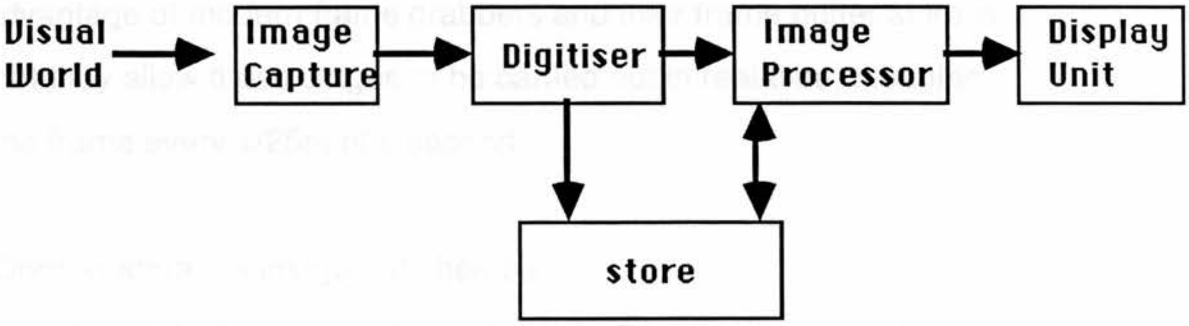


Fig. 1.3. Block diagram of image processing system

The scene is surveyed, usually by a camera which converts the optical image of the visual world into electrical video signals. Two common types of camera used are *vidicon* and *charge-coupled device* ( CCD ). In vidicon a light image is formed on the photoconductive target area of the tube which stores charge at what can be considered as discrete capacitor elements. The capacitors are later discharged by scanning the target with a low-velocity electron beam producing an output current. The CCD device consists of an array of photosensitive elements made up of electrode triplets on a silicon integrated circuit. Each element holds a charge proportional to the local light intensity received from the optical system. At the end of one integration time this pattern is transferred into a similar store section from which it is sequentially transferred to the on-chip charge detector amplifier which converts the charge signals into a voltage modulated video output.

The digitiser converts the analog signal into digital form suitable for storage and processing. The digitisation is done both spatially, where it is referred to as **image sampling**, and in amplitude where it is known as **quantisation**. The effects of different sampling and quantisation strategies will be considered in later chapters. The great advantage of modern frame grabbers and their frame buffer store is that they allow these stages to be carried out in real time, meaning one frame every 1/25th of a second.

Once in store the image can then be

a) processed as required using arithmetic or logical operators,

b) fed to a display unit where after conversion to analog it can be displayed on screen, and,

c) stored on some mass storage device such as hard disk.

The image processing system usually operates under the control of a digital computer which gives the opportunity through program control for greater flexibility.

### 1.3 An Image Model

The image has to be a carrier of information and for an optical image the signal is usually taken as proportional to the light energy. This is the model outlined below. However, Stockham [8] emphasised the importance of representation to transmission, storage and processing. His work on density representation is of interest. Its foundations are in photographic transparencies where the quantities of light transmitted are determined by the volume concentrations of the silver in the emulsion.

The more common light energy model as detailed by Gonzalez and Wintz [9] ( and others ) is a two-dimensional light intensity function, denoted by f(x,y), where the value of the function at spatial coordinates (x,y) gives the intensity of the image at that point. The restrictions on f are that it should be positive and finite given by

$$0 < f(x,y) < \quad m.$$

As images are formed by both incident light and reflected light this leads to two components of f, the illumination and reflectance denoted by i(x,y) and r(x,y) respectively. These components contribute as a product as given by

$$f(x,y) = i(x,y) . r(x,y).$$

However they carry two basic and separate kinds of information, i(x,y) holding information on the lighting of the scene and r(x,y) being an indication of the nature of the objects through their ability to reflect light.

For a monochrome image the intensity f at a point (x,y) is known as the ***grey level*** ( g ) of the image at that point. The grey level will lie between two finite values which are normally shifted to give the range

$$0 \leq g \leq L.$$

This range is known as the ***grey scale*** and the usual convention is $g = 0$ corresponds to black and $g = L$ corresponds to white with intermediate values having shades of grey. A useful way of visualising this model in three dimensions is as given in Fig. 1.4.

Binary or bilevel images have f restricted to the values 0 or 1, where these normally correspond to black and white.

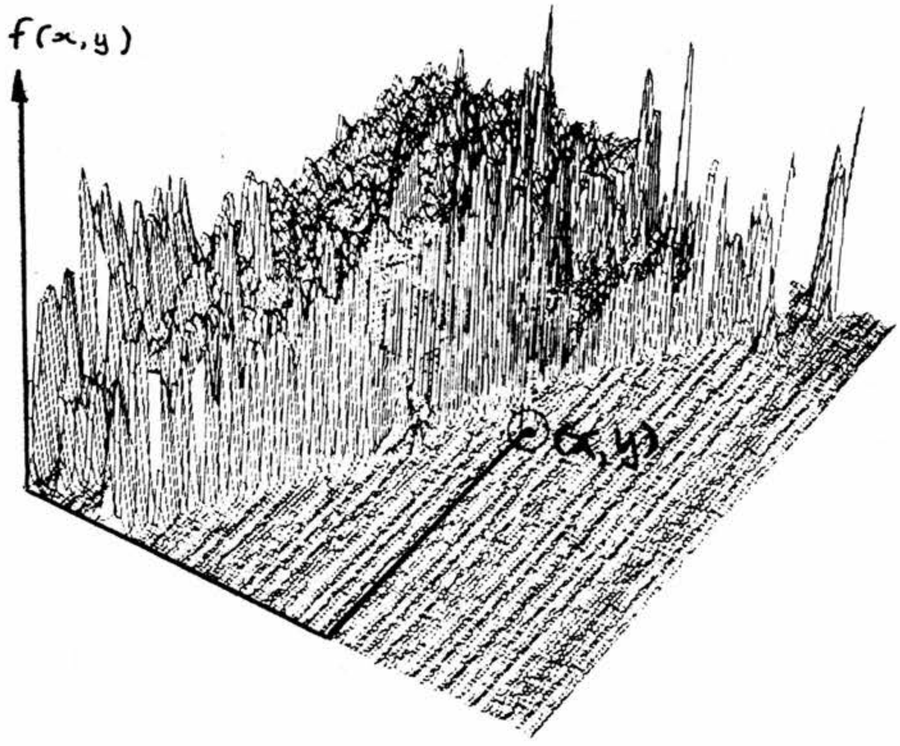


Fig. 1.4. A three dimensional representation of an image

## 1.4 General Data Compression Techniques

A data compression technique is one that reduces the bandwidth needed to transmit a given amount of information in a given time or the time needed to transmit a given amount of information in a given bandwidth. Techniques which fall into this broad definition are concerned with areas other than image communication but most can be applied to image transmission.
These have been categorised in many ways by writers in the literature and it does not seem within the scope of this report to look carefully at the many sometimes overlapping categorisations and their associated techniques. However, a limited number of examples should give sufficient flavour.

**1.4.1** An early categorisation of data handling due to Kortman [10] is as outlined in Fig. 1.5.

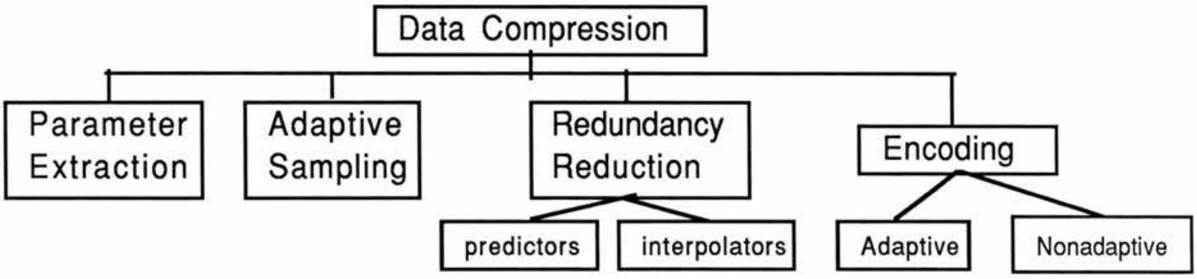


Fig. 1.5. Outline of data compression categorisation

a) **Parameter Extraction** is a technique that reduces the bandwidth required to transmit a given data sample by means of an information-describing irreversible transformation that extracts a particular characteristic or parameter of the signal. An example of this is the extraction of the power spectrum of a signal and transmission of the spectral components. Parameter extraction systems are very much moulded to meet the requirements of a particular application.

b) **Adaptive Sampling** adjusts the sampling rate of a given sensor to correspond to its information rate. A perfect match would mean there was no redundant information and compression would not be possible. However the complexity and accuracy of the activity detector and difficulty of implementation means that this is not possible. In practice adaptive or variable rate sensors have been successful. However there is no guarantee of compression and the overhead of tagging the data with a time or sensor tag may result in an expansion of data in certain circumstances.

c) **Redundancy Reduction** eliminates data samples that can be implied by examination of preceding or succeeding samples or by comparison with arbitrary reference patterns. Here the sampling rate is held constant and non-essential data eliminated later, unlike adaptive sampling where the sampling rate is varied to remove excess data.

There are many different reference patterns used to detect redundancy such as polynomials, exponentials and sine waves by which the raw data may be approximated.

There are two main sub-categories of redundancy reduction techniques, namely predictors and interpolators. In predictors redundancy is eliminated by estimating the value of each new sample based on the past performance of the data, while in interpolators after-the-fact polynomial curve fitting is used

.

d) **Encoding** transforms a given message into a corresponding sequence of code words. To be effective the sequential data should be well correlated. Thus an a priori knowledge of the data statistics is expected. Otherwise some adaptive method is used based on the most recent statistics of the encoder.

**1.4.2** Schreiber [7] gives a much simpler and widely used categorisation as follows:
a) Pure Statistical, b) Psychovisual and c) Hybrid.

a) Pure Statistical techniques rely on the statistical correlation between nearby elements, usually pixels. They are information preserving methods with the compression ratio measured by comparing the number of bits required for the original with those required for the coded image.

b) Psychovisual techniques alter the original image in order to compress the data, while at the same time maintaining an image of acceptable quality to the user. Any assessment of these methods must take into account many aspects of human vision and whether the images are for still or moving scenes.

c) Hybrid methods use combinations of a) and b).

## 1.5 Encoding of Digital Images

Attention is now focused on the encoding of **digital** images after capture, including initial sampling and quantisation, from the visual world. The image is held in an N x N array of pixels with m bits per pixel. It is the nature of digital images that they require a large number of bits for storage, a digital image 512 x 512 with 256 grey levels requires approximately 2.1 million bits. Hence the encoding procedure must compress this information as much as possible. It is helpful to split the process of encoding into three stages as shown in Fig. 1.6. The image in the pixel domain is mapped in a reversible step into a form that can be more easily compressed. A quantiser is then applied which may be uniform or nonuniform depending on the application. This is a nonreversible operation where the data from the mapping is approximated to the levels available in the quantiser causing an error with minimum value zero and maximum value of half the largest quantiser step. The resulting data is then coded to give a binary output suitable for transmission. For a given initial mapping the quantisation and coding need not be constant. The coding strategy can be varied to meet the nature of the output from the mapping. These steps are clearly seen in the more detailed discussion of several methods that follows later in this section.

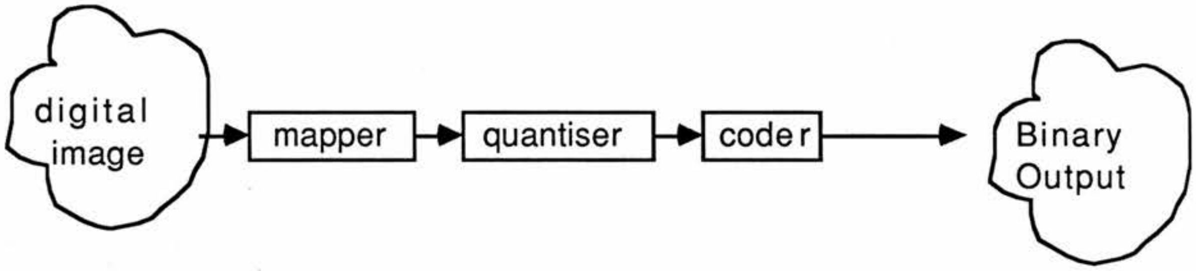


Fig. 1.6. Diagram of steps in encoding process

In a recent paper Kunt [11] loosely classifies the available compression techniques into 'first-generation' and 'second-generation'. The first-generation methods are those that emphasise the coder stage and make use of the spatial correlation between pixels. During the 1960s and early 1970s considerable progress was made using these, reaching maximum compression ratios of around 10:1. To go further than this, second-generation techniques have placed more stress on the mapper also making good use of recent findings on the nature, workings and limitations of human vision down to the neuron level. Substantial improvements in the compression ratio are claimed with several techniques giving figures around 50:1. The trade-off is 'quality' of the final image and especially at the higher compression ratios much information is discarded and the images might be judged poor or unacceptable for many applications. The reader is advised to view the details and images given in the reference above. An example, 'region growing', is given in section 1.5.3.5.

Before looking at individual methods some possible coding systems are considered.

### 1.5.1 Coding Choices

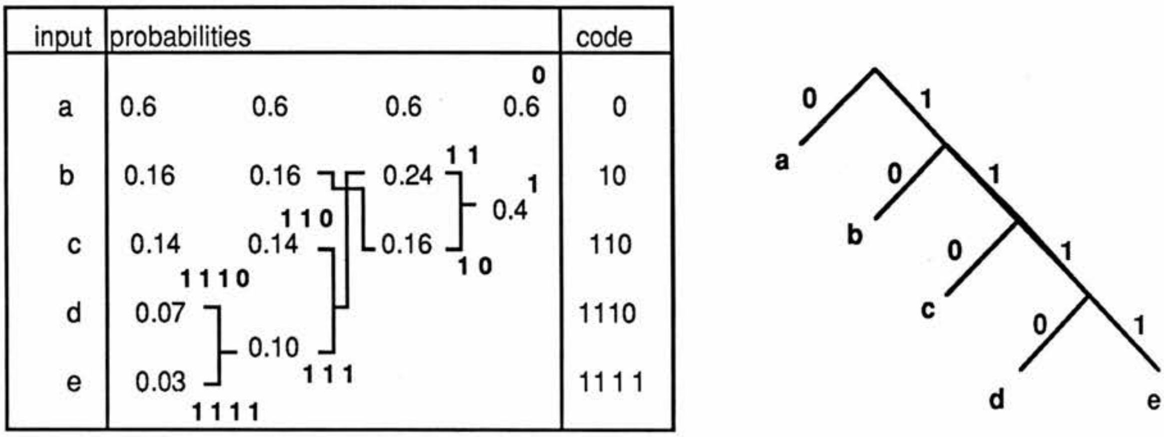
Coding using a binary alphabet gives code words made up of two symbols, normally 1 and 0. Possible codes can be classified into **fixed-length** and **variable-length**. An example of fixed-length coding is 3 bit natural binary code which gives 000,001,011,........., 110, 111. Another example of a fixed-length code is the particular Gray code known as reflected binary, which has the often advantageous feature that as we move through consecutive code words they change in only one digit. Examples of variable length codes are given in the following sections.

A **uniquely decodable** code is one in which the code words can be decoded in one way when received with no special space symbols. For instance alphabet $w_1$= 0, $w_2$=10, $w_3$=110, $w_4$=11 is not uniquely decodable as the sequence 110 could be $w_3$ or $w_4$, w1. However a fixed-length alphabet 00, 01, 10, 11 would be uniquely decodable. Another important classification is that of **instantaneous** and non-instantaneous according to whether or not it is possible to decode each word in sequence without reference to succeeding code symbols. An instantaneous code requires the condition that no code word be a prefix of another and has the advantage that decoding is simple.

When coding for compression of data we are interested in generating a code that would achieve the minimum number of bits. Using variable length codes with the most probable messages represented by the shortest codes should give greatest efficiency. Such codes are known as **compact** codes. If the probabilities are known then the entropy, as mentioned in an earlier section, gives a lower bound on the number of bits. However the probabilities are not always available and other factors such as simplicity and ease of implementation may give significant advantage to a suboptimum code.

### **Huffman Code** [12]

Huffman code is a compact code which is formed by the following process. The probabilities are first placed in descending order then the two smallest probabilities are added to produce a new set of probabilities which are again ordered. This is repeated until only two probabilities remain. A '0' or '1' is placed at each of the remaining two probabilities and the decomposition traced backwards repeating the '0' or '1' placement and carrying any previous allocation as a prefix. This construction and final codes are shown in Fig. 1.7.a. An alternative representation of codes as paths in a binary tree is given in Fig. 1.7.b.



| input | probabilities | | | | code |
|---|---|---|---|---|---|
| a | 0.6 | 0.6 | 0.6 | 0.6 | 0 |
| b | 0.16 | 0.16 | 0.24 | 0.4 | 10 |
| c | 0.14 | 0.14 | 0.16 | | 110 |
| d | 0.07 | 0.10 | | | 1110 |
| e | 0.03 | | | | 1111 |

(a) (b)

Fig. 1.7. Diagram showing the formation of Huffman codes.

### Continuation Bit Code ( B-codes )

This system has been found to be useful in run-length encoding where the number of short run lengths is large and as the run lengths increase the probability drops away quickly. The probability distribution is often close to a power law, the probabilities of the m coder inputs being given by

$$p_i = ( i )^{-k}$$

for i = 1,2,3, . . ,m, and k a positive constant.

The code word consists of two 'types' of bit, the continuation-bit ( c ) and the information-bit ( i ). When the continuation bit is not set, value 0, only the first group of digits is required for the message otherwise the continuation bit is set, value 1, and more groups of digits must be taken until a continuation bit set at 0 is found. The number of information bits is usually fixed for a given code. If the number of information bits is fixed at one it is a $B_1$-code, at two a $B_2$-code and so on. The common practice is for the continuation bit to precede the information bits. Examples giving the coding of possible run lengths are shown in Table 1.1 .

| run length | $B_1$-code | $B_2$-code |
|---|---|---|
| | CiCiCiCiC.... | CiiCiiCiiCii... |
| 0 | this is a special case applying only to the first run length being black or white | |
| 1 | 00 | 000 |
| 2 | 01 | 001 |
| 3 | 1000 | 010 |
| 4 | 1001 | 011 |
| 5 | 1100 | 100000 |
| 6 | 1101 | 100001 |
| 7 | 101000 | 100010 |
| 8 | 101001 | 100011 |
| 9 | 101100 | 101000 |

Table 1.1 Continuation bit coding.

### Shift Codes

This is a variable length code which is suitable for probability distributions which are monotonically decreasing. It also has the advantage that it is easy to implement.

The process is first to select a fixed length code to establish basic code words. Next one of these code words is selected as as a continuation code for out of range values. If an input is outwith the range of the three remaining information codes words, these words are shifted and prefixed by the continuation code. For example if the initial length is chosen as 2 the basic code words available are

00,01,10,11. Let the continuation code be 11 then the first eight inputs are coded as given in Table 1.2.

| input | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| code | 00 | 01 | 10 | 1100 | 1101 | 1110 | 111100 | 111101 |

Table 1.2 Input messages and possible shift codes.

### 1.5.2 Specific Data Compression Techniques

The range of available techniques and their extensions is vast. Thus a few fundamental methods with references are outlined below, followed by five selected examples in section 1.5.3.

**Pulse Code Modulation ( PCM )** [13]

The pulse code modulation ( PCM ) system is one of the earliest and most simple techniques often used as a reference for other methods. In this system the image signal is periodically sampled and the amplitude of each sample quantised and described by a fixed length binary code ready for transmission. Usually the number of levels of the quantiser is in the range from 8 to 256. The final image can suffer considerably from contouring if the number of levels is too low. The contouring can be broken up by addition of noise or dithering [14]. PCM does not take account of any properties of the human vision nor the spatial dependency between pixels with the final image quality depending on the sampling rate and number of quantisation levels.

### **Differential Pulse Code Modulation ( DPCM )** [15],[16]

Differential pulse code modulation is a predictive technique which uses previous pixel amplitude data to estimate the amplitude of the next. The difference between the prediction and the actual value is quantised and transmitted. For monochrome images with 256 grey levels the differences may range over the values -256 to 256. However, due to the correlation between adjacent pixels the majority of differences lie within a narrow band closely clustered around zero [9]. Therefore for many images the differences can be quantised using fewer levels than the original. The quantisation is often non-linear based on a tapered scale with small differences quantised more finely than large ones. Compression of data is thus obtained and can be improved by utilising adaptive techniques.

### **Delta Modulation** [17]

Delta modulation is a special case of DPCM where the quantiser has two levels ( 1 bit/pixel ). The main advantage of this system is its simplicity. Comparison of delta modulation and adaptive delta modulation with PCM is given by Abate[18]. Delta modulation suffers badly from *slope overload* and *granularity.* Slope overload occurs when there is a relatively large jump or discontinuity in the signal and the quantiser cannot respond quickly enough. Granularity is the false steps that are introduced in a signal by the quantiser fluctuating between levels in an approximately steady signal. These errors are shown in Fig 1.8.

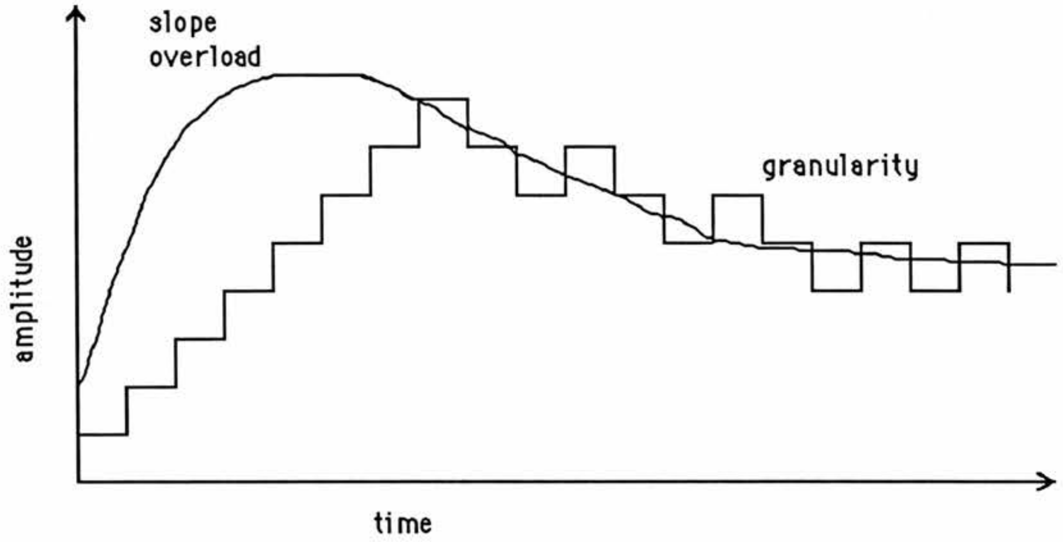


Fig. 1.8 A signal and its approximation by Delta Modulation.

## Transform Methods

There are many methods which can be included in the category of transform techniques where the original set of pixel data is replaced through a transformation by a set of coefficients. These coefficients should exhibit less correlation than the original data and can thus be considered more independent. The process first subdivides the image into separate blocks, often square. The transformation is then applied to this subimage and the resulting coefficients coded for transmission. The coding can be held constant for all subimages or altered to best suit the nature and contents of a particular subimage giving an adaptive form. It is not within the scope of this work to detail these techniques but the more common transform methods are Karhunen-Loeve (or Hotelling), Fourier, Hadamard, Haar, sine, cosine, etc.. The following references give a thorough coverage [19], [20], [21].

### Digital Television Inspired Techniques

A considerable amount of research has been stimulated by the compression of data for broadcast television. Methods that are good for still images or even in video conferencing are not necessarily as efficient in an environment where real-time operation is essential and a high fixed channel rate is available.

Techniques can be split into **intraframe** and **interframe.** Initially most work was on intraframe techniques which seek to take advantage of the spatial correlation in a frame as detailed by Connor et al [22]. With the availability of cheap frame store memories and with intraframe methods appearing close to their limit the emphasis switched to interframe applications. The work in this area was able to feed on early investigations at Bell Labs. on Picturephone [23] and on video conferencing research in general. Interframe coding tries to take advantage of the correlation between successive frames or corresponding fields in the case of interlacing. On its own interframe coding was going to be of little use in images where there was considerable movement of objects. Clearly the successive frames would change considerably. Interframe methods seemed best suited to frame sequences with little movement, not typical of T.V. transmissions. The solution was to use hybrid methods combining both intraframe and interframe coding in the appropriate circumstances. The requirement was now for a motion detector and criteria that would be able to categorise areas of an image and select the best coding technique. The paper by Mounts [24] introduces the idea of *conditional replenishment*. In conditional replenishment the differences between an incoming image and a reference image are compared and only those areas of an image which have changed significantly are transmitted. Papers of further interest in this area are Haskell et al [25] and Candy et al [26].

**note** Some of the methods to be reported in later chapters consider interframe applications to the transmission of images over a low bandwidth channel. The subject matter is restricted to suitable images with less violent movement such as human head and shoulders in a videophone style communication.

### 1.5.3 Selected Examples

Five examples are now given in more detail, they are

1. Run-length encoding
2. Quadtree encoding
3. Block Coding
4. Block Truncation Coding (BTC) and
   Colour Cell Compression (CCC) .
5. Contour-Texture Technique.

Run-length coding is given as it is the technique used in the work reported in later chapters. Quadtree and Block coding are different examples of area encoding. The combined example of BTC and its extension to CCC gives impressive colour images after considerable compression by an irreversible process. Lastly a contour-texture technique, using region growing, is a good example of a second-generation method

### 1.5.3.1. Run-length Coding

Initially the image is scanned to produce a sequence of integer pairs, giving the number of pixels with the same intensity and the value of the intensity. For black and white images a sequence of run lengths is sufficient if some convention such as first run length always corresponds to white pixels is used. If the first pixel is black then the first integer sent is zero. The ability to compress the image is dependent on the distribution of the run lengths produced and hence on the composition of the underlying image. Run-length coding has been used with considerable success in cases where a significant number of long runs are generated. However, as the number of long runs decreases the compression quickly falls away. The ease of encoding and decoding implementation is an advantage.

The bilevel techniques can be extended to grey level images by coding each bit plane. This will be considered in later chapters.

#### Scanning

A major influence on the compression is the ability of the scanning method to produce long run lengths. Therefore for a given image different scanning patterns can produce significant differences in the final data reduction. Two scanning methods to consider are a) the familiar **linear** scan and b) scanning by **space filling curve.** At this stage simple examples are given to show clearly the different sequences produced by these methods. A formal definition and more detailed discussion of space filling curves will be given in the following chapter.

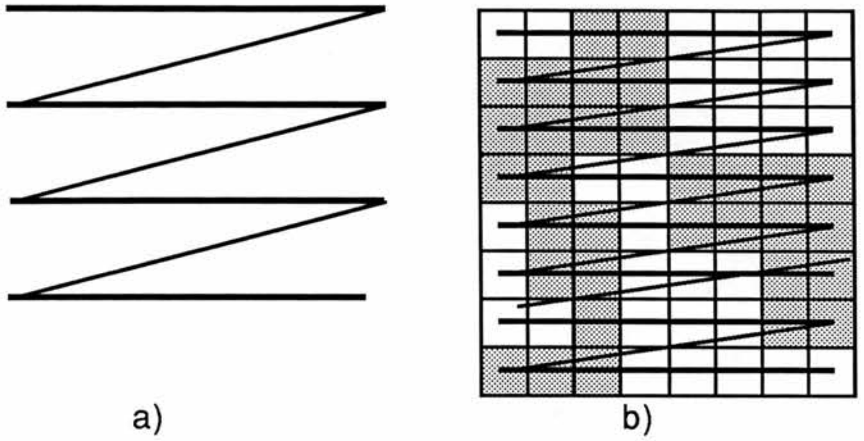
### a) Linear

Linear scanning scans from left to right and top to bottom with fly-back at the end of each scanline as shown in Fig.1.9. This is the basis of one-dimensional run-length coding and it takes advantage of the correlation between adjacent pixels on the scanline. There is a two-dimensional extension known as predictive differential quantisation (PDQ) which stores information on two scanlines of pixel data. This has been found to be more efficient than one-dimensional methods in the case of images with a few long run lengths.

### b) Space Filling Curves

Space filling curves can be applied to the scanning of images in two or three dimensions. For the present only the two-dimensional case is considered using one well known scan pattern due to Hilbert as shown in Fig. 1.10. One initial gain from the use of space filling curves is their ability to take advantage of the two-dimensional correlation between elements in a local area. Secondly, as we will see in chapter 2, there is great variety of scans and orientation which can be of assistance.

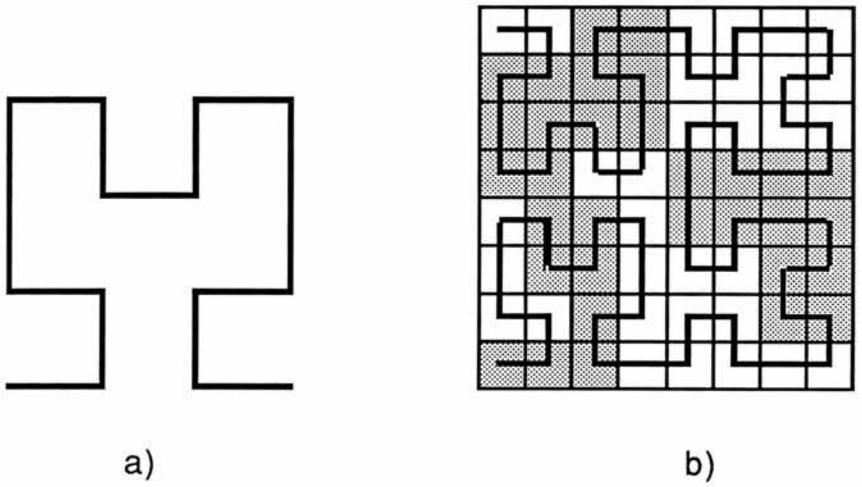
Run lengths are best coded with a variable length code as an equal length is clearly inefficient requiring $\log_2 M$ bits, where M is the maximum run length. In many images the run length distribution lends itself to continuation bit coding ( B-codes ).



2, 2, 4, 4, 4,
4, 4, 2, 2, 4,
1, 2, 1, 4, 1,
2, 3, 2, 2, 1,
3, 5, 5

c)

Fig. 1.9 a) Linear scan pattern b) scan of 8 x 8 image
c) run-length sequence



2, 7, 2, 5, 10,
3, 2, 2, 2, 7,
7, 2, 3, 4, 4, 2

c)

Fig 1.10
a) Hilbert curve order 2 b) scan of order 3
c) run-length sequence

### 1.5.3.2. Quadtree Encoding

This involves the formation of quadtrees by successively dividing a square digital image of side $2^m$ pixels into quadrants until all the quadrants are homogeneous, namely of the same colour or in the case of bilevel images all black or all white. These quadrants are represented by a tree structure, with the leaves of the tree corresponding to area of the same colour. The growth of the tree can be controlled by a resolution limit with the maximum value at the pixel level. A simple example of this is given in Fig. 1.11 with a linked tree representation each node below the root has four sons or is a leaf node.

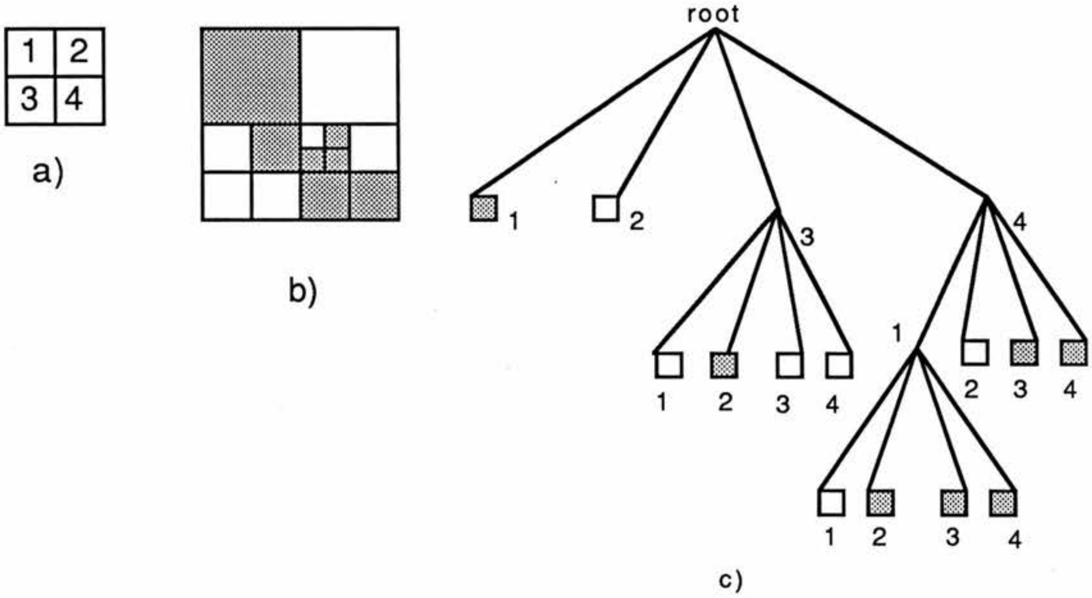


Fig. 1.11 a) quadrants numbered for clarity,
b) image & c) simplified quadtree representation of the image.

The quadtree structure in this form tries to exploit the coherence of many images and has been found to be a useful method for the storage of digital images. Klinger & Dyer [27] give examples of data compaction. This representation is also a suitable form for many image operations as reported by Hunter & Steglitz [28], Oliver & Wiseman [29] and others.

Many different data structures and methods have been developed for encoding, some with an emphasis on compaction [30] others more interested in manipulation, see references in the previous paragraph. A few of the more common cases are detailed below.

a) Regular Quadtrees ( RQT )

This system is based on a pointer representation between quadrants with links between father and four sons, and in some cases backwards and adjacent links to allow flexibility. While achieving compression there is a large overhead to store pointer information and in some examples expansion of data often occurred.

b) Explicit Quadtrees

Woodwark [31] avoided the need for links by allocating a location for every possible node. This ignored the coherence of an image and did not dramatically improve the storage requirements. However, it did give fast access for investigation of parts of the image and for modification.

c) Linear Quadtrees

Oliver & Wiseman [29] gave a more compact data structure where a traversal of the tree in depth first order gave a linear list of the nodes. The list also held the information associated with the nodes as in the RQT method, but without the storage overhead of pointers. However, this method was not general, as the original traversal of the tree dictated the order for future manipulation. Good results were obtained, but if a full traversal was required it could be inefficient. The technique was extended by compressing the linear node list as detailed by Woodwark [32]. The resulting Compressed Tree Codes(CTC) occupied 65% of the original treecode but were still no better than run-length encoding.

#### 1.5.3.3. Block Coding

This is a simple method that was originally applied to bilevel images and can easily be extended to grey level images. It can be applied as a reversible method with no data loss or with minor alteration, information can be discarded to produce better compression with a corresponding loss in quality. Studied in detail by Kunt and Johnsen [32] the method groups an image into blocks of pixels of size m x n. These blocks are then coded according to their probabilities of occurrence, using short code words for the most likely block configurations and long code words for the less likely block configurations. Thus compression is achieved. Initially Kunt developed block coding for two-dimensional images. It has since been improved using adaptive techniques and finally generalised for grey level images.

The number of possible arrangements in an m x n block is $2^{mn}$. These form the set of possible messages. The formation of code words from these blocks is known as block coding. For the highest compression the optimum Huffman code is used but for blocks greater than 3 x 3 the set of messages becomes very large and Huffman code is impractical. A suboptimum code is often used in these circumstances. The most common pattern, often a totally white block, is coded as 0 and all other patterns have the prefix 1 followed by 1's and 0's to indicate whether a pixel is on or off as shown in Fig. 1.12. The coding is information preserving at this stage. More compression can easily be obtained by coding blocks containing **k**, where **k** is a small positive integer, or less black pixels as if they were white blocks with a resulting degradation of quality. The amount of degradation is determined by the quality required for a particular application. In the adaptive technique the block size is varied depending on the statistics of the previous data.

Block coding is extended to grey level images by coding each of the bilevel bit planes. Further if the intensities are coded using an 8 bit canonical Gray code the bit-plane patterns are often more susceptible to block coding. The average compression ratio is found to be around 5:1.

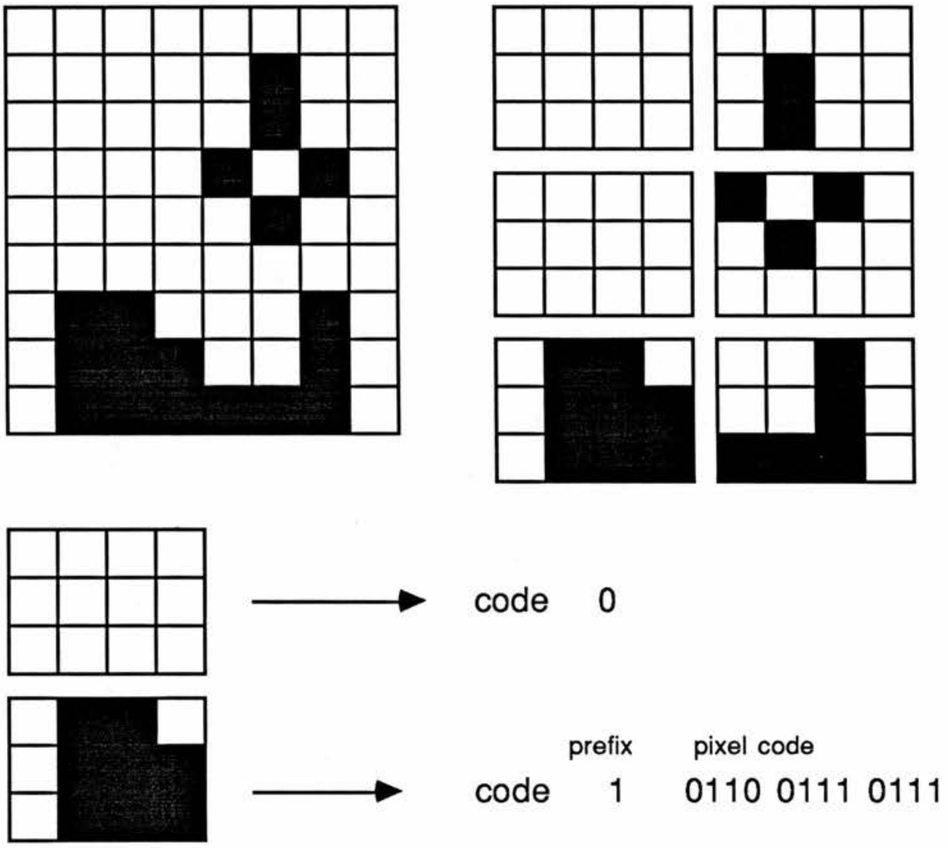


Fig.1.12 The technique of suboptimum block coding

## 4. Block Truncation Coding (BTC) & Colour Cell Compression( CCC )

### a) Block Truncation Coding

Block truncation coding ( BTC ) is quite different from the block coding given in the previous section. BTC is a non-information preserving method which attempts to retain important visual features of multilevel images. As it only requires one pass through the data it is easy to implement and found suitable for noisy data. It was developed by Delp & Mitchell [34] and extended to colour by Lena & Mitchell [35]. The image is divided into n x n blocks and the mean brightness level and variance are found for each block. Assuming a block of 4 x 4 , a 16-bit bit plane is formed using a threshold, often the sample mean. Each bit in the bit plane is set to a 1 if it is equal to or above the threshold and otherwise set to 0. The data transmitted is then 8-bit quantised values for mean and variance, and the16-bit bit plane giving 32 bits in total a rate of 2 bits/pixel. Brightness, contrast and most visible features within each block are retained. Mitchell & Delp [36] report that for certain bilevel images tested by Huang [37] run-length coding was better than BTC when time and memory are not restrictions, but BTC was considerably better for 32 grey-level examples.

The technique can be extended to colour by applying it to each of the R, G and B colour planes giving good quality images at 6 bits/pixel.

### b) Colour Cell Compression

Another method of interest based on BTC is Colour Cell Compression (CCC) developed by Campbell, DeFanti et al [38] where substantial compression is gained while maintaining what the human viewer would consider as very good quality images. The method takes advantage of the fact that the colour resolution of the human eye is usually happy with as little as 5 bits for each of the R, G and B components. The steps involved are as follows and as illustrated in Fig. 1.13.

step 1: A BTC method with cell size 4 x 4 is applied to each colour plane from the original image, giving a 16-bit bitmap dividing the cell into two regions. Two colours are then selected which best represents the original colours within each region, each colour being a 24-bit R, G, B value. At this stage this information is stored with the image, now represented by 4 bits/pixel.

step2: The R, G, B values are then quantised from the 8 bit form to 5 bits reducing the representation to 3 bits/pixel.

step3: Using the image representation from step 2 an ordered histogram of colours is formed and from this 256 suitable colours are selected which best represent the original. A "Median Cut" [39] method is used for selection. These colours are then stored in a colour look-up table with 8 bit values for R, G and B components.

step4: The stored form from step 1 is then adjusted so that each of the 24-bit colour vectors is replaced by an 8-bit pointer into the colour table position which best matches the original. This final form is a 2 bit/pixel representation which as shown in [38] produces colour images of better than expected quality.

There is considerable processing time to achieve the final result ,11 seconds for a 640 by 480 image on a VAX 750 is quoted.

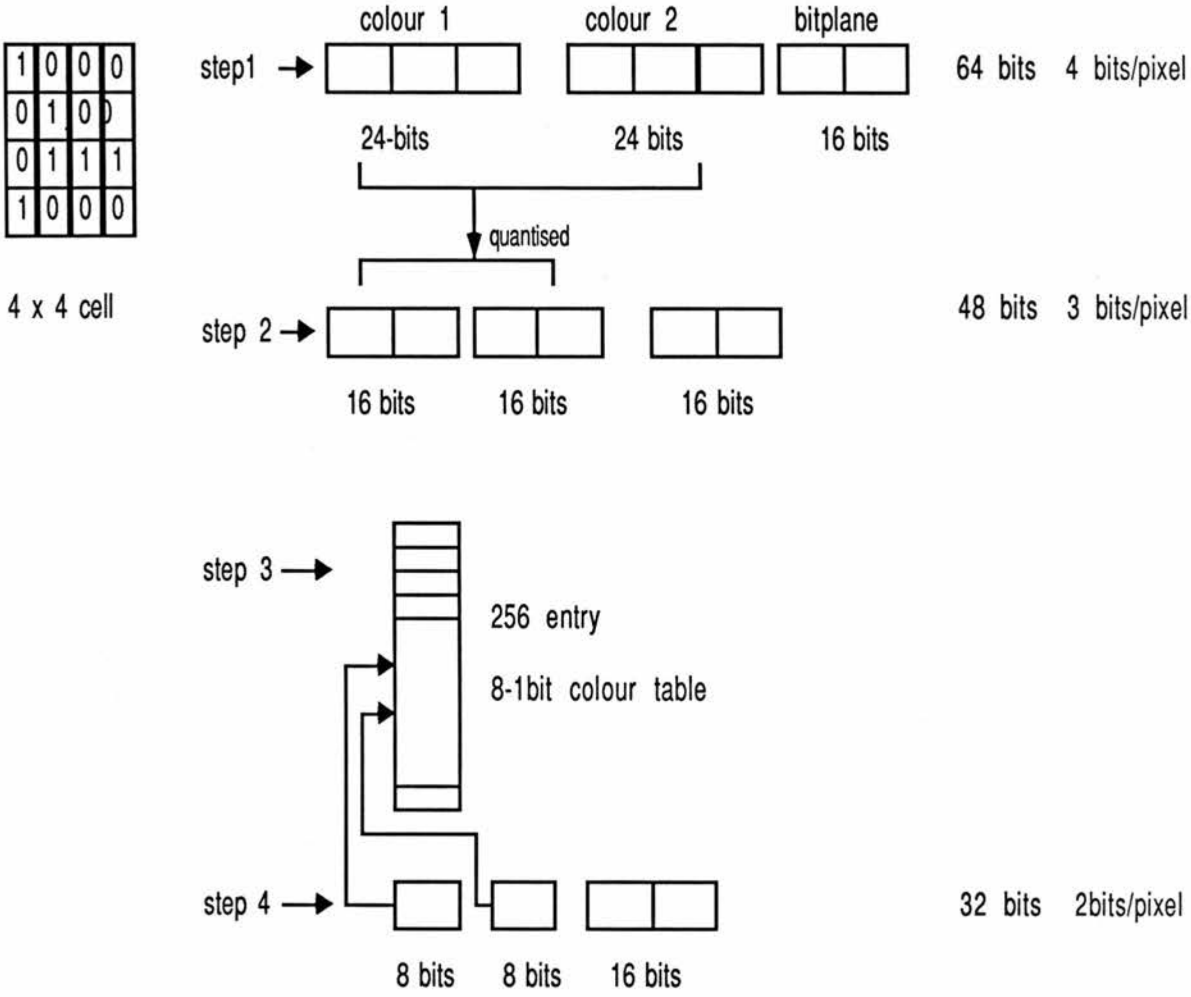


Fig. 1.13 Diagram showing main steps in CCC algorithm

### 1.5.3.5. A Contour-Texture Technique - Region Growing [11]

It is of interest to give brief details of a typical example of a second-generation method which achieves high compression ratios but still requires research work to improve the quality of the final image. For images with little detail or where the finer detail is not of major importance the final result is encouraging.
There are three major stages a) segmentation, b) contour coding and c) texture coding.

**Segmentation** breaks up the image into areas of similarly textured regions surrounded by contours where each pixel must be tagged as a 'contour' type or 'texture' type. These regions should be as close as possible to objects in the original image. In region growing the contours are closed regions that can be stored along with their associated properties. Segmentation can be further divided into preprocessing, region growing and elimination of artifacts.

Preprocessing should reduce the granularity of the original data and hopefully eliminate small contour areas that probably do not represent actual objects in the image. At the same time edges must be retained. These contradictory requirements are achieved by iterative application of a special filter which acts like a low-pass filter in areas without contours, otherwise as an all-pass filter.

Region growing takes the pixels from preprocessing and categorises them on property such as grey level. Regions are then grown by finding neighbouring pixels with a common property value. The selection of a suitable property is crucial to the success of the compression and often trial and error methods are used at this stage.

After region growing the resulting image is tidied up by removing open contours and contours two pixels wide. There is still a need to reduce the number of contours and a heuristic is applied which merges adjacent areas of low contrast and any small areas.

**Contour coding** must produce an efficient and precise description of the contours obtained from segmentation. After initial description of all border points the common border points which are described twice are identified and coded once. Following this the remaining contours are then described either by approximation to i) a line segment or

ii) a circle segment, or exactly by points.

**Texture coding** takes the regions from segmentation which should now have smooth variation of grey level and describes them initially by a two-dimensional polynomial function. The function may be altered in dimension after analysis of the resulting approximation errors and coding efficiency. As a last step the initial removal of granularity is rebalanced by the addition of some noise to give a more natural looking image.

Fig. 1.14 is an example that gave a compression ratio of 50:1 and shows more clearly some of the stages involved. The method is by no means perfect, suffering from some false regions that are not representative of any object in the original and considerable loss of detail. However, it does represent the major objects well and is promising in cases where the finer detail is of little importance.

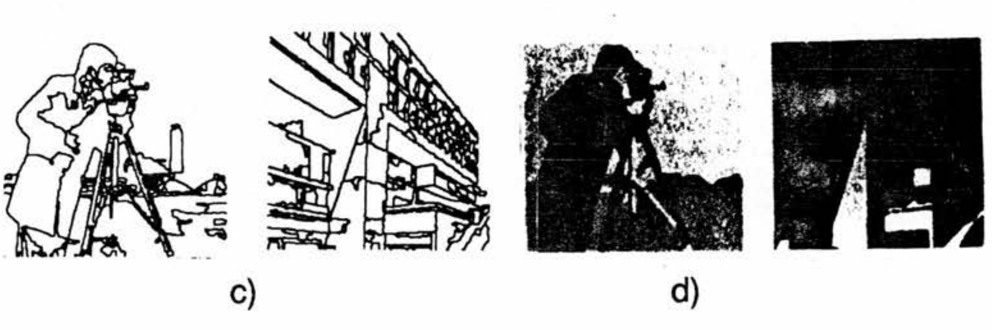


Fig. 1.14 Region growing [11]

a) original images, b) result after region growing,

c) result after segmentation and d) decoded pictures

## 1.6 Summary

A description of the basic concepts and several of the common techniques has been given. The motivation for data compression applied to the storage and transmission of digital images has been highlighted. It has been strongly suggested that the solution is of a visual engineering nature requiring the combination of a number of different tools. Assessment of a technique depends on a number of factors which vary from application to application. Some of the more important factors are data compression achieved, implementation efficiency and hardware requirements.

A few specific examples were outlined from a wide range of methods. If the reader was disappointed by the lack of depth, no apology is made. As many esteemed authors have pointed out the subject matter is now a vast and expanding area which, even if the reader follows all the references given, will still leave a 'life-time' of reading ahead. However, one topic that is of utmost importance and was reluctantly omitted is that of human vision. There seems little doubt that recent and on-going research into the workings of the human vision system will bring forth information that should boost work in data compression. Kunt [11] and many other writers referenced earlier give a summary of the mechanisms as they are understood at present.

# Chapter 2
# Peano Curves to Murray Polygons

In this chapter the development by Cole [40,41] from Peano's [42] original concept of a space filling curve to the definition of murray polygons and their associated methods via multiple radix arithmetic is described.

## 2.1 Space Filling Curves

The birth of what are now commonly called space filling curves was the discovery in 1890 by Guiseppe Peano [42] of a family of curves which scan n-dimensional space. At this time mathematicians were just accepting the definition of a 'curve' as the locus of points that satisfy equations of continuous functions. Intuitively a unique tangent could be drawn at any point on these curves. Peano produced an exception, which created turmoil and was regarded at the time as a non-intuitive 'monster'. He showed how to produce a curve by moving a single point continuously over a square, such that it passed at least once through every point on the square and its boundary. The curve produced was indeed continuous but it was impossible to draw unique tangents. That is, a continuous one-dimensional curve which at its limit filled a two-dimensional square but was differentiable nowhere.

Cole [40] used the underlying fundamental idea introduced by Peano of a continuous mapping of the unit line segment [ 0,1 ] on to the unit square. The original proof by Peano used a base three representation of the coordinates, from which he indicated its extension to any odd number base and to n-dimensions.

While based on the ideas introduced by Peano much recent work takes the alternative viewpoint given by Hilbert [43] that a space filling curve can be considered as a limit of polygons enclosed in the unit square. Hilbert generated a Peano curve with two free end points. Hilbert's curve is shown in Fig. 2.2 ( page 43) Moore[44] and Sierpinski [45] were others who were much involved in the early investigations of space filling curves. The mapping generated from Peano's transformation may also be regarded as a limit of polygons. Further details including more on Peano's original definition are given in [46].

Mandelbrot [47] classifies Peano curves as fractals and using his terminology Fig. 2.1 shows a) the original *generator*, which does not self-intersect but does self-contact, b) the unit square used as *initiator* and c) the curve formed after the first stage when each side of the square is replaced with the generator.

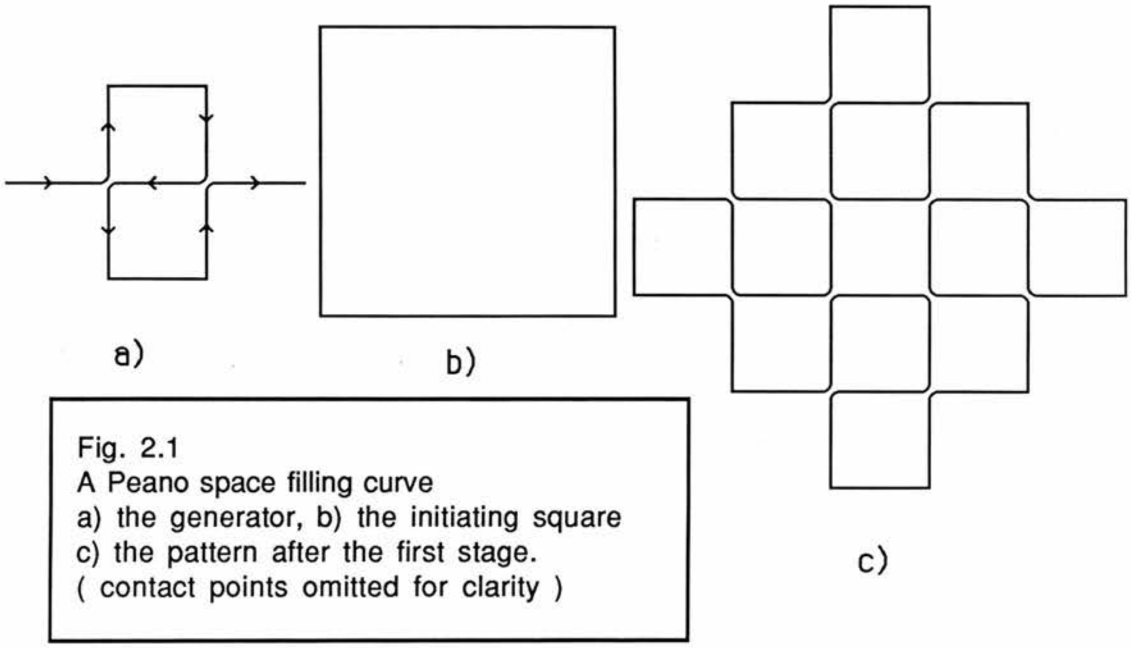
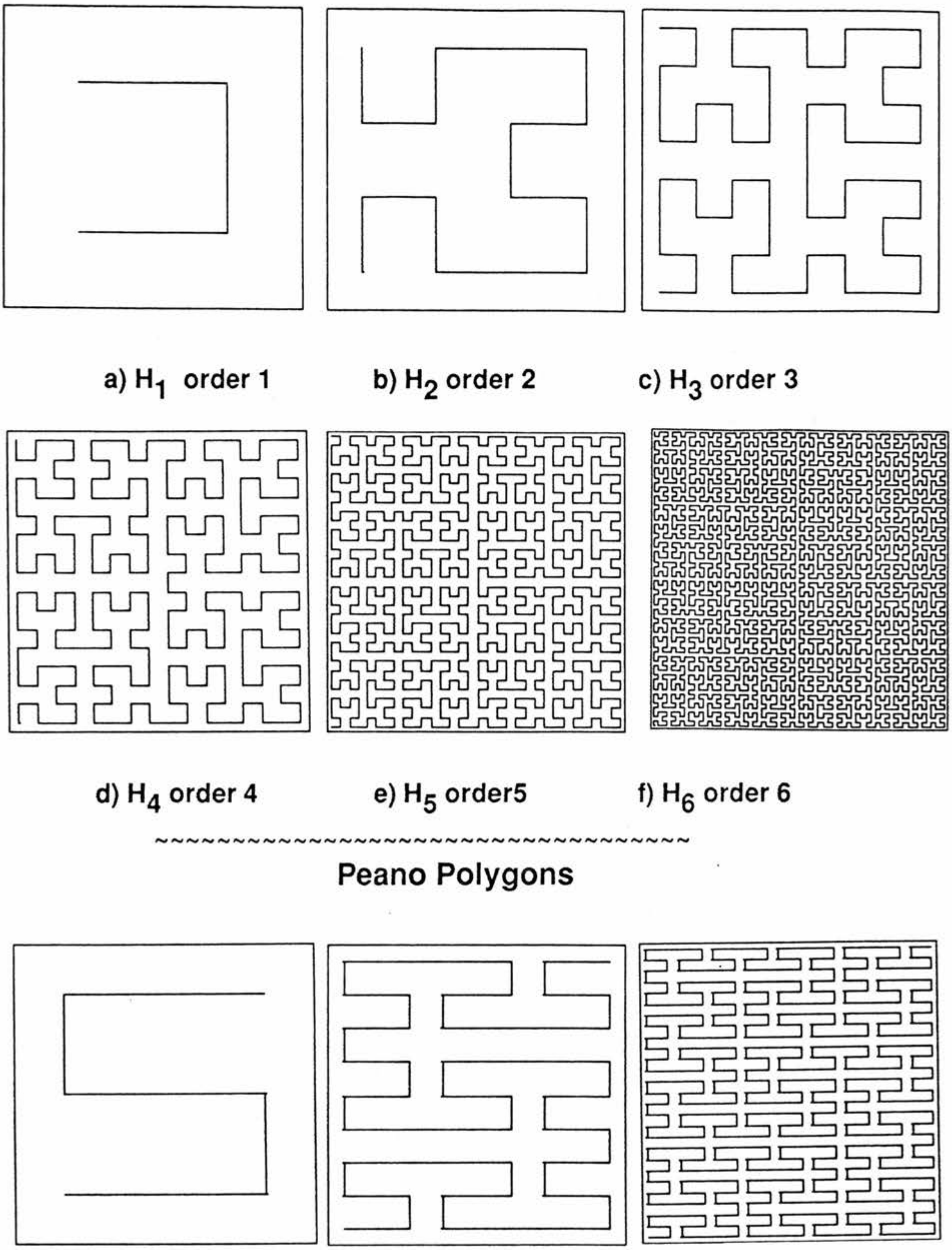


Fig. 2.1
A Peano space filling curve
a) the generator, b) the initiating square
c) the pattern after the first stage.
( contact points omitted for clarity )

While Fig. 2.1 shows the original examples it is much easier to visualise the idea of space filling curves and the limiting process by showing three of the most commonly quoted examples of polygons as given in Fig. 2.2 (the boundary squares are for reference only and are not part of the curves). These are all Peano polygons but are often referred to as **the** Peano, Hilbert and Sierpinski polygons. The first two are examples of open-polygons which do not self-contact while the later is a closed polygon.

Griffiths [48] investigated space filling curves and described a method for generating new ones. He considers the space filling curve in the unit square defined as the limit of a sequence $s_1, s_2, ....$ of continuous curves which pass through every point of the square. This can be viewed as a tessellation of square tiles all of which have the same pattern but with the orientation of the pattern varying. Firstly a tile has an n x n grid marked on it and the centres of each grid-square are taken as permissible points for the construction of a continuous open path that does not intersect. The resulting path must have endpoints such that $n^2$ tiles can be fitted together and the individual paths joined up with standard steps as shown in Fig. 2.3. Griffiths at this stage had shown how to generate new space filling curves which would traverse squares.

## Hilbert Polygons



a) $H_1$ order 1

b) $H_2$ order 2

c) $H_3$ order 3

d) $H_4$ order 4

e) $H_5$ order5

f) $H_6$ order 6

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Peano Polygons

a) $P_1$ order 1

b) $P_2$ order 2

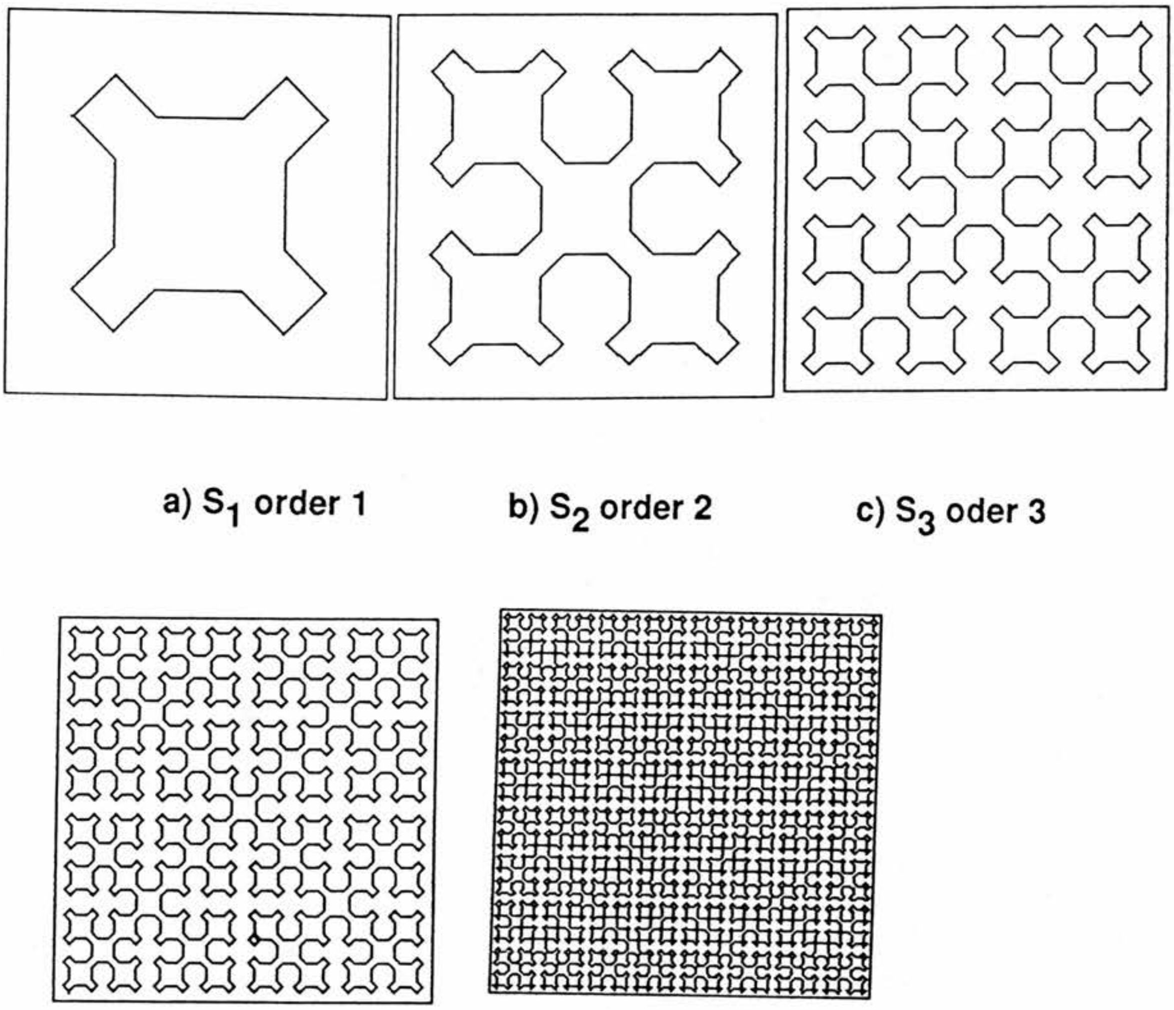
c) $P_3$ order 3

[contd]

**Fig. 2.2** Space filling curves

Peano curves order 1-3 and Hilbert curves order 1-6
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Sierpinski Polygons



a) $S_1$ order 1

b) $S_2$ order 2

c) $S_3$ oder 3

d) $S_4$ order 4

e) $S_5$ order 5

**Fig. 2.2** [contd] Space filling curves

Sierpinski curves order1-5

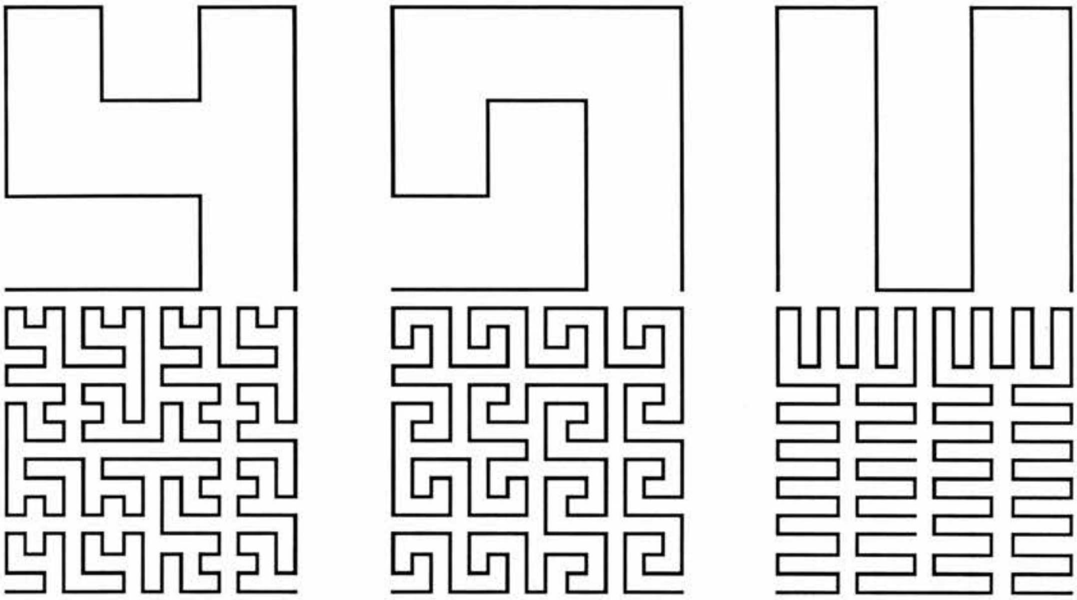


Fig 2.3 Three examples of basic tiles and second order polygons due to Griffiths[ 48]

Cole achieved two important steps forward, firstly he obtained a direct transformation for Peano polygons and later a general method to allow traversal of rectangular parallelepipeds with odd numbers of integer coordinates on each side. In developing these ideas Cole used multiple radix arithmetic, which gratefully he shortened to **murray arithmetic**, and from this he named the resulting curves **murray polygons.** This work is examined in the next sections.

## 2.2 Murray Polygons

As the basis of the research in later chapters is the work carried out by Cole [40,41,49], it seems correct to take some time to review the relevant parts of his work in detail. Cole became interested in space filling curves motivated by an argument with a colleague over the †classification of curves as either Hilbert or Peano polygons.

**†note:** This classification has troubled many writers in the field. The curves generated by Hilbert are a subset of the Peano curves but normal usage appears to name specific examples as given in Fig. 2.2.

After initial investigations he produced neat algorithms for drawing the common space filling curves, all three are obtained recursively from a single point [40]. The main procedure for the Hilbert curve is given in Fig. 2.4 the language used is the Outline System of PS-algol [50] , further details can be found in [41]. His consideration of the Peano polygon as having nine vertices by splitting each edge at its mid-point proved of significance later.

```
let draw.hilbert  = proc ( cint complexity )
begin
   let hilbert  = proc( cpic h ; cint order , width );nullproc
  hilbert := proc( cpic h ; cint order, width )
   if order = complexity then draw( h, 0, width, 0, width )
   else
   hilbert(
             (scale rotate h by -90 by -1,1          ^
              shift h by width + 2,0                 ^
              shift h by width + 2, width +2         ^
              shift scale rotate h by -90 by -1,1
                                  by width, 2 * width + 2 ),
          order + 1, 2 * width + 2  )

  hilbert ( [ 0,0 ],0,0 )
end
```

The initial call of hilbert starts with the Hilbert polygon of order zero $H_0$, which is in this case the point [ 0,0 ]. Assuming the complexity requested is greater than zero the single point is subject to the transformations detailed in the procedure with the ^ operator joining the four parts to obtain the first order polygon $H_1$. The reader is invited to follow through the transformations from $H_1$ to $H_2$ as given in Fig. 2.2.

**Fig 2.4** Main procedure for Hilbert polygons.

Peano's original definition had taken a point on the interval [0,1] and split it into two real base three numbers by taking all the odd indexed digits in their sequential order for the value for x and all the even ordered digits in their sequential order for the value for y to obtain the point ( x,y) of the square defined by $0 \le x \le 1, 0 \le y \le 1$.

It should be noted that to maintain the uniqueness of the transformation, the ambiguity of real number representation, where to base three 0.1211 and 0.121022222222....... are equally valid representations, was dealt with in a more complex manner, which need not be discussed in this report.

Cole considered application of a similar technique to define a mapping from the first $3^{2n}$ base three integers to the vertices of the nth Peano polygon. He had no success for some time before a breakthrough with a vital link, the application of cyclic progressive numbers.

### 2.2.1 The Contribution of Gray Codes

Gilbert [51] defines a Gray code as a means of quantising an angle and representing it in a binary alphabet. The encoding is such that angles in adjacent quantum intervals are encoded into n-tuples of binary digits which differ in just one place. These Gray codes are special cases of cyclic progressive number systems whose successive integers differ in only one digit. They are not restricted to binary representation but can take any number base. Cole[52] gives the following conversion rules from a pure number a) odd base and b) even base systems to Gray code.

Suppose an integer d in a pure number system with radix r is given by

$$d = d_n d_{n-1} \ldots\ldots\ldots\ldots d_3 d_2 d_1.$$

Note that any digit $d_i$ has **reduced radix complement** $r - 1 - d_i$.

a) odd base r

A digit $d_i$ remains unaltered if the **sum** of all its more significant ( i.e. left hand ) digits is even otherwise it is replaced by its complement.

b) even base r

A digit $d_i$ remains unaltered if its more significant ( i.e. left neighbour ) digit is even otherwise it is replaced by its complement.

To convert back to pure number form is exactly as given in the odd case above.

Table 1 gives some simple examples with different bases.

| base 3 | | base 10 | | base 5 | |
|---|---|---|---|---|---|
| Pure | Cyclic Prog. | Pure | Cyclic Prog. | Pure | Cyclic Prog. |
| 0000 | 0000 | 0000 | 0000 | 1443 | 1001 |
| 0001 | 0001 | 0001 | 0001 | 1444 | 1000 |
| 0002 | 0002 | ..... | ...... | 2000 | 2000 |
| 0010 | 0012 | 0009 | 0000 | ..... | ..... |
| 0011 | 0011 | 0010 | 0019 | ..... | ..... |
| 0012 | 0010 | 0011 | 0018 | 3434 | 3014 |
| 0020 | 0020 | ..... | ..... | 3440 | 3004 |
| 0021 | 0021 | 0019 | 0010 | 3441 | 3003 |
| 0022 | 0022 | 0020 | 0020 | 3442 | 3002 |
| 0100 | 0122 | 0021 | 0021 | | |

Table 2.1 Pure integers to various bases and their corresponding cyclic progressive form.

### 2.2.2 Direct Transformation for Peano Polygons

An important connection between the cyclic progressive number systems and space filling curves was seen by noting that in the case of Peano and Hilbert polygons consecutive vertices are one unit apart in either x or y but not both. After considering several possibilities and close to giving up Cole found the transformation he was seeking for the case of Peano polygons by using base three numbers ( further details follow in section 2.2.3.2 ). Further he realised the importance of the commutativity of conversion to Gray codes and reduced radix complementation to the mapping and why it was not possible therefore to use this method for Hilbert polygons. The proofs and detailed explanations to cover explicit mappings from the first $n^{2p}$ base n Gray code integers into the ordered vertices not only of a space filling curve but of the pth Peano polygon and vice versa are to be found in Cole [53]. The result now covers any odd based number system with radix r giving a generalised Peano polygon $P_m^{n,r}$ of type r in n dimensions which pass through all $r^{mn}$ points with integer coordinates in the n-dimensional cube of side $r^m - 1$ ( m = 1,2,3 .......). An illustration of the steps involved for part of a Peano polygon of order 2 is given in Fig. 2.5 and Table 2.2 .

The transformation to the pth Peano polygon is only true for the first $n^{2p}$ integers where n is odd. Soon after this Cole [54] produced a direct mapping between the first $2^{2p}$ integers and the ordered points of the pth Hilbert polygon. About the same time Fisher [55] derived a data-driven algorithm for the generation of Hilbert curves. These methods used table-driven algorithms and with minor modifications could be made equivalent. As with the Peano transformation the method could be extended to deal with Hilbert polygons in higher dimensional space for any even integer.

| Base 3 | | | | | | |
|---|---|---|---|---|---|---|
| Pure | Gray code | | | Pure | | coordinates |
| number | number | odd digits | even digits | odd digits | even digits | ( x, y ) |
| 0000 | 0000 | 00 | 00 | 00 | 00 | ( 0, 0 ) |
| 0001 | 0001 | 01 | 00 | 01 | 00 | ( 1, 0 ) |
| 0002 | 0002 | 02 | 00 | 02 | 00 | ( 2, 0 ) |
| 0010 | 0012 | 02 | 01 | 02 | 01 | ( 2, 1 ) |
| 0011 | 0011 | 01 | 01 | 01 | 01 | ( 1, 1 ) |
| 0012 | 0010 | 00 | 01 | 00 | 01 | ( 0, 1 ) |
| 0020 | 0020 | 00 | 02 | 00 | 02 | ( 0, 2 ) |
| 0021 | 0021 | 01 | 02 | 01 | 02 | ( 1, 2 ) |
| 0022 | 0022 | 02 | 02 | 02 | 02 | ( 2, 2 ) |
| 0100 | 0122 | 12 | 02 | 10 | 02 | ( 3, 2 ) |
| 0101 | 0121 | 11 | 02 | 11 | 02 | ( 4, 2 ) |
| 0102 | 0120 | 10 | 02 | 12 | 02 | ( 5, 2 ) |
| .. | .. | .. | .. | .. | .. | .. |
| 1112 | 1110 | 10 | 11 | 12 | 11 | ( 5, 4 ) |
| 1120 | 1120 | 10 | 12 | 12 | 10 | ( 5, 3 ) |
| 1121 | 1121 | 11 | 12 | 11 | 10 | ( 4, 3 ) |
| .. | .. | .. | .. | .. | .. | .. |
| 2220 | 2220 | 20 | 22 | 20 | 22 | ( 6, 8 ) |
| 2221 | 2221 | 21 | 22 | 21 | 22 | ( 7, 8 ) |
| 2222 | 2222 | 22 | 22 | 22 | 22 | ( 8, 8 ) |

Table 2.2 The transformation from base 3 numbers to the vertices of a Peano polygon.

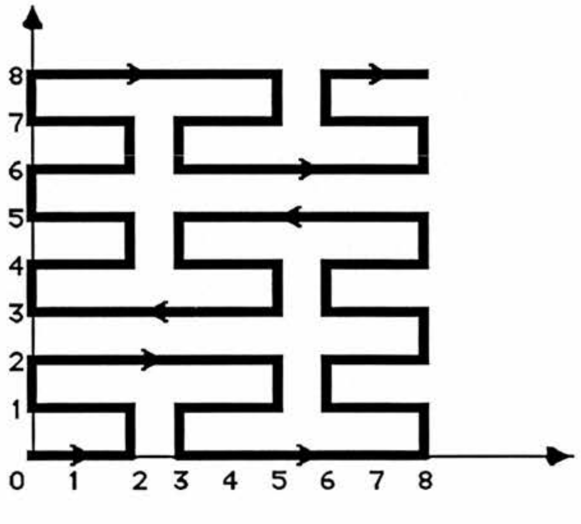


Fig. 2.5 The traversal of the second order Peano Polygon $P_2$.

### 2.2.3 Escape from the Square Cell

Cole had been aware that applications of the above techniques to problems in computer graphics looked promising. In particular data compaction for storage or transmission, object identification and others where the scanning of an area was required. These techniques, and quadtree methods ( section 1.5.3.2 ), were limited to squares with restrictions on their dimensions.

The tools to escape from the square cell were derived by Cole [41] using multiple radix arithmetic (murray arithmetic).

### 2.2.3.1 Murray arithmetic

Murray arithmetic is integer arithmetic in a number system in which each murray integer is defined as a sequence of digits

$$d_n d_{n-1} d_{n-2} \ldots\ldots d_1$$

together with a sequence

$r_n, r_{n-1}, \ldots\ldots\ldots, r_1$ of integers

where $r_i$ defines the radix associated with $d_i$ for i = 1,2, . . . ,n and such that for each i we have $0 \le d_i \le r_i - 1$.

The main operation required is addition which is defined as usual except that carry now takes place from the ith to the i + 1th digit when the sum in the ith place exceeds $r_i - 1$.

This leads to the following definitions for the initial murray integer

a) the reduced radix complement

$$d^* = b = b_n b_{n-1} \ldots\ldots b_1,$$

where $b_i = r_i - 1 - d_i$ ( i = 1,2, . . . . , n ),

b) the gray code

$d' = c = c_n c_{n-1} \ldots . c_1$ where

i) for all $r_i$ odd ,

$c_i = d_i$ if the sum of $d_{n,} d_{n-1}, \ldots, d_{i+1}$ is even or if i = n

and

$c_i = r_i - 1 - d_i$ that is, the reduced radix complement of $d_i$ otherwise.

ii) for all $r_i$ even

The rule is much simpler $d_i$ is replaced by its reduced radix complement if $d_{i+1}$ is odd or i = n, and unchanged otherwise.

iii) for $r_i$ even or odd

For any digit $d_i$ with corresponding radix $r_i$. Let j > i be the first integer such that $r_j$ is even ( It is always assumed that $r_{n+1}$ is even ).

let

$$p_{i,j} = ( \sum_{k=i+1}^{j} r_k ) \text{ rem } 2.$$

Then the cyclic progressive transform d' of d is

$$d' = c_n c_{n-1} \ldots c_1,$$

where

$$c_i = d_i \quad \text{if} \quad p_{i,j} = 0$$

and

$$c_i = r_i - 1 - d_i \quad \text{if} \quad p_{i,j} = 1.$$

note: If we consider two succesive murray integers there are two possible cases either they differ in only one digit the first $d_1$ or carry takes place from the first to the jth digit. As all the radices are odd the 1st to j -1 th are all zero and have not changed parity but the jth has changed parity as it increased by 1.

Cole named the murray integers where all the radices were odd **murray-o** integers and similarly those with all the radices even **murray-e** integers.

### 2.2.3.2 Murray Transformation

Concentrating on the murray-o integers Cole[41] proved that they can be transformed, using a method similar to that described in the section for Peano polygons, such that all the points with integer coordinates within a rectangle p by q, where p and q are odd integers, are traversed with all steps between consecutive points being of magnitude one. Importantly the murray transformation applies to each possible factorisation of p and q taken in any order. The resulting space filling curves he named murray polygons. He also extended this generalisation of the Peano polygon to higher dimensional space.

Cole almost immediately realised that murray polygons are not restricted to odd dimensions as the restriction on the radices being odd can be lifted for the first and last radices giving even sided rectangles in 2-dimensions ( see Fig. 2.9 ). Considering the two-dimensional case we now have an explicit transformation from the first n positive integers to the n points with integer coordinates in a rectangle containing exactly n such points.

The stages are outlined as follows:

1) Express the fixed base number d as a murray integer with given murray radices, $r_i$ say,

$$d = d_{2m}d_{2m-1} \ldots d_2d_1 \ , \ ( 0 \le d_i \le r_i\text{-}1, \ i = 1,2, \ldots ,2m ).$$

2) Convert this murray integer to a Gray code integer

$$d' = b_{2m}b_{2m-1} \ldots b_2b_1.$$

3) Split the Gray code number into parts x' and y' as below

$$x' = b_{2m-1}b_{2m-3} \ldots \ b_3b_1 \qquad y' = b_{2m}b_{2m-2} \ldots b_4b_2.$$

4) Convert Gray code x' and y' separately back into murray integers.

note:- Regarding stage4 it is not obvious that this will give consecutive points, indeed it is only true for odd radices. A counter example to base two is given to emphasise this point.

| binary code | gray code | gray | | binary | | |
|---|---|---|---|---|---|---|
| | | x' | y' | x | y | |
| 10110111 | 11101100 | 1010 | 1110 | 1100 | 1011 | (11,12) |
| 10111000 | 11100100 | 1010 | 1100 | 1100 | 1000 | (8, 12) |

5) convert the pair of murray integers from stage 4 into the original fixed base number pair ( x,y ).

The stages are shown clearly in Table 2.3 and Fig. 2.6.

| | step 1 | step2 | step3 | | step4 | | step5 |
|---|---|---|---|---|---|---|---|
| Pure | Murray | Gray code | | | Murray | | coords |
| integer | integer | integer | x' | y' | x | y | ( x,y ) |
| | 5353 | | | | 33 | 55 | |
| 74 | 1142 | 1142 | 12 | 14 | 11 | 10 | 3,5 |
| 75 | 1200 | 1042 | 02 | 14 | 02 | 10 | 2,5 |
| 76 | 1201 | 1041 | 01 | 14 | 01 | 10 | 1,5 |
| 77 | 1202 | 1040 | 00 | 14 | 00 | 10 | 0,5 |
| 78 | 1210 | 1030 | 00 | 13 | 00 | 11 | 0,6 |
| 79 | 1211 | 1031 | 01 | 13 | 01 | 11 | 1,6 |
| 80 | 1212 | 1032 | 02 | 13 | 02 | 11 | 2,6 |
| 81 | 1220 | 1022 | 02 | 12 | 02 | 12 | 2,7 |
| 82 | 1221 | 1021 | 01 | 12 | 01 | 12 | 1,7 |
| 83 | 1222 | 1020 | 00 | 12 | 00 | 12 | 0,7 |
| 84 | 1230 | 1010 | 00 | 11 | 00 | 13 | 0,8 |
| 85 | 1231 | 1011 | 01 | 11 | 01 | 13 | 1,8 |
| 86 | 1232 | 1012 | 02 | 11 | 02 | 13 | 2,8 |
| 87 | 1240 | 1002 | 02 | 10 | 02 | 14 | 2,9 |
| 88 | 1241 | 1001 | 01 | 10 | 01 | 14 | 1,9 |
| 89 | 1242 | 1000 | 00 | 10 | 00 | 14 | 0,9 |
| 90 | 2000 | 2000 | 00 | 20 | 00 | 20 | 0,10 |
| 91 | 2001 | 2001 | 01 | 20 | 01 | 20 | 1,10 |

Table 2.3 Murray transformation steps for radices 5 3 5 3.

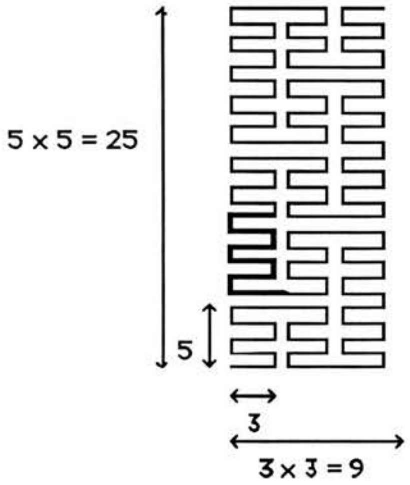


Fig 2.6 Muray polygon scanning a 9 x 25 rectangle with radices 5 3 5 3 . The section from Table 2.3 is highlighted.

The resulting dimensions of the bounding rectangle n1 by n2 are given by

$n1 = r_1 * r_3 * r_5 \ldots\ldots\ldots * r_{2m-1}$ ( product of odd radices )
$n2 = r_2 * r_3 * r_4 \ldots\ldots\ldots * r_{2m}$ ( product of even radices ).

Examples of several murray scans for different rectangles are given in Fig. 2.7. The figure also shows how the dimensions of each sub-tile can be found by considering pairs of adjacent radices. The pair $r_1$, $r_2$ giving the x and y dimensions of the smallest basic tile, $r_3 * r_1$, $r_2 * r_4$ the next and so on. These examples also highlight the effect of the order of the radices.

**Radix value 1**

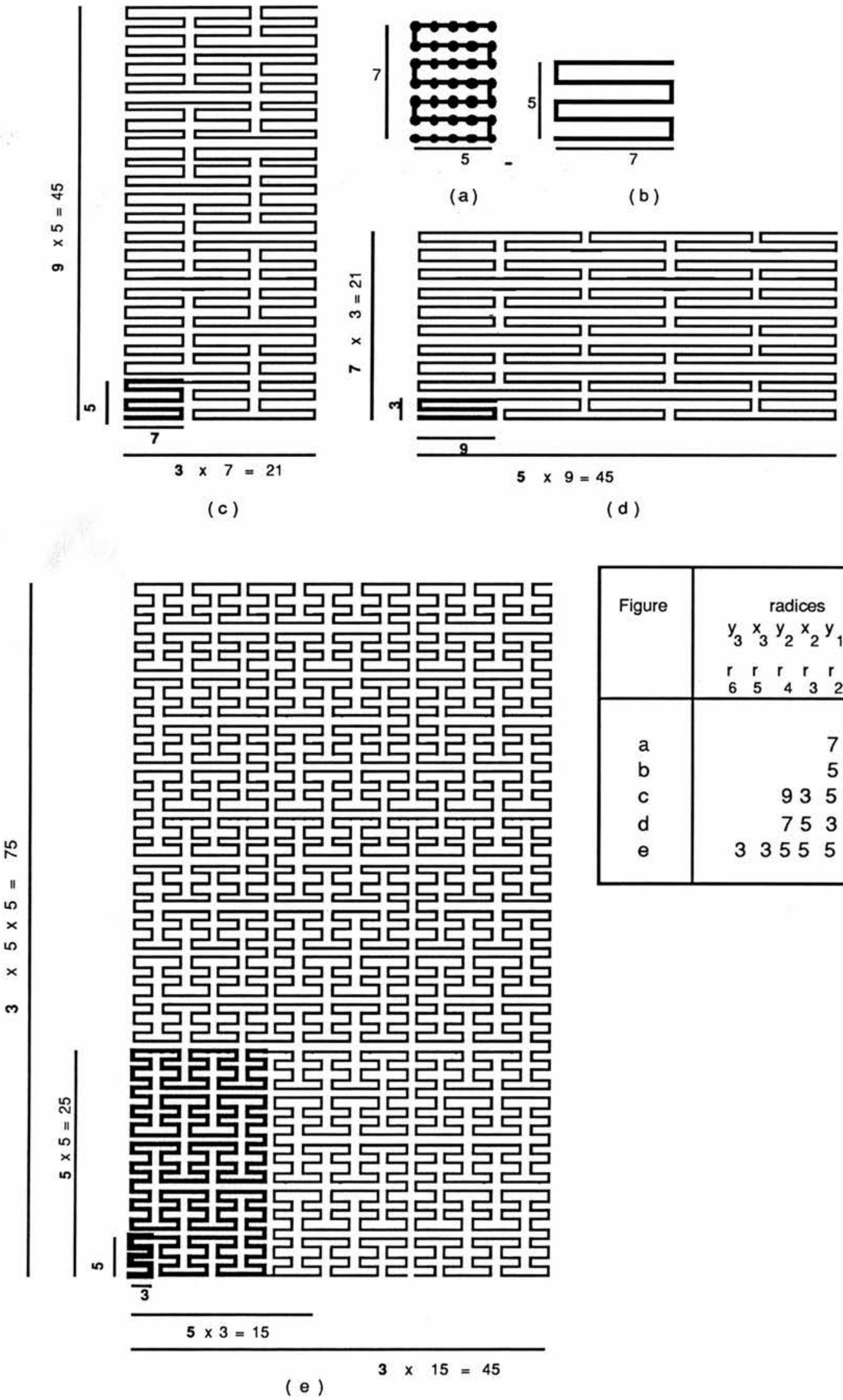
A radix value of 1 has two main uses as follows. Firstly it can be used to force movement in a particular direction. For instance for any tile pair $r_{2k}$, $r_{2k-1}$ if the least significant radix, namely the 'x radix' $r_{2k-1}$ has value 1 then all steps are forced to occur in the y direction, as shown in the basic tile of Fig. 2.8.c. Similarly movement can be restricted to the x direction by making the 'y radix ' take value 1 as shown in Fig. 2.8.b. The second use of radix value 1 is when the dimensions of the bounding rectangle cannot be factorised into an equal number of factors. The radices are packed with additional dummy radices of value 1. This is illustrated in Fig. 2.8. c, d & e. It should be noted that it is fortuitous that the fundamental algorithm works with a radix value of 1.

**Even Dimensions**

As mentioned earlier the restriction on the radices being odd can be relaxed for the first, normally x, radix and the last, normally y, radix. The effect of the first radix being even is to give the **basic** tile an even dimension. If the last radix is even then the number of horizontal scans of the largest tile is now even. These ideas are clearly seen in the diagrams of Fig. 2.9.

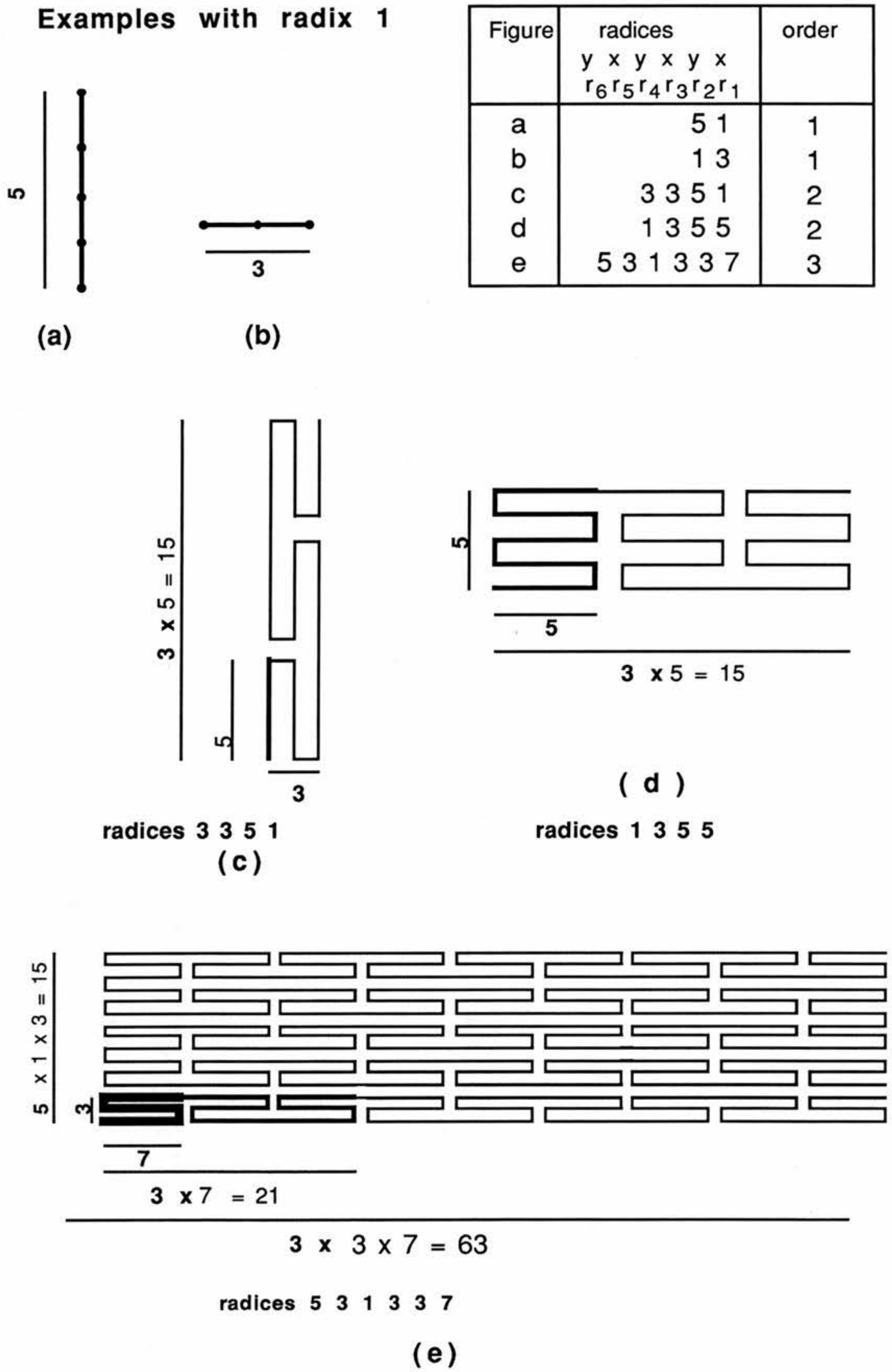
**Mixed Scans**

Cole [49] outlines how the murray polygon algorithm may be modified to allow switching of scan direction from tile to tile within a scan. He also combines this idea with the scan patterns described by



| Figure | radices | | | | | | order |
|---|---|---|---|---|---|---|---|
| | $y_3$<br>$r_6$ | $x_3$<br>$r_5$ | $y_2$<br>$r_4$ | $x_2$<br>$r_3$ | $y_1$<br>$r_2$ | $x_1$<br>$r_1$ | |
| a | | | | | 7 | 5 | 1 |
| b | | | | | 5 | 7 | 1 |
| c | | | 9 | 3 | 5 | 7 | 2 |
| d | | | 7 | 5 | 3 | 9 | 2 |
| e | 3 | 3 | 5 | 5 | 5 | 3 | 3 |

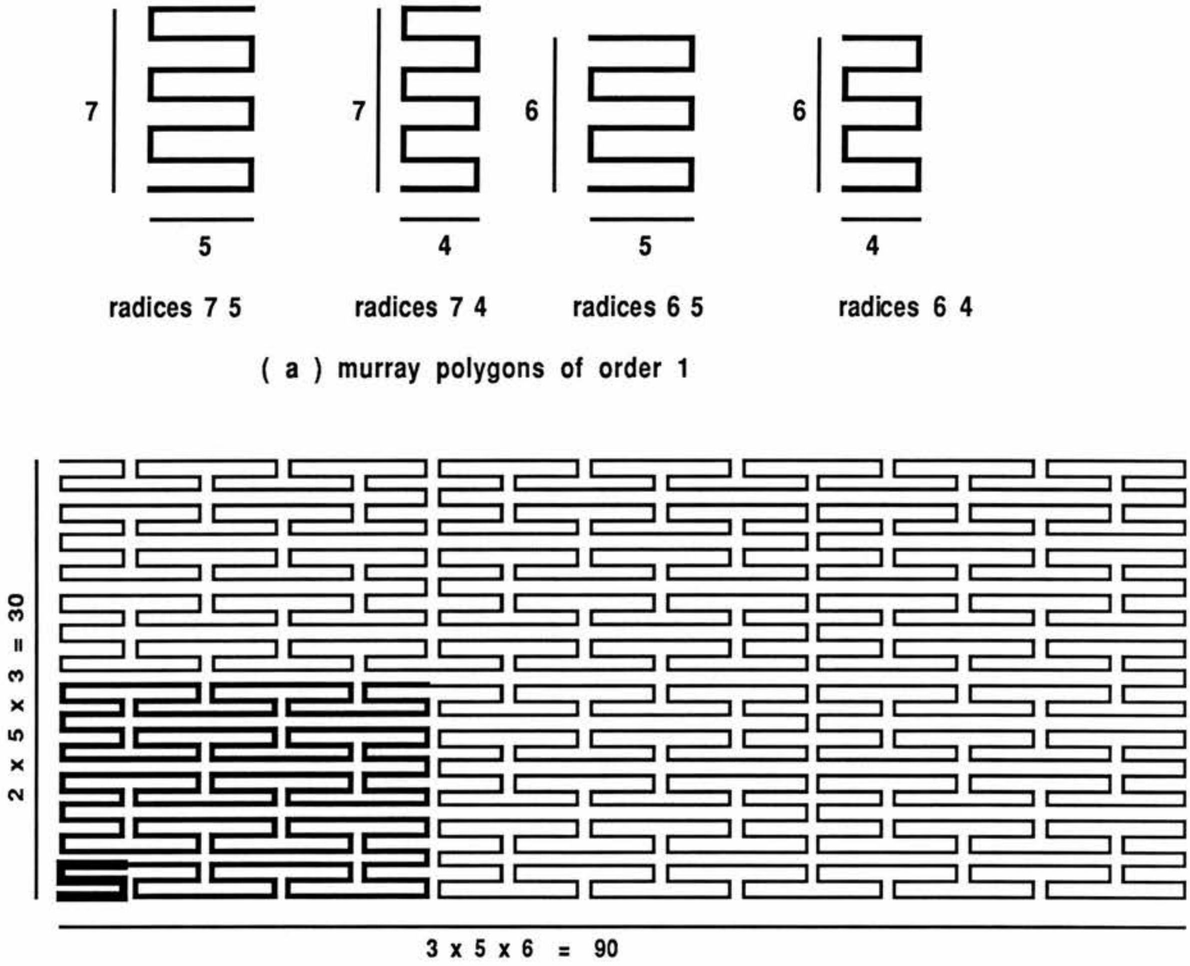
**Fig. 2.7** Various murray polygons as detailed in table above ( shading is only used to emphasise basic tile shapes ).



| Figure | radices | order |
|---|---|---|
| | y x y x y x | |
| | $r_6 r_5 r_4 r_3 r_2 r_1$ | |
| a | 5 1 | 1 |
| b | 1 3 | 1 |
| c | 3 3 5 1 | 2 |
| d | 1 3 5 5 | 2 |
| e | 5 3 1 3 3 7 | 3 |

**Fig. 2.8** Murray polygons showing the effect of radix value 1.

Murray polygons with even radices



( a ) murray polygons of order 1

(b) Murray polygon order 3 radices 2 3 5 5 3 6

Fig 2.9 Murray polygons with even radices

Griffiths [48] to give mixed Griffiths and murray scans. Cole is at present investigating the application of murray scans to produce bilevel hardcopy from grey scale image data giving results similar to half-toning. Mixed scans may be a useful tool in this application to reduce patterns caused by the standard scan. An example of a mixed scan is given below in Fig 2.10.

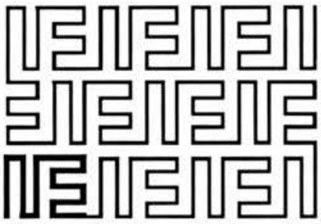


Fig. 2.10 Mixed scan from [49].

### 2.2.4 Implementation of murray scan

The original implementation by Cole is given in a) and his fast murray scan is given in b), in both cases the programming language is PS-algol[50]

**a) Original implementation**

The main procedures only are given, the steps referred to are those from the algorithm of section 2.2.3.2 .

The murray integer is held in an array of integers and this value is incremented to move from vertex to vertex on the scan. The main procedures required are ,

1. Conversion from pure integer to murray integer
   ***convert.to.murray***
2. Conversion from murray integer to pure integer
   ***convert.from.murray***
3. Increment murray integer by one.
   ***next.murray***

```
! Input parameters are an integer and an array of radices
! Output is the corresponding array of murray digits
let convert.to.murray = proc( int n; *int radices -> *int )
begin
      let murray.int = vector 1 :: upb( radices ) of 0
      let i := 1
      while n ~= 0 do
      begin
            murray.int( i ) := n rem radices( i )
            n := n div radices( i )
            i := i + 1
      end
      murray.int
end

! Input parameters are murray integer and radices arrays
! Output is the corresponding integer
let convert.from.murray = proc( *int murray.int,radices -> int )
begin
      let top = upb( murray.int )
      let n := murray.int( top )
      for i = top - 1 to 1 by -1 do
         n := n * radices( i ) + murray.int( i )
      n
end

! Input parameters are murray integer and radices arrays
! The murray integer array is incremented by 1
let next.murray = proc( *int murray.int, radices )
begin
      let i := 1
      while murray.int( i ) = radices( i ) -1 do
      begin
            murray.int( i ) := 0
            i := i + 1
      end
      murray.int( i ) := murray.int( i ) + 1
end

! Input parameters are murray.int and radices arrays
! murray.int is converted to the Gray code equivalent
let gray.code = proc( *int murray.int, radices )
begin
      let top = upb( murray.int )
      let parity :=(  ( murray.int( top ) rem 2 ) = 1 )
      for i = top - 1 to 1 by -1 do
      begin
            if parity do
                murray.int( i ) := radices( i ) -1 - murray.int( i )
            if ( murray.int( i ) rem 2 = 1 ) do
                parity := ~parity
      end
end
```

```
! Input is a murray integer
! Output is a structure holding the x and y murray integer arrays
let split.x.y = proc( *int murray.int -> pntr )
begin
        structure coords( *int a,b )
        let top = upb( murray.int )
        let x = vector 1 :: top div 2 of 0
        let y = vector 1 :: top div 2 of 0
        let i := 1
        for j = 1 to top - 1 by 2 do
        begin
                x( i ) := murray.int( j )
                y( i ) := murray.int( j + 1 )
                i := i + 1
        end
        coords( x,y )
end
```

### b) Fast murray scan algorithm and implementation

The original implementation required improvement in efficiency for complete scans. The parts of the first implementation that slow the algorithm down is the conversion to and from murray integers to pure integers and the transformation using the ***gray.code*** procedure.
By careful analysis of the parity changes it is possible to produce a much more efficient algorithm as described below.
Consider the murray integer and radices as below,

radices ----> $r_{2m}r_{2m-1}.........r_{i+4}r_{i+3}r_{i+2}r_{i+1}r_ir_{i-1}..........r_3r_2r_1$

murray integer ----> $d = d_{2m}d_{2m-1}........d_{i+4}d_{i+3}d_{i+2}d_{i+1}d_id_{i-1}........d_3d_2d_{1.}$

Let $p_i$ be the parity of the sum $d_{2m}$ + ..... $d_{i+3} + d_{i+2} + d_{i+1.}$

Let $s_i$ be the parity of the sum $d_k$ +..... $d_{i+6} + d_{i+4} + d_{i+2}$,
where k = 2m for even i otherwise k =2m - 1.

Let $q_i$ be the parity of the sum $d_k$ + ..... $d_{i+5} + d_{i+3} + d_{i+1}$,
where k = 2m for odd i otherwise k = 2m-1.

Considering the stages 2, 3 and 4 in the murray transformation of section 2..2.3.2 for odd radices. Assume **p** and **q** are boolean arrays holding the parities $p_i$ and $q_i$ respectively.

Stage 2: transformation from murray integer to gray code.

If $p_i$ is false( even sum ) $d_i$ is unchanged otherwise
$p_i$ is true ( odd sum ) and $d_i$ is replaced by $r_i - 1 - d_i$.

Since $r_i$ is odd $d_i$ and $r_i - 1 - d_i$ have the same parity there is no change in any of the parities **p**.

Stage 3: Split Gray code integer into x' and y' parts.

Stage 4: Convert x' and y' parts from Gray code to murray integers.

The parity of i identifies whether the x part or y part is under consideration ( i odd or even corresponds to x or y respectively ). The change in a digit now follows the rules as in stage 2. However the change in a digit depends on the parity of $s_i$.

if $s_i$ is false ( even sum ) $d_i$ is unchanged otherwise
$s_i$ is true ( odd sum ) and $d_i$ is replaced by $r_i - 1 - d_i$.

The result of stages 2, 3 and 4 clearly depend on the value of $p_i$ and $s_i$. However it is easy to see, reference Fig 2.11, that the change can be found by considering $q_i$ alone.

The change can be controlled as follows,

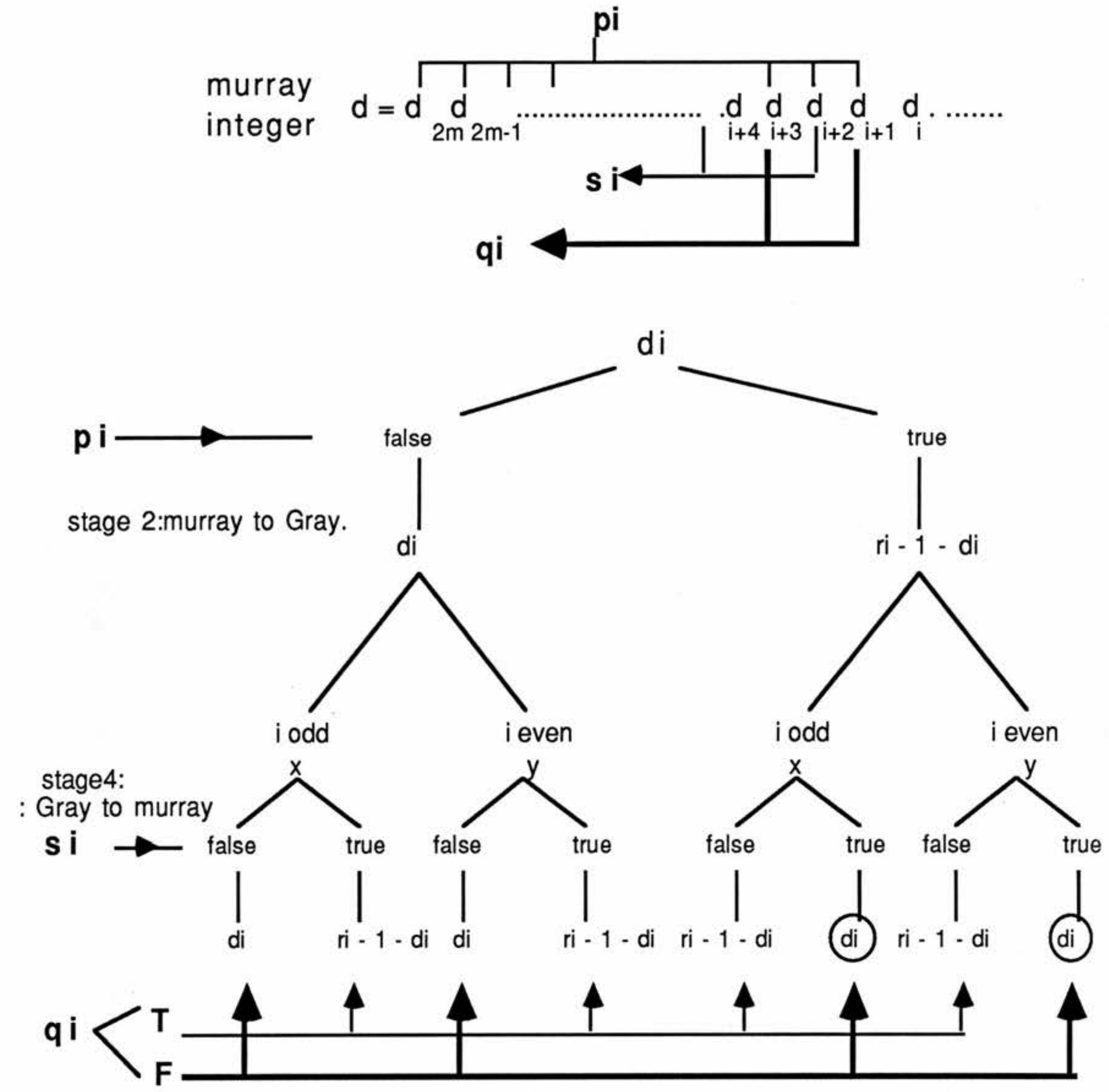
If $q_i$ is false ( even sum ) then $p_i$ and and $s_i$ must have been of equal parity and thus $d_i$ is unchanged otherwise $q_i$ is true and $d_i$ is replaced by $r_i - 1 - d_i$

Consider now the case of a murray integer about to be increased by one. This will cause a change in parity of digit $d_i$. Either $d_i$ is the first digit or carry has taken place in one or more positions and all the digits to the right of $d_i$ will become zero as shown in the simple cases below.

| | | | |
|---|---|---|---|
| radices ---> | 5 9 5 7 5 3 | radices ---> | 5 9 5 7 5 3 |
| | 3 2 4 0 1 1 | | 3 2 4 6 4 2 |
| **i = 1** | 3 2 4 0 1 2 | **i = 4** | 3 3 0 0 0 0 |

Now 0 and $r_i$-1 are both even hence the only digit to change parity is $d_i$. Thus the parities $q_{i-1}$,$q_{i-3}$, ....... will have changed.

We now see that when a murray integer increases by one there is only



| | | | | |
|---|---|---|---|---|
| pi | T | F | T | F |
| si | T | F | F | T |
| qi | F | F | T | T |

di - ri - 1 - ( ri -1 - di ) = di

T - true ( odd sum )

F - false ( even sum )

**Fig 2.11** Diagram showing steps of digit transformation leading to the use parities qi in fast algorithm.

one digit which changes parity and its position determines whether a change has occurred in x or y, and also the q values to change. The only remaining information to determine is whether the change is +1 or -1. This can be found from the value of parity $q_i$ with false or true corresponding to +1 or -1 respectively.

The fast algorithm can now be stated as follows, assuming the murray integer is stored in a vector of integers and the parities in **q** a vector of booleans ( boolean array ).

1. Increment the murray integer by one,
2. find the position i of the leftmost digit to change ( i.e. which digit changed parity ),
3. alter appropriate parities in **q**, namely $q_{i-1}$,$q_{i-3}$ .....,
4. select x or y to change based on the parity of i,
5. increment the selected coordinate by +1 or -1 as indicated by parity $q_i$.

The main procedures and important lines used for the fast algorithm implementation are as below.

```
! Input parameters are murray integer and radices array.
! Output is the position of the digit to change parity.
let increment := proc( *int d, r ; int i -> int );nullproc
increment := proc( *int d, r ; int i -> int )
      if d( i ) < r ( i ) - 1 then { d( i ) := d( i ) + 1; i }
      else { d( i ) := 0; increment ( d, r, i+1 ) }
```

```
! Input parameters are parities boolean array and position integer
! The parities being updated accordingly
let change.parities = proc( *bool q; int startpos )
      for i = startpos - 1 to 1 by -2 do q( i ) := ~q( i )
```

```
! The lines which determine the position of the change and
! updates the parities.
! Whether the change is +1 or -1 and
! the coordinate to change.
let pos = increment( digits,radices,1 )
change.parities( parities,pos )
let inc = if parities( pos ) then -1 else 1
if i rem 2 = 1 then x := x + inc else y := y + inc
```

## 2.3 Hardware Implementation

Following on from the ideas used in the implementation of the fast algorithm outlined in the previous section, Cole [56] suggests some hints on hardware implementation. The basic suggestion is that the function of the array of integers holding the murray integer is taken over by a bank of shift registers. Each register would have a capacity corresponding to the radices selected and be initially set with value 1. If a register is cleared by a shift operation it then resets to zero and forces a shift in the next register. If the register does not clear then the parity of the register number will identify whether the movement is in the x or y direction. There would also be a number of parity bits which can be toggled appropriately and will determine if the step is -1 or +1.

## 2.4 Applications of murray scans to image data.

The major application to data compression is the subject matter of the following chapters. However some general points on scanning and other applications are mentioned briefly.

### 2.4.1 Scanning

Witten & Wyvill [64] suggested the use of space filling curves to scan an image. A graphics screen can be considered as a finite rectangular array of pixels with integer coordinates. A picture or image may likewise may be considered split into a finite number of cells. These arrays can be scanned in total or part by an appropriate murray polygon. Cole calls this process a **murray scan**.

There are several expected advantages from a murray scan when compared to a linear scan. Firstly, as the murray scan by its nature will pass through many points close to each other it will be able to take advantage of any local correlation between pixels. This should be a considerable advantage as many images have strong local correlation. Secondly, the murray scan will in general change direction frequently within a relatively small and compact area, thus reducing the common patterning resulting from the more regular linear scan. Lastly, murray scans have considerable flexibility allowing change of basic tile pattern, scan order, scan direction and even dimension of scan.

Another common representation is by quadtrees. This is included as a special case of a Hilbert scan and murray scans include data storage similar to quadtree 'scanning' based on the number of subdivisions of a basic tile. Murray scan's ability to cover rectangular areas immediately without modification is a major advantage over quadtree representation.
A disadvantage of the murray scan is when adjacent points in the image are a long way apart on the scan sequence.

The result of the scanning process is a series of run lengths and corresponding colour or grey scale data. Bilevel images will only require run lengths as they alternate between black and white.

### 2.4.2 Applications

Major application areas appear to be:

1. data compaction for storage and transmission of exact and non-exact images,
2. object identification,
3. operations on images using run lengths and
4. bilevel representation of monochrome images.

An important feature of the method is the ability to carry out calculations and operations on the run lengths themselves without returning to the original image.

Operations on images and object identification are mentioned in more detail in Cole[41, 56] and will be discussed in the later sections of this report.

### Summary

The historical background to space filling curves has been given. The path of development by Cole from Peano's early work to murray polygons has been traced out in some detail. The advantages of explicit transformation, flexibility and local 'area' scanning were detailed. The use of murray scans to run-length encode an image follows in the next chapter.

# Chapter 3

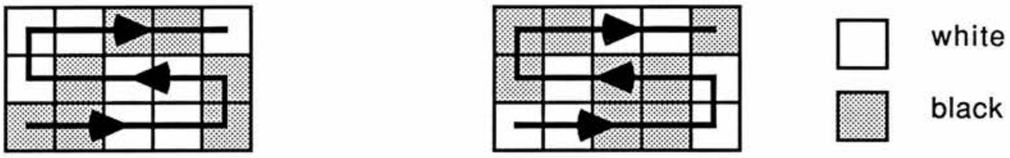
## Application of Murray Polygons to Data Compression

Cole[57] suggested various techniques for the compression of data using the space filling curves of Peano and Hilbert. Later Cole [58] considered the application of murray polygons to data compaction. Some of the main ideas arising are introduced and expanded in this chapter as they form the background to the work reported in the next few chapters.

### 3.1 Reversible error-free compression.

The data compression is achieved by run-length coding. The run length sequences and their associated colour information are produced by scanning an image with a murray scan. This exhaustively passes through each pixel recording the colour information and the number of successive pixels with the same value. The sequence of colour and run length are coded to minimise the data required to describe the image. The maximum data compression occurs when an image is made up of a small number of long run lengths. The technique depends on consecutive pixels having the same value. In the case of grey scale images with 256 possible levels the chance of long run lengths occurring is not high. The image is usually split into bit planes and each bit plane considered separately as a black and white image. In the case of black and white images there is no need for colour information as the run lengths alternate between black and white. However, the first output in the sequence must obey a convention, usually taken as first run length is white. If the first run length is one of black pixels then a zero is output first. With this information an image can be unambiguously described and thus reconstructed from the run lengths at any required position.

For example, the simple rectangular images below, Fig. 3.1, have been scanned with a murray polygon of first order with x-radix 5 and y-radix 3. In case a) the leading zero signifies that there are no white pixels initially, thus the first two pixels are black, whereas the same sequence without the leading zero corresponds to a different image as given in case b).



a)

0,2,2,2,2,1,3,2,1

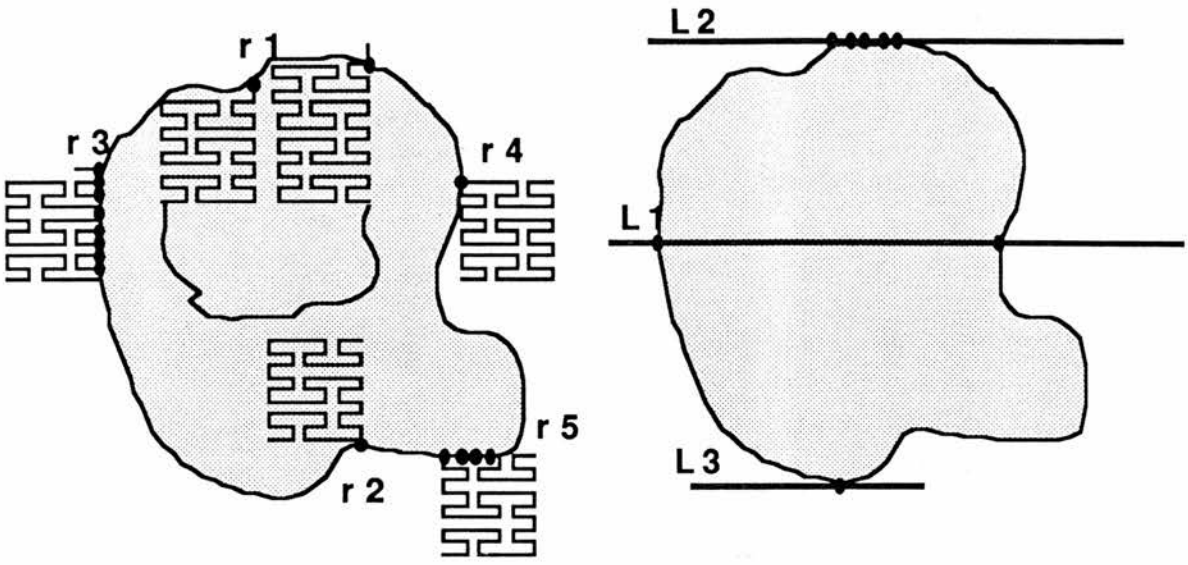
b)

2,2,2,2,1,3,2,1

Fig. 3.1 Murray scan and resulting run-length sequences.

It is expected that the murray scan will be able to take advantage of any inherent structure in the image. The fact that the murray scan is covering an 'area' before moving on means that it should have a better chance of capturing any dependence within local areas of the image and hence produce better compression than the standard linear scan. The success of the final data compression depends on the distribution of the run lengths generated from the image and the total number of run lengths. It might be expected that the murray scan with its localised scanning patterns would lead to long runs and hence produce a significantly smaller number of run lengths than a linear scan in homogeneous areas of an image. Fortunately the former is true and significantly more long run lengths are produced but the number of runs is not always substantially less. The explanation for this is as follows. Consider a relatively large homogeneous area with reasonably well defined boundaries as shown in Fig. 3.2. Each linear scan will in general pass through two boundary points (L1). Exceptions to this occur when the scan-line is a tangent (L3) when only one point is used or when the boundary runs parallel to the scan-line (L2) using a number of points. The exceptions tend to cancel each other out and the number of run lengths is approximately proportional to half the number of boundary points. This also applies to murray scans despite the differences in the scanning pattern and the resulting exceptions. A murray scan will have a more tortuous route through a region but will still in general enter by one point and exit cleanly by another point (r1). There are obvious exceptions illustrated by r2 - r5. These cover cases where the scan approaches the boundary, either from the inside (r2) or the outside ( r3,r4,r5 ), and then touches it in one point and retreats, using one point, or runs along the boundary using up several points.

It has been found that the above exceptions tend to cancel each other out with in most cases the murray scan giving marginally less runs than the linear scan. In addition to producing a smaller number of total runs the murray scan produces more long runs as it moves around within the area. Associated with a homogeneous area we therefore expect long runs, a fact that can be applied to basic object identification. If there are long runs generated from the area then equally there must be a large number of short runs if the number of runs is to remain proportional to half the boundary points. This will lead to a particular pattern in the run-length distribution which will hopefully be helpful to the task of data compression.



a) Murray scan

b) Linear scan

Typical boundary point possibilities for homogeneous area

Fig. 3.2 Diagram showing relationship of scan-lines to boundary points.

Cole [57] investigated some computer generated black and white images of a geometric nature using Peano, Hilbert and linear scans. He found that both Peano and Hilbert scans gave a significantly smaller number of run lengths for his tested images, see Fig. 3.3. These differences are higher than expected from the discussion in the previous paragraph. This may be due to the white boundary areas and the well defined boundaries of the geometric shapes used. In the examples reported later in this work the differences in number of runs for the images tested is still apparent but much smaller supporting the description given in the previous paragraph.

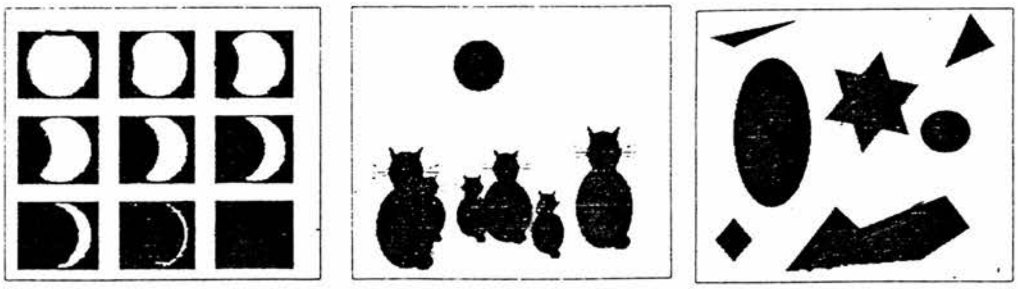


Fig. 3.3 Some images investigated by Cole [57]

The possibility of data compression seems to lie with the distribution of the run lengths. This will obviously depend on the composition of the image under consideration and as mentioned in earlier chapters the number of possible images is very large indeed. Run length coding has been used successfully in many situations reported in the literature and there is a large set of images that produce suitable run-length distributions. Given this the work reported in later chapters will investigate the possibility that for a given image the run-length distribution produced by a murray scan will be more promising than existing methods.

### 3.2 Irreversible Coding

### Compression with reduction of image quality

Considerable reduction in the required data can be achieved if it is acceptable to lose information. This is achieved by merging short run lengths into the surrounding run lengths. A short run length can have any small integer value from one upwards. A sequence of three run lengths a, n, b is replaced by a single run length of value ( a + n + b ). This results in the central run length changing colour and being merged with the surrounding colour. The amount of information lost can be controlled by varying the value of n and advantages to data reduction weighed up against the quality of the resulting image. If the value of n is increased then the process is most usefully applied by consecutive application removing runs of length 1,2 and so on up to the required value. The method is applied to black and white images and can be extended to grey scale images via application to individual bit planes.

The quality of the resulting images has been promising even with a substantial loss of data. One of the reasons for this is the nature of the run length distribution from the murray scan. As mentioned in section 3.1 we expect a murray scan to produce a significant number of short run lengths to balance the long runs. Many of these short runs are not isolated points in the image but arise from the scan just touching the boundary of a homogeneous area. Therefore the perceived deterioration of the images depends on the nature of the boundaries. In smooth regular boundaries the deterioration may become unacceptable far more quickly than in an image with rougher less well defined boundaries.

A simple case is shown below in Fig. 3.4 to emphasise the technique. More meaningful and detailed examples will be given in the next chapter. The method is image dependent and it is not difficult to select pathological examples such as a chess board pattern which is reduced to a single coloured square with the merging of runs of length 1. However, the examples given later will show that in many cases the length of runs removed can be increased to 2, 3 and further for many situations and still give acceptable results.

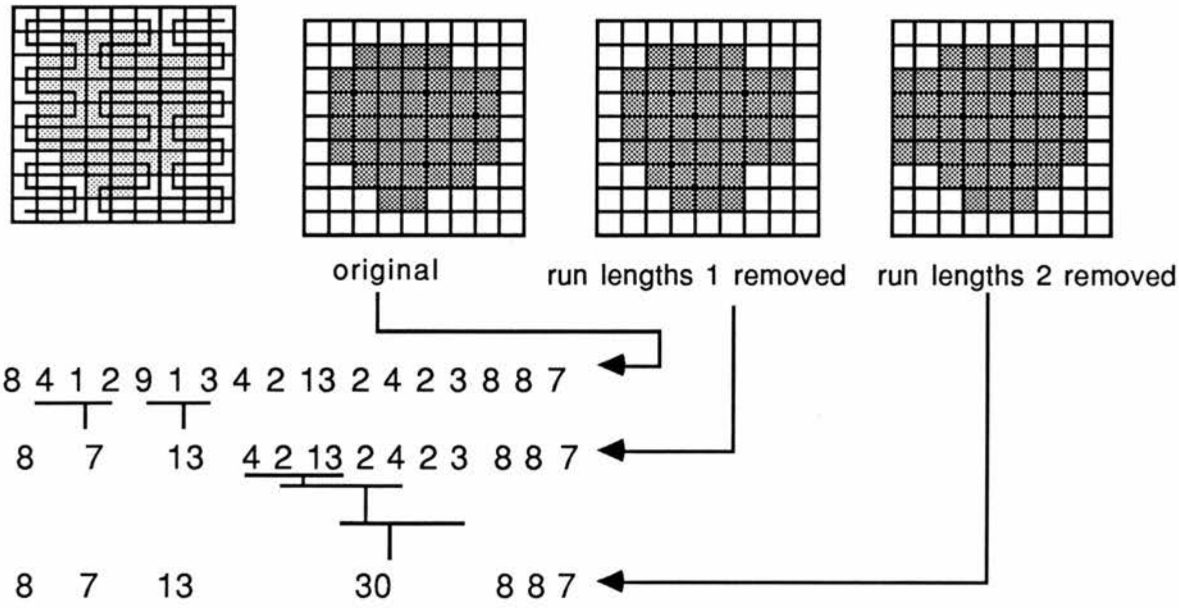


Fig. 3.4 Diagrams showing the effect of merging short run lengths.

## 3.3 Rough Frames - Exact Pictures

It is possible to combine the methods of sections 3.1 and 3.2 to produce a sequence of run lengths such that part of the image can be reconstructed exactly while the rest is reduced in quality by the removal of short run lengths. This can be done by manipulation of the original run length sequence away from the original image or directly from the original as a first step.

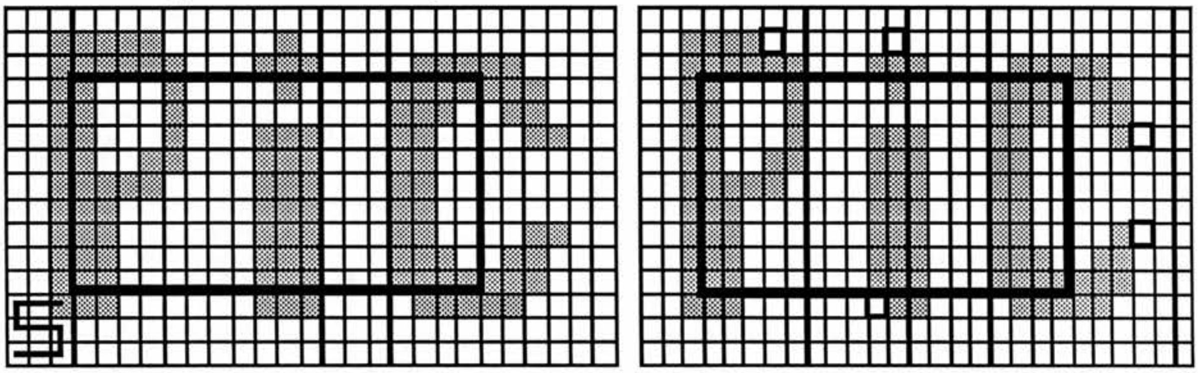


**Fig. 3.5 Diagrams showing the effect of removal of run lengths of length 1 in the 'border' of image ( scan radices 5 9 3 3 ).**

It is thought that an exact 'picture' and approximate 'frame' might be a possible application with the geometrical layout as given above in Fig. 3.5. The central area would be of more interest to the viewer or may contain information that is more likely to change if we are considering motion in a 'head and shoulders' image.

## 3.4 Coding

It was initially felt that at this stage of the investigation the coding strategy should be relatively simple but such that it gave a good idea of the success of the scanning technique. A very poor choice of coding strategy, fixed -length coding for example, could obscure any advantage gained from the murray scan. A variable length code is obviously required as certain values of run length will be much more likely than others. Gonzalez & Wintz [9] suggest that the distribution of run lengths for many images is approximately exponential and that B-codes ( see section1.5.1 ) are nearly optimal for these cases. The B-codes are exactly the same as the 'continuation-bit' or 'extension-bit' coding suggested by Cole [2.21] . Some variation on the standard B-codes is also investigated where the number of information bits may vary within the code. A B1 code is of the form CiCiCi....., B2 code CiiCiiCiiCiiCii.... variations that may be useful are CiCiCiiCiiCiiiCiiiCiiii ...., CCiCiCiiCiiiCiiii .... and others, where **C** is a continuation-bit and **i** an information-bit. The main advantage of this coding strategy is its simplicity and ease of implementation.

**Summary**

The application of murray scans to data compression both for reversible and irreversible coding has been explained in detail. Murray scans are an alternative to linear scanning and through their ability to capture any local correlation in an area or block produce very suitable run length distributions. These distributions offer the opportunity for easy and reasonably efficient coding by continuation bit codes. The tiles of the murray scan effectively break the image into blocks as does quadtree encoding, but also provide an explicit transformation and considerable flexibility allowing rectangular areas to be considered. The opportunity is also present to vary the scan for a given area by selecting a different set of radices or changing the order of the radices.

Data compression via resolution approximation is very straightforward, using the method of short run length removal. The resulting images suffering very mild distortion of area boundaries and removal of isolated areas, leaving the main features virtually unscathed.

The method of short run length removal can easily be extended to provide more drastic data reduction up to any level with the corresponding reduction in image quality.

# Chapter 4

## Applications and results for Black and White Images.

The work covered in this chapter can be split up as follows:-

a) Various black and white images are scanned by murray scan and the run length distributions are found;

b) The run lengths generated by the scans are coded by

i) a reversible information preserving coding and

ii) an irreversible coding. The compression details are found for both of these cases;

c) Examples are given of images split into exact and non-exact regions;

d) Comparison of murray scan and linear scan data.

### 4.1 The Images

The images to be used are from various sources.

1. **'World'** plates of size 256 by 256 available from demonstration software as shown in Fig. 4.1. This set of 30 images gave a wide range of composition as illustrated by the eight examples shown in Fig. 4.1. The variation is from large well formed homogeneous areas with few small or isolated elements such as Fig. 4.1.1 to far more scattered and broken patterns as in Fig. 4.1.5. They are of course atypical in that all include a circular boundary this could be overcome by clipping portions for investigation.

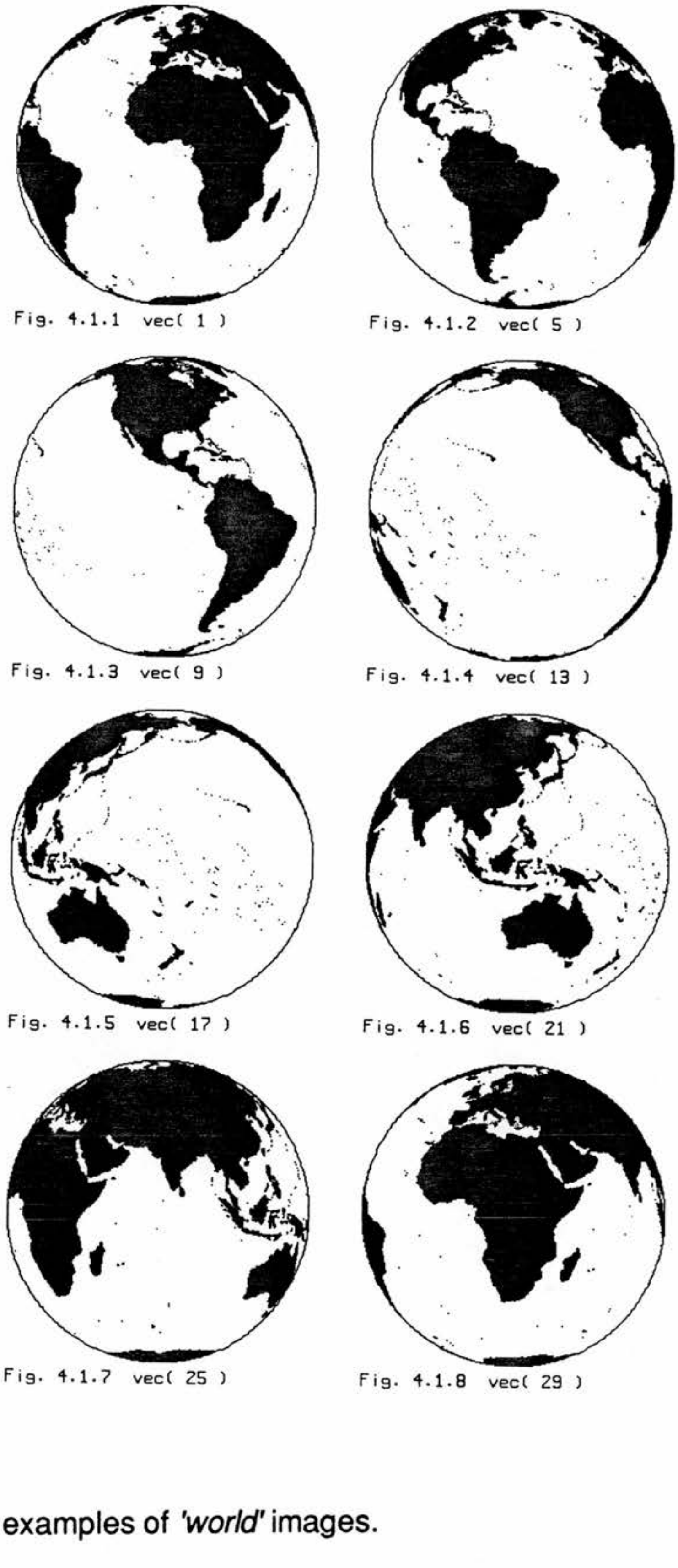


Fig. 4.1.1 vec( 1 )

Fig. 4.1.2 vec( 5 )

Fig. 4.1.3 vec( 9 )

Fig. 4.1.4 vec( 13 )

Fig. 4.1.5 vec( 17 )

Fig. 4.1.6 vec( 21 )

Fig. 4.1.7 vec( 25 )

Fig. 4.1.8 vec( 29 )

Fig. 4.1 Various examples of *'world'* images.

2. Images grabbed from the real world using a frame grabber. The hardware used is described in detail in later chapters. It consists of a camera, either Ferguson colour plumbicon or EEV photon CCD, and low-cost frame grabber controlled by an IBM PC/AT. The images grabbed are of size 512 by 512 with each pixel having a grey scale intensity value from 0-255. The analog to digital conversion is performed using an 8-bit flash converter. These images are to be used later in our discussion of application to grey scale images but initially each bit plane is considered as a separate black and white image available for investigation. Figs 4.3 - 4.5 show subimages of size 225 by 225 from some of the images used in this category. For reference they are referred to as '*ajc*' ( Fig. 4.3 ), '*fred*' ( Fig. 4.4 ) and ' *imb*' ( Fig. 4.5 ). These images were the first to be tested from the real world and gave interesting composition for investigation. They are dependent on a number of factors in the initial image capture but before considering those it is important to be clear about the information the images contain as they are used considerably in later investigations. If the three-dimensional model of a grey scale image is considered, as described in section 1.3, and the particular case taken where the model formed is a cone with the vertex value of 255 ( white ) and base value of 0 ( black ), this corresponds to a two-dimensional image in the form of a circular disc with a white centre and smooth linear transition radially to black at the circumference. Each of the eight bit planes contains information about the status of a particular bit for the stored intensity value from 0-255. This is shown clearly in Fig 4.2 where the cone model and the corresponding black and white images for bit planes 0 - 2 are given. Table 4.1 shows the relationship between white and black and the corresponding intensity values for these images.

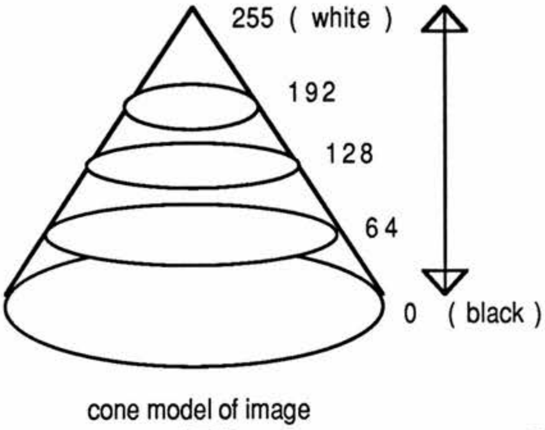
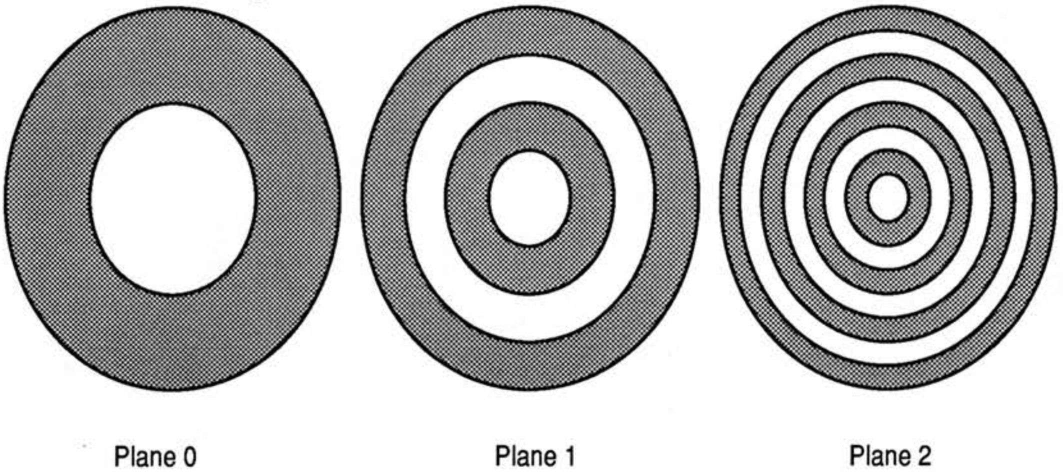


**Table 4.1**

| plane | black | white | bit position |
|---|---|---|---|
| 0 | 0 - 127 | 128 - 255 | ▨□□□□□□□ |
| 1 | 128 - 191<br>0 - 63 | 192 - 255<br>64 - 127 | □▨□□□□□□ |
| 2 | 192 - 223<br>128 - 159<br>64 - 95<br>0 - 31 | 224 - 255<br>160 - 191<br>96 - 127<br>32 - 63 | □□▨□□□□□ |

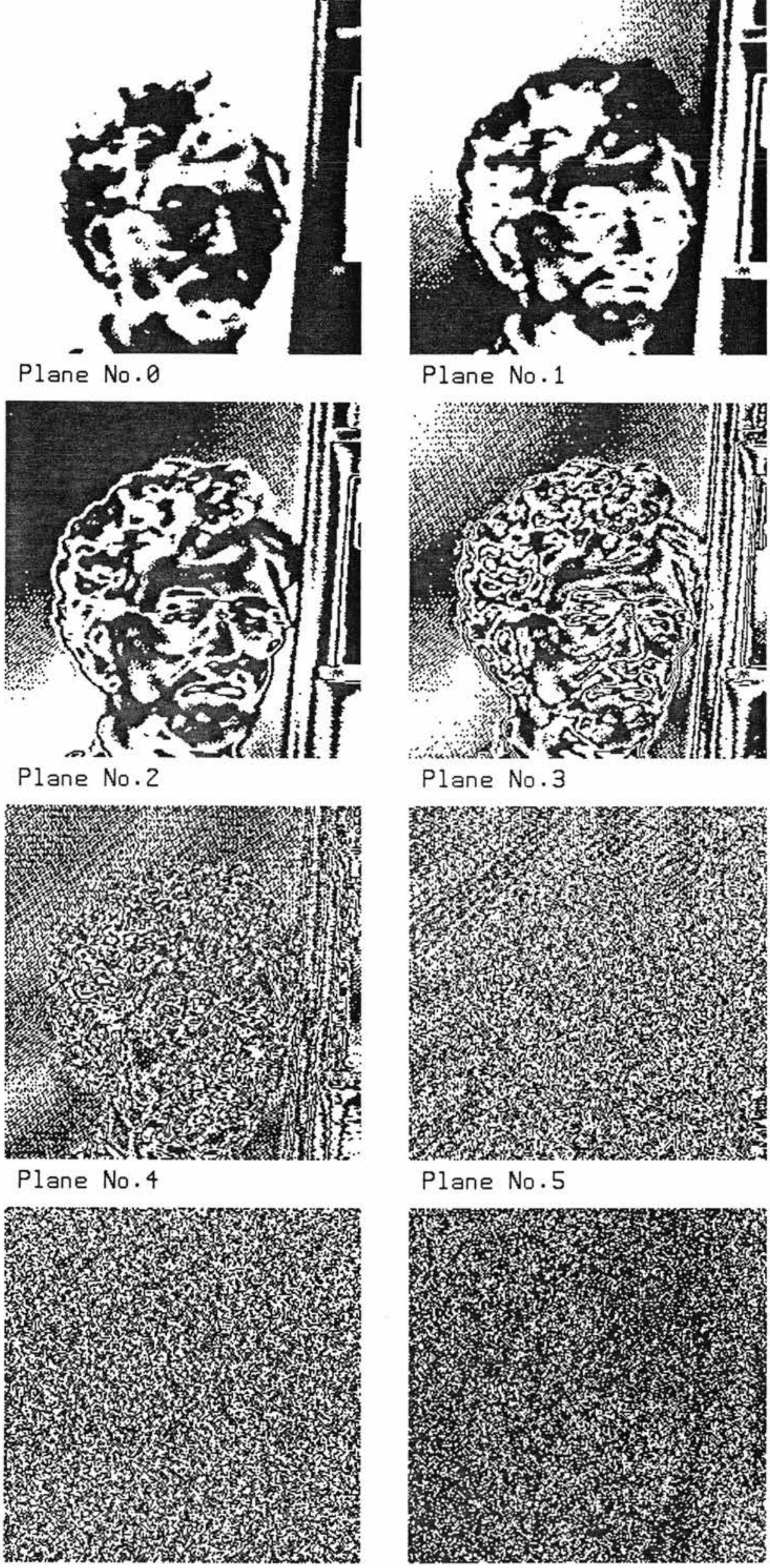


Black and white bit planes for cone model as detailed in table 4.1.

**Fig 4.2 Diagram showing three most significant bit planes for image given by cone model.**

Plane 0 in Figs 4.3-4.5 will therefore show all pixels with intensity values greater than or equal to 128 as white and others as black. It is stressed at this stage that the bit plane images are used as examples of black and white images, and as we will see later for grey scale applications there is no need to consider all eight planes. Good quality grey scale images can be obtained with 5 or 6 planes.

It was surprising to find a strong 'feeling' for the form of the subject presenting itself in plane 7 ( Fig 4.4 and 4.5 especially ) which



Plane No.0

Plane No.1

Plane No.2

Plane No.3

Plane No.4

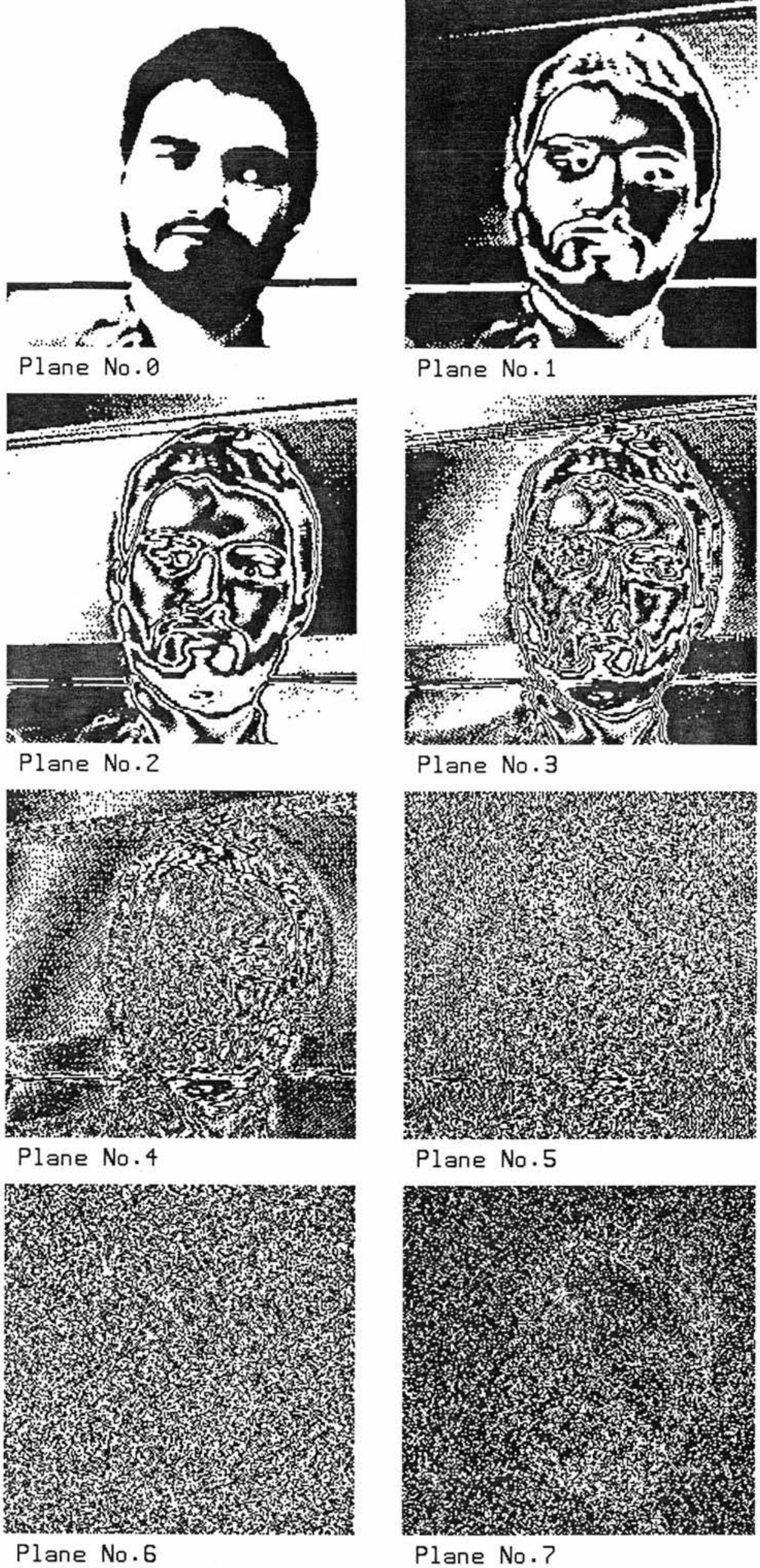
Plane No.5

Plane No.6

Plane No.7

xcoord = 130 ycoord = 190 xdim = 225 ydim = 225

Fig. 4.3 The bit planes of the grey level image *' ajc '*.



Plane No.0

Plane No.1

Plane No.2

Plane No.3

Plane No.4

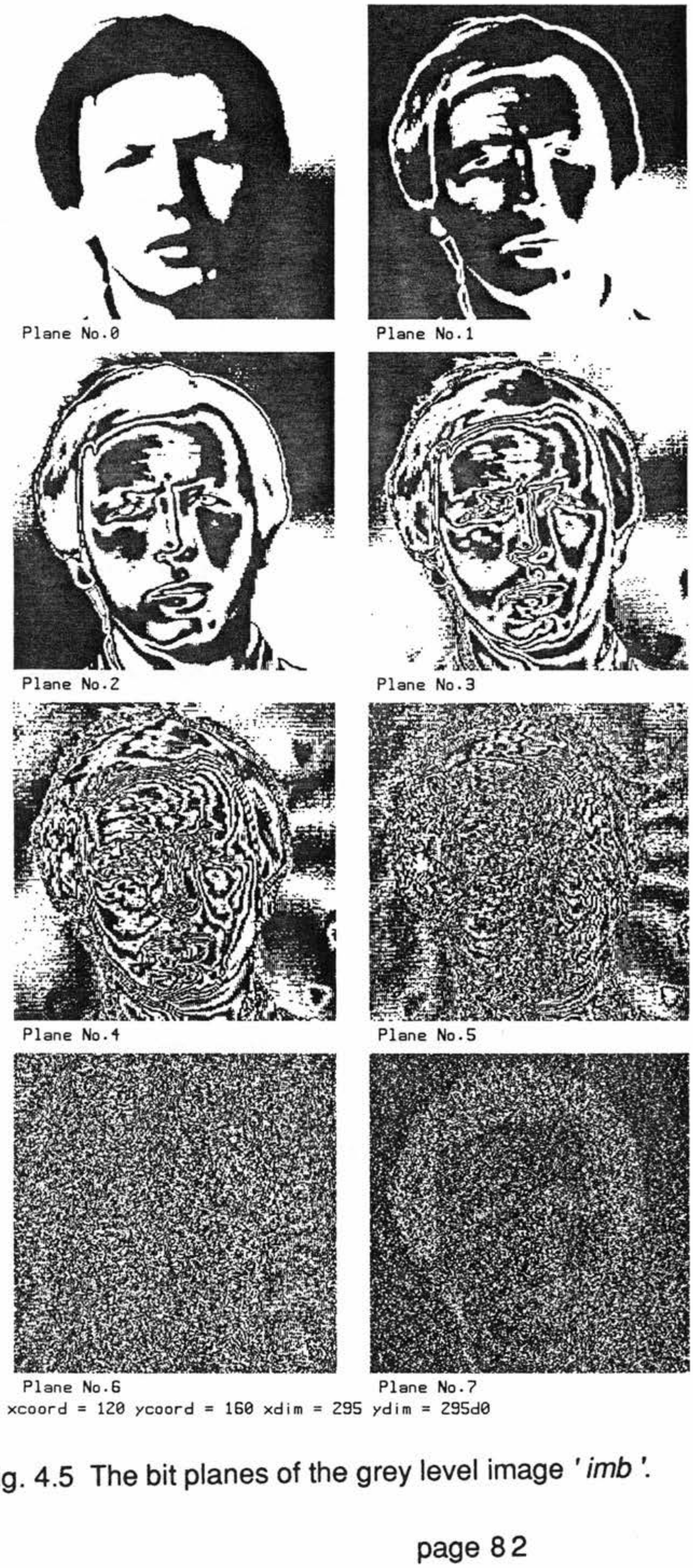
Plane No.5

Plane No.6

Plane No.7

xcoord = 130 ycoord = 190 xdim = 225 ydim = 225

Fig. 4.4 The bit planes of the grey level image *' fred '*.



Plane No.0

Plane No.1

Plane No.2

Plane No.3

Plane No.4

Plane No.5

Plane No.6

Plane No.7

xcoord = 120 ycoord = 160 xdim = 295 ydim = 295d0

Fig. 4.5 The bit planes of the grey level image *' imb '*.

corresponds to the least significant bit. The pattern in this plane should reflect only whether the intensity value was even or odd. This generated much discussion, with the best explanation being inaccuracies of the analog to digital converter. The accuracy of the least significant bit is suspect and in tests carried out it was found to be heavily biased to producing even intensities values with the ratio of even to odd values increasing rapidly as the intensity value increased.

Subimages can easily be clipped from any of the above images for test purposes.

### 4.1.1 Image storage

In general the images for investigation are stored in one of two forms.

a) As PS-algol [50, 59] data type **image**, written as **#pixel**, held in a persistent store. The image is now a three dimensional object made up of a rectangular grid of pixels. Pixels have a depth to reflect the number of planes they possess and images have an X and Y dimension to reflect their size. In its most degenerate form a pixel is one spot which is either **on** or **off**. For example

**let** one.pixel = **on**

creates a pixel *one.pixel* with depth 1. A pixel with depth 8 could be formed by

**let** another.pixel = **on** & **on** & **off** & **off** & **on** & **on** & **on** & **off**.

To form an image with dimensions greater than 1x1 the following

**let** rect.image = **image** 3 **by** 9 **of on** & **on** & **off** & **on**

creates an image *rect.image* of depth 4 with 3 pixels in the X direction and 9 in the Y direction.

The standard function **Pixel** allows an image to be interrogated on a pixel basis. The arguments given are image and position, thus

**Pixel**( rect.image , 2, 3 )

returns the pixel at position 2,3 of image rect.image, namely **on** & **on** & **off** & **on**. This function is used in the case of black and white images of depth 1 to examine whether a pixel is black or white corresponding to **off** or **on.** This is illustrated by the program lines in Fig. 4.6.

b) The image captured by the frame grabber is stored in a file with a header holding various information on the image followed by intensity values for each pixel in linear scan order.

### 4.2 Run Lengths from Images

The run lengths are found by scanning the image using the murray scan fast algorithm of section 2.2.4.b and as the scan proceeds a count is kept of the number of pixels of the same value. When a change occurs the value of count is output to a file and count reset to 1. The run lengths are then available in file, along with the radices and order of the scan which have already been stored, for future analysis and manipulation. The procedures for the scan are those given on page 60 and only the new program lines required for output are given in Fig. 4.6.

```
let x1 := 0; let x := 0; let y1 := 0; let y := 0
! oldpoint and newpoint are vectors of integers holding the coordinates
if Pixel( image,x,y ) = off do output out.file, "0"     ! first pixel black
for point = 1 to no.of.points - 1 do
begin
    newpoint := next.point( murray,radices,parities, oldpoint )
    x := newpoint( 1 )
    y := newpoint( 2 )

    if Pixel( this.image,x,y ) = Pixel( this.image,x1,y1 )
        then count := count + 1
    else { output out.file, count ; count  := 1 }

    x1 := x ; y1 := y
end
```

**Fig 4.6 Program lines to output run lengths from an image**

### 4.3 Run Length Distributions

Having scanned the images using murray polygons the next step was to look at the run length distribution. The maximum run length would occur when the image was either all black or all white. For the images considered the maximum was rarely above 5000 and usually 80% of the runs were of length 100 or less. At this stage a very rough idea of the distribution was all that was required as final detailed compression figures would be calculated after coding. Initially a simple printout of data with information on run length, frequency, cumulative frequency and percentage of total run lengths was produced. A pictorial view of this in a histogram with cumulative frequency superimposed is shown in Fig. 4.7 and 4.8.

Fig. 4.7 shows the distribution for the *'world'* images from fig 4.1. There is little obvious difference in the overall pattern.

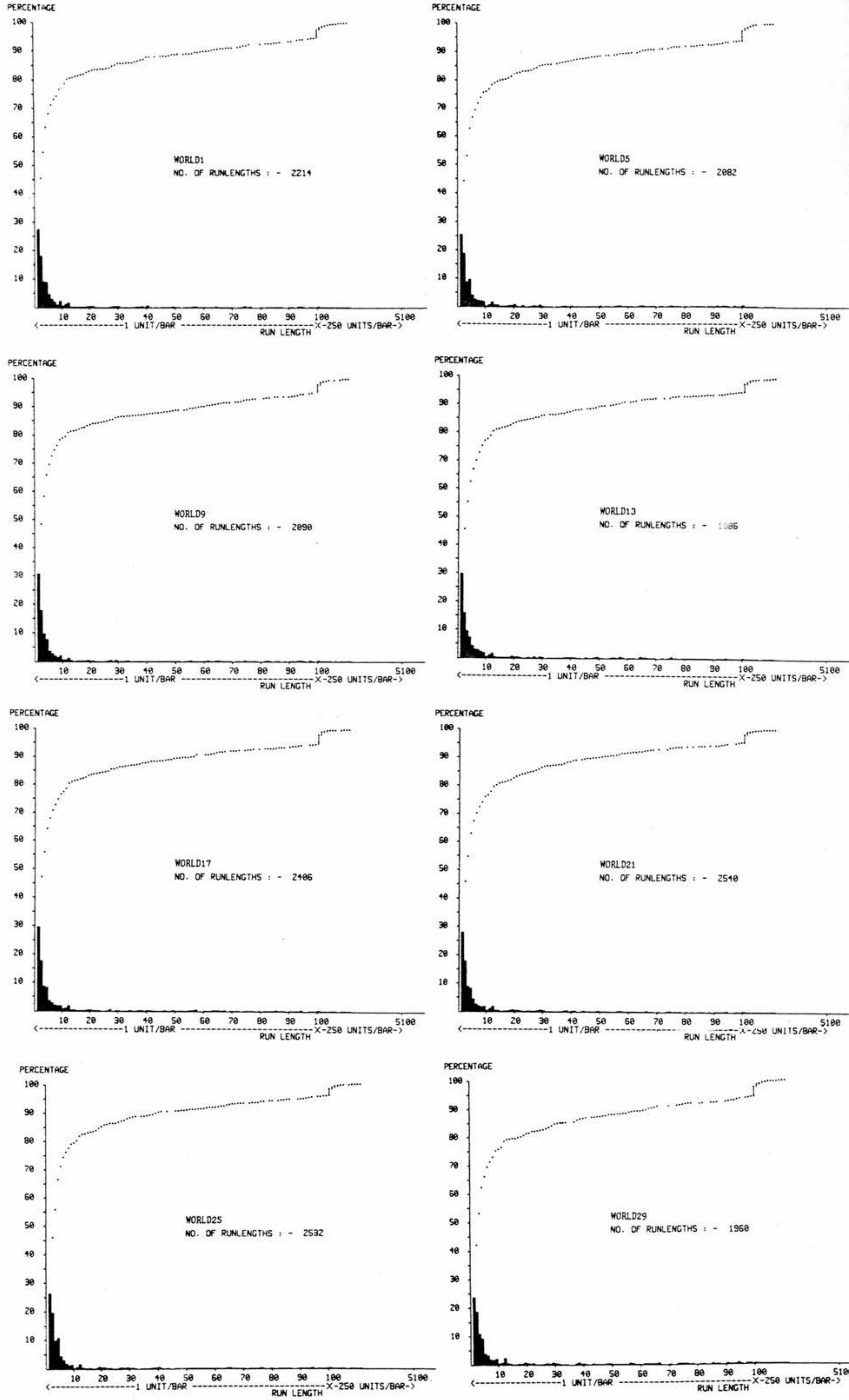


Fig 4.7 The run length distributions of Fig. 4.1 images.

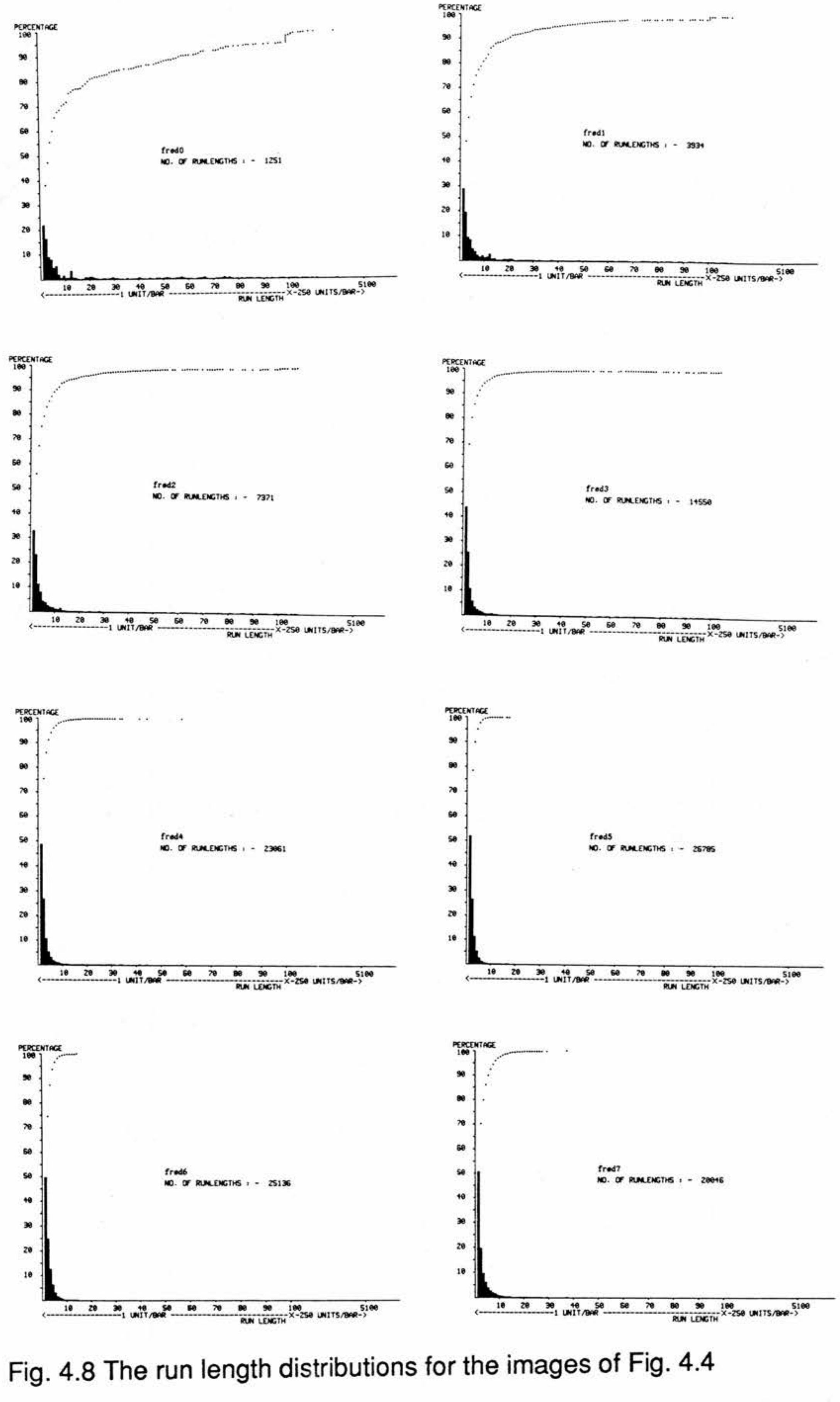


Fig. 4.8 The run length distributions for the images of Fig. 4.4

However, on close examination of the number of runs, percentages of short runs and the detailed printout the larger number of runs and higher percentages of small runs can be clearly seen to correspond to the images with a more fragmented pattern, e.g. Fig. 4.1.6, and longer runs are to be found in those images with large homogeneous areas Fig. 4.1.8.

Fig. 4.8, the distributions for the *'fred'* images clearly show the sharp increase in the number of run lengths and corresponding decrease in long runs as the image becomes very fragmented as we progress from the most to least significant planes, with the latter being less susceptible to compression.

From these it looked as if the distribution in many cases was promising and that the coding strategy considered could be fruitful.

### 4.4 The Compression

The compression was to be measured relative to the bitmap. In other words one bit per pixel for the black and white images. The coding used was 'continuation-bit' coding as described in section 3.4. It is possible to consider varying either the **order** of the scans applied, and for any given order the values of the radices themselves, or the arrangement of continuation and information bits within the coding. Initially the order and radix values are held constant for a given set of images with the proviso that the radices increase in magnitude from least to most significant position. This gives a tighter scan in terms of remaining in the same local area for a longer period. Various coding bit arrangements are investigated briefly.

The compression is given in two forms defined as follows

$$\text{compression ratio} = \frac{\text{total number of bits for bitmap at 1 bit/pixel}}{\text{number of bits in compressed form}}$$

$$\text{percentage of bitmap} = 100 / \text{compression ratio.}$$

The results for the *'world'* images given in Fig. 4.1 gave compression ratios from 8 to 6.4 using the radices and coding pattern given in Table 4.2.

| image | radices | code pattern | comp<br>ratio | % of<br>bitmap | no.of<br>runs | no.of<br>points |
|---|---|---|---|---|---|---|
| world1 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 7.2 | 13.8 | 2214 | 74529 |
| world5 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 7.5 | 13.4 | 2082 | 74529 |
| world9 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 7.8 | 12.8 | 2090 | 74529 |
| world13 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 8.0 | 12.5 | 1986 | 74529 |
| world17 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 6.7 | 14.9 | 2406 | 74529 |
| world21 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 6.4 | 15.7 | 2540 | 74529 |
| world25 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 6.6 | 15.1 | 2532 | 74529 |
| world29 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 7.8 | 12.8 | 1960 | 74529 |

**Table 4.2 Compression details for images of Fig. 4.1**

The coding pattern was varied with the radices held constant to determine the effect on the compression. The results with the *'world'* images and others tested showed that the the pattern given in Table 4.2 was rarely improved on. Some typical results for the image of Fig 4.1.1 'world1' with various coding patterns are given in Table 4.3.

| image | radices | code pattern | comp<br>ratio | % of<br>bitmap | no.of<br>runs | no.of<br>points |
|---|---|---|---|---|---|---|
| world1 | 13 13 7 7 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 7.2 | 13.8 | 2214 | 74529 |
| world1 | 13 13 7 7 3 3 | 2 2 2 2 2 2 2 2 2 2 2 2 | 6.9 | 14.4 | 2082 | 74529 |
| world1 | 13 13 7 7 3 3 | 3 3 3 3 3 3 3 3 3 3 3 3 | 6.2 | 16.0 | 2090 | 74529 |
| world1 | 13 13 7 7 3 3 | 4 4 4 4 4 4 4 4 4 4 4 4 | 5.6 | 17.9 | 1986 | 74529 |
| world1 | 13 13 7 7 3 3 | 1 1 2 1 3 5 2 2 2 2 2 2 | 7.0 | 14.3 | 2406 | 74529 |
| world1 | 13 13 7 7 3 3 | 1 1 1 3 4 5 2 2 2 2 2 2 | 7.0 | 14.2 | 2540 | 74529 |
| world1 | 13 13 7 7 3 3 | 0 1 1 2 2 3 3 4 4 5 5 5 | 6.8 | 14.7 | 2532 | 74529 |

**Table 4.3 The compression details when the coding pattern is varied for the image of Fig. 4.1.1 ' world1'.**

The results at this stage were encouraging as applications planned for grey scale images and frame sequences for moving pictures may be concerned with images whose inherent structure would compress better than those tested. With this in mind attention was now focused on the images grabbed by the frame grabber. The opportunity was also taken to briefly assess the effect of increasing the order of the scan from three to four while maintaining the same lower order radices. The results for the planes of Fig. 4.1.3 *'fred'* are given in Table 4.4. From these it can be clearly seen that the compression ratio drops off rapidly as the image becomes more fragmented. The increase in order had almost no effect on the compression figures indicating as suspected that the first two tile levels were of paramount importance in this case.

| image | radices | code pattern | comp ratio | % of bitmap | no.of runs | no.of points |
|---|---|---|---|---|---|---|
| fred0 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 8.0 | 12.5 | 1251 | 50625 |
| fred1 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 3.2 | 31.0 | 3934 | 50625 |
| fred2 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 2.0 | 49.9 | 7371 | 50625 |
| fred3 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 1.2 | 82.3 | 14550 | 50625 |
| fred4 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | - | 117.3 | 23061 | 50625 |
| fred5 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | - | 129.7 | 26785 | 50625 |
| fred6 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | - | 126.2 | 25136 | 50625 |
| fred7 | 15 15 5 5 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | - | 109.2 | 20046 | 50625 |
| order increased to 4 | | | | | | |
| fred0 | 5 5 5 5 3 3 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 8.1 | 12.3 | 1249 | 50625 |
| fred1 | 5 5 5 5 3 3 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 3.2 | 31.0 | 3936 | 50625 |
| fred2 | 5 5 5 5 3 3 3 3 | 1 1 1 1 1 1 1 1 1 1 1 1 | 2.0 | 49.9 | 7359 | 50625 |

**Table 4.4 Compression details for images 'fred' from Fig. 4.3.**

These results for fred4 - fred7 are given for comparison only, obviously these would be more efficiently transmitted by the bitmap itself. Full analysis of the effect of variation of the coding pattern and the radices is beyond the scope of the present work.

## 4.5 Irreversible coding by short run length merging.

The compression ratios while encouraging can be improved dramatically by the removal of short run lengths by merging them into the surrounding run lengths as described in section 3.2. The trade-off is increase in compression ratio against quality of image. The resulting *'ajc'* images formed as run lengths of length 1, 3 and 5 are removed are given in Fig. 4.10 and the *'fred'* images for run lengths 1 and 3 removed in Fig 4.11. Note the run lengths are removed successively. Thus for removal of short runs of length 5 it can be assumed that those of 1, 2, 3 and 4 have already been merged. The alternative to removing the run lengths successively is to remove them simultaneously. Simultaneous removal gives rise to more changes of colour and less acceptable deterioration in the image structure, as illustrated in [57]. This is seen in the simple example below where run lengths up to 5 are removed.

| | black | white | black | white | |
|---|---|---|---|---|---|
| Original sequence .... | 6 | 5 | 1 | 7 .... | |
| Successive removal ... | 6 | 13 ........ | | | 1 pixel changed |
| Simultaneous removal ... | 12 | 7 ......... | | | 5 pixels changed |

The images used are the first 5 bit planes for the images of *'ajc'* and '*fred*' from Fig. 4.3 and 4.4. The resulting images show a deterioration as longer run lengths are merged and as the plane considered becomes less significant. Both of these are expected but the resulting images are still easily recognisable and will be acceptable for many applications. The deterioration, as predicted in the discussion of section 3.2, is clearly seen to be the removal of small isolated areas and around the boundaries of the large homogeneous areas. This means that the main structures of the image are still clear in images typified by the planes 0 - 3. Table 4.5 gives a summary of the resulting increase in compression from run length removal for Fig 4.10 planes 0 - 3. The radices used are 15 15 5 5 3 3 and code pattern 1 1 .........1 1 1.

As can be seen clearly from Table 4.5 this reduced the amount of data required to approximately one third of the original for planes 0-3.

| image | comp ratio | % of bitmap | no.of runs | image | comp ratio | % of bitmap | no.of runs |
|---|---|---|---|---|---|---|---|
| ajc0 | 4.7 | 21.5 | 2436 | ajc1 | 2.7 | 37.4 | 5262 |
| ajc0rem1 | 6.6 | 15.2 | 1316 | ajc1rem1 | 4.5 | 22.3 | 2166 |
| ajc0rem3 | 9.1 | 10.9 | 684 | ajc1rem3 | 6.8 | 14.6 | 990 |
| ajc0rem5 | 10.4 | 9.6 | 560 | ajc1rem5 | 8.2 | 12.1 | 764 |
| ajc2 | 1.9 | 52.3 | 7593 | ajc3 | 1.1 | 89.3 | 15390 |
| ajc2rem1 | 3.1 | 32.1 | 3455 | ajc3rem1 | 2.2 | 46.1 | 5620 |
| ajc2rem3 | 4.8 | 20.7 | 1565 | ajc3rem3 | 3.6 | 27.5 | 2378 |
| ajc2rem5 | 5.9 | 17.0 | 1215 | ajc3rem5 | 4.8 | 21.0 | 1614 |

**Table 4.5 Compression details for images from 'ajc' Fig. 4.7.**

The main procedure used to merge the short run lengths is given in the listing on page 95. The original run lengths are stored in the file *in.file* and are read into variables *first ,second* and *third* for processing. The boolean *more* catches the end of input data. The length of run length to be merged is input via the parameter *remove* and the new run lengths output to the file *out.file*. The language is again PS-algol [50].

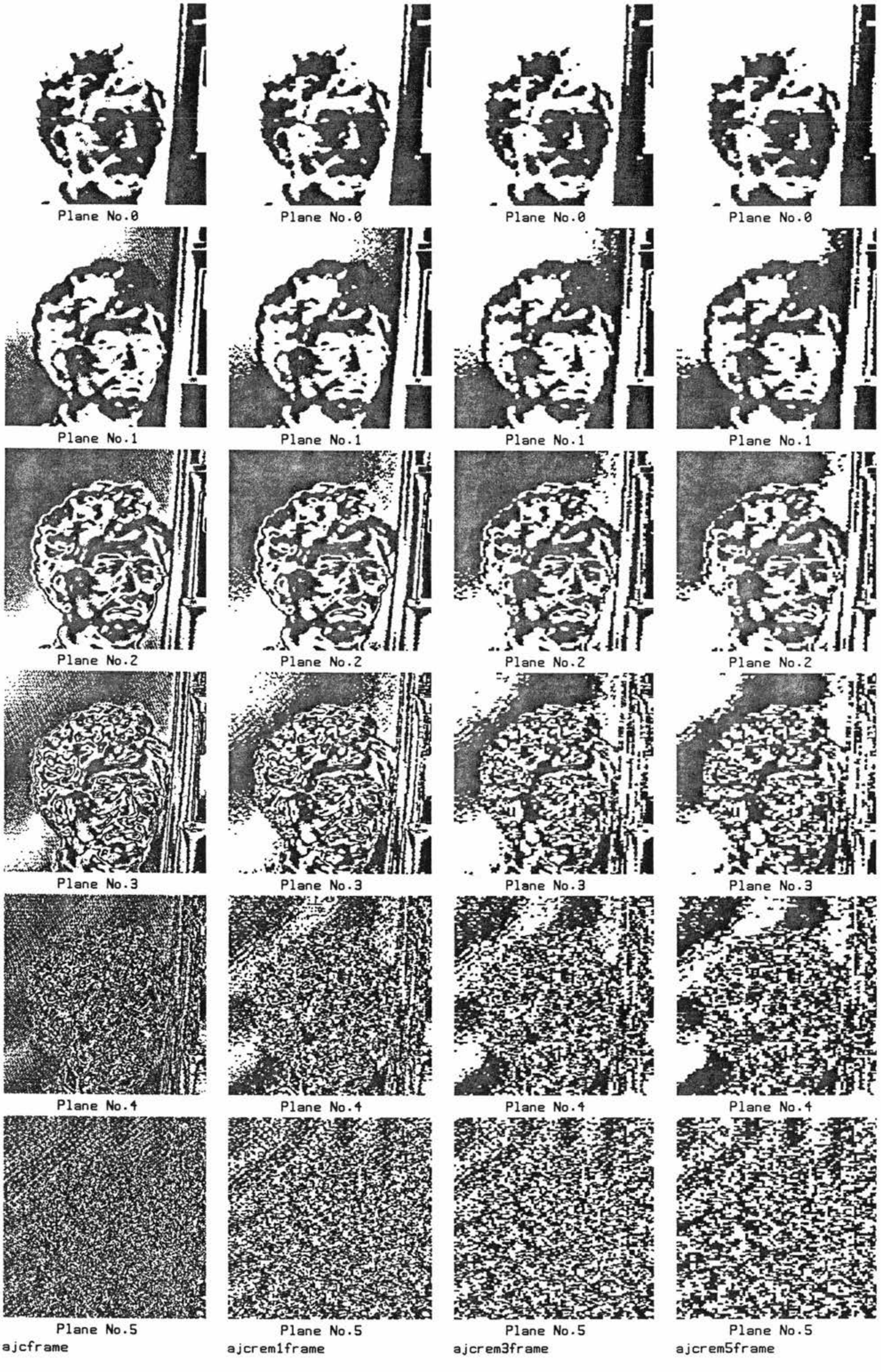


**Fig. 4.10** '*ajc* ' images with a) original, b) runs of length 1 removed, c) runs up to length 3 removed and d) runs up to length 4 removed.

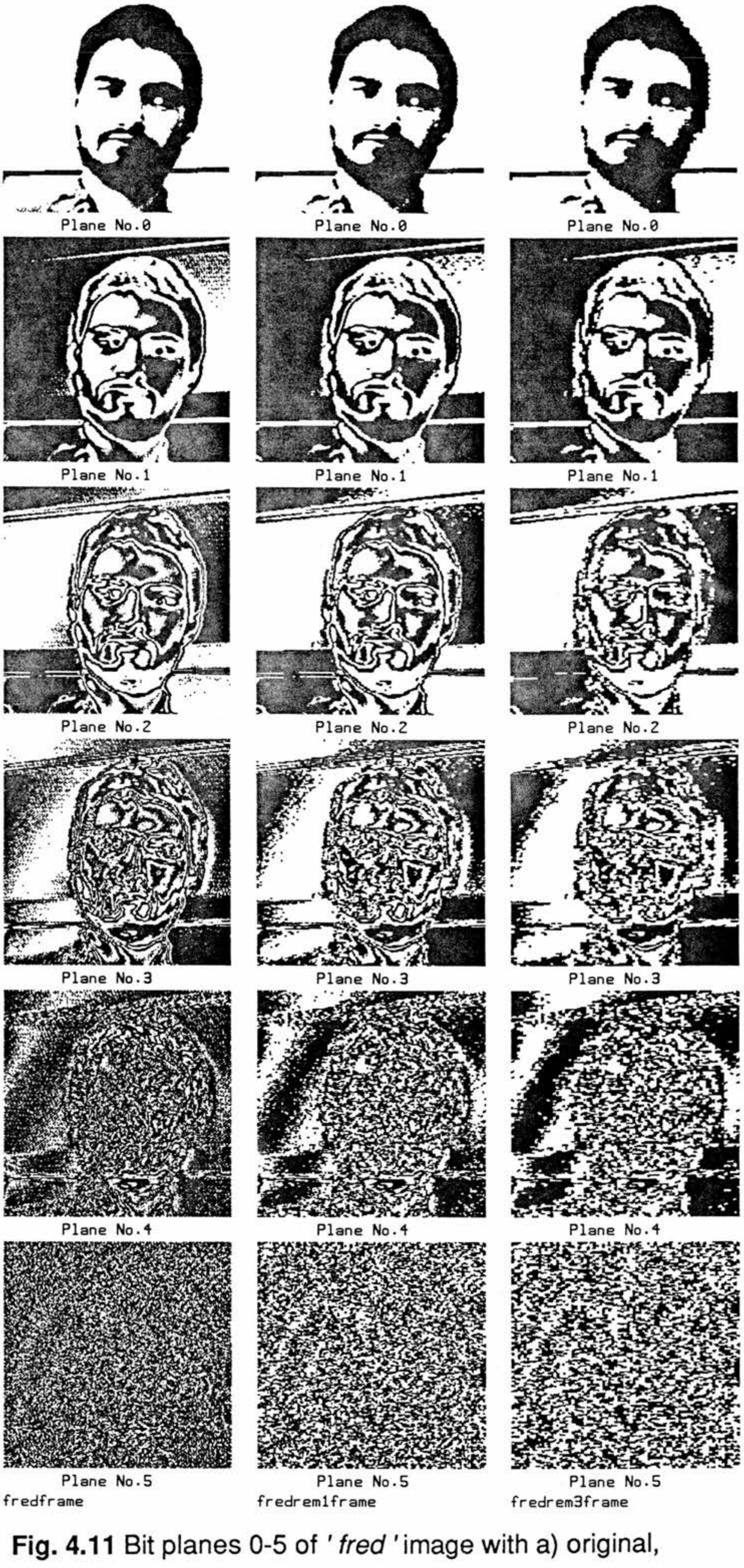


**Fig. 4.11** Bit planes 0-5 of *' fred '* image with a) original, b) runs of length 1 removed and c) runs up to length 3 removed.

```
let compact = proc( cfile in.file,out.file; int remove )
begin
        let complexity = readi( in.file )
        output out.file, complexity
        for i = 1 to 2 * complexity do
                        output out.file, readi( in.file )

        let more := true                ! boolean to test for more input data
        let first := readi( in.file )
        let second := readi( in.file )
        more := ~eoi( in.file )
        let third := -1

        if more do
        begin
                        third := readi( in.file )
                        more := ~eoi( in.file )
        end

        if first = remove do
        begin
                        second := second + remove
                        first := 0
        end

        while more do
        begin
                        if second = remove then
                        begin
                            first := first + second + third
                            second := readi( in.file )
                            more := ~eoi( in.file )
                            if more then
                            begin
                                        third := readi( in.file )
                                        more := ~eoi( in.file )
                            end
                            else   third := -1
                        end
                        else
                        begin
                            output out.file,first
                            first := second
                            second := third
                            third := readi( in.file )
                            more := ~eoi( in.file )
                        end
                        if third = -1 then                  ! exit with second as last run length
                             if second = remove then output out.file, first + remove
                            else output out.file, first, second
                        else
                            if second = remove then output out.file, first + second + third
                            else output out.file,first, second, third

        end
end
```

## 4.6 Examples of mixed exact and non-exact coded images.

As described in section 3.3 the ability to use a reversible ( exact ) coding for part of an image and an irreversible ( non-exact ) coding for the remainder could be of use in several situations. For example if the information to be sent for an exact coding of the full image is beyond the limits of the available channel then areas of the image can be approximated to reduce the data required appropriately. This may happen in the case of moving pictures if excess movement occurs. The method then becomes adaptive. Another case would be where the nature of the image is such that less importance is attached to certain areas, leading to an approximate coding being acceptable in this region - for instance the background of a 'head and shoulders' image. The murray scan technique lends itself to the coding of regions in such a manner by manipulating the run lengths away from the image.

In the implementation the new step required is to determine whether a point is in the compaction area or not . In the test implementation the compaction frame is defined by the number of rows ( *nrow* ) and the number of columns ( *ncol* ) of largest tiles that are to be removed giving a symmetrical layout as shown in Fig. 4.12. The result is an exact central area and approximate 'frame', but it is not essential that the pattern be symmetrical . Taking the *'world'* image from Fig. 4.1.1 various frame shapes and approximation levels are shown in Fig. 4.13 with details on the values of nrow and ncol and the maximum number of runs removed in non-exact area. Other than removal of small 'island' groups and the circumference of the *'world '* the main features are still easily recognisable.

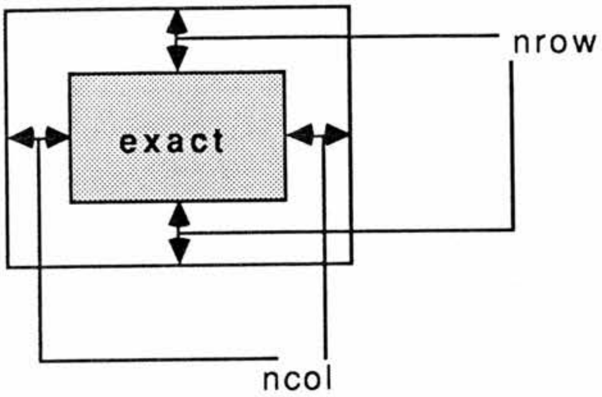
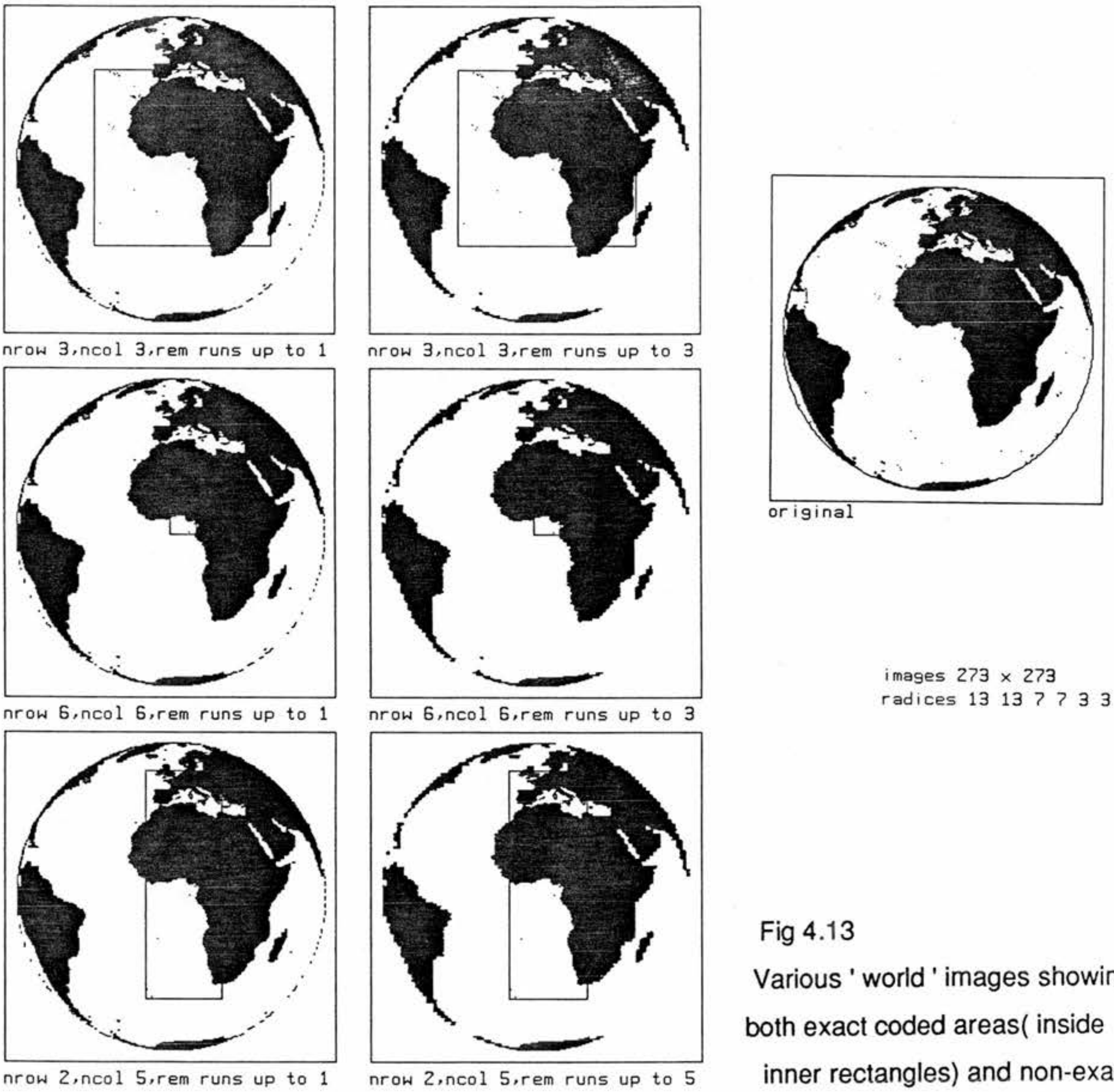


Fig. 4.12 Schematic diagram showing position of exact coded area.


nrow 3,ncol 3,rem runs up to 1

nrow 3,ncol 3,rem runs up to 3

original

nrow 6,ncol 6,rem runs up to 1

nrow 6,ncol 6,rem runs up to 3

images 273 x 273
radices 13 13 7 7 3 3

nrow 2,ncol 5,rem runs up to 1

nrow 2,ncol 5,rem runs up to 5

Fig 4.13

Various ' world ' images showing both exact coded areas( inside inner rectangles) and non-exact areas

## 4.6 Results with linear scan

The murray scan simulation of linear scanning was used to obtain data for comparison. That is, for an image of size m x n a murray scan is used with radices $r_4,r_3,r_2,r_1$ given by n, 1, 1, m. This forces the scan to move across the full width of the image before a unit change in the y direction occurs ( see chpt 2 page 58 ) resulting in a scan pattern as given below in Fig. 4.14.

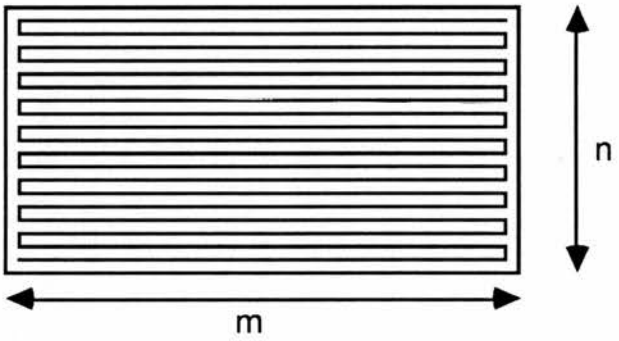


Fig. 4.14 Linear scan simulation by murray scan with radices n 1 1 m.

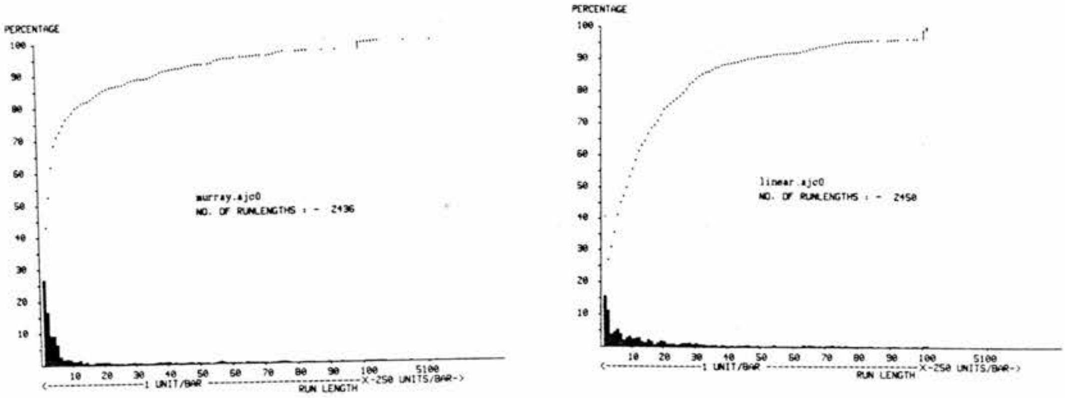
This does not have fly-back and runs are allowed to wrap round from one scanline to the next thus giving better results than systems with maximum run length limited to a scanline length.

From the images tested the compression possible for run-length sequences generated by murray scan was greater than those formed by linear scan. This was particularly true for images typified by the composition of images '*ajc*' ( planes 0 -2 in Fig. 4.3 ) as given in Table 4.12, which are those in general best suited to run-length coding. Savings of up to 30% of the linear scan compression being made. Note the number of run lengths are very similar.

| run length sequence | | % of bitmap | | increase in % of bitmap | no. of runs | |
|---|---|---|---|---|---|---|
| murray | linear | murray | linear | | murray | linear |
| ajco | Lajco | 21.5 | 28.0 | 30% | 2436 | 2450 |
| ajc1 | Lajc1 | 37.3 | 43.0 | 15% | 5160 | 5202 |
| ajc2 | Lajc2 | 52.5 | 57.1 | 9% | 7353 | 7563 |
| ajc3 | Lajc3 | 89.2 | 89.9 | 1% | 15312 | 15400 |

Table 4.12 Results for compression using murray scans and linear scans on the images *ajc* planes 0 - 3 from Fig. 4.3.

The run-length distribution was such that the murray scan, as expected, gave a larger number of long run lengths and a larger number of very short run lengths while the overall number of run lengths was only slightly different. Considering the image '*ajc*' plane 0 from Fig. 4.3 the number of run lengths from the murray scan was 2436 and from the linear scan 2450. However, the corresponding figures for runs of length 1 was 633 ( ≈ 26% ) for murray scan and 387 ( ≈ 16% ) for linear scan, and the longest run was 5670 for murray scan and 392 for linear scan. The distributions are given in Fig 4.15 and the resulting number of run lengths after removal of short runs given in Table 4.6. The reduction in number of run lengths in the murray scan data is considerably better than that for linear scan data and is reflected in the better compression results.



a) b)

Fig. 4.15 The run-length distributions for image '*ajc*' plane0

a) murray scan and b) linearscan

| run length sequence | | % of bitmap | | increase in % of bitmap | no. of runs | |
|---|---|---|---|---|---|---|
| murray | linear | murray | linear | | murray | linear |
| ajco | Lajco | 21.5 | 28.0 | 30% | 2436 | 2450 |
| rem1 | Lrem1 | 15.1 | 24.5 | 62% | 1316 | 1832 |
| rem2 | Lrem2 | 12.2 | 22.1 | 81% | 852 | 1616 |
| rem3 | Lrem3 | 11.0 | 20.6 | 87% | 610 | 1304 |
| rem4 | Lrem4 | 10.1 | 19.7 | 95% | 560 | 1170 |
| rem5 | Lrem5 | 9.6 | 18.0 | 87% | | |

Table 4.6 Figures for image *ajc* plane 0 from Fig. 4.3 showing results for murray and linear scan as short run lengths are removed.

## Summary

The results for the black and white images tested have been given both for reversible and irreversible coding. A comparison is given between murray scan and linear scan data, showing the murray scan has a significantly better performance for these images. The results, in comparison to linear scan, produce around the same number of runs but a more favourable run-length distribution which leads to better

data compression. This advantage can be around 30% improvement for many images but as the image becomes very fragmented the advantage is much less.
When the approximation is made via run length removal then it follows that the murray scan will give considerably less run lengths with a corresponding improvement in data reduction. The data used and results given are typical examples from the real world source described and not preselected to suit the scanning method or run length encoding. The results were encouraging enough to extend the work to monochrome images as reported in the next chapter.

Examples have been given which show the versatility of the murray scan in identifying different areas within the image for resolution reduction and the promising quality of the resulting images.

# Chapter 5

## Application to Grey-Level Images

In previous sections we have been concerned with black and white images. The black and white examples used have been the bilevel bit planes of a grey-level image and as such are now relevant to the following discussion which focuses on grey-level images themselves.

For monochrome images the intensity is quantised into a finite number of levels. Each pixel has a grey-level value g, which lies in the range defined by the grey-scale $0 \leq g \leq L$. The number of levels taking values such as 256, 128, 64, 32, 16, 8, 4 and 2 with the corresponding 8, 7, 6, 5, 4, 3, 2 and 1 bit representation. The 1-bit representation being the bilevel black and white image. The number of levels required varies with such things as subject matter, viewing conditions and sampling strategy but useful applications have been found with between 16 - 256 levels.

### 5.1 Bit Plane Coding

The images grabbed by the frame grabber gave 256 levels corresponding to an 8-bit representation from the A/D converter, giving a grey-level value representing intensity in the range 0 - 255. By considering the status of each of the eight bits individually a bilevel image, the bit plane, is obtained ( see section 4.1). The original grey-level image can now be considered as eight separate black and white bit planes, each of which can be compressed by any method available for bilevel images. An example of these bit planes is given in Fig. 5.1 for immediate reference, further examples are to be found earlier in Figs. 4.3 - 4.5. Clearly run length coding with any scanning technique is not appropriate for the less significant planes 5 - 7 as the patterns are very fragmented.

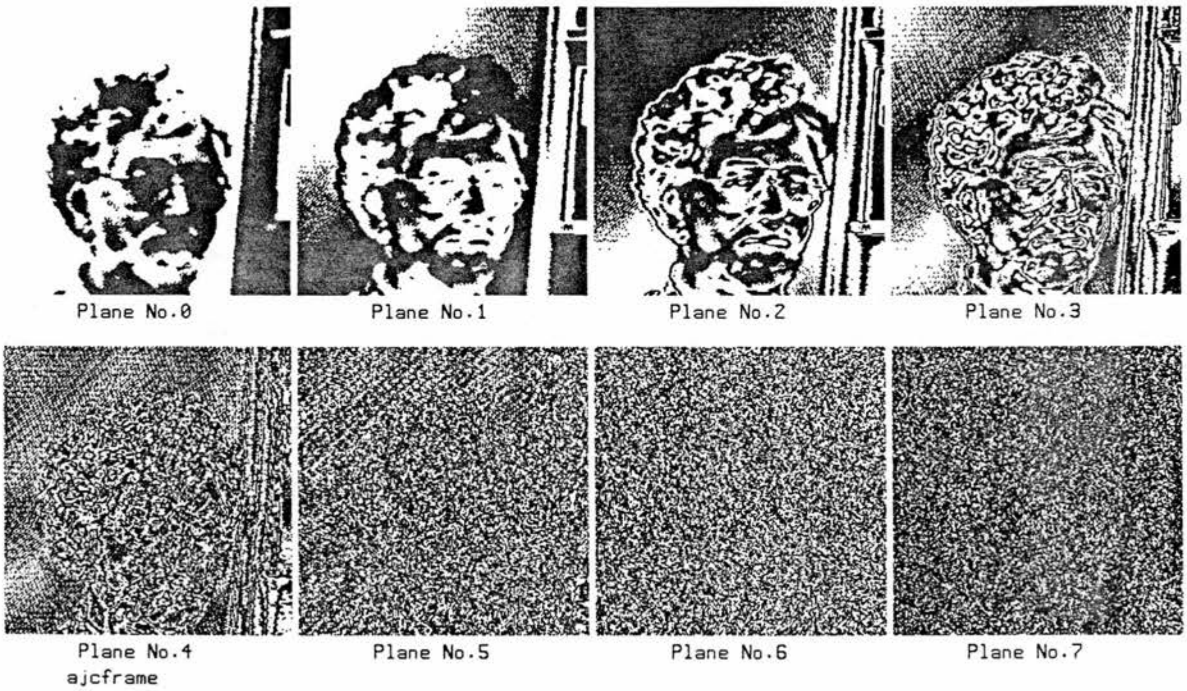


Fig 5.1 The eight bit planes for a grey-level image with 256 levels.

### 5.1.1 Reduction of grey levels

For the ' head and shoulders' images we are considering it is the nature of the subject matter that the most significant bit plane is often the most suitable for run-length coding with suitability dropping off as the less significant planes are considered. By plane 4 data reduction is small and the extremely fragmented nature of the planes 5-7 means that run-length coding rarely leads to data reduction. One obvious move is therefore to reduce the number of grey-levels by eliminating the planes of lower significance. However, the reduction must not reach a level such that contouring noise becomes unacceptable. The lower bound appears to be 16 grey-levels (4 bits ) where contouring can sometimes become a serious problem.

Contouring is the visibility of discontinuities in the image producing false contours or jumps in areas where the subject was undergoing a smooth change in intensity. This is caused by low-frequency quantisation error due to the coarse quantiser levels. The nature of the differential sensitivity of the eye means that these contours are less noticeable in areas of the image where detail exists and also in lighter areas. Fig. 5.2 gives some photographs showing the effect on a grey-scale image of considering only the more significant planes. The examples given are photographs of screen images and as such suffer from developing distortions. The reader should not attempt to read anything from these other than evidence of the contouring effect. From investigation of the reconstruction of grey-level images we found that the top five planes ( 32 levels ) gave images very close to the originals and contouring was not a problem. The results reported consider the information contained within these first 5 planes ( 0-4 ). As the main investigation is an initial assessment of the application of murray polygons to data compaction no attempt to use tapered or adaptive quantisers was made.

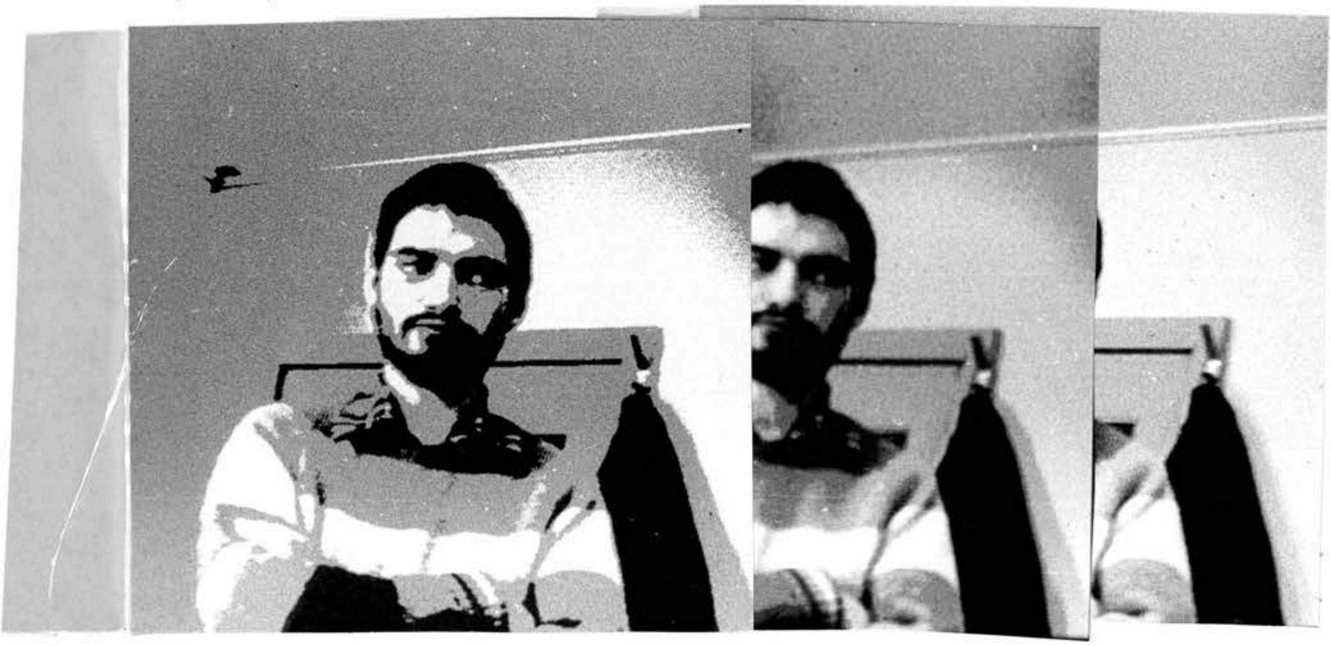


Fig. 5.2 Grey-level images showing contouring a) planes 0-1 b) planes 0-2 and c) planes 0-3.

Contouring is the visibility of discontinuities in the image producing false contours or jumps in areas where the subject was undergoing a smooth change in intensity. This is caused by low-frequency quantisation error due to the coarse quantiser levels. The nature of the differential sensitivity of the eye means that these contours are less noticeable in areas of the image where detail exists and also in lighter areas. Fig. 5.2 gives some photographs showing the effect on a grey-scale image of considering only the more significant planes. The examples given are photographs of screen images and as such suffer from developing distortions. The reader should not attempt to read anything from these other than evidence of the contouring effect. From investigation of the reconstruction of grey-level images we found that the top five planes ( 32 levels ) gave images very close to the originals and contouring was not a problem. The results reported consider the information contained within these first 5 planes ( 0-4 ). As the main investigation is an initial assessment of the application of murray polygons to data compaction no attempt to use tapered or adaptive quantisers was made.

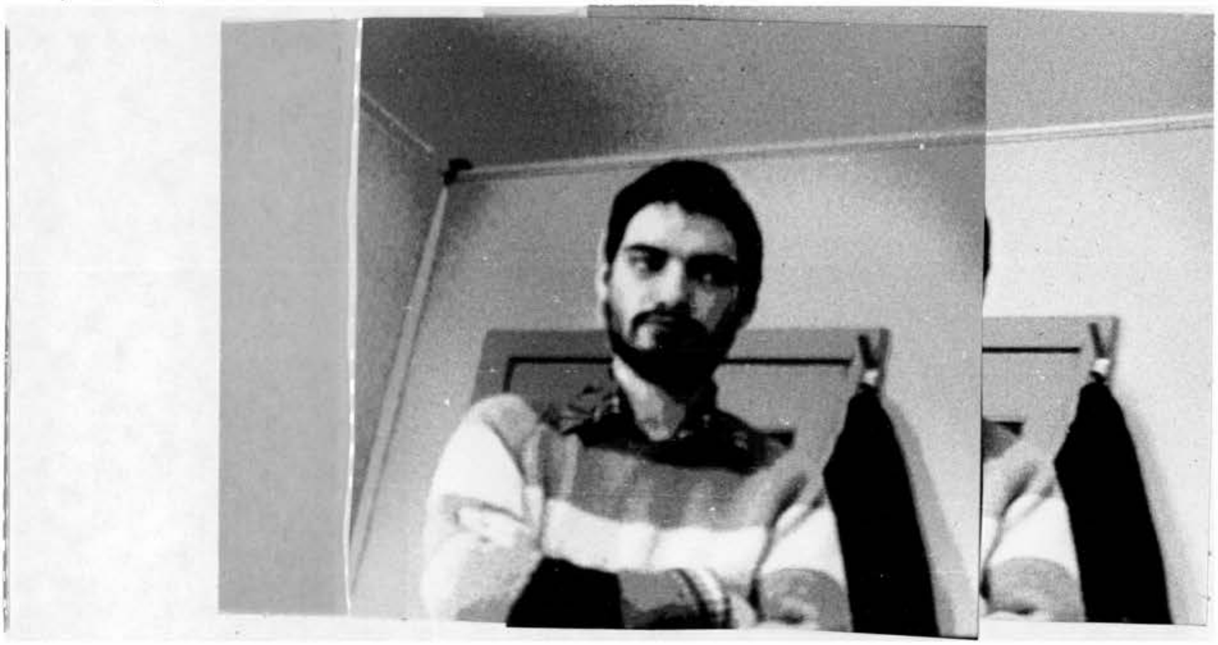


Fig. 5.2 Grey-level images showing contouring a) planes 0-1 b) planes 0-2 and c) planes 0-3.

Contouring is the visibility of discontinuities in the image producing false contours or jumps in areas where the subject was undergoing a smooth change in intensity. This is caused by low-frequency quantisation error due to the coarse quantiser levels. The nature of the differential sensitivity of the eye means that these contours are less noticeable in areas of the image where detail exists and also in lighter areas. Fig. 5.2 gives some photographs showing the effect on a grey-scale image of considering only the more significant planes. The examples given are photographs of screen images and as such suffer from developing distortions. The reader should not attempt to read anything from these other than evidence of the contouring effect. From investigation of the reconstruction of grey-level images we found that the top five planes ( 32 levels ) gave images very close to the originals and contouring was not a problem. The results reported consider the information contained within these first 5 planes ( 0-4 ). As the main investigation is an initial assessment of the application of murray polygons to data compaction no attempt to use tapered or adaptive quantisers was made.

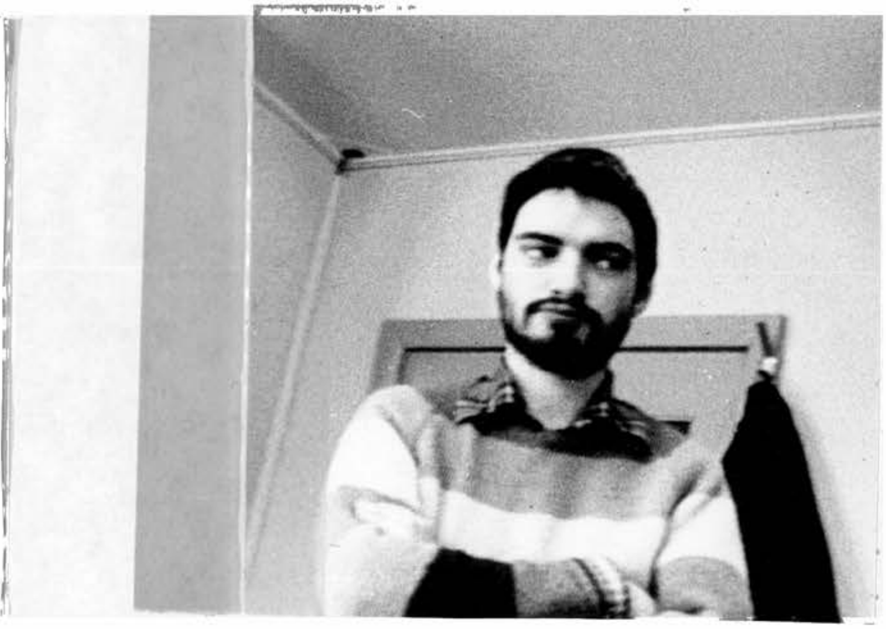


Fig. 5.2 Grey-level images showing contouring a) planes 0-1 b) planes 0-2 and c) planes 0-3.

Using the results available from the analysis of the bit planes in chapter 4 we can quickly find the average compression for exact coding of the top 4 or 5 planes of the images of Fig. 4.2 - 4.4. These are summarised for three examples in Table 5.1.

| plane | compression as % of bit map | | |
|---|---|---|---|
| | image | | |
| | ajc | fred | imb |
| 0 | 21.5 | 12.5 | 11.0 |
| 1 | 37.4 | 31.0 | 18.8 |
| 2 | 52.3 | 49.9 | 34.3 |
| 3 | 89.3 | 82.5 | 59.0 |
| 4 | 100.0 | 100.0 | 89.8 |
| average ( 0 - 4) | 60.1 | 55.2 | 42.6 |
| average ( 0 - 3 ) | 50.1 | 41.8 | 30.8 |

Table 5.1 Compression figures as percentages of bit map.

The variation in compression between images is accounted for by the variation in bit plane patterns. The image *'imb'* gives considerably better results than *'ajc'* as the bit planes in the former have larger well defined homogeneous areas leading to a more suitable distribution of run lengths. One underlying reason for this is the range and distribution of the intensity values in the original image at capture. These can be influenced by the lighting conditions. For instance if the image was made darker overall such that no values of intensity existed over 127 then the most significant bit plane would be totally black. No attempt was made to investigate possibilities of tight control on lighting conditions at image capture, but the possible effect should be noted.

The results suggested that the data could be reduced to around 50% of the original bit map data for typical images of the type described.

### 5.1.2 Short run length removal

As described in chapter 4 (section 4.5) the technique of merging short run lengths into the surrounding run lengths offered opportunities for further compression. Dramatic improvement was available using this technique but whether the resulting distortion would be too severe or not was not clear. To test this run lengths up to and including length 5 were successively removed and the resulting grey-level images assessed. The assessment at this stage was simply to subjectively test if the resulting images looked acceptable to the human viewer. This raises the questions. Which human viewer ? In which circumstances ? There is a very wide range of possible applications where the circumstances and expectations of the human viewer differ. Such variation covers images of almost T.V. quality, situations where it is sufficient to recognise a face and simple assembly line object identification. Therefore it seems legitimate to leave this for later study and investigation.

We were pleasantly surprised at the quality of the resulting images even with run lengths of length 5 removed. Fig. 5.3 gives a photographic <u>impression</u> of the grey-level screen images. The screen images themselves were of considerably better quality. Some of the compression figures for these images are given in Table 5.2.

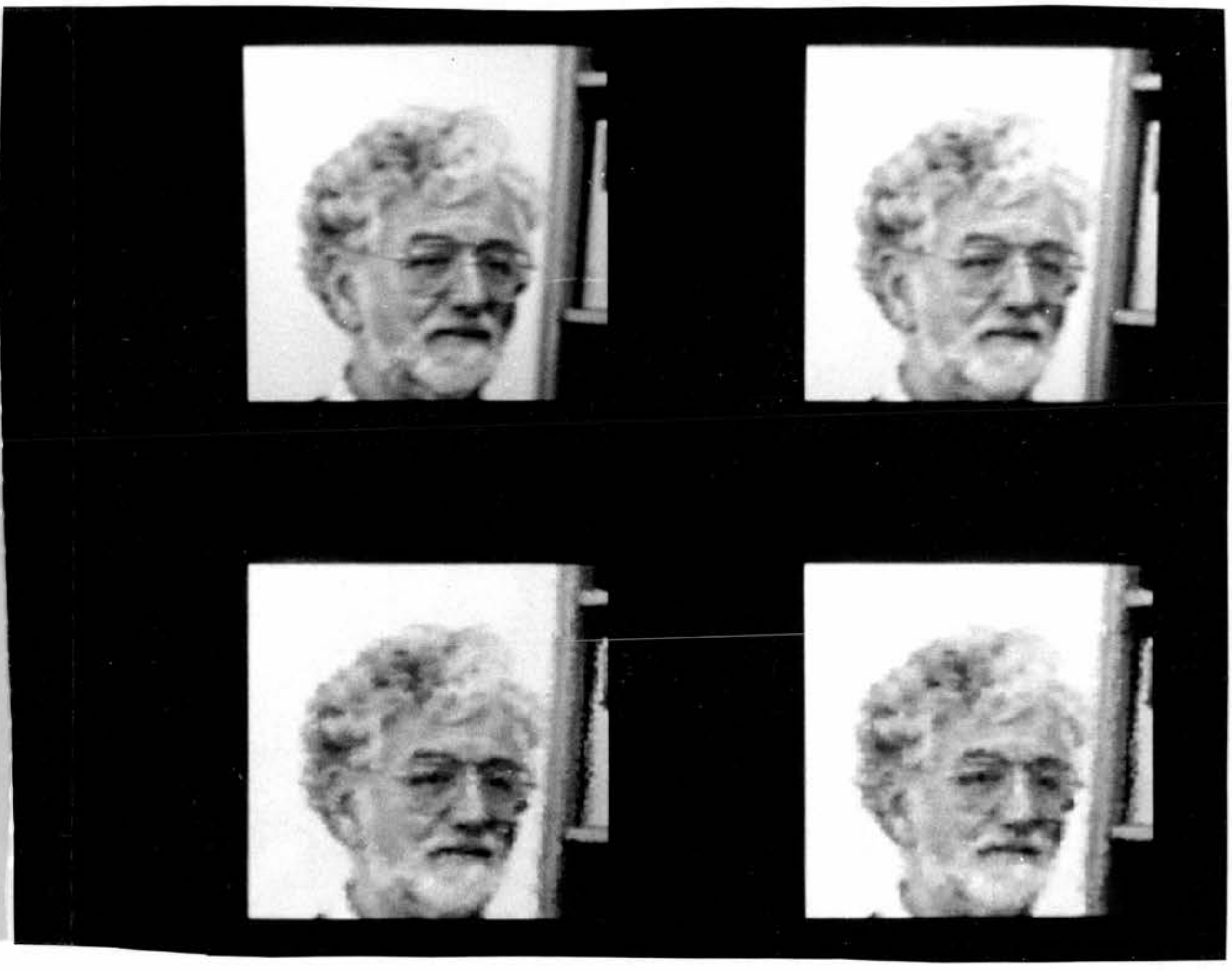


Fig. 5.3 Images showing the effect of the removal of short run lengths. From left to right and top to bottom a) original, b) runs up to 1, c) runs up to 3 and d) runs up to 5 removed

By removing run lengths of length 1 the data for the 5 planes is reduced by around 40% of the exact coded data, columns *'ajc'* and *'fred'* in Table 5.2, with relatively little distortion in the image. The reduction for each plane varying from approximately 25% for the most significant plane to almost 50% for plane 4. This variation is explained by the run length distribution from the respective bit planes.

| plane | compression as % of bit map | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | image | | | | | | | |
| | ajc | rem1 | rem3 | rem5 | fred | rem1 | rem3 | rem5 |
| 0 | 21.5 | 15.1 | 10.9 | 9.6 | 12.5 | 9.3 | 7.0 | 6.0 |
| 1 | 37.4 | 22.3 | 14.6 | 12.2 | 31.0 | 21.3 | 14.0 | 11.7 |
| 2 | 52.3 | 32.0 | 20.6 | 17.0 | 49.9 | 29.6 | 19.2 | 15.3 |
| 3 | 89.3 | 46.1 | 27.4 | 21.0 | 82.5 | 41.0 | 24.4 | 18.5 |
| 4 | 100.0 | 59.3 | 34.9 | 25.8 | 100.0 | 53.5 | 30.7 | 22.7 |
| average ( 0 - 4) | 60.1 | 35.0 | 21.7 | 17.1 | 55.2 | 30.9 | 19.0 | 14.8 |
| average ( 0 - 3 ) | 50.1 | 28.9 | 18.4 | 15.0 | 41.8 | 25.3 | 16.2 | 12.9 |

Table 5.2 Compression figures planes 0-4 as % of bit map

With the fragmented patterns increasing from plane 0 -4 there was a corresponding increase in the number of short run lengths. Overall for the exact case the compression ratio is around 2 ( 2.5 bits/pixel or 50% of the bit map ) and in the case of run lengths up to length 5 merged the compression ratio is about 6-7 ( 0.6 bits/pixel or 13 % of the bit map ).

It is important to note that the method of varying the resolution through run length removal in specific areas, as described in section 3.3, can easily be applied to grey-level images. These areas are easily identified by using the murray digits and suitable radix pairings to select appropriate tiles.

### 5.1.4 Gray code representation of intensities

Kunt & Johnsen [33] suggested that for grey-level images to be coded by run-length coding the bit planes, it may be an advantage to consider the intensities stored by Gray [51] code values (see section 2.2.1 ). The advantage here is that the nature of the black and white bit plane images are changed giving larger homogeneous areas more suitable for run-length coding. This was done for several images and indeed the resulting bit planes were more promising in composition. The original and Gray code patterns for the image *'ajc'* are given in Fig. 5.3. Considering the first 5 planes the data is reduced from 60% to 40% of the bitmap, the details are given in Table 5.3. This is a promising improvement and had it been realised earlier in the investigations would have been given more prominence. It should be noted and given careful consideration for future work.

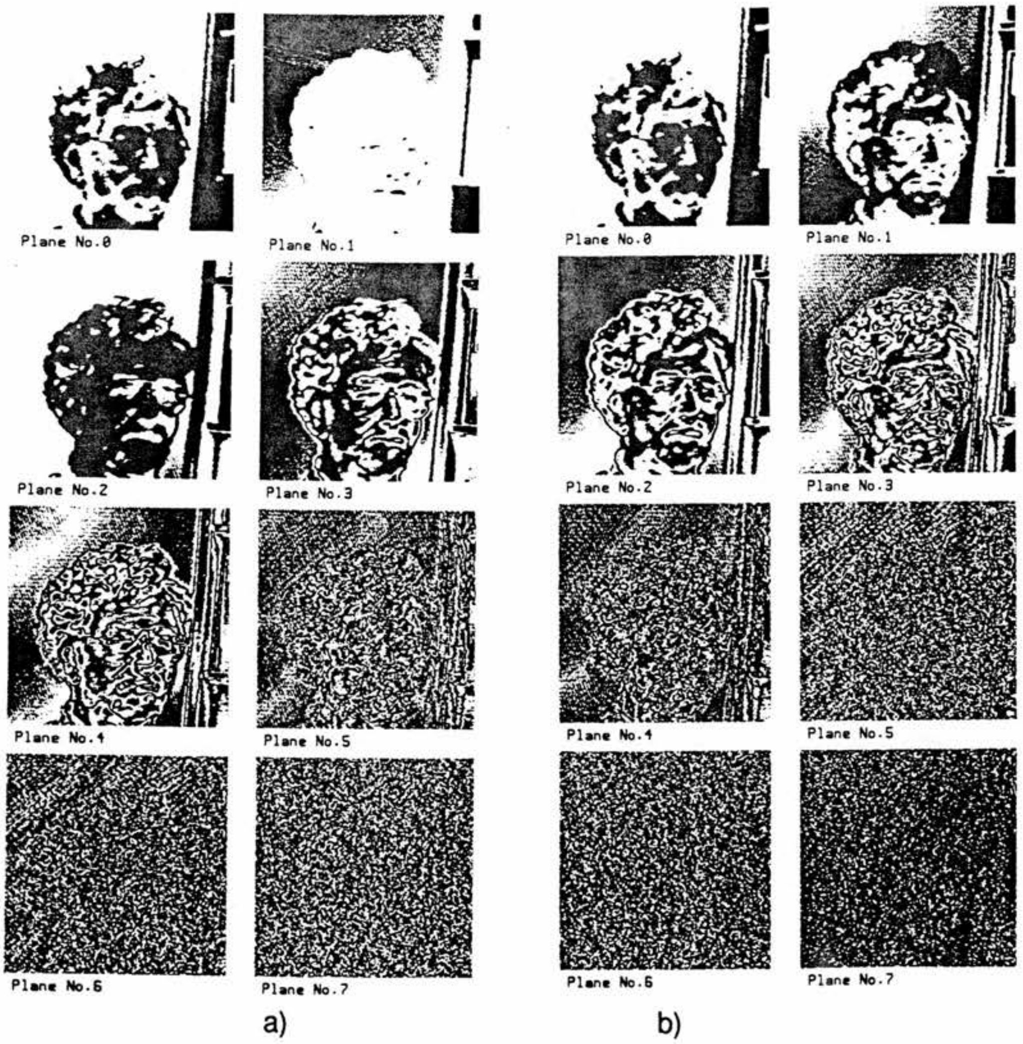


Fig. 5.3 The bit planes for a) intensities stored as Gray codes and b) intensities stored as pure numbers.

| plane | compression | | | |
|---|---|---|---|---|
| | image | | | |
| | ajc | | gray ajc | |
| | % of bit map | compression ratio | % of bit map | compression ratio |
| 0 | 21.5 | 4.7 | 21.5 | 4.7 |
| 1 | 37.4 | 2.7 | 17.5 | 5.7 |
| 2 | 52.3 | 1.9 | 21.6 | 4.6 |
| 3 | 89.3 | 1.1 | 54.6 | 1.8 |
| 4 | 100.0 | 1.0 | 84.6 | 1.2 |
| average ( 0 - 4) | 60.1 | 1.7 | 40.0 | 2.5 |
| average ( 0 - 3 ) | 50.1 | 2.0 | 28.8 | 3.5 |

Table 5.3 Compression figures comparing intensities stored in pure number form with those in Gray code.

## 5.2 Coding of grey-level images by intraframe pixel differences

The work reported in this paper considers the coding of the grey-level value by taking each bit plane separately, along with continuation bit coding. However, other well know intraframe methods concerned with pixel differences used in conjunction with a murray scan are relevant and are discussed briefly at this stage.

### 5.2.1 Shift coding the differences

In the exact coding of grey level images such as the images from satellite monitoring of the Earth an equal length code requires 7 bits for images with 128 grey levels. It has been found, Gonzalez & Wintz [9], that many of the possible grey levels did not have a high occurrence. This led to the possibility of coding the difference in grey level between adjacent pixels. Namely an image with n pixels described by the sequence of grey levels $g_1,g_2,g_3$ ... $g_i$,........ ,$g_n$ can be described by the new sequence $g_1,g_2$-$g_1,g_3$-$g_2$,..... $g_i$ - $g_{i-1}$,......,$g_n$ - $g_{n-1}$. This now means that there are 256 possible differences from -127 to +127, which requires 8 bits for an equal length code. However, when the distribution of the differences is investigated usually most of the differences lie within a relatively small range of values, typically -8 to +8. The differences are also found to be highly peaked about zero. Thus if a variable length code is used in conjunction with a double shift ( see section 1.5.1 ) for values out of range then considerable saving should be possible. For instance with code words available c1,c2.......c15,c16 then c2-c15 could be used to code the range -7 to +6 and c1 to signify a shift below -7 and c16 a shift above 6. Examples are given in Fig 5.4 .

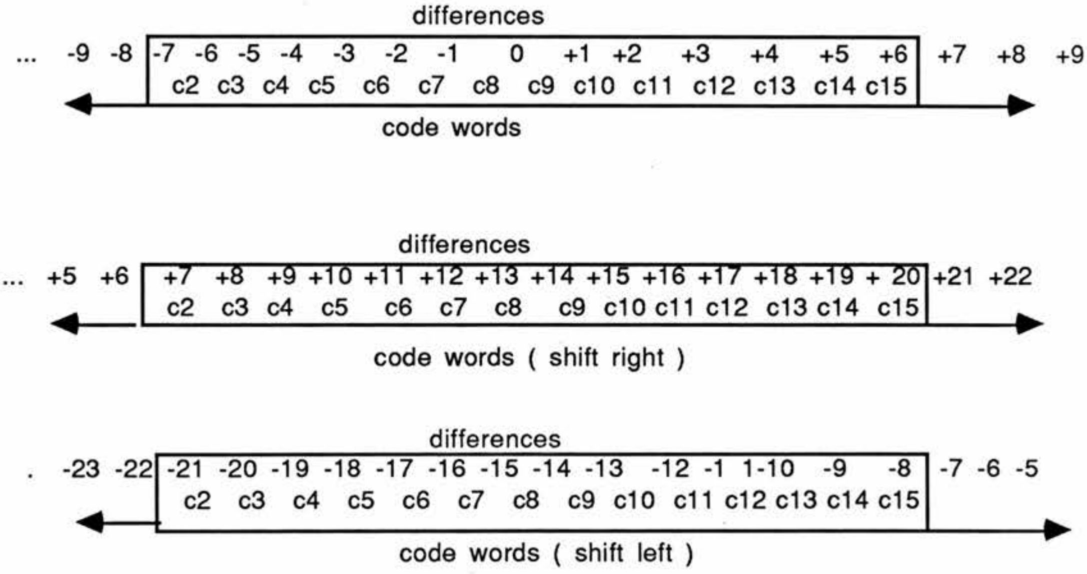


Fig. 5.4 Possible coding for grey-level differences showing effect of shifts left and right

The use of a murray scan could give a more suitable distribution of differences than linear scan and lead to better compression.

### 5.2.2 Differential Pulse Code Modulation ( DPCM )

This technique introduced in section 1.5.2 takes advantage of the expected local correlation between pixel values in many grey-level images. For any pixel intensity $g_i$ along the scan the adjacent pixel is likely to have a value close to $g_i$. This can be seen from investigation of the distribution of intensity differences as mentioned in the previous section. In DPCM from the known history of previous pixel values a prediction of the next value is made and the difference between the prediction and actual value is found. This difference is usually smaller than the intensity value of the pixel and can therefore be coded in a fewer number of bits. This technique combined with the local area scanning by murray scan could be fruitful.

### Conclusion

The progress made and compression achieved for grey-level images for image data from the real world has been reported. Restricting the number of grey-levels to 32 gave a compression ratio of around 2 (50% of the bit map) on average for exact coded images. However, this could be improved considerably by reducing the resolution through run length removal. With runs up to length 5 removed the compression ratio could be increased to values in the range 5-7 ( 20-14 % of the bit map). It should be stressed that these results are from the basic applications at this initial stage and by dovetailing some of the many refinements now regularly used in data compaction more progress should be expected. Such methods include spatial sub-sampling with interpolation, adaptive quantisation and Gray coded intensities.

In addition the investigation of combining murray scans with DPCM is suggested as an area for future work.

# Chapter 6
# Motion from Frame Differences

As well as a general investigation of the application of murray polygons to image data compaction, an early thought as suggested in chapter 1 was the application of the results to motion - in particular the video telephone image requirements for transmission over low bit rate digital links. This explains the nature of the black and white, and grey-level images considered in chapters 4 and 5. It is well known that successive frames in motion transmission are often very similar and in transmitting each frame separately a large amount of redundant data is included. The amount of redundant data will of course depend on the amount of movement in the particular situation and the scanning frame rate. In general a frame rate of around 25 frames/sec is considered adequate to capture motion but to avoid flicker the viewing rate must be around 50 frames/sec. Methods such as frame repeating and interlacing have been used to counteract flicker. Interpolation between sampled values is another technique used in moving areas. Mounts [24] introduced the idea of conditional replenishment, where only information on those elements that have changed significantly is transmitted. A significant change was measured by a threshold value, usually about 1.5 % of the peak signal, and both the address and new PCM intensity value were sent for changed elements. A considerable amount of further work and refinements have been added to the original conditional replenishment scheme, such as predictive coding, cluster coding by bridging small gaps between changed elements, sub-sampling and removal of isolated differences. Work in this area is well documented and the reader is directed to the following references [60,26]. Pease and Limb [61] investigated the effect of sub-sampling both spatially and temporally and the option of switching between the two when appropriate. In spatial sub-sampling the sample rate was reduced along the scan line with the unsampled pixels replaced by interpolating between the known values. This was used with fast moving objects but with slower moving objects they found that the temporal resolution could be reduced without impairing the image quality.

The properties of the human eye are of crucial importance. Levine [62] gives a detailed account of the biological vision system. The eye system must take light patterns and process them to produce perceptions. Levine considers this as a three stage process. Firstly the optical stage as light passes through the lens system producing a focused image at the retina. Secondly the retinal stage uses a photochemical process to produce electrical signals which, in the third stage, are passed to the brain where they are further processed at the neuron level to produce a final perception.
In particular the required and tolerable resolution needs and their associated motion levels need careful consideration. As suggested above, when there is considerable motion the resolution can normally be reduced without the human viewer finding it objectionable. The ability of murray scans to vary the resolution within different areas of the picture ( see section 3.3 ) may be usefully employed in such circumstances.

One case where there is a high level of redundancy is in the transmission of a human 'head and shoulders' type of image, such as seen in news programmes. The movement here is usually restricted to eye and mouth movements along with more gentle head movement against a relatively stationary background. It is the frame image found in this situation that is investigated further in this chapter.

### 6.1 Acquisition of successive frames

The Data Translation ( DT2853) frame grabber available was capable of storing two images 512 x 512 x 8 for future analysis. The block diagram is given in Fig. 6.1.

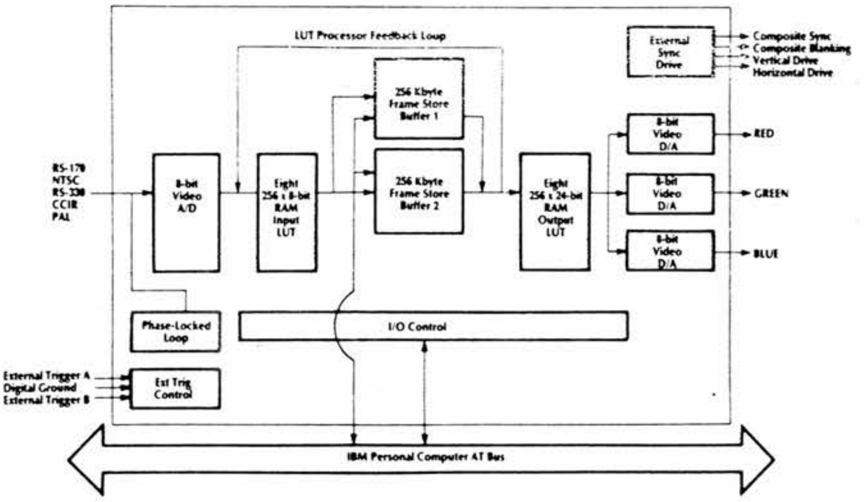


Fig. 6.1 Block diagram of the DT2853 frame grabber.

The software allowed the acquisition of two successive frames in real time ( 1/25 sec ) with 8-bit A-D flash converters giving binary values 0-255 from the analog signal. These were stored in frame buffers for later comparison and manipulation.

The images once captured could be manipulated using real time arithmetic and logic image processing operations including AND, OR, XOR, frame averaging, frame addition and subtraction, and multiplication and division. These operations are accomplished on-board through the use of a look-up table ( LUT ) processor.

Initial investigations were carried out using a plumbicon colour camera with a chrominance filter in the system to remove the colour information. If the colour information is not removed it causes interference patterns in the digitised image. When consecutive frames of a motion sequence were analysed on a preliminary basis the differences found were much greater than expected. Both the number of pixels changing in value and the magnitude of the changes exceeded those suggested in the literature. The reasons for this were not obvious and much time was spent with tests, often on a trial and error basis, to determine why this was happening. Even with successive frames of still objects the number and magnitude of the changes in areas of similar shading was greater than would have been expected due to noise. Haskell et al [25] had suggested that on average only about 9 % of the image area changed from frame to frame in sequences with movement. We were not able to determine the exact cause of the problem and work was held up for some time.

Progress was only made when a CCD camera was available on loan from EEV ( see acknowledgements ). Using the charged coupled device the results were closer to those expected and investigations in this area were resumed. The details follow in the next section.

## 6.2 The Frame Differences

The investigation focused on the bit plane patterns produced by differences in successive frames. These patterns were formed by comparing the bit planes of successive frames using an XOR raster operation with pixels that have changed showing up as white in the examples given. If two bit planes had not changed then the bit plane difference pattern would be totally black. Some examples of the results are shown in the two cases in Fig. 6.2. The first in Fig. 6.2.a is a case where considerable movement occurred in eyes, mouth and head, while Fig.6.2.b is an example where there was little movement. These patterns show several features typical of those generated. Significant movement results in relatively large homogeneous areas of white corresponding to considerable changes in the grey-level values. In particular this is visible around the edges of the hair, where the intensity value would change from a low value on the hair (dark object ) to a high value in the background ( light object ). The edges are less well defined in Fig. 6.2.b where the changes although located in similar positions to those in Fig. 6.2.a are far fewer in number and rarely occur in consecutive groups. All planes have been shown for completeness, but as mentioned previously the least significant bit was inaccurate as the A/D converter was heavily biased towards even values giving unexpected structure in plane 7.

It is normal practice in interframe methods to ignore isolated changes and to use a threshold to determine if a change is classified as significant or not. This was applied to the original image data using a threshold of 4/256 of the maximum signal and the resulting difference patterns are those shown in Fig 6.3. These are the difference patterns used for subsequent analysis. In a motion sequence the effect of this approximation at the low threshold level would not be noticed by the human viewer. Comparing the bit plane patterns of the differences from Fig. 6.2 and 6.3 there is significant change with the resulting patterns of Fig. 6.3 being suitable for run length encoding. In the cases of the frames where little movement occurred the change is very dramatic indeed. Despite the substantial change in the difference patterns from Fig. 6.2 to Fig. 6.3 the results in practice are not disturbing.

The hardware was not available to assess the effect of this approximation in our examples but we did combine the processed difference patterns with the first frame using standard raster options. The result was viewed on a high resolution graphics screen and gave very acceptable results.

Note:- the blotches appearing in the images above and to the right of the head are caused by dirty patches on the CCD array.

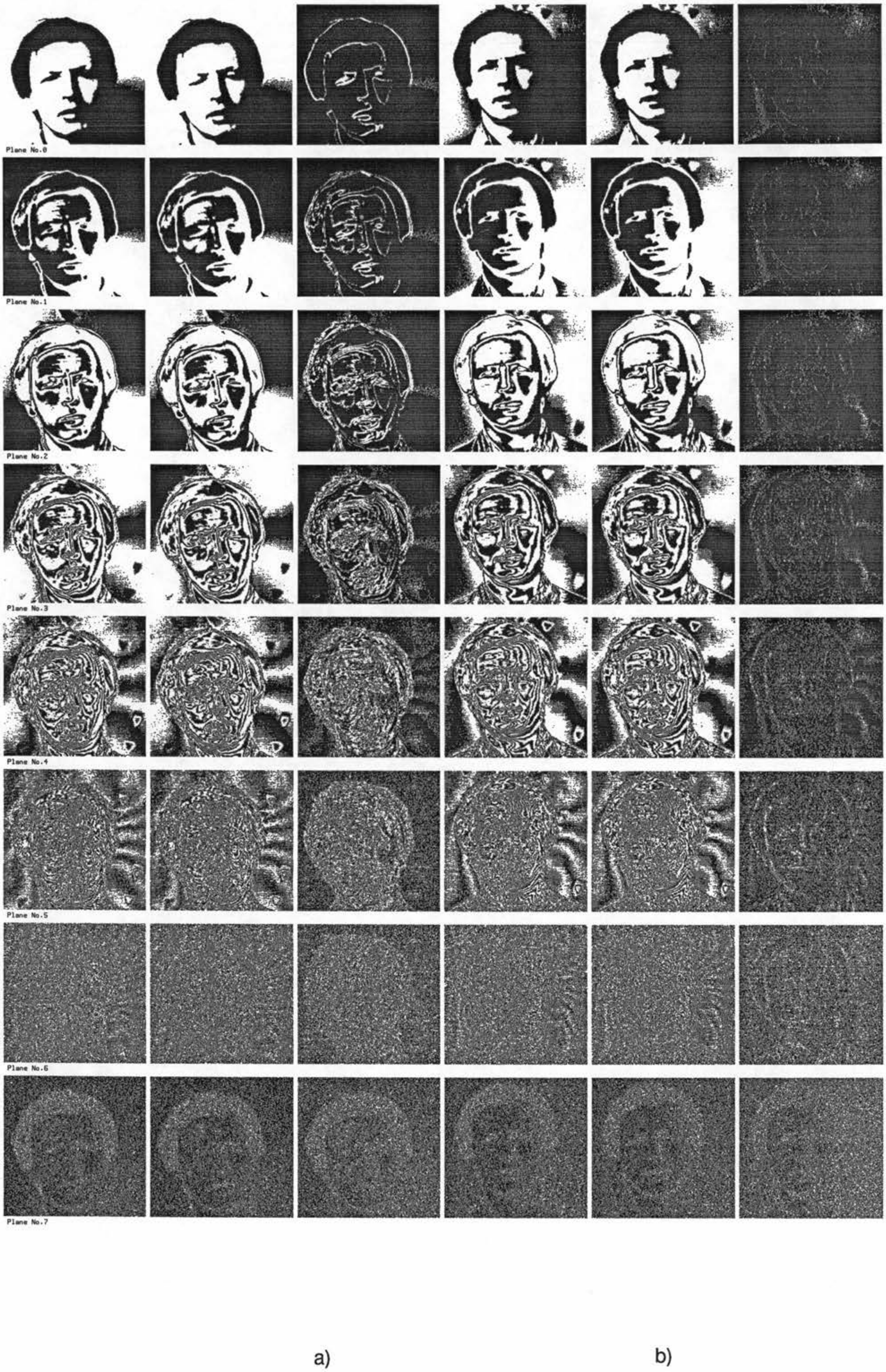


Fig. 6.2 Bit planes for consecutive frames and the corresponding difference patterns a) Considerable movement and b) little movement.

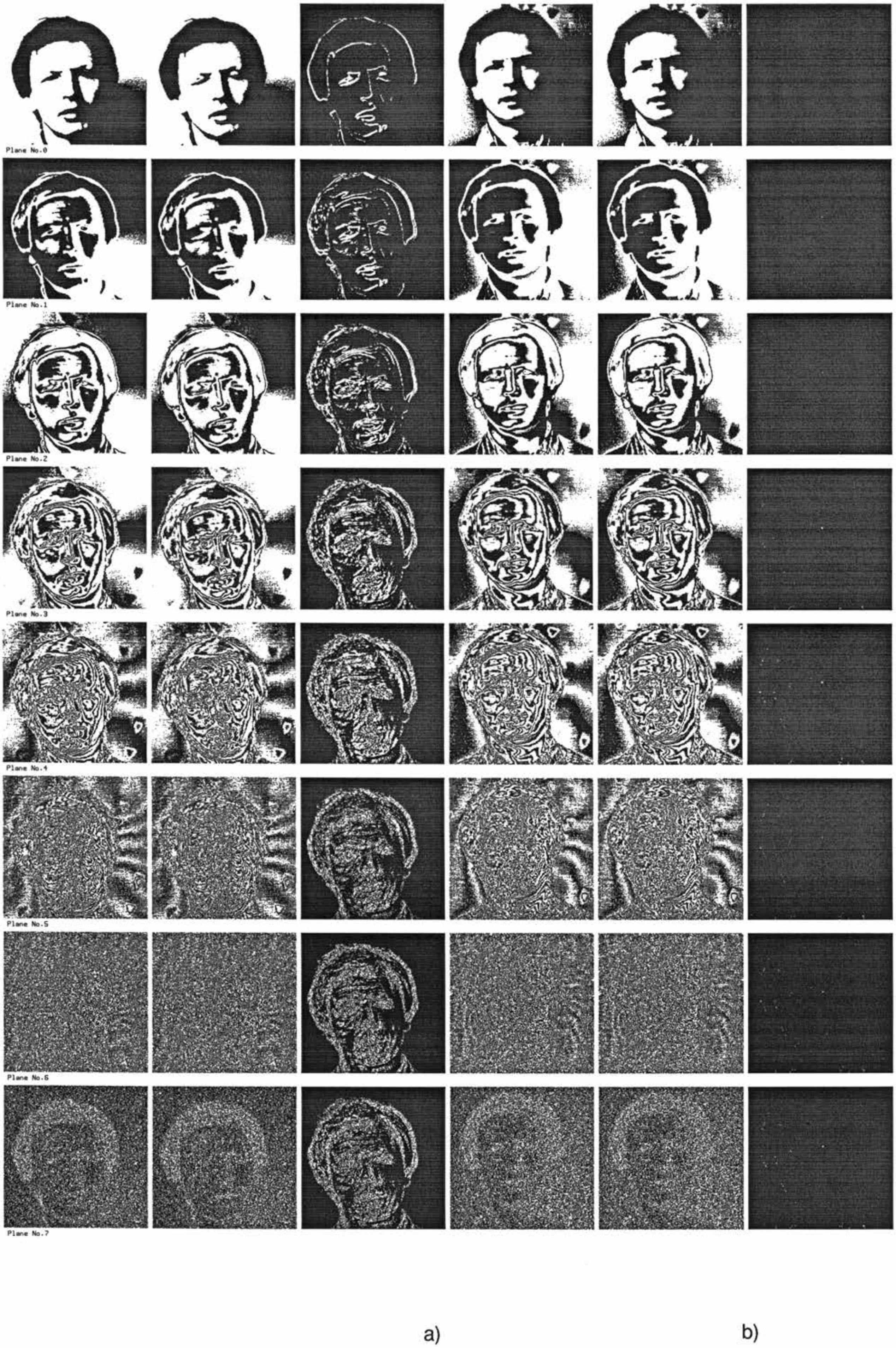


Fig. 6.3 Bit planes for consecutive frames and corresponding difference patterns after changes below a threshold of 4/256 of max signal and isolated points are removed ( compare with Fig. 6.2 ).

## 6.3 Coding the differences

By observing the resulting bit planes the expected compression should be substantial. Using Murray scans and run-length encoding with the continuation bit codes described previously some of the results for the difference planes of Fig. 6.3 are as follows.

The compression possible for planes 0-4 of the differences of Fig. 6.3 is given in table 6.1.

Considering the case with considerable motion from Fig. 6.3.a the compression ratio for plane 0 was 11:1 ( 9% of the bit map ) and decreased to just over 2:1 ( 46% of the bit map ) for plane 4.

| plane | compression as % of bit map | |
|---|---|---|
| | image | |
| | nod<br>( Fig. 6.2.a ) | still<br>( Fig. 6.2.b ) |
| 0 | 9.0 | 0.19 |
| 1 | 16.9 | 0.33 |
| 2 | 25.5 | 0.41 |
| 3 | 37.3 | 0.60 |
| 4 | 46.2 | 0.89 |
| average<br>( 0 - 4) | 27.0 | 0.48 |
| average<br>( 0 - 3 ) | 22.1 | 0.38 |

Table 6.1 Compression figures for the difference patterns between successive frames as given in Fig. 6.3.

This gave an average compression ratio of just less than 4 :1( 27% of the bit map ) which was approximately half the data required to send the second frame separately. A substantial saving with negligible distortion.

The case with very little movement, resulting in the the bit plane patterns of Fig. 6.3.b, will clearly give very good compression figures indeed. The average for the first 5 planes is approximately 200 :1 ( 0.5% of the bit map ).

These examples highlight the range of results that might be expected at different motion levels between successive frames.

**Conclusion**

The results and progress made with frame differences from real world images have been reported. The compression achieved by murray scan and continuation bit coding is promising with compression ratios in the range 5-200 for varying motion content.
No account has been taken of error recovery but at this stage we have neither applied coding refinements nor gone to the limits of resolution approximation. Both of these should give gains that easily deal with error recovery and still improve significantly on the data reduction.

Whether these results are sufficient to transmit useful motion sequences over low bit rate lines depends on quantifying the amount of movement that might occur. Considerable work has been done on this subject and is well reported by Haskell [60], who points out that despite considerable analysis much work still has to be done to achieve a good understanding of the phenomena involved in interframe coding.

A useful yardstick for future work is that of an image 256 x 256 with 32 grey-levels transmitted at 25 frames/sec would require an average compression ratio of around 130:1 for transmission over 64Kbits/sec lines. The frame difference patterns alone without any other standard techniques are producing data reduction of this order.

# Chapter 7

## Summary, Applications and Future Work

The development of murray polygons by Cole from the work of Peano was outlined in chapter 2. From there the work reported has focused on one application of murray polygons, namely digital image data compression.

### 7.1 Summary

Murray polygons applied to the scanning of a digital image and subsequent run-length encoding, offer another tool to workers in any field that requires storage, transmission or processing of digital images. This tool has been shown to have several advantages over the common linear scan. The main advantages are :-

1. Local 'area' gathering and describing ability that can take advantage of any local correlation that exists in the image. The resulting run length distributions are promising.
2. Flexibility.

   The complexity of the scan is easily altered.

   The scan pattern can be varied by altering the order of the radices.

   The values of the radices can be changed.

   The scans are not restricted to squares.

It is clear from the varied methods already in use and others being vigorously investigated that there is no single solution to image data compression but the task calls for a visual engineering solution. In particular the methods under investigation will clearly be unsuitable for images that are not conducive to run length encoding in general.

Using murray polygons explicit transformations are available in conjunction with simple and clear data structures which encapsulate the image data. In addition the flexible bounding rectangles and scanning patterns make it appealing and useful.

The ability, using short run length removal, to provide improved compression with acceptable loss of resolution and retention of major features gives the added reduction required to widen the range of applications. A further feature of this approximation is the ability to vary the quality of the final image either in total or in particular regions. Therefore the opportunity is present for some adaptive tailoring of the system to meet the changing demands of a particular situation.

The work reported has dealt with bilevel and grey-level images but the underlying methods can be extended to colour.

**Bilevel images**

For bilevel images the real world data mostly used was the bit planes of grey-level images of a similar type. However, as can be clearly seen from these images they represent quite different patterns which give a variety of composition. Each bit plane will therefore be typical of a wider set of bilevel images and need not be tied to the original grey-level source. The run-length distributions for many of the images were considerably better with murray scans than the corresponding results for linear scans. This being especially true for images characterised by the more significant bit planes. For exact coding compression ratios around 10 : 1 ( 10% of the bit map ) were achieved. This figure decreased as the images characterised by the less significant planes were considered. For very fragmented patterns often no compression was achieved but these were cases where no run-length method would be suitable.

It should be noted that compression ratios better than those obtained could have been achieved, if the purpose had been to apply the method to situations already known to give reasonable results when run-length encoded. For instance images which usually contain a limited number of large homogeneous areas, such as satellite data which has been processed to give areas of a particular vegetation.

By short run length removal the compression ratios were further improved. Those images with compression ratios around 10 : 1 being improved to 14 : 1 and those images that previously were not compressed gave ratios of 2 : 1. Further reductions with runs up to length 5 removed gave compression ratios up to 16 : 1.

**Grey-level images**

For grey-level images with intensities represented by 5-bit values (32 levels) overall compression by *exact* encoding each bit plane was typically 2.4 : 1 to 1.7 : 1 ( 42 - 60% of the bit map ). By coding the intensities using Gray codes, results suggested that the data required could be further reduced.

*Approximate* coding using short run length removal could take the compression to around 3:1 for runs of length 1 and up to 6:1 for runs up to length 5 removed.

**Interframe data**

The initial investigation of bit plane frame differences for the grey-level 'head and shoulders' images previously described, showed that considerable data reduction was possible. Applying minor approximations compression ratios varying over a wide range ( 5:1 to 200 : 1) were achieved, the value depending mainly on the amount of motion between frames. The images considered were limited to motion of the type expected from head, eye and mouth movement and did not consider extremely violent motion, or the differences caused by a complete change of scene. Further work in conjunction with more sophisticated interframe techniques is required to fully analyse the application to motion, but initial results look promising.

### 7.2 Applications

The applications are too numerous to mention, in the sense that they are to be found wherever a digital image storage/transmission need exists. The growth of communication technology, the introduction of digital links, the use of fibre optics and the availability of real-time image processing using frame store facilities are some of the reasons why the applications are extensive and still growing rapidly. Some of the more common areas are:

1. Storage and transmission of digital image data such as satellite data, x-ray images and general database requirements for pictorial information;
2. Video conferencing and telephony;
3. Simulations of various sorts, flight simulation being the most common;
4. Graphical art work in commercial and noncommercial applications.

Some applications that were rejected long ago are once again possible. For instance video telephony which was investigated in some detail before television, is now receiving new interest. The video phone is at present available to consumers in Japan and America. In

Britain, video phone systems are already used by the police to send photographs from station to station and by hospitals to store and transmit x-ray images. The creation of digital image databases demanding efficient storage coupled with transmission on low bit rate lines is attracting considerable commercial interest.

There is little doubt that the movement of information describing a picture source, from that source to the receiver/user will more and more comprise of digital links. The main constraints, which vary in importance from application to application, can be drawn from or are combinations of:

1. available bit rate
2. hardware complexity
3. cost of network equipment and transmitter/receiver hardware.
4. image quality requirements.

The work reported suggests an alternative tool for both still and moving image data transmission and storage. Some ideas about the application of the work reported in this thesis to transmission of digital image data on low bit rate links ( 64Kbits/s ) is given by Cole & Buntin [63]. Further work is required involving a hardware implementation to assess more precisely the performance in comparison with existing methods and applications.

### 7.3 Further work

While progress has been made in the 13 months work reported in this thesis, the author is hopeful that some hardware expertise may be applied to future investigations of murray scans and the transmission of moving pictures.

The report has not fully investigated the advantages to data compression that can be found by careful consideration of the properties of the human visual system. This would require work on both the objective and subjective assessment of final image quality.

The investigation of the results when differential pulse code modulation (DPCM) is applied is another area that merits further attention.

Lastly other work and techniques, presently being investigated, using murray polygons could add considerably to specific areas of digital image handling. A scaling method, reduction and enlargement, is available that can easily be applied to the run lengths from bilevel images with drastic reductions in data and little obvious impairment in the resulting image at the receiver. Halftoning methods amalgamated with mixed scan patterns appear to give promising bilevel renditions of monochrome images.

## References :- Alphabetic

18 Abate, J. E. [1967].
" Linear and Adaptive Delta Modulation ", Proc. IEEE, vol. 55, no. 3, pp. 298- 307, March.

26 Candy, J. C., Franke, M. A., Haskell, B.G. & Mounts, F.W. [1971].
" Transmitting Television as Clusters of Frame-to-Frame Differences ", The Bell System Technical Journal, vol. 50, pp. 1889-1917, July-August.

38 Campbell, G., DeFanti, T. A., Frederiksen, J., Joyce, S.A., Leske, L.A., Lindberg, J.A. and Sandin, D.J. [1986].
" Two bit/pixel full color encoding ", SIGGRAPH '86, vol. 20, no. 4, pp. 215-220.

52 Cole, A. J. [1966].
" Cyclic Progressive Number Systems ", Math. Gazette, vol. L, no. 372, pp. 122-131.

40 Cole, A. J. [1983].
" A Note on Space Filling Curves ", Software-Practice and Experience, vol. 13, pp.1181-1189.

53 Cole, A. J. [1985].
" A Note on Peano Polygons and Gray Codes ", Intern. J. Computer Math, vol 18, pp. 3-13.

41 Cole, A. J. [1986].
" Multiple Radix Arithmetic and Computer Graphics ", Bulletin Inst. of Math. and its Applications, vol. 22, May/June, pp. 71-75.

54 Cole, A. J. [1986].
" Direct Transformations between Sets of Integers and Hilbert Polygons ", Intern. J. Computer Math., vol. 20, pp.115-122.

57 Cole, A. J. [1987].
" Compaction Techniques for Raster Scan Graphics using Space-filling Curves ", The Computer Journal, vol. 30, no. 1, pp. 87-92.

58 Cole, A. J. [1987].
" Data compaction using murray polygons ",Computer Graphic Technology & Systems conference, CG87, London, pp. 185-194.

49 Cole, A. J. [1988].
" Direct transformations for a class of space filling curves", internal publication, CS/88/1, Univ. of St. Andrews.

56 Cole, A. J. [1988].
" Murray Polygons as a Tool in Raster Scan Graphics ", ICONCG '88, Singapore, Sept. 88.

63 Cole, A.J. & Buntin, I.M. [1988]
" Some ideas about the low speed transmission of moving pictures ", Proc. Computer Graphics '88 Conference, pp 33 - 42, October .

22 Connor, D. J., Brainard, R. C. and Limb, J. O. [1972].
" Intraframe Coding for Picture Transmission ", Proc. IEEE, vol. 60, no. 7, pp. 779-791, July.

23 Crater, T. V. [1971].
" The Picturephone service:Service standards ",The Bell System Technical Journal, vol. 50, pp. 235-270.

15 Cutler, C. C. [1952] .
" Differential Quantisation of Communication Signals ", U.S.Pattent 2 605 361, July.

17 DeJager, F. [1952].
" Delta Modulation : A Method of PCM Transmission Using a One-Unit Code ", Phillips Res. Rep., vol. 7, pp. 442-466.

34 Delp, E. J. and Mitchell, O. R. [1979].
" Image compression using block truncation coding ", IEEE Trans. Commun., vol. COM-27, Sept.

5 Fano, R. M. [1961].
Transmission of Information, Cambridge, Mass. :
M.I.T. Press.

55 Fisher, A. J. [1986].
" A New Algorithm for Generating Hilbert Curves ",
Software - Practice and Experience, vol. 16(1), pp. 5-12,
January.

51 Gilbert, E. N. [1957].
" Gray Codes and Paths on the n-Cube ", The Bell System
Technical Journal, vol. 37, no. 1, pp. 815-826.

9 Gonzalez, R. C. and Wintz, P. [1987].
Digital Image Processing : Addison-Wesley Publications .

13 Goodall, W. M. [1951].
" Television by Pulse Code Modulation ", The Bell System
Technical Journal, January, pp. 33-49.

48 Griffiths, J. G. [1986].
" An Algorithm for Displaying a Class of Space-Filling
Curves ", Software-Practice and Experience, vol. 16(5),
pp. 403-411, May.

60 Haskell, B. G. [1979].
" Frame Replenishment Coding of Television ", Advances in
Electronics and Electron Physics, Suppl. edited by
W. K. Pratt, pp.189-215

25 Haskell, B. G., Mounts, F. W. & Candy, J. C. [1972].
" Interframe Coding of Videotelephone Pictures ",
Proc. IEEE, vol. 60, no. 7, pp. 792-800, July.

39 Heckbert, P. [1982].
" Color image quantization for frame buffer display ",
SIGGRAPH 1982 Proceedings, pp. 297-307.

37 Huang, T. S. [1977] .
" Coding of two-tone images ", IEEE Trans. Commun., vol.
COM-25, pp.1406-1425, Nov.

43 Hilbert, D. [1891].
" Ueber stetige Abbildung einer Linie auf ein Flachenstuck ",
Math. Annln. vol. 38, pp. 459 - 460.

12 Huffman, D. A. [1952].
" A Method for the Constructuion of Minimum Redundancy Codes ", Proc. IRE, vol. 40, no. 10, pp. 1098-1101 .

28 Hunter, G. M. and Steiglitz, K. [1979].
" Operations on images using quadtrees ", IEEE Trans. on Pattern Analysis and Machine Intell., PAMI-1(2), pp.145-153.

4 Jain, A. K. [1981].
" Image Data Compression: A Review ", Proc. IEEE, vol. 69, no. 3, March, pp. 349-389.

27 Klinger, A. and Dyer, C. R. [1976].
" Experiments in Picture Representation using Regular Decomposition ", Computer Graphics and Image Processing, no. 5, pp. 68-105.

10 Kortman, C. M. [1967].
" Redundancy Reduction - A Practical Method of Data Compression ", Proc. IEEE, vol. 55, no. 3, pp. 253-263.

11 Kunt, M., Ikonomopoulud, A. & Kocher, M. [1985].
" Second-Generation Image-Coding Techniques ", Proc. IEEE, vol. 73, no. 4, pp. 549-573 .

33 Kunt, M. and Johnsen, O. [1980].
" Block Coding of Graphics: A Tutorial Review ", Proc. IEEE, vol. 68, no. 7, pp. 770-786.

35 Lena, M. and Mitchell, O. R. [1984].
" Absolute Moment Block Truncation Coding and its Application to Color Images ", IEEE Trans. Commun., vol. COM-32, no. 10, pp. 1148-1157, October.

62 Levine, M. D. [1985].
Vision in Man and Machine : McGraw-Hill Publications.

1 McFarlane, M. D. [1972 ].
" Digital Pictures Fifty Years Ago", Proc. IEEE, vol. 60, no. 7, pp. 768-770.

61 Pease, R. F., and Limb, J. O. [1971].
" Exchange of Spatial and Temporal Resolution in Television Coding ", The Bell System Technical Journal, pp.191-199, Jan..

2 Pender, H. and McIlwain, K. [1936 ].
Electrical Engineers' Handbook, vol. V, Electrical Communication and Electronics, 3rd ed. New York: Wiley pp.11-35,14-27.

50 [1985] PS-algol Reference Manual, Persistent Programming Research Report, PPRR-12, Dept. of Computational Science, University of St. Andrews.

14 Roberts, L. G. [1962].
" Picture Coding using Pseudo-Random Noise ", IRE Trans. Informat. Theory, vol IT-8, pp.145-154, January.

20 Roese, J. A. [1979].
" Hybrid Transform/Predictive Image Coding ", Advances in Electronics and Electron Physics, Suppl. edited by W.K.Pratt, pp.157-187.

30 Samet, H. [1985].
" Data Structures for Quadtree Approximation and Compression ", Communications of the ACM, vol. 28, no. 9, pp. 973-993.

3 Shannon, C. E. and Weaver, W. [1949].
The Mathematical Theory of Communication, Urbana, Ill. : University of Illinois Press.

7 Schreiber, W. F. [1967].
" Picture Coding ", Proc. IEE, vol. 55, no. 3, pp. 320-330, March.

45 Sierpinski, W. [1912].
" Sur une nouvelle courbe qui remplit toute une aire plaine ", Bull. Acad. Sic. Cracovie, Serie A, pp. 462-478.

8 Stockham, T. G. [1972].
" Image Procesing in the Context of a Visual Model ", Proc. IEEE, vol. 60, no. 7,pp. 828-841, July.

19 Tescher, A. G. [1979].
" Transform Image Coding ", Advances in Electronics and Electron Physics, Suppl. edited by W.K.Pratt, pp.113-154.

21 Wintz, P. A. [1972].
" Transform Picture Coding ", Proc.IEEE, vol. 60, no. 7, pp. 809-820, July.

31 Woodwark, J. R. [1982].
" The explicit quad tree as a structure for computer graphics ", The Computer Journal 25(2), pp. 235-238.

32 Woodwark, J. R. [1984].
" Compressed Quad Trees ", The Computer Journal, vol. 27, no. 3, pp. 225-229.

64 Witten, I. H. & Wyvill, B. [1983]
"On The Generation and Use Of Space-filling Curves", Software-Practice and Experience, vol 13, pp519-525.

## References :- Numeric

1 McFarlane, M. D. [1972 ].
"Digital Pictures Fifty Years Ago ", Proc. IEEE, vol. 60, no. 7, pp. 768-770.

2 Pender, H. and Mcllwain, K. [1936 ].
Electrical Engineers' Handbook, vol. V, Electrical Communication and Electronics, 3rd ed. New York: Wiley pp.11-35,14-27.

3 Shannon, C. E. and Weaver, W. [1949].
The Mathematical Theory of Communication, Urbana, Ill. : University of Illinois Press

4 Jain, A. K. [1981].
" Image Data Compression: A Review ", Proc. IEEE, vol. 69, no. 3, March, pp. 349-389.

5 Fano, R. M. [1961].
Transmission of Information, Cambridge, Mass. : M.I.T. Press.

6 Pearson, D. E. [1967].
" A Realistic Model for Visual Communication Systems ", Proc. IEEE, vol. 55, no. 3, pp. 380-389, March.

7 Schreiber, W. F. [1967].
" Picture Coding ", Proc. IEE, vol. 55, no. 3, pp. 320-330, March.

8 Stockham, T. G. [1972].
" Image Procesing in the Context of a Visual Model ", Proc. IEEE, vol. 60, no. 7, pp. 828-841, July.

9 Gonzalez, R. C. and Wintz, P. [1987].
Digital Image Processing : Addison-Wesley Publications .

10 Kortman, C. M. [1967].
" Redundancy Reduction - A Practical Method of Data Compression ", Proc. IEEE, vol. 55, no. 3, pp. 253-263.

11 Kunt, M., Ikonomopoulud, A. & Kocher, M. [1985].
" Second-Generation Image-Coding Techniques ", Proc. IEEE, vol. 73, no. 4, pp. 549-573 .

12 Huffman, D. A. [1952].
" A Method for the Constructuion of Minimum Redundancy Codes ", Proc. IRE, vol. 40, no. 10, pp. 1098-1101.

13 Goodall, W. M. [1951].
" Television by Pulse Code Modulation ", The Bell System Technical Journal, January, pp. 33-49.

14 Roberts, L. G. [1962].
" Picture Coding using Pseudo-Random Noise ", IRE Trans. Informat. Theory, vol IT-8, pp.145-154, January.

15 Cutler, C. C. [1952].
" Differential Quantisation of Communication Signals ", U.S.Patent 2 605 361, July.

16 O'Neal, J. B. Jr. [1966].
" Predictive quantizing systems( differential pulse code modulation) for the transmission of television signals ", Bell Syst. Tech. J., vol. 45, pp. 689-721.

17 DeJager, F. [1952].
" Delta Modulation : A Method of PCM Transmission Using a One-Unit Code ", Phillips Res. Rep., vol. 7, pp 442-466.

18 Abate, J. E. [1967].
" Linear and Adaptive Delta Modulation ", Proc. IEEE, vol. 55, no. 3, pp. 298- 307, March.

19 Tescher, A. G. [1979].
" Transform Image Coding ", Advances in Electronics and Electron Physics, Suppl. edited by W.K.Pratt, pp.113-154.

20 Roese, J. A. [1979].
" Hybrid Transform/Predictive Image Coding ", Advances in Electronics and Electron Physics, Suppl. edited by W. K. Pratt, pp. 157-187.

21 Wintz, P. A. [1972].
" Transform Picture Coding " ,Proc.IEEE, vol. 60, no. 7, pp. 809-820, July.

32 Woodwark, J. R. [1984].
" Compressed Quad Trees ", The Computer Journal, vol. 27, no. 3, pp. 225-229.

33 Kunt, M. and Johnsen, O. [1980].
" Block Coding of Graphics: A Tutorial Review ", Proc. IEEE, vol. 68, no. 7, pp. 770-786.

34 Delp, E. J. and Mitchell, O. R. [1979].
" Image compression using block truncation coding ", IEEE Trans. Commun., vol. COM-27, Sept.

35 Lena, M. and Mitchell, O. R. [1984].
" Absolute Moment Block Truncation Coding and its Application to Color Images ", IEEE Trans. Commun., vol. COM-32, no. 10, October.

36 Mitchell, O. R. and Delp, E. J. [1980].
"Multilevel Graphics Representation Using Block Truncation Coding," Proc. IEEE, vol. 68, no. 7, July.

37 Huang, T. S. [1977].
" Coding of two-tone images ", IEEE Trans. Commun., vol. COM-25, pp.1406-1425, Nov.

38 Campbell, G.,DeFanti, T. A., Frederiksen, J., Joyce, S.A., Leske, L.A., Lindberg, J.A. and Sandin, D.J. [1986].
" Two bit/pixel full color encoding ", SIGGRAPH '86, vol. 20, no. 4, pp. 215-220.

39 Heckbert, P. [1982].
" Color image quantization for frame buffer display ", SIGGRAPH 1982 Proceedings, pp. 297-307.

40 Cole, A. J. [1983].
" A Note on Space Filling Curves ", Software-Practice and Experience, vol. 13, pp.1181-1189.

41 Cole, A. J. [1986].
" Multiple Radix Arithmetic and Computer Graphics ", Bulletin Inst. of Math. and its Applications, vol. 22, May/June, pp. 71-75.

42 Peano, G. [1890].
'Sur une courbe, qui remplit toute une aire plaine ', Math. Annln., vol 36, pp.157-160.

43 Hilbert, D. [1891].
" Ueber stetige Abbildung einer Linie auf ein Flachenstuck ", Math. Annln. vol. 38, pp. 459-460.

44 Moore, E. H. [1900].
" On Certain Crinkly Curves ", Trans. Am. Maths. Soc., vol 1, pp. 72-90.

45 Sierpinski, W. [1912].
"Sur une nouvelle courbe qui remplit toute une aire plaine ", Bull. Acad. Sic. Cracovie, Serie A, pp. 462-478.

46 Null, A. [1971].
"Space filling curves, or how to waste time with a plotter " Software-Practice and Experience, vol1, pp. 403-410.

47 Mandelbrot, B. B. [1977].
" The Fractal Geometry of Nature ", Publisher : W.H.Freeman and Co..

48 Griffiths, J. G. [1986],
" An Algorithm for Displaying a Class of Space-Filling Curves ", Software-Practice and Experience, vol. 16(5), pp. 403-411, May.

49 Cole, A. J. [1988].
" Direct transformations for a class of space filling curves ", internal publication, CS/88/1,
Univ. of St Andrews.

50 [1985] PS-algol Reference Manual, Persistent Programming Research Report, PPRR-12, Dept. of Computational Science, University of St. Andrews.

51 Gilbert, E. N. [1957].
"Gray Codes and Paths on the n-Cube ", The Bell System Technical Journal, vol. 37, no. 1, pp. 815-826.

52 Cole, A. J. [1966].
"Cyclic Progressive Number Systems ", Math. Gazette, vol. L, no. 372, pp.122-131.

53 Cole, A. J. [1985].
" A Note on Peano Polygons and Gray Codes ", Intern. J. Computer Math, vol18, pp. 3-13.

54 Cole, A. J. [1986].
" Direct Transformations between Sets of Integers and Hilbert Polygons ", Intern. J. Computer Math., vol. 20, pp.115-122.

55 Fisher, A. J. [1986].
" A New Algorithm for Generating Hilbert Curves ", Software - Practice and Experience, vol. 16(1), pp. 5-12, January.

56 Cole, A. J. [ 1988].
" Murray Polygons as a Tool in Raster Scan Graphics ", Proc. ICONCG '88, Singapore, Sept..

57 Cole, A. J. [1987].
" Compaction Techniques for Raster Scan Graphics using Space-filling Curves ", The Computer Journal, vol. 30, no. 1, pp. 87-92.

58 Cole, A. J. [1987].
" Data compaction using murray polygons ", Computer Graphic Technology & Systems conference, CG87, London, pp.185-194.

59 Morrison, R., Brown, A. L., Dearle, A. and Atkinson, M. P. [1988].
" An Integrated Graphics Programming Environment" , Persistent Programming Research Report, PPRR-14.

60 Haskell, B. G. [1979].
"Frame Replenishment Coding of Television ", Advances in Electronics and Electron Physics, Suppl. edited by W. K. Pratt, pp.189-215

61 Pease, R. F., and Limb, J. O. [1971].
" Exchange of Spatial and Temporal Resolution in Television Coding",The Bell System Technical Journal, pp.191-199,Jan..

62 Levine, M. D. [1985]. Vision in Man and Machine : McGraw-Hill Publications.

63 Cole, A.J. & Buntin, I.M. [1988]
" Some ideas about the low speed transmission of moving pictures ", Proc. Computer Graphics '88 Conference, pp 33 - 42, October.

64 Witten, I. H. & Wyvill, B. [1983]
"On The Generation and Use Of Space-filling Curves", Software-Practice and Experience, vol 13, pp519-525.