TITLE Image Data Compression Based On a Multiresolution Signal Model

AUTHOR *Martin Peter Todd*

INSTITUTION and DATE The University of Warwick 1989

# Image Data Compression
# Based On a Multiresolution Signal Model

*Martin Peter Todd* B.Sc.

A thesis submitted to
The University of Warwick
for the degree of
Doctor of Philosophy

November 1989

**Image Data Compression**
**Based On a Multiresolution Signal Model**

*Martin Peter Todd B.Sc.*

A thesis submitted to
The University of Warwick
for the degree of
Doctor of Philosophy

November 1989

Summary

Image data compression is an important topic within the general field of image processing. It has practical applications varying from medical imagery to video telephones, and provides significant implications for image modelling theory.

In this thesis a new class of linear signal models, linear interpolative multiresolution models, is presented and applied to the data compression of a range of natural images. The key property of these models is that whilst they are non-causal in the two spatial dimensions they are causal in a third dimension, the scale dimension. This leads to computationally efficient predictors which form the basis of the data compression algorithms. Models of varying complexity are presented, ranging from a simple stationary form to one which models visually important features such as lines and edges in terms of scale and orientation. In addition to theoretical results such as related rate distortion functions, the results of applying the compression algorithms to a variety of images are presented. These results compare favourably, particularly at high compression ratios, with many of the techniques described in the literature, both in terms of mean squared quantisation noise and more meaningfully, in terms of perceived visual quality. In particular the use of local orientation over various scales within the consistent spatial interpolative framework of the model significantly reduces perceptually important distortions such as the blocking artefacts often seen with high compression coders. A new algorithm for fast computation of the orientation information required by the adaptive coder is presented which results in an overall computational complexity for the coder which is broadly comparable to that of the simpler non-adaptive coder. This thesis is concluded with a discussion of some of the important issues raised by the work.

**Key Words**

multiresolution, data compression, orientation, rate distortion

**Acknowledgements**

This work was supported by UK SERC and British Telecom Research Labs (BTRL), and conducted within the Image and Signal Processing Research Group in the Department of Computer Science at Warwick University.

I should like to thank all the staff at both the Computer Science Department and BTRL. In particular, thanks go to all the researchers in the Image and Signal Processing Group, Abhir Bhalerao, Andrew Calway, Simon Clippingdale, Roddy McColl and Edward Pearson, for their day to day advice and flow of ideas.

I should also like to thank my supervisor at BTRL, Dr. Charles Nightingale for his ideas and support.

Finally I am indebted to my supervisor, Dr. Roland Wilson, without whose guidance, ideas and encouragement this work would not have been possible.

**Table of Contents**

## List of Figures

**List of Photographs**

Note: the PSNR values are in dB.

**List of Tables**

**List of Graphs**

**Notation used in this Thesis**

$\wedge$

logical AND

$\vee$

logical OR

$\delta_{rs}$

Kronecker delta function i.e. $\delta_{rs} = \begin{cases} 1 & \text{iff } r = s \\ 0 & \text{else} \end{cases}$

CARD[•]

The cardinality of a set.

E[•]

The expected value of a random variable.

VAR[•]

The variance of a random variable.

$\lceil a \rceil$

The ceiling of $a$, i.e. the minimum integer $\geq a$, for $a \geq 0$

$\square$

End of Proof.

## 1. Introduction

The increasing use of digital rather than analogue signal representations for signal processing, and in particular for image processing, is derived from the tractability of many digital techniques which would be impracticable if not impossible to perform in an analogue form. The techniques opened up by digital image representations cover a broad spectrum of applications. In image analysis the aim is to derive significant parameters describing the content and structure of an image or sequence of images with a view to object detection, recognition and classification. In image enhancement, the aim is to make the image more accessible to an observer (either human or machine) by techniques such as image restoration, where distortions of the image are to be reduced, or pseudo-colour, where the structure of the image is made more visible to a human observer for applications such as medical or satellite imaging.

An additional advantage of digital representation in communication is the ability to distinguish reliably between information and noise. This leads to greatly enhanced signal quality for long distance or inherently noisy communication channels, often without requiring any extra bandwidth to be made available. Indeed, where the digital representation makes otherwise impractical compression techniques available, there may be a reduction in bandwidth, or equivalently of the storage capacity, required for image data. It is the purpose of this thesis to describe the development of some new methods for accomplishing such data compression of images.

### 1.1. A General Communications System Model

The use of data compression techniques to reduce the bandwidth of digital images has applications which include transmission of single frame images over a limited channel, where compression can reduce the time required to transmit the image; transmission of image sequences in real time for applications such as video telephones, where compression may be desirable in order to reduce the necessary channel capacity and therefore the cost; and the storage and retrieval of images on hard discs, magnetic tapes or video discs, where compression can increase the number of images that can be stored. The compression technique may be invertible, in that an exact copy of the original can be derived from the compressed data, or it may be non-invertible, so that the image derived from the compressed data has a degree of distortion with respect the original image.

Such applications can be represented by the general communications model [96] [8] shown in fig 1.1.1.

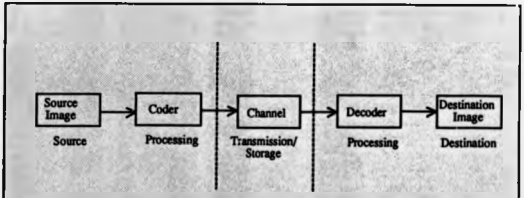



Fig 1.1.1 - A General Communications System Model

The model consists of five separate components,

(1) Image Source

The source image is a digital signal representation of an image or sequence of images which is produced by some well defined source process. Since this work only considers single frame images, the source is treated as a process which produces a single image for transmission.

In this thesis, it is assumed that the process produces a source image which is a regular cartesian lattice of picture elements or pixels with each pixel position relating directly to a spatial position in the image and that each such pixel has associated with it a discrete data value (or discrete data vector) representing the image intensity (or colour) at the appropriate spatial position in the image.

An important part of the image compression problem is to devise a statistical model which describes the source images in an adequate manner. The properties of this source model should be closely related to those of the image source [51]. From this model various significant parameters of an image or class of images can be derived, as well as optimal or sub-optimal techniques for processing such images.

(2) Channel

The function of the channel is to store or transmit the formatted data from the coder. The channel has a particular capacity, which is a measure of the amount of information is can handle. The aim of the coder/decoder pair is to ensure that the rate of the image data is less than or equal to this channel capacity. Depending on the application, the capacity may be fixed

or it may be specified by the systems designer. In general, however, an increase in channel capacity is usually associated with an increase in system cost.

The physical nature of the channel may be anything from an electrical or optical cable, a telecommunications network, a satellite system, to a magnetic tape or disc. Sources of noise will be present in all physical channels, introducing distortions into the data in transmission across the channel.

(3) Destination

The purpose of the whole system is to construct at the destination an image which is an adequate copy of the source image. The criteria for adequacy will depend on the application and some fidelity measure is devised to determine the quality of the result. The work in this thesis assumes that the application involves the destination image being viewed by a human observer and therefore the fidelity measure should reflect this. However as discussed in section 1.2, all too often the fidelity measure is simply assumed to be mean squared error. The fidelity measure should be used in the derivation of optimal or sub-optimal coding techniques.

(4)(5) Coder/Decoder (codec)

The fundamental task of the coder is to transform the source image data into the form best suited for transmission across the channel and of the decoder to 'invert' this received data back into a form which is as close as possible to that of the original image data. Berger [8] splits this fundamental task of communication into two parts:

*"Problem 1: What information should be transmitted ?"*

*"Problem 2: How should it be transmitted ?"*

The first problem is the source coding problem: find the most efficient way to represent important information from the source; the second problem is channel coding: protect it against the effects of errors introduced into the received data by channel noise. The significance of these distortions will depend on the coding technique (and therefore the source model) and the distortion measure.

In [8] it is shown that, in principle, channel coding can be treated as a separate issue. Unfortunately, realisable systems cannot afford the unlimited resources required for this result to hold, particularly if only a single image is to be transmitted.

Thus in practice the optimal channel coding should be derived with respect to the source model and fidelity measure in conjunction with source coding. This is often not considered in the design of compression systems however, either because the channel characteristics are not known or because the channel is assumed to be a general purpose channel, with its own error protection coding. In particular, it is assumed in this thesis that the channel can be regarded as having virtually no noise, so that it introduces minimal distortions into the source coded data. A good example of this is the case where image data are stored on the disc of a general purpose computer system, where the computer system has the responsibility of ensuring the data are not distorted.

While there are applications for which this is not a reasonable assumption, the work in this thesis is intended to address the issues of source coding and not those of the channel coding.

The source coding techniques used may be invertible, in which case the maximum compression achievable will be limited by the redundancy in the source image, or it may be non-invertible, in which case a larger compression will be achievable at a cost of some degradation of the source image. The source coder should be directly related to, or derived from, the source model. Indeed it is sometimes possible to derive an optimal source coder with respect to the source model and a given fidelity criterion.

Taken together the coder and decoder represent the source coding process and are referred to as a codec. The design of a codec is derived explicitly or implicitly from a particular source model and fidelity measure. The overall performance of this codec will depend on a good choice of the source model and fidelity measure. If the source image has significantly different properties from the source model, or the fidelity criterion for the application is significantly different from the fidelity measure, then the performance of the derived compression technique may be poor. The properties of image sources and fidelity criteria are discussed further in section 1.2. Note that although it is sometimes possible to derive an optimal codec it is not always practical to implement it - often the optimal codec will require either excessive memory or excessive computation, and a sub-optimal codec must be used.

The source model and fidelity measure can be used to derive a function which expresses the maximum compression which can be achieved for a given amount of distortion. This function, formulated by Shannon in 1959 [97] is known as the Rate Distortion Function and its properties are thoroughly investigated in the book by Berger [8]. The function is formulated in terms of the statistical parameters derived from a class of images.

The work in this thesis is aimed at considering some of the issues involved in image modelling, fidelity measures and derived compression techniques, in particular with respect to the importance of scale and local orientation as properties of natural source images. A source model for images with the two properties mentioned is developed, as well as simple sub-optimal coders for the model and approximations to the rate distortion functions for related models. In the remainder of this chapter some of the basic properties of the source images are considered, along with some fundamentals of information theory and fidelity measures, before going on to a brief review of previous work in the area.

### 1.2. Image Structure, Information and Fidelity

As mentioned in the previous section the work in this thesis assumes the digital representation to be a regular cartesian lattice of $N$ by $M$ pixels. This is not the only possible representation, but is the most common. In this work, for example, images having either $N = M = 512$ or $N = M = 256$ are used.

Each pixel in the lattice has a value, or set of values, which represent the characteristics of the image at that pixel. For a monochrome image the value is simply a discrete variable representing the quantised light intensity at that pixel. In this thesis the intensities are assumed to be quantised into 256 equidistant levels, therefore requiring 8 bits for a simple binary representation. Again this is not the only possible representation but is the most common. Photos 1.1-1.3 show three original 512 by 512 by 8 bit monochrome images: GIRL, BOATS and LAKE used as typical examples in this work.

For colour images each pixel has a set of three discrete variables representing the quantised colour at the pixel. The most common form of colour

Photo 1.1 - Original GIRL Image

Photo 1.2 - Original BOATS Image

Photo 1.3 - Original LAKE Image

representation is the RGB form, which expresses the colour in terms of a RED component, a GREEN component, and a BLUE component. Each of these is represented in 8 bits (i.e. 256 levels). There is a considerable degree of correlation between these three variables, which is mainly to do with brightness, and an alternative representation which reduces the correlation between these values

is often used. It is known as the YUV representation [85] [71] and consists of a *Luminance* component, Y, and two *Chromaticity* components, U and V, with each being stored as 8 bits. However, the U and V components are often subsampled by a factor of two with respect to the Y component i.e. $N_U = N_V = \frac{1}{2}N_Y$, and $M_U = M_V = \frac{1}{2}M_Y$. The colour example images used in this work have $N_U = M_U = N_V = M_V = N_Y = M_Y = 256$ with 8 bits per pixel per plane, giving 24 bits per pixel. However the originals came from a source which had been subsampled in the U and V planes, and therefore a more realistic value is $8 + \frac{8}{4} + \frac{8}{4} = 12$ bits per pixel. Photos 1.4-1.6 show the original colour images: MISS, TREV and SPLIT and their respective Y, U and V planes.

Photo 1.4 - Original MISS Image

Noiseless data compression is possible for such images because of the inherent structure within the images and the correlation between local pixel values which is implied by this structure. This correlation means that there is some

Photo 1.5 - Original TREV Image

Photo 1.6 - Original SPLIT Image

redundancy within the data and the total number of bits required to represent the image can be reduced by techniques which reduce this redundancy. In this section a number of simple measures are defined which give the minimum number of bits required to represent exactly a particular set of image data. Each measure depends on a particular model of the image data and clearly the better the

model the better the measure.

The simplest is the zeroth order entropy $E^0$ function formulated by Shannon in 1948 [96], which assumes that the data is a sequence of independent data values taken from an alphabet of possible values $\{i\}$, with a given probability distribution $\{P(i)\}$. It is defined as

$$E^0 = -\sum_{i=1}^{L} P(i) \log_2\Big[P(i)\Big] \qquad \text{bits per pixel (bpp)} \qquad (1.2.1)$$

where $L$ is the size of the alphabet - the number of distinct symbols.

However the data values are often not independent, and the first order entropy $E^1$ is a measure which reflects this by relating the occurrence of a given data value to the previous value

$$E^1 = -\sum_{j=1}^{L} P(j) \sum_{i=1}^{L} P(i\,|\,j) \log_2\Big[P(i\,|\,j)\Big] \qquad \text{(bpp)} \qquad (1.2.2)$$

where $P(i\,|\,j)$ is the probability that the symbol $i$ will occur given that the symbol $j$ has just occurred.

A alternative measure to $E^1$ is the differential entropy $E^D$ which acts not on the data symbols but the differences between successive data symbols. Assuming that the data symbols represent equally spaced integer values, and by performing the differences modulo $L$ (with complete invertibility), the alphabet of these differences is of size $L$, and representing these symbols by $e$, and their probability distribution by $P(e)$, the zeroth order differential entropy is given by

$$E^D = -\sum_{e=1}^{L} P(e) \log_2 \left[ P(e) \right] \qquad \text{(bpp)} \qquad (1.2.3)$$

Table 1.2.1 gives the values of $E^0$, $E^1$ and $E^D$ for the three monochrome original images shown in photos 1.1-1.3.

| Entropy | Image | | |
|---|---|---|---|
| | GIRL | LAKE | BOATS |
| $E^0$ | 7.25 | 7.64 | 7.07 |
| $E^1$ | 4.65 | 5.55 | 4.80 |
| $E^D$ | 4.92 | 5.99 | 5.06 |

Table 1.2.1

Table 1.2.2 gives the values of $E^0$, $E^1$ and $E^D$ for the Y, U and V planes of the three example colour images, together with the sum (T) of the values over the three planes.

| Entropy | Image | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MISS | | | | TREV | | | | SPLIT | | | |
| | Y | U | V | T | Y | U | V | T | Y | U | V | T |
| $E^0$ | 6.00 | 5.61 | 4.67 | 16.28 | 6.55 | 3.66 | 4.11 | 14.34 | 7.16 | 4.81 | 4.75 | 16.72 |
| $E^1$ | 3.90 | 2.20 | 2.22 | 8.32 | 4.43 | 1.81 | 2.19 | 8.43 | 4.39 | 2.02 | 2.04 | 8.45 |
| $E^D$ | 4.30 | 2.44 | 2.28 | 9.02 | 4.67 | 1.91 | 2.28 | 8.86 | 4.72 | 2.26 | 2.26 | 9.24 |

Table 1.2.2

These entropies are derived for noiseless (invertible) compression, where the data are represented exactly. In order to achieve higher compression ratios, however, techniques which allow some distortion of the data are required. Furthermore, the statistical structure which they express is of low order - only neighbouring pixels enter into the expressions (1.2.2), and (1.2.3). Clearly, natural images have correlations over much longer distances than this. One of the aims of this thesis is to introduce a class of image models which possess

such long range structure.

The fidelity measure used to assess the degree of distortion in the image should accurately reflect the application of the compression system. Where the application involves a human observer at the destination the fidelity measures should reflect the human processes of quality assessment. Obviously, the measure cannot be expected to be an exact model of the entire human visual system, both because of the complexity of the latter, and since the processes of vision are still far from understood. However certain properties of the human visual system are known [69] [38] and these could be incorporated into such a measure [91] [100]. These properties include edge masking [77], where the presence of an edge makes noise less visible, and spatial frequency and local orientation selectivity [11] [44] [45] [46] [76].

The work described in this thesis does not attempt to define or use such a measure explicitly, but the models developed in chapter 4 share certain properties with the visual system. In particular they reflect the importance of local orientation (anisotropy) and this, together with the spatially interpolative nature of the model, leads to derived codecs which produce results with good visual quality. However, for quantitative comparison with other work (and for the rate distortion work) the fidelity measure that is used is mean squared error. If $s(x,y)$ are the pixel intensities of the original image and $\hat{s}(x,y)$ the pixel intensities of the resulting decoded image then the mean squared error is defined as

$$\text{mse} = \frac{1}{N \times N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} |s(x,y) - \hat{s}(x,y)|^2 \qquad (1.2.4)$$

This measure is often quoted in the literature for image data compression in the form of the peak signal to noise ratio (PSNR) which is defined as

$$\text{PSNR} = 10 \log_{10}\left[\frac{P^2}{\text{mse}}\right] \quad \text{dB} \tag{1.2.5}$$

where $P$ is the peak or maximum value the signal can have. For this thesis, as in most applications, this value is 255, the maximum intensity possible for the 8-bit representation used.

Despite the fact that it is the most common measure used in the literature, it is not a particularly good choice of fidelity measure with respect to applications involving a human observer. Take for example the case where each intensity value in an image is increased by a value of 10. This corresponds to brightening the image slightly, however virtually no distortion will be apparent to an observer. Yet the PSNR will be $10 \times \log_{10}\frac{255^2}{100} = 28$ dB, implying some not insignificant drop in quality.

A more subtle comparison is given in [56] where two images made up of an equal amount of white noise are filtered. One is filtered with the filters aligned at 90 degrees to the local feature orientation of an original image, and the other at 0 degrees to these orientations. The result is a considerable difference in the perception of the image (and therefore the quality of the image) even though they both have the same degree of noise.

It is clear that mean squared error could not be used to represent either of these significant properties of perceived quality, but it is still the most commonly used measure. There are three main reasons for this preference. The first is its

mathematical familiarity. Since mean squared error has been used in many fields of mathematics, it is well known and many of its properties within other models have been thoroughly investigated. Secondly it is often mathematically tractable, so that many problems formulated with respect to mean squared error have relatively simple solutions. Finally, there is no specific obvious alternative which would be universally accepted. Although it is possible to derive alternatives based on known properties of the visual system such as frequency weighted mean squared error [68], the lack of an adequate model for vision prevents a definitive choice of the best measure, since it would change as models improve. Therefore, in order to provide some form of universally comparable distortion measure, mean squared error and PSNR are usually quoted for coding results and a more accurate judgement of quality is made by a visual inspection of the resulting images. This is clearly not a particularly good solution to the problem, since it assumes that such results can be adequately reproduced in publications, which is often not the case. Thus, although the results in this work quote PSNR's and the rate distortion functions are with respect to mean squared error, the photographic prints of the results give a better indication of the success or otherwise of the work.

Finally, the practical codecs described in this thesis have been implemented in the "C" programming language on a SUN 4/180 running UNIX†. The codecs were split into separate coder and decoder entities, with the source coded data from the coder being stored in an intermediary file. All the bit rates quoted in this thesis (as opposed to entropies) were calculated by taking the physical size of this file and dividing by the number of image pixels. The decoder

† UNIX is a trademark of AT&T Bell Laboratories.

reconstructed the image from the coded source data in the intermediary file. All the peak signal to noise ratios and mean squared errors quoted in this thesis were calculated between the original image and this reconstructed image. Thus the codec results presented represent an upper bound on the rate it is possible to achieve with these codecs, i.e. they can perform at least as well as the results presented.

The photographic results were taken on a Dunn Instruments MultiColor unit, using Kodak TMAX 100 ASA film on the green channel, with an exposure time of 2.36 seconds, for the monochrome results and Kodak GOLD 100 ASA film, with exposure times of Red: 2.85 seconds, Green: 2.36 seconds and Blue: 0.83 seconds, for the colour images.

### 1.3. A Review of Image Data Compression Techniques

This section is intended to be a brief review of relevant work in the area. For a full review of data compression techniques, the papers by Netravali and Limb [77], Jain [50] and Kunt, Ikonomopoulos and Kocher [61] and books by Hall [41] and Clarke [19] provide comprehensive coverage of the area.

The basic framework of the codecs described in this thesis is predictive in nature, a technique which has been extensively investigated since the early work of Oliver [80] and Elias [30]. More recently it has been combined with a number of techniques such as edge masking [111] [15], multiresolution [110] and arithmetic coding [4] [36] [73] [74] to provide codecs which are generally efficient and fairly simple to implement. The main drawback of predictive coding, however, has been that it is essentially causal, whereas the image signals are non-causal: there is no preferred direction in an image, unlike a one

dimensional signal.

Transform codecs take a non-causal approach, approximating the eigenvector transform of the image autocorrelation function [51] and, partly because of this non-causal nature, generally achieve higher compression ratios than predictive techniques, but usually at the expense of greater complexity. Two dimensional transform coders for images were introduced by Andrews and Pratt [5] and a large volume of work has been published on these techniques [115] [84] [90] [17] [18] [68], including the book by Clarke [19]. Recent work, such as that of Chen and Pratt [17], has been based on threshold techniques and there have been several combinations of transform codecs with other techniques such as the vector quantiser of Ho and Gersho [43] and Nasrabadi and King [54], the pyramid transforms of Wang and Goldberg [108], and selective codecs which choose between predictive and transform techniques according to their performance locally within the image [111].

The significance of the properties of scale within images has been recognised by the use of multiresolution or pyramid techniques [72]. These have been incorporated into other coding techniques such as the predictive quadtree codecs of Wilson [110] [111] and the transform codecs of Nasrabadi and King [54] and Wang and Goldberg [108]. Recursive Binary Nesting, which is the predictive multiresolution framework of the codecs described in this thesis, has been investigated by Tricker et al [107] and Cordell [24] [25] [26] as well as previously published work by the author [104] [105] [106]. Multiresolution coding is related to the subband coding techniques of Burt and Adelson [2], Woods and O'Neil [118], Smith and Barnwell [98], Westerink et al [109] and Kronander [60], where a signal is separated into a sequence of frequency bands and each band is coded separately.

Vector quantisation techniques, which can be formulated as an optimal solution to the coding problem, have been studied by a number of people [35] [33] [34], and various adaptive forms have been formulated [54]. However as mentioned in section 1.2 these optimal codecs are often impractical and so sub-optimal codecs are devised [86] [103] [42] [12] [55].

There are a number of algorithms which exploit the work on the orientation sensitivity of the human visual system by Hubel and Wiesel [44] and others [11] [76], including the work of Ikonomopoulos and Kunt [49] and Wilson, Knutsson and Granlund [112].

Other work in the area includes the sketch based coding of Carlsson [16], region based techniques [59] [71], the representation of images by higher order polynomials [29] [95], and the so-called 'model-based' methods [28] [32] for restricted applications such as 'head and shoulder' video-telephone images.

### 1.4. Thesis Outline

In chapter 2, a homogeneous multiresolution model, the Linear Interpolative Multiresolution (LIM) model, is formulated for images. It is developed into a particular form, the LIM RBN model, which is related to the Recursive Binary Nesting algorithm developed at BTRL [107] for use in single frame image data compression.

In chapter 3 various RBN codecs are derived for the LIM RBN model of chapter 2, and their performance is discussed. The implementation of the necessary segmentation, prediction, quantisation and entropy coding is described, and also the modifications necessary for adaptive codecs.

In chapter 4 adaptive models are devised to represent the scale selectivity of the codecs of chapter 3, and the model is then expanded to include the significant property of local orientation (anisotropy). Several rate distortion functions are presented for related multiresolution models.

In chapter 5 the techniques needed to implement codecs based on the models of chapter 4 are described, including orientation estimation, a 'single consistent orientation' measure, anisotropic prefiltering, boundary coding, and oriented interpolation. The performance of these codecs is discussed.

Conclusions and possibilities for further research are discussed in the final chapter.

# 2. Linear Signal Models

## 2.1. Introduction

In section 1.1 the importance of deriving the codec from an appropriate source model was noted. In this chapter some of the issues involved in modelling the source for natural images are considered and a homogeneous multiresolution linear signal model for such images is developed. The properties and limitations of this homogeneous model are considered and a specific form of the model, the Recursive Binary Nesting Model is defined.

As a result of the growing importance of applications for digital images since the 1960's, there has been interest in attempts to model the source images both in order to enhance various image processing techniques and provide a general theoretical background within which their advantages and limitations can be considered.

Many of the early models were one dimensional, either as a consequence of being derived from work on one dimensional signals such as speech, or based on the raster line scan which is an integral part of most practical image capture and display systems. These models force the essentially two dimensional source images into a one dimensional signal, and although the models have had a degree of success they lack the facility to take account of many of the properties inherent in the two dimensional nature of images. In the field of data compression, one set of techniques which have used one dimensional models extensively are Differential Pulse Code Modulation (DPCM) predictive codecs [50] [37] [78] [79] [4] [39] [40].

The causal linear predictive models which are associated with such codecs take the form of a predictor and an innovator and essentially exploit the correlation between successive data values. The predictor seeks to predict the value of a given data point from the values of a set of neighbouring data points, and the innovator provides a correction or innovation to this prediction. If the form of the predictor is chosen so as to minimise the variance of the innovation terms then it is known as the minimum variance predictor [50]. For such a predictor the innovation terms are uncorrelated with the data values, a property which is desirable in data compression since it also minimises the entropy of the innovation terms [50]. If the data are assumed to be Gaussian, then the minimum variance predictor is simply a linear combination of the set of neighbouring data points, hence the name Linear Signal Model.

In order to apply a one dimensional model such as the causal linear predictive model to an image, a path or time scan must be defined which traverses the image in a one dimensional manner. The most common scan is the raster scan, because of its use in capture and display equipment, but other scans such as the Peano-scan [82] which have some improved properties over the raster scan, have also been used. All of these one dimensional scans, however, share the property that the sequence in which they visit each pixel has a specific time direction, i.e. for a given pixel there is a certain set of pixels which come before it in the scan and a certain set which come after it. If the estimate of a given pixel is based solely on pixels that come before it in the scan, as is the case in the above models, then the model is said to be causal. If the estimate requires any pixels after it in the scan, then it is said to be a non-causal model. Causality is an essential property of conventional linear predictive models, since it is a natural requirement for any predictive coder based on such a model that all terms used to estimate the value of a data point must be known at the decoder.

i.e. they must already have been transmitted.

As an alternative to using a parametric model such as the linear predictive model, coding can be based simply on the correlation properties of the image. Such an approach is the basis of a variety of orthogonal transform codecs [19] and is inherently non-causal. The advantage of such an approach over the causal models is that it can take account of the correlations between data points in both the forward and backward direction of the signal. This means that codecs derived from such models can generally achieve a higher compression ratio than those derived from causal models. However these codecs involve an intrinsic time delay, since all the data points must be accessible before the coefficients can be calculated. Furthermore the non-causal models (and their derived codecs) generally involve a larger degree of computation than the causal models because they cannot be expressed in a fast recursive form.

As stated earlier, these one dimensional models lack some of the essential two dimensional properties of images; in particular the enforced one dimensional scan of a two dimensional image cannot account for correlations in an arbitrary spatial direction within a local area. Thus a one dimensional scan passing over a line or edge feature in an image may treat it as virtually uncorrelated whereas in two dimensions it will be highly correlated along the direction of the feature. Two dimensional DPCM and transform codecs may be derived from causal, non-causal and semi-causal (causal in one axis and non-causal in the other) [51] two dimensional models.

These two dimensional models have the same basic trade off between complexity and compression as their one dimensional counterparts. However, whereas in the case of one dimensional signals such as speech there may be some

justification in the assumption of causality, (i.e. speech generation is essentially a causal process) there is no such justification for images. They are inherently non-causal in the image plane and the imposition of a causal scan across the image will restrict the effectiveness of such a method.

The model presented in this chapter may be seen as a compromise between the simplicity of a causal model and the advantages of a non-causal model. It is a linear signal model which is non-causal in the two dimensions of the spatial plane but causal in a third dimension, the scale or resolution dimension [21] [20] [22]. This permits a fast recursive implementation, from which a simple predictive coder can be derived.

## 2.2. One Dimensional Linear Signal Models

Representing the signal by a sample of size $N$ taken from an infinite one dimensional sequence of data values $x(n)$

$$\{x(n)\ ;\ 0 \le n < N\} = \{x(0), x(1), \cdots, x(N-1)\}$$

a set of predictor coefficients for each $n$ is defined by

$$\{a_i(n)\ ;\ 1 \le i \le R\} = \{a_1(n), a_2(n), \cdots, a_R(n)\}$$

and an innovation term for each $n$ by

$$b(n)w(n)$$

The one dimensional linear signal model can then be defined by the recursive equation

$$x(n) = \sum_{i=1}^{R} a_i(n)x(n-i) + b(n)w(n) \qquad 0 \le n < N \qquad (2.2.1)$$

where each data value is represented as a linear combination of $R$ previous data values plus an innovation term. The values of $\{x(n)\ ;\ -R \le n < 0\}$ used in equation (2.2.1) are unknown, and may be taken as a set of arbitrary initial conditions.

The innovation term $b(n)w(n)$ consists of two parts, $b(n)$, which is a coefficient used to weight the innovation and is often taken to be unity, and $w(n)$ which is the innovation value used to augment or 'update' the estimate of $x(n)$.

The innovation $w(n)$ is modelled as a sample from a zero-mean random process which is uncorrelated with $x(n)$

$$E[w(n)x(n)] = 0 \qquad (2.2.2)$$

The random variable $w(n)$ may be taken from a white noise process

$$E[w(n)w(m)] = \sigma^2\delta_{nm} \qquad (2.2.3)$$

where $\sigma^2$ is the variance of the noise process.

For this model it is easy to show [51] [81] that the minimum variance predictor of $x(n)$ in terms of $\{x(n-i)\ ;\ 1 \le i \le R\}$ is simply the first term of equation (2.2.1)

$$\hat{x}(n) = \sum_{i=1}^{R} a_i(n)x(n-i) \qquad (2.2.4)$$

In general the entropy of the innovation values $w(n)$ will be less than the entropy of the data values $x(n)$. Indeed, if the data are Gaussian the entropy of $w(n)$ is minimised by the choice of the minimum variance predictor. This property, together with the fast implementation afforded by equation (2.2.1) leads to the use of DPCM codecs [37] [78] [79] [4] based on the model. In such a codec, the prediction error $\tilde{w}(n)$ is derived for each $n$ from

$$\tilde{w}(n) = x(n) - \hat{x}(n) \qquad (2.2.5)$$

Note that whereas in the model $w(n)$ is produced by a random process, $\tilde{w}(n)$ in the codec is derived from the signal. A quantiser is generally applied to the prediction errors to give

$$\tilde{w}_q(n) = Q\left[\tilde{w}(n)\right] \qquad (2.2.6)$$

and these quantised approximations to the innovation sequence are then coded, with the causality of the model ensuring that the quantised data sequence $x_q(n)$ can be reconstructed from this data by

$$x_q(n) = \sum_{i=1}^{R} a_i(n)x_q(n-i) + b(n)\tilde{w}_q(n) \qquad 0 \le n < N \qquad (2.2.7)$$

Note that the addition of the quantiser requires the minimum variance predictor to be expressed in terms of $x_q(n)$ rather than $x(n)$ to ensure that the quantisation noise is introduced separately at each datum rather than summing across all the elements, so that equation (2.1.4) becomes

$$\bar{x}(n) = \sum_{i=1}^{R} a_i(n) x_q(n-i) \qquad (2.2.8)$$

The number of recursions of equation (2.2.1) is equal to the length of the data sequence which for applications to an image of dimensions $M$ by $M$ will be equal to $M^2$. The number $R$ of previous data points in the estimate is known as the order of the model.

In the following section, the model is expanded to a two dimensional form which is non-causal in the two spatial dimensions but causal in a third dimension, the scale dimension.

## 2.3. Two Dimensional Linear Multiresolution Signal Models

### 2.3.1. General Linear Multiresolution Model

In this section a general class of two dimensional linear multiresolution signal models is defined. For this class of models it is convenient to define $s(n)$ to be a two dimensional vector representing the image at a particular level of resolution $n$

$$s(n) = \begin{bmatrix} s_{00}(n) & . & . & s_{0M}(n) \\ . & . & . & . \\ . & . & . & . \\ s_{M0}(n) & . & . & s_{MM}(n) \end{bmatrix}$$

and $w(n)$ to be a two dimensional vector of innovation values

$$w(n) = \begin{bmatrix} w_{00}(n) & . & . & w_{0M}(n) \\ . & . & . & . \\ . & . & . & . \\ w_{M0}(n) & . & . & w_{MM}(n) \end{bmatrix}$$

An estimation operator $A(n)$ is a linear operator acting on $s(n)$ and an innovations operator $B(n)$ a linear operator acting on $w(n)$. The two dimensional multiresolution linear signal model is then based on a sequence of the two dimensional image vectors $\{s(n) \; ; \; 0 \le n \le N\}$ representing increasing levels of resolution (decreasing scale), with the full or original image being represented at level $N$ by $s(N)$. The transition from one level to the next is defined by a simple recursive equation

$$s(n) = A(n)s(n-1) + B(n)w(n) \qquad 0 < n \le N \tag{2.3.1}$$

with the initial conditions

$$A(0) = 0 \qquad s(0) = B(0)w(0) \tag{2.3.2}$$

The form of equation (2.3.1) is identical in structure to that of equation (2.2.1) for the one dimensional case, but the variable $n$ now represents scale rather than

spatial position and the elements being predicted are entire images rather than single data values. The linear operator A(n) is effectively a linear predictor of the image at resolution $n$ from the image at resolution $n-1$, and the linear operator $B(n)$ is a linear weighting of the innovations image $w(n)$, determining the innovations made at level $n$ to the estimate produced by $A(n)$. The form of these linear operators can be made clear by rewriting equation (2.3.1) explicitly as

$$s_{xy}(n) = \sum_{pq} A_{xypq}(n) s_{pq}(n-1) + \sum_{rs} B_{xyrs}(n) w_{rs}(n) \qquad (2.3.3)$$

The range of $pq$ and $rs$ may differ if the model represents the levels of resolution by images of different sizes, as in the case of quadtree or pyramid models [20] [22]. As in the one dimensional case, the innovation terms are modelled as a zero mean random process which is independent in $x,y$ and $n$

$$E[w_{xy}(n) w_{pq}(m)] = \sigma^2 \delta_{xp} \delta_{yq} \delta_{nm} \qquad (2.3.4)$$

The innovations vector is also uncorrelated with the signal vector

$$E[w_{xy}(n) s_{pq}(m)] = 0 \qquad m < n \qquad (2.3.5)$$

As in the one dimensional, case the minimum variance predictor for $s(n)$ in terms of $s(n-1)$ is given by the first term of equation (2.3.1) [see appendix A]

$$\hat{s}(n) = A(n) s(n-1) \qquad (2.3.6)$$

In one dimensional models, the number of recursions in equation (2.2.1) was equal to $M^2$ for an $M$ by $M$ image. In the two dimensional models, the number of recursions is given by $N = \log_m(M)$ where $m$ is a scale factor representing the increase in resolution (decrease in scale) between each level of the model. For computational efficiency and simplicity $m$ is usually chosen to be 2. For this value of $m$, the spatial resolution of the image increases by a factor of 4 (in terms of area) between levels $n$ and $n+1$.

The model is non-causal in the two spatial dimensions ($x,y$ in $s_{xy}(n)$), which allows it to take account of local correlations in all spatial directions but causal in the scale dimension ($n$ in $s_{xy}(n)$), allowing for a fast implementation of the estimator $A(n)$. This is the most general form of the model and the rest of the chapter is devoted to looking at particular forms of the model which are achieved by particular definitions of $A(n)$ and $B(n)$.

### 2.3.2. Quadtree Models

Since their introduction in 1975 [101], quadtree techniques have been widely used as a method of processing images over different scales [48]. The quadtree linear signal models which are a subset of the general linear signal models have been thoroughly investigated by Wilson and Clippingdale [21] [20] [22] for their application to image restoration. In the quadtree model, the size of each level of resolution within the model varies by a factor of 2 as shown in fig 2.3.1.

The quadtree is structured by linking the nodes $s_{xy}(n)$ across neighbouring levels. Each parent node at level $n$ has a link to 4 child nodes at level $n+1$, thus the element node $s_{xy}(n)$ is the parent of the set of nodes $\{s_{2x\,2y}(n+1), s_{2x+1\,2y}(n+1), s_{2x\,2y+1}(n+1), s_{2x+1\,2y+1}(n+1)\}$ as shown in the familiar

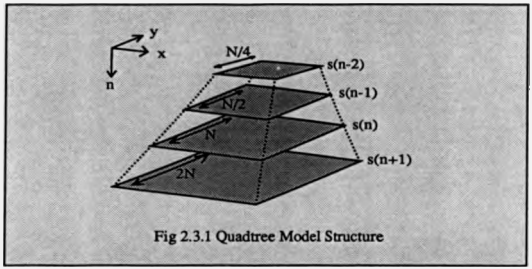



Fig 2.3.1 Quadtree Model Structure

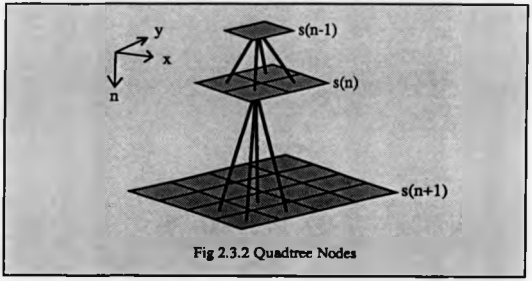



Fig 2.3.2 Quadtree Nodes

diagram of fig 2.3.2.

The area of the image increases by a factor of 4 with each successive level of resolution and thus the indices *rs* in equation (2.3.3) have twice the range of the indices *pq*. The linear estimator $A(n)$ for the quadtree model can be defined by

$$A_{xypq}(n)=\left[\delta_{x\,2p}+\delta_{x\,2p+1}\right]\times\left[\delta_{y\,2q}+\delta_{y\,2q+1}\right] \qquad (2.3.6)$$

i.e. the estimate for each child is simply the value of its parent.

The innovations operator $B(n)$ can be defined by

$$B_{xyrs}(n)=\delta_{xr}\delta_{ys} \qquad (2.3.7)$$

i.e. an innovation is made to each child to update the estimate produced by $A(n)$.

The use of quadtree structures generally involves at least a two-pass algorithm, with the first pass building the structure and the second pass performing some processing. The quadtree structure is built from an image by recursion up the tree from the highest resolution (smallest scale) to the lowest resolution (largest scale) assigning at each level the parent node $s_{xy}(n)$ to be the average of its four child nodes $\{s_{2x\,2y}(n+1), s_{2x+1\,2y}(n+1), s_{2x\,2y+1}(n+1), s_{2x+1\,2y+1}(n+1)\}$.

Quadtrees have been used in both image compression [110] [111] and image restoration [20] [21] [22] applications and have the advantages that they are fast and exact minimum mean squared error estimates can be derived from such models [21]. Invariably, blocking effects occur when these models are used for natural images, which must be overcome if they are to be effective [21].

### 2.3.3. Laplacian Pyramid Models

The Laplacian pyramid model [13] [94] has an identical structure to that of the quadtree in the sense that the size of the image at each level of resolution varies by a factor of 2, and the structure has the same definition for $B(n)$. However, it differs in the definition of $A(n)$ and the manner in which the model parameters are built from an original image. It uses a more refined interpolation function to remove the blocking effects referred to in the previous section.

The estimator $A(n)$ uses an interpolation function (usually an approximation to a Gaussian) to estimate each node at level $n$ from a local set of $R$ by $R$ 'child' nodes at level $n-1$ and can be defined as

$$A_{x,y,n}(n) = \begin{cases} w_g(x-2p, y-2q) & \text{iff } (|x-2p|<R) \wedge (|y-2q|<R) \\ 0 & \text{else} \end{cases} \qquad (2.3.8)$$

where $w_g(\bullet, \bullet)$ is a two dimensional interpolation function.

In order to estimate the model parameters, each node at level $n$ is formed by a low-pass filter function (usually the same function as the interpolation function $w_g(\bullet, \bullet)$) applied to a set of $R$ by $R$ child nodes in a local area of the image at level $n+1$, rather than simply the block average, as in the quadtree model. The child set may include an arbitrary number of nodes, and these child sets may be overlapping (i.e. each child node may be linked to several parent nodes).

Note from these properties that the quadtree model is a subset of these pyramid models which is simpler and more efficient to implement but which does not have the same anti-aliasing properties .

The Laplacian pyramid has been used in image compression [13] by coding the innovation terms which are effectively the difference between successively more low-pass images, i.e. subbands of the image [109] [118]. However in the implementation of [13], the quantisation of these successive sub-bands is performed separately and thus the quantisation noise is summed over the bands. This contrasts with a conventional predictive coder in which the quantised data rather than the original data are used for prediction, with the consequence that only the quantisation error from the current datum appears at the output of the decoder.

## 2.4. Linear Interpolative Multiresolution Models

### 2.4.1. General Structure

The Linear Interpolative Multiresolution (LIM) models are a subset of the general linear multiresolution models which are defined as having certain extra properties. The first property is that they have a constant spatial size across the levels of resolution. Thus rather than the tree or pyramid structure of the two previous models they have more of an 'office block' structure as shown in fig 2.4.1. A further property of the LIM model is that at each level of resolution a fixed subset of the pixels in the image is updated. This updating process consists of adding an innovation to an estimate of the pixel value. Any given pixel is updated at one level of resolution only and once it has been updated its value remains fixed for all higher levels of resolution. Pixel values which have been derived in this manner at the current or previous levels of resolution are used as the basis of the estimator which predicts the values of all pixels not yet updated. Defining the set of pixels which are updated at level $n$ of the resolution structure by $\Lambda_n$

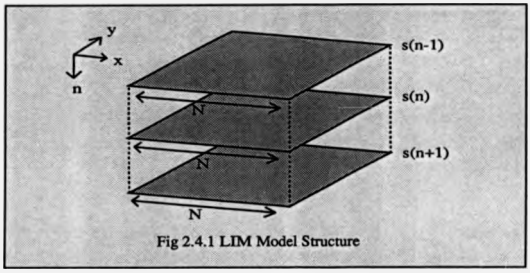


Fig 2.4.1 LIM Model Structure

$$\Lambda_n = \{(x_{n1}, y_{n1}), (x_{n2}, y_{n2}), \cdots, (x_{nr}, y_{nr})\} \qquad (2.4.1)$$

$$0 \le n \le N$$

and the set of all pixels in the image by $\Lambda$

$$\Lambda = \bigcup_{0 \le x,\, y < M} (x, y) \qquad (2.4.2)$$

Then two of the basic conditions defining a LIM model can be expressed as

$$\Lambda = \bigcup_{0 \le n \le N} \Lambda_n \qquad (2.4.3)$$

$$\Lambda_n \cap \Lambda_m = \varnothing \quad \text{iff } n \ne m \qquad (2.4.4)$$

Equation (2.4.3) specifies that every pixel in the image will be updated as part of an innovation set at some level of resolution, i.e. the union of the innovation sets over all levels $n$ of the resolution is the entire image.

Equation (2.4.4) specifies that any pixel which is a member of the innovation sets at level $n$ will not be a member of the innovation sets at any other level $m$, i.e. each pixel will be updated only once.

The distribution of the pixels in the image amongst the sets $\Lambda_n$ defines a specific implementation of the LIM model. This distribution may be such that a fixed set is obtained at each level regardless of the characteristics of a particular image - a homogeneous model - or the sets may be dependent on particular local image characteristics - an inhomogeneous model. The particular distribution which forms the Recursive Binary Nesting Model is defined in section 2.5.

The innovations sets can be used to provide the conditions for a LIM model on the linear operators $A(n)$ and $B(n)$. On the innovations operator $B(n)$ it imposes the condition

$$B_{xyrs}(n) = 0 \quad \text{if} \quad \Big[(x,y) \notin \Lambda_n\Big] \vee \Big[(r,s) \notin \Lambda_n\Big] \tag{2.4.5}$$

This is the explicit specification that an innovation is made only once for each pixel. On the estimation operator $A(n)$ it imposes two conditions

$$A_{xypq}(n) = \delta_{xp}\delta_{yq} \quad \text{if} \quad (x,y) \notin \bigcup_{m<n} \Lambda_m \tag{2.4.6}$$

$$A_{xypq}(n) = 0 \quad \text{if} \quad (p,q) \notin \bigcup_{m<n} \Lambda_m \tag{2.4.7}$$

Equation (2.4.6) specifies the condition that once a pixel has been updated at level $n$ it is left unaltered by the estimator at all higher levels. Equation (2.4.7) specifies the condition that the estimation is based only on pixels which have already been updated at a lower level of resolution. This is a requirement of the causality in $n$ of the model, necessary for the derivation of predictive coders.

From the definition of the innovations operator, it can be seen that a fully interpolated image at level $n$ is given by $A(n+1)s(n)$, $0 \le n < N$, since $s(n)$ only contains point innovations, and these innovations are to be interpolated across a local area. The reason for formulating the innovations operator this way is its mathematical simplicity.

An important property of the LIM models is that every level $s(n)$ can be constructed directly from the full image at level $s(N)$ by a simple linear operator. This property underlies the efficiency of the predictive coding strategy based on the model, and it is useful to prove this result formally.

**Theorem:**

For all n there exists a linear operator $c(n, N)$ such that

$$c(n, N)s(N) = s(n) \tag{2.4.8}$$

**Proof:**

Define an index limiting operator $I(\Lambda_n)$ by

$$\left[I\left(\Lambda_n\right)\right]_{xypq} = \begin{cases} \delta_{xp}\,\delta_{yq} & \text{iff} \quad (x,y) \in \Lambda_n \\ 0 & \text{else} \end{cases} \tag{2.4.9}$$

$I(\Lambda_n)$ is a linear operator which selects only those pixels which are elements of the set $\Lambda_n$, leaving their value unaltered and sets the value of all other pixels to zero. From equations (2.4.2) and (2.4.3), the identity operator $I$ can be expressed as

$$I = \sum_{n=0}^{N} I(\Lambda_n) \tag{2.4.10}$$

and from equation (2.4.6) it follows that

$$I(\Lambda_n)s(n) = I(\Lambda_n)s(m) \qquad n < m \tag{2.4.11}$$

which is an alternative way of stating that the values of the pixels in the set $\Lambda_n$ remain constant at all levels $m$ greater than $n$. Now suppose there exists a linear operator $c(n, m)$ such that

$$c(n, m)s(m) = s(n) \qquad n < m \tag{2.4.12}$$

Then it follows from equation (2.3.1) that

$$s(n+1) = A(n+1)c(n, m)s(m) + B(n+1)w(n+1) \tag{2.4.13}$$

and then from equations (2.4.4) and (2.4.11) that

$$s(n+1) = (I - I(\Lambda_{n+1}))A(n+1)c(n, m)s(m) + I(\Lambda_{n+1})s(m) \tag{2.4.14}$$

so that $s(n+1)$ can be obtained from $s(m)$, $m>n$ using the operator given by equation (2.4.15).

$$c(n+1, m) = (I - I(\Lambda_{n+1}))A(n+1)c(n, m) + I(\Lambda_{n+1}) \qquad (2.4.15)$$

But from equations (2.3.2) and (2.4.11) it follows that

$$s(0) = I(\Lambda_0)s(m) \qquad (2.4.16)$$

and hence that

$$c(0, m) = I(\Lambda_0) \qquad (2.4.17)$$

Hence from equations (2.4.15) and (2.4.17), $c(n, N)$ can be constructed for any $n$.

□

A property of LIM models is that every pixel $s_{xy}(N)$ in the final image can be expressed as a linear combination of a set of other pixels plus an innovation as shown in equation (2.4.18).

$$s_{xy}(N) = \sum_{(p,\, q) \in \bigcup_{m>n} \Lambda_m} \alpha_{xypq} s_{pq}(N) + v_{xy} \qquad (x,y) \in \Lambda_n \qquad (2.4.18)$$

where $v_{xy}$ is independent of $s_{pq}(N)$ i.e.

$$E[v_{xy}s_{pq}(N)] = 0 \qquad (x,y)\in \Lambda_n \qquad (p,q)\in \bigcup_{m<n} \Lambda_m \qquad (2.4.19)$$

This is the linear interpolative property of the model from which it takes its name. However, it is the specific definitions of the sets $\Lambda_n$ which give it the multiresolution structure and characteristics. For a multiresolution model, the sets are defined to have two specific properties: periodicity and locality. For n>1 the linear operators $A(n)$ and $B(n)$ are defined to have a periodic structure, i.e. there exists $m_n$ such that

$$A_{(x+im_n)(y+jm_n)(p+sm_n)(q+jm_n)}(n) = A_{xypq}(n) \qquad (2.4.20)$$

So for a given level $n$ the operator may be defined by the top left block of size $m_n$ by $m_n$ and the rest is simply a periodic repetition as given by (2.4.20). In addition, it is defined to have a locality property, in that there is some distance $r_n > 0$ for which

$$A_{xypq}(n) = 0 \qquad \left[\,|x-p|>r_n\right] \vee \left[\,|y-q|>r_n\right] \qquad (2.4.21)$$

i.e. the estimator is based only on a local area of the image.

Moreover as $n$ increases there is a passage from global to local structures in that $m_n > m_{n+k}$ and $r_n > r_{n+k}$ for $k>0$, as shown in fig 2.4.2, and the cardinality of the innovation set increases (i.e. $\mathrm{CARD}(\Lambda_{n+k}) > \mathrm{CARD}(\Lambda_n)$ ). It is these properties which give the model a multiresolution element and provide the basis for the codecs derived from it.

### 2.4.2. Summary of Homogeneous LIM Model Properties

The following is a list of the essential properties of the LIM models considered above

(1) Exclusivity

$$\Lambda_n \cap \Lambda_m = \varnothing \quad \text{iff} \quad n \neq m \tag{2.4.22}$$

(2) Completeness

$$\Lambda = \bigcup_{0 \le n \le N} \Lambda_n \tag{2.4.23}$$

(3) Single innovation

$$B_{xyrs}(n) = 0 \quad \text{if} \quad \Big[(x, y) \notin \Lambda_n\Big] \vee \Big[(r, s) \notin \Lambda_n\Big] \tag{2.4.24}$$

(4) No change

$$A_{xypq}(n) = \delta_{xp}\delta_{yq} \quad \text{if} \quad (x, y) \in \bigcup_{m<n} \Lambda_m \tag{2.4.25}$$

(5) Innovations basis

$$A_{xypq}(n) = 0 \quad \text{if} \quad (p, q) \notin \bigcup_{m<n} \Lambda_m \tag{2.4.26}$$

(6) Periodicity

$$A_{(x+im_n)(y+jm_n)(p+im_n)(q+jm_n)}(n) = A_{xypq}(n) \qquad (2.4.27)$$

$$B_{(x+im_n)(y+jm_n)(p+im_n)(q+jm_n)}(n) = B_{xypq}(n)$$

(7) Locality

$$A_{xypq}(n) = 0 \qquad \text{if} \quad \left[\, |x-p| > r_n \right] \vee \left[\, |y-q| > r_n \right] \qquad (2.4.28)$$

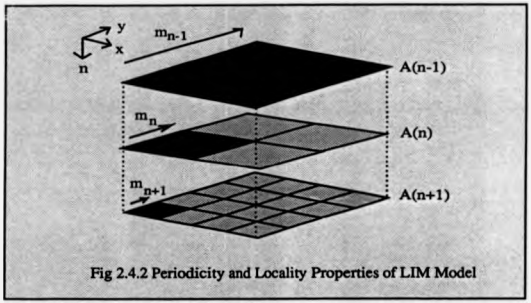



Fig 2.4.2 Periodicity and Locality Properties of LIM Model

Finally, it should be noted that all these properties assume the model is homogeneous. However, to derive efficient codecs an inhomogeneous model is required. In the inhomogeneous model, the locality property of fig 2.4.2 is selected at different scales across the image, according to the image

characteristics, giving the structure shown in fig 2.4.3 where different areas have a different degree of locality.

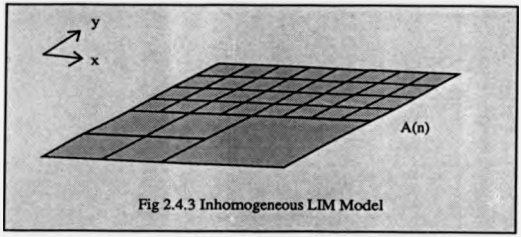



Fig 2.4.3 Inhomogeneous LIM Model

This clearly breaks the periodicity property. There is more on inhomogeneous models in chapter 4, but it should be noted that the more efficient coders derived in chapter 3 are inhomogeneous.

### 2.5. Recursive Binary Nesting LIM Model

One of the most efficient and computationally simple forms of the LIM model is the Recursive Binary Nesting (RBN) model. The RBN algorithm was developed at BTRL [107] as a means of compressing single frame images and has been used in a number of coding schemes [6] [24] [25] [26]. The simplest form of the RBN model uses a linear interpolator based on four corner pixels of a square block as the estimator $A(n)$. The corner pixels of these blocks form the elements of the sets $A_n$ and each recursion into equation (2.2.1) involves splitting each block into four edge-sharing subblocks with each subblock having one quarter of the the area of the old block.

The new pixels required to define the corners of these new subblocks form the elements of the innovation set $\Lambda_n$ as illustrated in fig 2.5.1 and can be defined for an image of size $2^N+1$ by $2^N+1$ as

$$\Lambda_n = \left\{ \bigcup_{\substack{0<i\le 2^{n-1} \\ 0\le j\le 2^n}} (i2^{N-(n-1)} - 2^{N-n},\ j2^{N-n}) \right\} \cup \left\{ \bigcup_{\substack{0\le i\le 2^n \\ 0<j\le 2^{n-1}}} (i2^{N-n},\ j2^{N-(n-1)} - 2^{N-n}) \right\}$$

$$0 < n \le N \qquad (2.5.1)$$

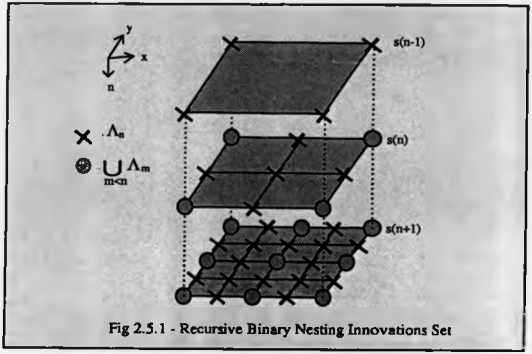



Fig 2.5.1 - Recursive Binary Nesting Innovations Set

The initial set $\Lambda_0$ is the four corner pixels of the image $\{(0, 0), (0, 2^N), (2^N, 0), (2^N, 2^N)\}$. Each successive innovation set $\Lambda_n$ is the set of pixels required to define the corners of the new edge-sharing quadrants of the blocks in the previous image $s(n-1)$. Thus the area of the blocks is reduced by a factor of four at each level as $n$ increases. The estimating operator $A(n)$ is a bilinear interpolation for each block based on its four corner pixels as shown

in fig 2.5.2, which shows the top left hand block $0 \le x,y \le 2^{N-n}$ of the image $s(n)$ at level $n$. The value of each pixel in the block (inclusive of the boundaries) shown in fig 2.5.2 is given by

$$\hat{s}_{xy}(n) = \frac{x}{2^{N-n}} f_y + (1 - \frac{x}{2^{N-n}}) e_y \qquad (2.5.2)$$

with

$$f_y(n) = \frac{y}{2^{N-n}} d + (1 - \frac{y}{2^{N-n}}) c \qquad (2.5.3)$$

and

$$e_y(n) = \frac{y}{2^{N-n}} b + (1 - \frac{y}{2^{N-n}}) a \qquad (2.5.4)$$

giving

$$\hat{s}_{xy}(n) = \frac{1}{2^{2N-2n}} \left[ a(2^{2N-2n} - x2^{N-n} - y2^{N-n} + xy) \right] + \qquad (2.5.5)$$

$$\frac{1}{2^{2N-2n}} \left[ b(y2^{N-n} - xy) \right] + \frac{1}{2^{2N-2n}} \left[ c(x2^{N-n} - xy) \right] + \frac{1}{2^{2N-2n}} \left[ d(xy) \right]$$

On any edge of the block this reduces to a single interpolation between the two corner pixels defining the edge. Thus neighbouring blocks share an identical interpolation along their shared edges giving a continuity of pixel intensities across block boundaries. However, with the bilinear interpolation there is generally no continuity in the intensity gradient across block boundaries, which can lead to blocking artefacts in codecs derived from the model.

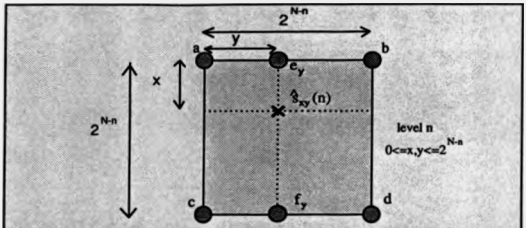



Fig 2.5.2 - Bilinear Block Interpolation

From equation (2.5.5) the linear operator $A(n)$ can be derived for the RBN model by defining it for the top left hand block $0 \le x,y \le 2^{N-n}$ and noting the periodicity of equation (2.4.20)

$$A_{xypq}(n) = \begin{cases} 2^{2n-2N}(2^{2N-2n}-x2^{N-n}-y2^{N-n}+xy) & \text{iff } (p=0) \wedge (q=0) \\ 2^{2n-2N}(y2^{N-n}-xy) & \text{iff } (p=0) \wedge (q=2^{N-n}) \\ 2^{2n-2N}(x2^{N-n}-xy) & \text{iff } (p=2^{N-n}) \wedge (q=0) \\ 2^{2n-2N}(xy) & \text{iff } (p=2^{N-n}) \wedge (q=2^{N-n}) \\ 0 & \text{else} \end{cases} \qquad 0 \le x,y \le 2^{N-n} \tag{2.5.6}$$

Since the innovations operator $B(n)$ for the RBN model makes a single unweighted innovation at each point in the innovations set, it can be defined as

$$B_{xypq}(n) = \delta_{xp}\delta_{yq} \qquad (x,\ y) \in \Lambda_n \tag{2.5.7}$$

Having defined $A(n)$ and $B(n)$ for the LIM RBN model, the image $s(N)$ for the model can be built from equations (2.3.1) and (2.3.2). This requires that the variance of the zero mean white noise process $w(n)$ is specified. The variance of this process can be estimated from a particular image or set of images.

It is assumed that the variance at each level $n$ of resolution is different and that these variances can be estimated from the pixels in the innovation sets $\Lambda_n$ at each level. The innovation terms $\hat{w}_{xy}(n)$ at level $n$ are derived from a particular image, and assumed to have zero mean. The variance at level $n$ of the random process producing $w_{xy}(n)$ is then estimated from equation (2.5.8)

$$V_n = \frac{1}{\text{CARD}[\Lambda_n]} \sum_{(x,y)\in\Lambda_n} \left[\hat{w}_{xy}(n)\right]^2 \qquad (2.5.8)$$

A table of variances estimated from the innovation terms for the GIRL image is given in table 2.5.1. Note that they are generally decreasing with level from level 3 onwards. The variances at levels 1, 2 and to some extent 3, are based on very few samples (5, 16 and 56 respectively), and are thus to some extent unreliable estimates. Also, given that the pixel variance for the image is 2405.78, these three estimates seem to be quite high. This is probably because although the model assumes some correlation between pixels even at the largest scales, in the actual image there is virtually no correlation between pixels at these levels. This implies that, not surprisingly, the model does not realistically represent the image at larger scales. Photo 2.1 shows the image $s(N)$ derived from the model using a random process with these variances.

Certain correlation properties of the LIM RBN model can be determined by considering a model in which each pixel in the final image $s(N)$ has unit

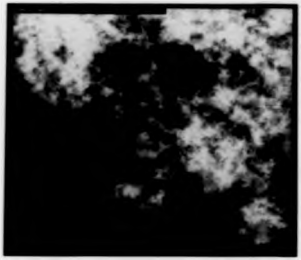


Photo 2.1 - Homogeneous Isotropic

LIM RBN Model

| level $n$ | variance | level $n$ | variance |
|---|---|---|---|
| 0 | - | 5 | 1056 |
| 1 | 1764 | 6 | 535 |
| 2 | 1670 | 7 | 235 |
| 3 | 2764 | 8 | 96 |
| 4 | 1701 | 9 | 27 |

Table 2.5.1

variance

$$\mathrm{VAR}[s_{xy}(N)] = 1 \tag{2.5.9}$$

By explicitly expanding the recursions of the model, it is possible to define any pixel $s_{xy}(N)$ in the image as a linear combination of the innovation terms $v_{pq}$ of a set of parent nodes $\Lambda_{xy}^{P}$ as shown in equation (2.5.10). Note that the four

elements of the set $\Lambda_0$ are considered to be uncorrelated.

$$s_{xy}(N) = \sum_{(p,q)\in\Lambda_{xy}^{P}} \alpha_{pq} v_{pq} + v_{xy} \qquad (2.5.10)$$

If for a given pixel $(x,y)$, the set $\Lambda_{xy}^{P}$, the terms $\alpha_{pq}$ and the variances of the terms $v_{pq}$ are known, then, because the innovation terms are independent, the variance of the term $v_{xy}$ can be calculated from equation 2.4.10.

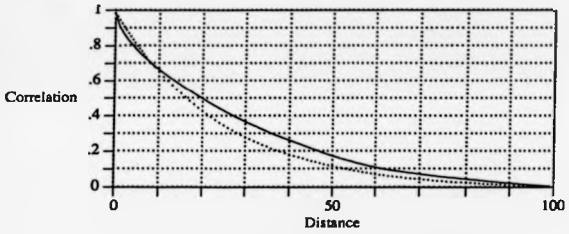
$$\mathrm{VAR}[s_{xy}(N)] = 1 = \sum_{(p,q)\in\Lambda_{xy}^{P}} \alpha_{pq}^{2}\,\mathrm{VAR}[v_{pq}] + \mathrm{VAR}[v_{xy}] \qquad (2.5.11)$$

The parent set $\Lambda_{xy}^{P}$ and the relation factors $\alpha_{pq}$ can be derived by a recursive algorithm. The variances of the terms $v_{pq}$ can be calculated recursively within this algorithm, since for any pixel $(x,y)\in\Lambda_n$ all the elements in $\Lambda_{xy}^{P}$ will also be in $\bigcup_{m<n}\Lambda_m$, and their variances will already have been calculated. Thus it is possible to calculate the variances of the innovation terms for every pixel in the image.

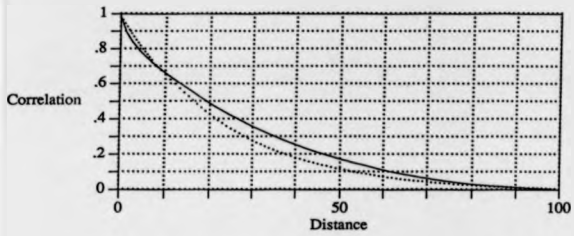
Additionally, given two parent sets $\Lambda_{x_1,y_1}^{P}$ and $\Lambda_{x_2,y_2}^{P}$ for any two pixels $(x_1,y_1)$, and $(x_2,y_2)$, together with the relation factors and the variances of the innovation terms for the elements of these sets, it is possible to calculate the correlation between the two pixels. Since the innovation terms are independent, the correlation is simply given by equation 2.5.12

$$\rho_{x_1,y_1,x_2,y_2} = \frac{\sum_{(p,q)\in\Lambda_{x_1,y_1}^{P}} \sum_{(r,s)\in\Lambda_{x_2,y_2}^{P}} \delta_{pr}\delta_{qs}\alpha_{pq}\alpha_{rs}\mathrm{VAR}[v_{pq}]}{\sqrt{\mathrm{VAR}[v_{x_1,y_1}]}\sqrt{\mathrm{VAR}[v_{x_2,y_2}]}} \qquad (2.5.12)$$

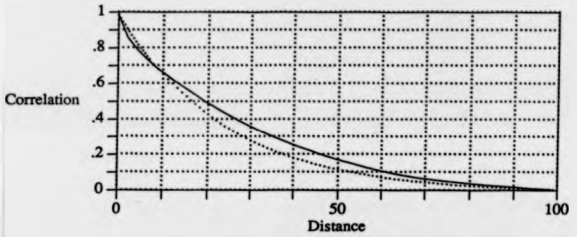
Performing this computation on a 129 by 129 image and averaging the correlations over the image to give the correlation as a function of the Euclidean distance between the two pixels gives a graph of the one dimensional correlation function for the model. Noting that a 129 by 129 image corresponds to level 2 in table 2.5.1, and that the variances in the table suggest that the pixels are virtually uncorrelated down to level 3 or 4, it was decided to treat the pixels down to level 3 (i.e. 64 by 64) as independent, and thus the function is calculated with no correlation between the pixels in sets $\Lambda_0$ and $\Lambda_1$. Graph 2.5.1 shows this function for correlations along the direction of the x-axis (or y-axis since the functions are identical). Graph 2.5.2 shows this function for correlations along the direction of the vector $\begin{bmatrix}1\\1\end{bmatrix}$ (i.e. along the diagonal). Graph 2.5.3 shows this function for correlations along the direction of the vector $\begin{bmatrix}2\\1\end{bmatrix}$.




Graph 2.5.1 - RBN Correlation Structure along $\begin{bmatrix}1\\0\end{bmatrix}$




Graph 2.5.2 - RBN Correlation Structure along $\begin{bmatrix}1\\1\end{bmatrix}$




Graph 2.5.3 - RBN Correlation Structure along $\begin{bmatrix}2\\1\end{bmatrix}$

The interpolation estimator $A(x)$ given in equation (2.5.6) has a slight bias towards the $x$ and $y$ axis, but it is essentially isotropic, as can be seen in the model shown in photo 2.1, and in graphs 2.5.1-2.5.3. Thus this model is referred to as the isotropic LIM RBN Model, to distinguish it from the anisotropic model described in section 4. It can be seen directly from graphs 2.5.1 to 2.5.3 that the model possesses the long range correlations typical of many

natural images. For the sake of comparison, a conventional Gaussian model, the separable Gauss-Markov model [8] [51] for the same image has the correlation function

$$R(x,y) = \rho^{|x|}\rho^{|y|} \tag{2.5.13}$$

where $\rho = 0.96$ is a typical value for an image of size 128 by 128. This function is given as a dotted line in graphs 2.5.1 to 2.5.3. Note that the RBN model correlation falls more rapidly at small distances but more slowly at longer distances. Arguably this provides a better fit to the correlation properties of natural images than does the Gauss-Markov model.

If the estimator $A(n)$ provides an 'acceptable' estimate of the original image at level $n$ for a given block it is not necessary to split that block or provide any innovations within it. This property is used in codecs derived from the model to improve the compression. However, it negates some of the equations and properties defined in this section since the operators $A(n)$ and $B(n)$ vary from image to image. This inhomogeneity is dealt with further in chapter 4. In summary, the new models are attractive computationally, due to their simple recursive structure, and provide an effective basis for modelling the type of long range statistical structure which is apparent in many natural images.

# 3. Multiresolution Predictive Coding

## 3.1. Recursive Binary Nesting Codec

### 3.1.1. Introduction

The basic Recursive Binary Nesting codec involves a hierarchical segmentation of the image into edge sharing blocks as shown in fig 3.1.1.

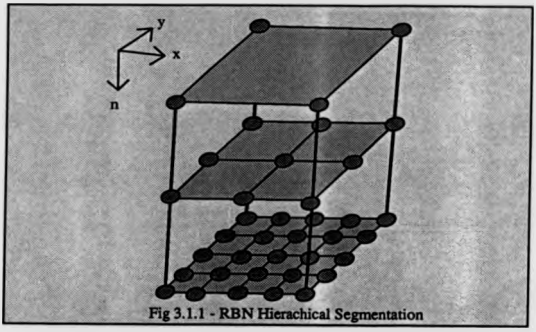



Fig 3.1.1 - RBN Hierachical Segmentation

The intensities $\hat{s}_{xy}(n)$ of the pixels within each block at level $n$ (including the boundary pixels) are represented by a bilinear interpolation from the intensities of the four corner pixels as shown in fig 3.1.2, and described in section 2.5.

These estimates $\hat{s}_{xy}(n)$ are compared to the intensities of the pixels in the original image and a quality measure is applied to the differences between the two in

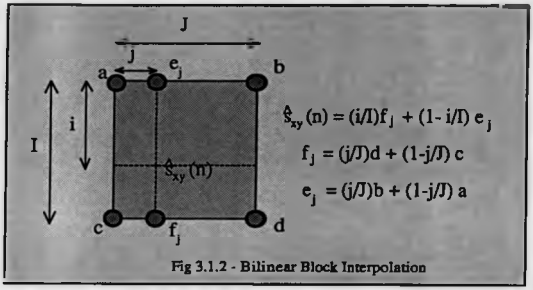



Fig 3.1.2 - Bilinear Block Interpolation

order to determine whether the estimates adequately represent the original image. If the quality measure is satisfied, then no further processing is done on the block and the pixels in the block are represented by the estimate. This process is referred to as termination. If the block fails the quality measure then it is split into four edge sharing quadrants as shown in fig 3.1.3.

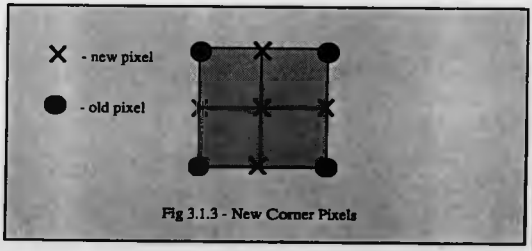



Fig 3.1.3 - New Corner Pixels

Five new pixels are needed at such a split to define the corners of the new blocks and these five pixels are coded as prediction errors between the original values and the estimate provided by the bilinear interpolation. Note that because the blocks are edge-sharing, each pixel on an edge may have already been coded during the processing of a neighbouring block as shown in fig 3.1.4 where only three of the five pixels require coding.

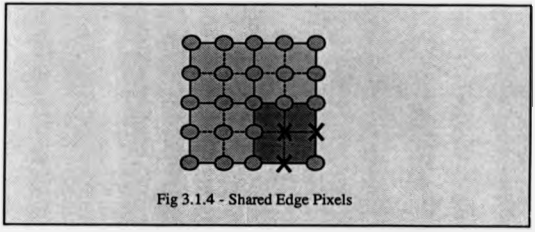


Fig 3.1.4 - Shared Edge Pixels

The algorithm gets the 'recursive' part of its name from the order in which it deals with the processing of the four subblocks at any level. Each separate block is processed completely including all levels of subblocks within the block before the algorithm proceeds to the next block. This sequence for processing the blocks is demonstrated in fig 3.1.5. The process is initialised by coding the four corner pixels of the image and treating the whole image as one block. A schematic flowchart for the basic RBN algorithm is shown in fig 3.1.6. The order in which the four subblocks are processed may be fixed as shown in fig 3.1.5, or it may be altered at each level of recursion as shown in fig 3.1.7. This gives a scan across the image which is spatially somewhat more local than the fixed RBN scan, spatial locality being an important property in the estimation of

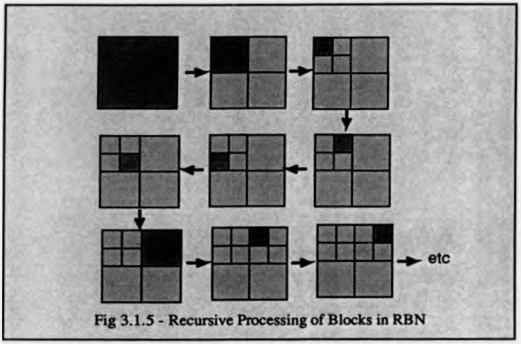


Fig 3.1.5 - Recursive Processing of Blocks in RBN

non-stationary properties of the image such as the probability distribution of the prediction errors (as described in section 3.2). Across each level $n$ the scan across the blocks is effectively a Peano-scan [82] [117], but because of the multiresolution structure it is not true to say that the sequence in which the pixels are coded is a Peano scan. The Peano-RBN scan has four ways of ordering the four subblocks. For each of these ways the ordering at the next level of resolution can be defined for each subblock as shown in fig 3.1.8.

### 3.1.2. Prediction Errors and Noiseless Coding

From section 2.2 the minimum variance predictor for $s(n)$ from $s(n-1)$ is given by

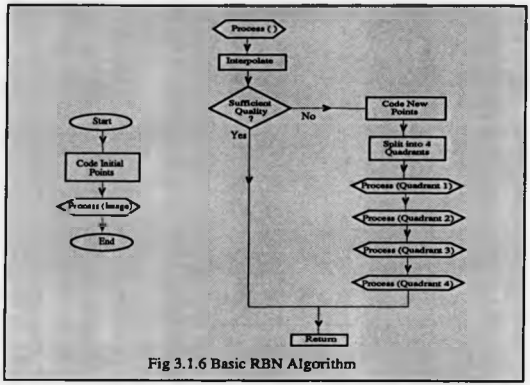



Fig 3.1.6 Basic RBN Algorithm

$$\hat{s}(n \mid n-1) = A(n)s(n-1) \qquad n>0 \qquad (3.1.1)$$

and therefore the prediction errors for the pixels in the innovation set $\Lambda_n$ at level $n$ are given by

$$e_{xy}(n) = \left[ s_{xy}(n) - \hat{s}_{xy}(n \mid n-1) \right] \qquad n>0 \qquad (x,y) \in \Lambda_n \quad (3.1.2)$$

Consider a noiseless coding of the image, i.e. one where the decoder can reconstruct the image exactly, with no quantisation and no termination of the segmentation process. The initial set of pixels $\Lambda_0$ are coded at a cost of 8-bits per

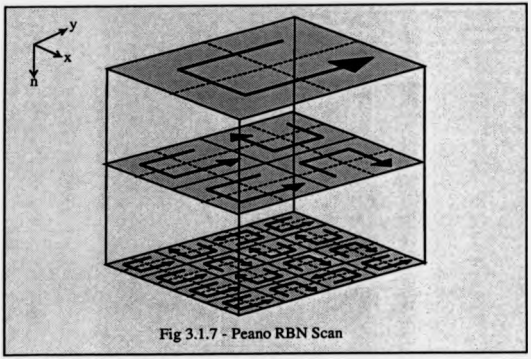



Fig 3.1.7 - Peano RBN Scan

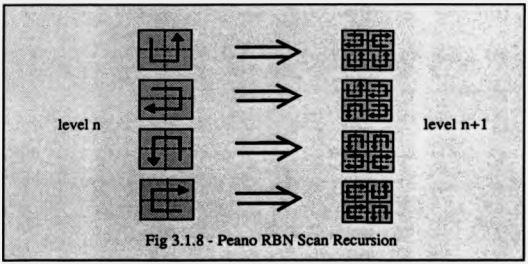



Fig 3.1.8 - Peano RBN Scan Recursion

pixel, i.e. the values $s_{xy}(0)$, $(x,y)\in \Lambda_0$ are coded. Then the elements of $\{e_{xy}(n); (x,y)\in \Lambda_n, 1 \le n \le N\}$ are coded according to either the fixed RBN scan or the Peano RBN scan sequences. If the probability distribution $P(e_{xy}(n))$ over all $(x,y)\in \Lambda_n$, $1 \le n \le N$, is calculated for a particular image of

size $M$ by $M$, then the stationary RBN entropy $E_s$ for the noiseless coder is given by

$$E_s = \frac{1}{M^2}\left[4\times 8 - \sum_{n=1}^{N}\sum_{(x,y)\in A_n}\log_2 P(e_{xy}(n))\right] \quad \text{(bpp)} \qquad (3.1.3)$$

Table 3.1.1 shows this entropy for a variety of images and can be compared with the zeroth order and differential entropies given in tables 1.2.1 and 1.2.2.

| Image | Entropy $E_s$ (bpp) |
|---|---|
| GIRL | 4.42 |
| BOATS | 4.74 |
| LAKE | 5.74 |

Table 3.1.1

Equation 3.1.3 and table 3.1.1 assume that the probability distribution $P(e_{xy}(n))$ of the prediction errors $e_{xy}(n)$ is stationary across $x,y$ and $n$. However, for natural images the distribution is likely to be non-stationary both in the spatial dimensions and the scale dimension. Section 3.2 deals with a non-stationary entropy coder in more detail. As a simple comparison, table 3.1.2 shows the entropy $E_n$ achieved by calculating the probability distribution $P_n(e_{xy}(n))$ separately for each scale as shown in equation (3.1.4).

$$E_n = \frac{1}{M^2}\left[4\times 8 - \sum_{n=1}^{N}\sum_{(x,y)\in A_n}\log_2 P_n(e_{xy}(n))\right] \quad \text{(bpp)} \qquad (3.1.4)$$

| Image | Entropy $E_s$(bpp) |
|---|---|
| GIRL | 4.37 |
| BOATS | 4.70 |
| LAKE | 5.69 |

Table 3.1.2

Note that the entropies are slightly lower than those of table 3.1.1 indicating that there is indeed some variation with respect to scale. Finally table 3.1.3 gives a noiseless coding result (rather than pure entropy calculations) using the spatially and scale non-stationary entropy coder of section 3.2. Here a very small difference in the locality between the fixed RBN and Peano RBN scans shows up.

| Image | Fixed RBN Bit Rate (bpp) | Peano RBN Bit Rate (bpp) |
|---|---|---|
| GIRL | 4.29 | 4.27 |
| BOATS | 4.61 | 4.59 |
| LAKE | 5.68 | 5.65 |

Table 3.1.3

Comparing the stationary entropies of table 3.1.1 and the scale and space nonstationary rates of table 3.1.3 it can be seen that the nonstationarity gives a percentage saving for the GIRL of $\frac{4.42-4.27}{4.42} \times 100 = 3.4\%$, for the BOATS of $\frac{4.75-4.59}{4.74} \times 100 = 3.2\%$, and for the LAKE image of $\frac{5.74-5.65}{5.74} \times 100 = 1.6\%$.

### 3.1.3. Quantisation

A quantising function $Q_n(\bullet)$ (which will in general be nonlinear) acts on the prediction errors $e_{xy}(n)$ to give the quantised prediction errors $e^q_{xy}(n)$

$$e^q_{xy}(n) = Q_n(e_{xy}(n)) \tag{3.1.5}$$

The quantiser is defined as a function of $n$ so that it can vary with scale. It is well established [68] that quantisation errors introduced are less visible at higher spatial frequencies, and in section 3.1.4, scale selection is introduced, whereby areas of high spatial frequencies are represented at small scales, whereas areas of low spatial frequencies are represented at larger scales. Hence the quantisation can be coarser at smaller scales than it can at larger scales. The effect of the quantiser on the image is to reduce the bit rate whilst introducing some quantisation noise given by the two dimensional array $v_q(n)$. From equation (3.1.2) the quantised image at level $n$ is given by

$$\begin{aligned} s^q(n) &= e^q(n) + \hat{s}(n \mid n-1) \\ &= s(n) + v^q(n) \end{aligned} \tag{3.1.6}$$

This quantised image will be the basis for the estimator $A(n+1)$ at the next level of recursion so that equation (3.1.1) must be altered to

$$\hat{s}(n \mid n-1) = A(n)s^q(n-1) \tag{3.1.7}$$

Only those prediction errors $e_{xy}(n)$ which are in the set $\Lambda_n$ are quantised at level $n$ so that an additional condition on $Q_n(\bullet)$ is

$$(Q_n(e(n)))_{xy} = \begin{cases} e^q_{xy}(n) & \text{if} \quad (x,y) \in \Lambda_n \\ e_{xy}(n) & \text{if} \quad (x,y) \notin \Lambda_n \end{cases} \tag{3.1.8}$$

The choice of $Q_n(\cdot)$ will determine the extent and effect of the quantising error. There are well established theories on optimising such quantisers with respect to a mean squared error criterion [70]. The mean squared error measure is not, however, a good measure of the perceived effect of such errors. Thus the quantising functions should be judged by their effect on the human perception of the image. The quantisation function used in this codec was derived by modifying an optimal Max quantiser, based on judgement of visual quality.

### 3.1.4. Termination or Scale Selection

In order to introduce termination into the RBN scan, a quality measure must be defined. In this work, the simple maximum error criterion defined by equation (3.1.9) was found to be effective. This specifies a decision variable for a block (i,j) at level $n$, $d_{ij}(n)$ as

$$d_{ij}(n) = \begin{cases} 1 & \text{iff } \left[\max_{\Psi} |e_{xy}(n)|\right] \le d_n \\ 0 & \text{else} \end{cases} \qquad d_n \ge 0 \qquad (3.1.9)$$

where

$$\Psi = \left\{ \bigcup_{\substack{L_i \le x \le U_i \\ L_j \le y \le U_j}} (x,y) \right\} \cap \left\{ \bigcup_{(x,y) \notin \bigcup_{m=n}^{N} \Lambda_m} (x,y) \right\} \qquad (3.1.10)$$

and if $d_{ij}(n) = 1$ then the block $(i,j)$ is not subdivided. $L_i$, $L_j$, $U_i$, $U_j$ are the lower and upper spatial indexes for a block $(i,j)$ in the image at resolution $n$. A

better measure may well exist, but it would be more complex, to take into account the perceptual mechanisms discussed in section 4.4.

Equation (3.1.9) states that the block splitting decision variable $d_{ij}(n)$ is equal to 0 (i.e. the block is split) if the maximum prediction error of any pixel within the block which has not yet been coded is greater than some level dependent threshold $d_n$. The threshold $d_n$ is made level dependent since it allows for the possibility of having larger errors at small block sizes where they are less perceptually noticeable. While $d_n$ will increase with $n$, it is neither simple to define any optimal function for this scale variability nor what quality of image any given $d_n$ will give, except within fairly broad bands. Thus the choice of $d_n$ for a given quality of image was based on a series of test runs of the coder and a subjective judgement of the quality it achieved.

The indices $i$, $j$ of the splitting decision variable $d_{ij}(n)$ will in general not form a regular lattice, since decisions for the subblocks of any block which is not split are never required. The splitting decision variable is incorporated into the coding algorithm as a binary variable which is coded after each splitting decision is made by the coder. At the decoder the value of this variable is determined and used to determine the splitting decision.

### 3.2. Entropy Coding

The basic isotropic RBN coding algorithm produces two sets of data,

1) $\{e^q_{xy}(n)\,,\ (x,y)\in \Lambda_n\,,\ 1 \le n \le N\}$: the quantised prediction errors.

2) $\{d_{ij}(n)\}$: the addressing information for the block splitting decisions.

In order to minimise the number of bits required to code these data sets, an entropy coder is used. The entropy coder used in this work is a binary arithmetic coder with a non-stationary estimation of the probability distribution called the Q-CODER, which was developed by IBM [73] [74] [75] for image compression. This section provides an account of the theory of arithmetic entropy coding and a brief description of the Q-CODER. The results of incorporating the Q-CODER within the RBN algorithm are then presented.

### 3.2.1. Arithmetic Coding

Consider a data stream $X$ consisting of a sequence of $M$ discrete data items

$$X = \{x_0, \cdots, x_m, \cdots, x_M\} \tag{3.2.1}$$

Each data item $x_m$ can take one of $N$ possible values with each value assigned a unique symbol $i_n$.

The set of all possible data values $\{i_n\}$ is known as the alphabet for the data stream. Given that the cardinality of the set $\{i_n\}$ is $N$, then a simple set of binary codes can be devised for each symbol with a fixed number of bits per symbol $\left\lceil \log_2(N) \right\rceil$ as shown in example 3.2.1.

Of course, this is not necessarily the most efficient set of codes. It may be possible to achieve a compression over the simple binary code by using a variable number of bits per symbol. If the probability of a symbol $i_n$ is given by $P(i_n)$ then the set $\{P(i_n)\}$ is the probability distribution for the alphabet $\{i_n\}$. If $P(i_n)$ varies with $i_n$, i.e. the distribution is non-uniform, then it is possible to achieve a compression over the simple binary code by devising a code which assigns

**Example 3.2.1**

$$N=6 \quad , \quad \log_2(N) = 2.4, \quad \lceil \log_2(N) \rceil = 3$$

| Symbol | Simple Binary Code |
|---|---|
| $i_0$ | 000 |
| $i_1$ | 001 |
| $i_2$ | 010 |
| $i_3$ | 011 |
| $i_4$ | 100 |
| $i_5$ | 101 |

fewer bits to higher probability symbols and more bits to lower probability symbols.

The theory of determining codewords according to the probability distribution of the alphabet is known as entropy coding and a great deal of work has been done on the subject [96] [47] [1] [52] [120] [27] [89] [88] [62] [65] [116] [64]. The most important property is that the minimum number of bits for a given symbol can be shown to be given by

$$I(i_n) = -\log_2(P(i_n)) \quad \text{bits} \tag{3.2.2}$$

$I(i_n)$ is known as the self information of $i_n$ and measures the amount of information inherent in the event $i_n$. The entropy $E$ of the data stream is the average information transmitted per symbol in the data stream and is given by

$$E = -\sum_n P(i_n)\log_2(P(i_n)) \quad \text{bits per event} \tag{3.2.3}$$

It is the average cost per event for the data stream with alphabet $\{i_n\}$. The aim of an entropy coder is to achieve this cost. The best known entropy coder is the Huffman coder [47], which is optimal for a fixed single symbol code. A table of codewords is devised from the probability distribution $\{P(i_n)\}$ with each symbol $i_n$ having a fixed integer number of bits $R_n$. The symbols are unique and invertible i.e. the symbols can be decoded from a concatenation of the individual codewords. Example 3.2.2 shows a Huffman code for an alphabet with four symbols.

**Example 3.2.2**

$N=4$

| Symbol | $P(i_n)$ | $-\log_2(P(i_n))$ | Simple Binary Code | Huffman Code | $R_n$ |
|---|---|---|---|---|---|
| $i_0$ | 0.8 | 0.32 | 00 | 0 | 1 |
| $i_1$ | 0.1 | 3.32 | 01 | 10 | 2 |
| $i_2$ | 0.05 | 4.32 | 10 | 111 | 3 |
| $i_3$ | 0.05 | 4.32 | 11 | 110 | 3 |

The simple binary code in example 3.2.2 has a fixed rate of 2 bits per symbol whereas the Huffman code has a rate given by

$$\sum_n P(i_n) R_n = 1.3 \quad \text{bits per event} \tag{3.2.4}$$

Note that this is a lossless compression of $\frac{2.0-1.3}{2.0} \times 100 = 35\%$ over the simple binary code. It is still not optimal however since the entropy for the set $\{i_n\}$ is given by

$$-\sum_n P(i_n) \log_2(P(i_n)) = 1.02 \quad \text{bits per event} \tag{3.2.5}$$

This non-optimality stems from the restriction of having to use an integer number of bits for each codeword whereas the optimal number of bits is generally a non-integer. This effect is particularly significant for binary data sources, where the alphabet has only two symbols and the Huffman coder is forced to assign a codeword of 1 bit to each symbol regardless of the probability distribution.

Instead of considering just single events, an arbitrarily large sequence of successive events $\{ i_{n1}, i_{n2}, \cdots, i_{nM} \}$ can be considered as a symbol in an extended alphabet of size $N^M$. The entropy per event $E_M$ of this $N^M$ alphabet is given by

$$E_M = -\frac{1}{M}\sum P(i_{n1} \cdots i_{nM}) \log_2 P(i_{n1} \cdots i_{nM}) \qquad (3.2.6)$$

where the sum is taken over all possible sequences of $M$ events, and

$$P(i_{n1} \cdots i_{nM}) = P(i_{n1})\, P(i_{n2} | i_{n1})\, P(i_{n3} | i_{n1} i_{n2}) \qquad (3.2.7)$$
$$\cdots P(i_{nM} | i_{n1} \cdots i_{nM-1})$$

If the events are independent equation (3.2.7) reduces to

$$P(i_{n1} \cdots i_{nM}) = P(i_{n1})\, P(i_{n2}) \cdots P(i_{nM}) \qquad (3.2.8)$$

If the M-sequence is the entire length of the data stream, then theoretically a Huffman code can be devised for the table of all such possible sequences. The entropy corresponding to the sequence of actual events is then a unique Huffman code representing it with arbitrarily close to the optimal number of bits

(there is still one integer approximation but as N increases this becomes insignificant). However, the standard Huffman coder algorithm is not computationally feasible for large sequences and in addition not all entries in the table are required, but only the one entry that actually occurs. Thus an algorithm which computes a unique codeword of length $-\log_2(P(x_1x_2\cdots x_M))$ given the sequence $\{x_1x_2\cdots x_M\}$ is required.

One efficient algorithm is known as arithmetic coding [87] and in recent developments [63] [73] [74] [75], it has been enhanced into a simple and efficient practical coder. The simplest model to describe arithmetic coding is the segmentation of the real number line between 0.0 and 1.0 into a set of non-overlapping segments whose union is the entire line. Each segment represents one of the symbols $i_n$ and the lengths of the segments are their probabilities. A symbol is coded by selecting the segment associated with it and recursively segmenting this segment into new segments, one for each possible symbol. This is illustrated in fig 3.2.1.

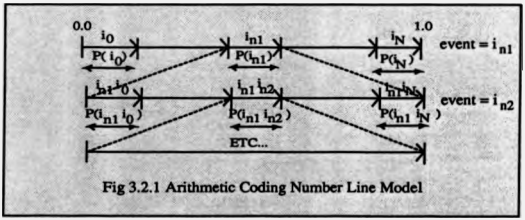


Fig 3.2.1 Arithmetic Coding Number Line Model

The expansion in scale at each level in the coding process in fig 3.2.1 is for convenience of viewing rather than any intrinsic part of the algorithm, although

this expansion does have a counterpart in any efficient implementation of the algorithm. All the addressing of the line is done using binary fractions (i.e. 0.75 decimal is 0.11 binary) in order to achieve a final binary codeword. A codeword is selected by choosing any point within the current segment. Note that if a segment has length $A$ then a unique codeword can be chosen within the segment with a binary precision of $\lceil -\log_2(A) \rceil$ bits. That the codeword for a sequence $(i_{n1} \cdots i_{nM})$ is invertible can be seen from the fact that the segments are non-overlapping. Note also that at each stage the size of the interval is given by

$$A_M = \prod_{1 \le m \le M} P(i_{nm}) \quad , \quad M > 0 \tag{3.2.9}$$

and the starting point $C_M$ of the interval by

$$C_M = \sum_{1 \le m \le M} \left[ A_{m-1} \sum_{p < nm} P(i_p) \right] \tag{3.2.10}$$

From (3.2.9), it can be seen that at each stage a unique codeword $C$ can be chosen within the current interval to represent the sequence $(i_{n1} .. i_{nM})$ with a precision given by

$$P_M = [-\log_2(A_M)] = [-\log_2(P(i_{n1})P(i_{n2}) \cdots P(i_{nM})] \tag{3.2.11}$$

assuming the events are independent. Thus $P_M$ is equal to the entropy for the sequence i.e. it is optimal. This final codeword $C$ is unique to the particular sequence of events, and, given that the probabilities $\{P(i_{n1}), \cdots, P(i_{nM})\}$ are

known at the decoder, the events $\{i_{n1}, \cdots, i_{nM}\}$ can be determined by an identical segmentation of the number line as shown in fig 3.2.2.

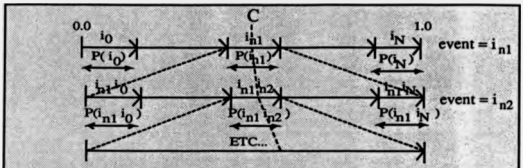



Fig 3.2.2 Arithmetic Decoding Number Line Model

A simplification of the arithmetic coding process is achieved by restricting the alphabet to two symbols. This binary arithmetic coder leads to a simple algorithm and is still generally applicable, since a unique sequence of binary events can be derived for any multiple symbol alphabet by simply using a binary tree as shown in fig 3.2.3, with a probability $Q_i$ for each binary event.


M = 6
Multiple Symbol Alphabet
{a,b,c,d,e,f}
Binary codes
{00,01,100,1010,1011,11}
Binary Event Probabilities
$Q_n$ , 1<=n<=R
Note: R = M-1

Fig 3.2.3 - Multiple Symbol to Binary Symbol Tree

If the binary events are labeled MPS for the more probable symbol, with a

probability of $Q_i$, and LPS for the less probable symbol, with a probability of $(1-Q_i)$, then fig 3.2.4 illustrates the segmentation at any stage.

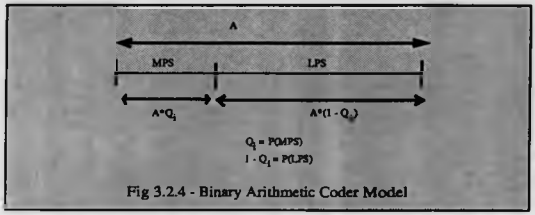



Fig 3.2.4 - Binary Arithmetic Coder Model

This leads to a simple algorithm shown in fig 3.2.5

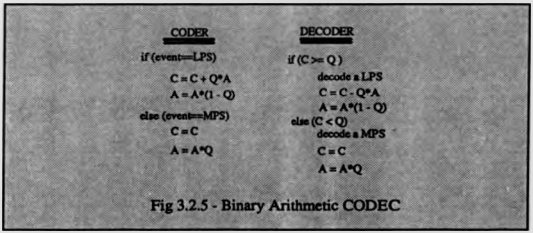



Fig 3.2.5 - Binary Arithmetic CODEC

Note that $Q_i$ can be altered at each stage of the algorithm (i.e. after each event), provided it is altered at the same stage on both the coder and decoder. There are two obvious problems in implementing this algorithm:

1) the fact that it contains a multiplication of real numbers and at each such multiplication the precision necessary to hold the result doubles.

2) The codeword variable C requires increasing precision, and the interval size variable A becomes increasingly small, as the algorithm proceeds. Thus an unlimited amount of buffering is apparently required.

However, both these problems are superficial and are solved with simple techniques which lead to a negligible loss of optimality [62] [75], and allow the decoder to decode the events from the codeword whilst it is still being built by the coder. Obviously there must be a slight time delay between the coder and decoder.

### 3.2.2. Adaptive Estimation of the Probability Distribution

In order for the algorithm to be optimal, the probabilities $Q_i$ must be known. This implies that they must either be preset, or estimated from the data. Although it obviously depends on the compression technique used, the probabilities $Q_i$ for the data produced by image coding algorithms generally vary from image to image, and indeed often vary within a given image. These probabilities can be estimated from the data and passed as side information to the decoder. However the decoder requires these probabilities in order to decode the events, and thus there is an intrinsic time delay if the estimation is performed non-causally. However if it is performed causally, ie. the probability estimate for a given event depends only on previous events, then the probabilities need not be sent as side information, and no time delay is involved.

The estimation of such non-stationary properties of a signal is limited by some form of uncertainty, in that the signal sample over which the parameter is estimated is assumed to be stationary, and the larger this signal sample is, the more reliable the signal parameter estimate is, but the less valid is the assumption of stationarity (eg [113]). In terms of estimating the probability distribution of a signal, the smaller the section (or window) of data over which the probability distribution is estimated the more likely the probability distribution is to be stable, but the less data are available to estimate the distribution accurately. This trade off between spatial locality and accuracy of the estimate is a wide field of research in its own right, but greater knowledge of the source model allows for a more accurate estimation of the parameters. The Q-coder [74] provides adaptive causal estimation techniques which dynamically 'track' the probabilities $\hat{Q}_i$ based on the relative frequency of LPS and MPS events occurring in the recent past.

For the basic RBN algorithm the probability distribution of $e^n_{xy}(n)$ is assumed to be different for each $n$ and also to be varying for $x,y$. Noting from fig 3.2.3 that $R = M - 1$ binary contexts (probabilities) are required to represent the multisymbol alphabet of $M$ symbols used to represent the quantised prediction errors. Thus the coder requires $N$ levels of the $R$ binary probability estimates $\hat{Q}^r_{nr}$.

$$0.0 < \hat{Q}^r_{nr} < 1.0 \qquad 1 \le n \le N \qquad 1 \le r \le R \qquad (3.2.12)$$

Since the number of quantisation symbols $M = 25$, the codec requires 24 contexts per level. This may be a large number of contexts, requiring a large amount of memory or complexity, but in practice many of the contexts handle

an insignificant fraction of the total coded data, and these contexts could be merged with little loss of efficiency. Note that $\tilde{Q}_t$ is not allowed to be either 0.0 or 1.0 since this implies that an event can either never occur or must always occur, and an actual occurrence (or non occurrence) of such an event will carry with it an undefined amount of information.

The addressing information $d_{ij}(n)$ is already a binary event but its probability distribution too can be expected to differ across levels of scale $n$ and to vary spatially with $x,y$, so that it requires N probability contexts as given in equation (3.2.13).

$$0.0 < \hat{Q}_n^* < 1.0 \qquad 1 \le n \le N \qquad (3.2.13)$$

### 3.2.3. Arithmetic Coding Results

Tables 3.2.1 and 3.2.2 give the zeroth order entropies $E_n$ for the prediction errors at each level $n$ for the coding results shown in photo 3.1 and photo 3.3, together with the number of bits $R_n$ produced by the Q-coder at each level $n$.

Note that the number of bits produced by the Q-CODER is less than the zeroth order entropy for the quantised prediction errors. This is due to the adaptation of the probability estimation within the Q-CODER. It is more noticeable at the lower levels of resolution for two main reasons. First, the predictions are based on a more local area and therefore it is more likely to contain locally consistent but globally varying distributions, and secondly there is sufficient data for the dynamic estimation to reach an initial probability estimate (as is not the case for the first few levels).

| level $n$ | $E_n$ | $R_n$ | $\frac{R_n}{E_n} \times 100\%$ |
|---|---|---|---|
| 0 | - | - | - |
| 1 | 11.6 | 25 | 215.5 |
| 2 | 48.5 | 78 | 160.8 |
| 3 | 227.7 | 265 | 116.4 |
| 4 | 791.8 | 828 | 104.6 |
| 5 | 2614.4 | 2601 | 99.5 |
| 6 | 7677.3 | 7316 | 95.3 |
| 7 | 18878.1 | 16775 | 88.9 |
| 8 | 36349.7 | 29173 | 80.2 |
| 9 | 13861.4 | 11354 | 81.3 |
| sum | 80460.6 | 68815 | 85.0 |

Table 3.2.1

| level $n$ | Prediction Errors | | | Addressing | | |
|---|---|---|---|---|---|---|
| | $E_n$ | $R_n$ | $\frac{R_n}{E_n} \times 100\%$ | $E_n$ | $R_n$ | $\frac{R_n}{E_n} \times 100\%$ |
| 0 | - | - | - | - | - | - |
| 1 | 11.6 | 25 | 215.5 | 0.0 | 1 | - |
| 2 | 48.5 | 79 | 162.9 | 0.0 | 4 | - |
| 3 | 227.7 | 264 | 115.9 | 0.0 | 8 | - |
| 4 | 782.7 | 815 | 104.1 | 7.4 | 23 | 310.8 |
| 5 | 2309.2 | 2345 | 101.6 | 146.5 | 149 | 101.8 |
| 6 | 5474.6 | 5569 | 101.7 | 757.0 | 700 | 92.5 |
| 7 | 9301.9 | 9424 | 101.3 | 2455.4 | 2020 | 82.3 |
| 8 | 10093.6 | 10353 | 102.6 | 4635.1 | 4244 | 91.6 |
| 9 | 1890.5 | 1987 | 105.1 | 2229.8 | 2223 | 99.7 |
| sum | 30140.3 | 30861 | 102.4 | 10231.1 | 9372 | 91.6 |

Table 3.2.2

Note that arithmetic coding is inherently unstable in channel noise, since once a single bit is corrupted, the entire sequence of following symbols may be (and usually will be) apparently randomly distorted. Because of this problem in

practice a less optimal but more robust entropy coder, such as a Huffman coder may be preferable, or the arithmetic coder must be structured (e.g. by blocking it) so as to reduce the effects of errors.

## 3.3. Vector Quantisation

### 3.3.1. Introduction

Equation 2.2.2 assumes that the prediction errors $\{e_{x,y}^{n}(n)\ ,\ (x,y)\in\Lambda_n\}$ at level $n$ are independent of each other, but there is generally some local correlation between these values. The vector quantiser attempts to reduce this redundancy by quantising groups of prediction errors as a single vector rather than separately. The vector quantiser can be shown to perform at least as well as a scalar quantiser [35] and may perform better where there is correlation between the elements of the vector.

Each vector is composed of a set of $n$ data points (prediction errors) in a specific order and it is quantised by having a limited number $N$ of possible codeword vectors. Each data vector is assigned the codeword whose vector is closest (in a mean squared error sense) to the data vector, and the data vector is then represented by the codeword vector, with the difference between the two being quantisation noise. The set of $N$ codewords from which the quantiser can choose is determined by training the quantiser using samples from the data, the codewords representing the centroids of an $N$-region partition of the entire vector data set. The partition is chosen such that the mean squared error from all data vectors to the corresponding centroid is a minimum. It is common to introduce symmetries into the vector codewords by splitting the magnitude from the vector via some normalisation process or by a rotation or reflection symmetry.

One advantage of this is that it reduces the bias introduced by the particular training set having a bias towards one or other of these symmetries.

For the RBN algorithm, a natural choice of a 5-point vector is suggested by the new pixels required at each split decision as shown in fig 3.3.1.

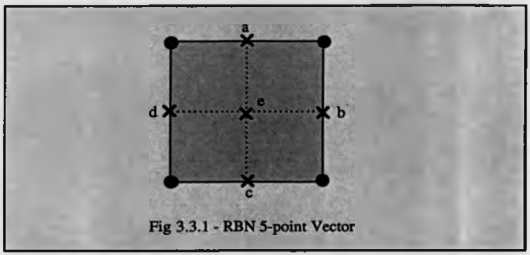



Fig 3.3.1 - RBN 5-point Vector

The 5-point vector of prediction errors is given by equation 3.3.1.

$$v_{ij}(n) = \begin{bmatrix} e_a(n) \\ e_b(n) \\ e_c(n) \\ e_d(n) \\ e_e(n) \end{bmatrix} \qquad (3.3.1)$$

This vector is normalised to give

$$v_{ij}(n) = |v_{ij}(n)| = \left[ e_a^2(n) + e_b^2(n) + e_c^2(n) + e_d^2(n) + e_e^2(n) \right]^{\frac{1}{2}} \qquad (3.3.2)$$

and

$$v_{ij}^{N}(n) = v_{ij}(n) \times \frac{1}{v_{ij}^{m}(n)} \tag{3.3.3}$$

The magnitude $v_{ij}^{m}(n)$ is quantised separately using a scalar quantiser and in the case where it is quantised to zero no information about the vector is coded. When the quantised magnitude is not zero, the vector $v_{ij}^{N}(n)$ is coded by matching it to the nearest vector codeword and transmitting the index of that codeword.

The $N$ codewords are determined from the training vector sequence such that they minimise the mean of the squared distance from each training vector to its nearest codeword. The process of finding a minimum from the set of training data is an iterative process based on the K-means clustering algorithm [35]. The process involves forming a N-partition of the training data into a number of sets, and calculating the centroid for each set. The training data is then repartitioned into the sets based on these centroids, such that the distance between each training vector and the centroid of the set to which it is assigned is a minimum. This new partition gives a new set of centroids, and the process is iterated until a stable partition is achieved. The centroids of this stable partition represent a minimum solution for the codeword vectors.

A good starting partition can be estimated by forming a 1-partition of the training data and iterating the partition and centroid calculation until it stabilises (naturally, this case requires only one iteration to give a stable partition). The centroid is then split into two vectors (slightly offset from each other), which are used as an initial guess at the centroids of a 2-partition of the training data,

and the partition and centroid calculating process is then iterated until it gives a stable 2-partition. The N-partition is achieved by repeating the splitting process until the required number of centroids is achieved.

The process will iterate to a local minimum, but this is not necessarily the global minimum, and only an exhaustive search of all local minima will achieve the global minimum. This set of code vectors must then be made available to both the coding and decoding processes. They may be sent as initialisation information at the start of a coding, or they may be fixed at both the coder and decoder by training the vectors from a set of typical images.

Once the set of code vectors has been fixed, the coding process consists of comparing the normalised data vectors formed from each set of prediction errors with all the code vectors and selecting the code vector which has the minimum distance from the data vector. The magnitude of the data vector is quantised and coded separately from the normalised data vector, and if it is zero there is no need to code the vector information. The code vector is coded by its index and the decoder selects the vector from the set of code vectors by this index, rescales it using the quantised magnitude and uses the components of the code vector as the prediction errors in reconstructing the image.

### 3.3.2. Missing pixels

There is a problem in the RBN coder in that since the blocks are edge-sharing, the vectors may consist of less than 5 components, i.e. one or more components may already have been coded as part of a neighbouring block (see fig 3.1.4). If the vector quantiser takes no account of this, then each pixel on an edge of a block will be coded twice leading to two problems:

1) When decoded each such pixel will have two possible values to select from. One strategy would be to take the average of the two, but then some of the consistency in intensities at block edges would be lost. A second strategy would be to ignore the second coded value of the pixel and so keep the edge continuity.

2) Effectively each such pixel is being coded twice, and therefore costs twice as much to code. Since this overlapping occurs at all 4 edge pixels within each vector, a considerable proportion of the pixels are coded twice, adding a excessive amount to the bit rate of the coder.

The problem is overcome by labelling all pixels as they are coded the first time and assigning a flexible value to all pixels which have already been labelled. This flexible value is defined to have a distance of zero from any code vector component, so that it adds zero to the overall error between the data vector and any codeword vector. Thus the codeword vector is selected on the basis of the other components only. This flexible value is also used in the training of the codewords so that it can eliminate any structure introduce by the particular ordering of the RBN scan i.e. for a fixed RBN scan there will be a bias in which pixels in the vector will be assigned the flexible value.

An alternative to the above approach would be to have a set of codewords for all possible combinations of 1-d, 2-d, 3-d, 4-d and 5-d vectors, but this would be a considerably more complex solution to the problem.

### 3.3.3. Rotation and Reflection Symmetries

Noting that the vector has 8 rotational and reflectional symmetries as shown in fig 3.3.2, it is natural to remove these symmetries from the vector in order to prevent any one of them from predominating in the training of the vectors. In other words, the feature it represents is expected to be equally likely to occur in all 8 symmetries.

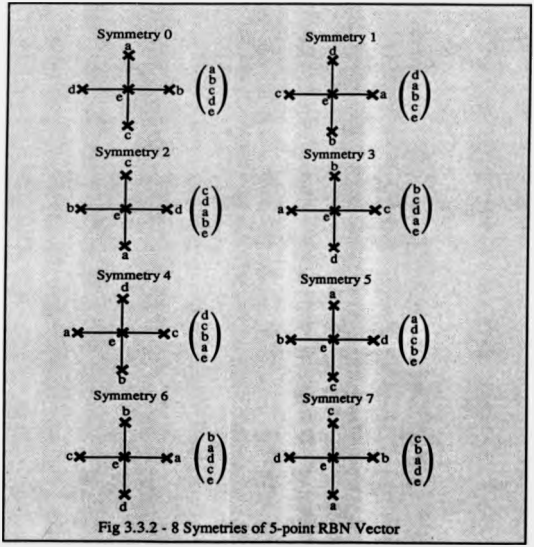



Fig 3.3.2 - 8 Symetries of 5-point RBN Vector

This is done by comparing each data vector with all possible symmetries of all the codeword vectors and chosing the combination of codeword and symmetry which has the least distance from the data vector. The index of the symmetry is then transmitted along with the index of the codeword. The training of the codewords proceeds in a similar manner, with each training vector being assigned to the codeword and symmetry which is closest to it and the centroid of each partition calculated from the training vectors in that partition after they have been adjusted to the appropriate symmetry. This process effectively multiplies the number of codewords by a factor of eight, but provides a less biased set than simply having eight times as many simple codewords. A similar process is used to remove the sign of each vector, and transmit that separately.

Each component is entropy coded using the arithmetic coder described in section 3.2. The magnitude of the vectors is quantised to 25 levels and therefore it requires 24 probability contexts for each level, i.e.

$$0.0 \le \hat{Q}_{nr}^{m} \le 1.0 \qquad 1 \le n \le N \qquad 1 \le r \le 24$$

The 32 codewords require 31 contexts at each level, i.e.

$$0.0 \le \hat{Q}_{nr}^{c} \le 1.0 \qquad 1 \le n \le N \qquad 1 \le r \le 31$$

The 8 symmetries require 7 contexts at each level, i.e.

$$0.0 \le \hat{Q}_{nr}^{s} \le 1.0 \qquad 1 \le n \le N \qquad 1 \le r \le 7$$

The sign requires 1 context at each level, i.e.

$$0.0 \le \hat{Q}_n^i \le 1.0 \qquad 1 \le n \le N$$

### 3.3.4. Vector Quantisation Results

The RBN vector quantised coder (RBN VQ) performed as well as the scalar quantised RBN scheme, but it was not noticeably better, and considering its additional complexity, it is not used in any of the following codec results.

## 3.4. Isotropic Coder Results

Table 3.4.1 gives some coding results for a full (no termination) RBN codec. Photos 3.1 and 3.2 show the reconstructed images for the Peano-RBN Scan coding of the GIRL and BOATS images given in table 3.4.1. The artefacts introduced by the coding are most clearly seen in the BOATS image, where the ropes are considerably distorted.

Photos 3.3 and 3.5 show the reconstructed images for the GIRL and LAKE images, and photos 3.4 and 3.6 show the scale selection for these results by displaying those pixels at which an innovation has been made (i.e. the set $\bigcup_n \Delta_n^i$ as described in section 4.2 on the inhomogeneous model). Note the disturbing artefacts around the edges on the reconstructed GIRL image. Note also from the sample pixel image that the pixels are concentrated around the edges. Table 3.4.2 gives a set of coding results for the terminated RBN codec, together with figures quoted in other published work for comparable (i.e. 512 by 512 by 8 bit) images. Note that the PSNR results are comparable to other published

work. The PSNR's are greater than 30 dB at rates above 0.15 bpp, giving it comparable performance to other techniques in terms of mean squared error. However, it does suffer from blocking artefacts around edges at lower rates, something which is not present in [112]. These blocking artefacts mean that the visual quality is not acceptable. This defect is corrected in the anisotropic coders of chapter 5.

| Image | Fixed RBN | | | Peano RBN | | |
|---|---|---|---|---|---|---|
| | mse | PSNR | bpp | mse | PSNR | bpp |
| GIRL | 39.26 | 32.19 | 0.27 | 39.26 | 32.19 | 0.26 |
| BOATS | 53.84 | 30.82 | 0.42 | 53.84 | 30.82 | 0.41 |
| LAKE | 106.59 | 27.85 | 0.74 | 106.59 | 27.85 | 0.73 |

Table 3.4.1

Photo 3.1 - Homogeneous Isotropic RBN Result (GIRL)

Photo 3.2 - Homogeneous Isotropic RBN Result (BOATS)

| Image | Isotropic Coder Results | | Published Results | | |
|---|---|---|---|---|---|
| | Rate | PSNR | Rate | PSNR | Authors |
| GIRL | 4.27 | noiseless | 5.07 | noiseless | Ho & Gersho * [43] |
| | 1.70 | 41.39 | " | " | " |
| | 1.25 | 40.18 | " | " | " |
| | 1.08 | 38.12 | 1.06 | 37.0 | Wilson [111] |
| | | | 1.0 | 31.55 | Ramamurthi & Gersho [86] |
| | | | 1.0 | 34.8 | Cohen and Woods [23] |
| | 0.71 | 35.84 | " | " | " |
| | 0.66 | 35.79 | " | " | " |
| | 0.37 | 32.70 | 0.50 | 28.55 | Ramamurthi & Gersho [86] |
| | | | 0.50 | 30 | Wilson et al [112] |
| | | | 0.45 | 32.5 | Cohen and Woods [23] |
| | 0.27 | 31.70 | 0.28 | 30.61 | Ho & Gersho [43] |
| | | | 0.27 | 27 | Wilson et al [112] |
| | | | 0.25 | 32.8 | Wilson [110] |
| | | | 0.25 | 30 | Kim et al [53] |
| | 0.19 | 30.75 | 0.19 | 29.58 | Feng and Nasrabadi † [31] |
| | 0.15 | 30.35 | " | " | " |
| | 0.12 | 29.30 | | | |
| | 0.07 | 27.06 | 0.06 | 24.70 | Ho & Gersho [43] |
| LAKE | 0.55 | 27.68 | 0.37 | 25.8 | Wilson [110] |
| BOATS | 4.59 | noiseless | 5.26 | noiseless | Ho and Gersho * [43] |
| | 0.28 | 29.73 | 0.38 | 30.40 | Ho and Gersho [43] |

Table 3.4.2

† Feng and Nasrabadi quote figures for the green plane of the GIRL image (whereas the image used in this work is believed to be the Y plane).

* Ho and Gersho use a progressive transmission technique and therefore the noiseless figures are not a very good comparison.

Photo 3.3 - Inhomogeneous Isotropic RBN Result (GIRL)

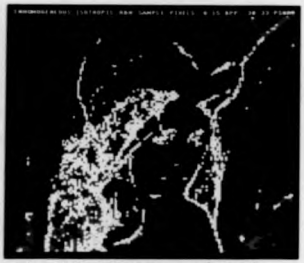


Photo 3.4 - Inhomogeneous Isotropic RBN Innovations Sets (GIRL)

Photo 3.5 - Inhomogeneous Isotropic RBN Result (LAKE)

Photo 3.6 - Inhomogeneous Isotropic RBN Innovations Sets (LAKE)

# 4. Adaptive LIM RBN Models

## 4.1. Introduction

In the final section of chapter 2, the properties of a homogeneous isotropic LIM RBN model were considered. In this chapter, the LIM RBN model is extended to various adaptive forms, where the linear operators $A(n)$ and $B(n)$ are dependent on local properties of a particular image. One of these corresponds to the adaptive codecs of the previous chapter. These adaptive models are considerably better models of the source images, and thus the codecs derived from them are more efficient, as demonstrated in chapter 5.

Two principal sources of adaptation are considered,

1) Scale Selection, where the termination of the splitting process occurs at varying levels of resolution across the image.

2) Local Anisotropy, where the local orientation of image features is incorporated into the interpolation operator $A(n)$.

In a strict sense, both these sources produce inhomogeneity in the spatial and scale dimensions. However, those models which use local anisotropy, but which do not have any scale selection, (i.e. the RBN splitting process is always carried out to the lowest level) are referred to in this chapter as homogeneous models.

The inclusion of these two properties leads to a considerable improvement in the LIM RBN models, since natural images are clearly inhomogeneous in the spatial dimensions and perhaps less clearly inhomogeneous in the scale

dimension. In this chapter the properties of these adaptive multiresolution models are considered in more detail, and in particular the importance of local anisotropy is considered. Related inhomogeneous multiresolution rate distortion functions are derived, and these functions are presented for parameters estimated from the GIRL image.

## 4.2. Inhomogeneous Isotropic LIM RBN Model

### 4.2.1. Scale Selection

The scale selection or termination process introduces inhomogeneity into the LIM RBN model by allowing the structure of $A(n)$ and $B(n)$ to vary with spatial position. If a block is terminated at level $n$, then no further innovations are made to any pixels within that block at levels $m > n$. If $\Omega_n$ is defined as the set of all pixels within the blocks which are terminated at level $n$, excluding those pixels which are members of an innovation set at a previous level, then the operators $A(n)$ and $B(n)$ are additionally conditioned by

$$A_{xypq}(n) = \delta_{xp}\,\delta_{yq} \quad \text{if} \quad (x,y) \in \bigcup_{m<n} \Omega_m \tag{4.2.1}$$

and

$$B_{xyrs}(n) = 0 \quad \text{if} \quad (x,y) \in \bigcup_{m \le n} \Omega_m \tag{4.2.2}$$

The termination process produces a new innovations set $\Lambda_n^T$ by removing from the set $\Lambda_n$, defined by equation (2.5.1), all pixels which are in the set $\bigcup_{m \le n} \Omega_m$.

i.e. all pixels which do not require an innovation

$$\Lambda_n^I = \Lambda_n - \bigcup_{m \le n} \Omega_m \qquad 0 \le n \le N \tag{4.2.3}$$

This is shown in fig 4.2.1.

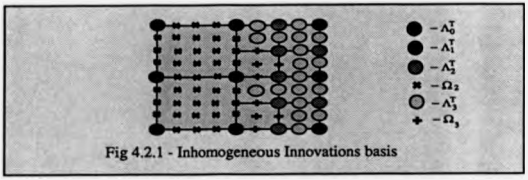



Fig 4.2.1 - Inhomogeneous Innovations basis

The inhomogeneity of this LIM RBN model alters some of the properties defined in the summary of the homogeneous isotropic LIM RBN models, in particular the periodicity property no longer applies. A summary of the corrected inhomogeneous properties is given below.

### 4.2.2. Summary of Inhomogeneous LIM RBN Model Properties

(1) Exclusivity

$$\Lambda_n^I \cap \Lambda_m^I = \varnothing \quad \text{iff} \quad n \neq m \tag{4.2.4}$$

$$\Omega_n \cap \Omega_m = \varnothing \quad \text{iff} \quad n \neq m$$

$$\Lambda_n^I \cap \Omega_m = \varnothing$$

(2) Completeness

Certain pixels in the image are now represented by an interpolation only and have no innovation to the interpolation and hence do not appear in the innovation set. Completeness is redefined by

$$\Lambda = \bigcup_{0 \le n \le N} (\Omega_n \cup \Lambda_n^T) \tag{4.2.5}$$

(3) Single innovation

$$B_{xyrs}(n) = 0 \quad \text{if} \quad \left[(x, y) \notin \Lambda_n^T\right] \vee \left[(r, s) \notin \Lambda_n^T\right] \tag{4.2.6}$$

(4) No change

$$A_{xypq}(n) = \delta_{xr}\delta_{ys} \quad \text{if} \quad (x, y) \notin \bigcup_{m<n} (\Lambda_m^T \cup \Omega_m) \tag{4.2.7}$$

(5) Innovations basis

Pixels in the set $\Omega_n$ may be used at levels $m > n$ as the basis for the linear estimator $A(m)$ as shown in fig 4.2.1, since they are now fixed and will not be altered at any later stage. This means that the specification of the innovations basis is now

$$A_{xypq}(n) = 0 \quad \text{if } (p,q) \notin \bigcup_{m<n} (\Lambda_m^T \cup \Omega_m) \tag{4.2.8}$$

(6) Periodicity

The structure of $A(n)$ and $B(n)$ can no longer be defined on a periodic

basis.

Photo 4.1 shows $s(N)$ for this inhomogeneous isotropic LIM RBN model, with the particular inhomogeneity taken from the GIRL image, i.e. the scale selection in photo 4.1 reflects that of the GIRL image. The variances for the various levels of the zero mean normal white noise source $w(n)$ were estimated from the GIRL image using only those pixels in the innovations sets $A_n^T$, and are given in table 4.2.1. The variances in the table generally decrease with level, like the homogeneous model parameters given in table 2.5.1. However, from level 4 onwards, the variances of the inhomogeneous model are larger than those of the homogeneous model and at the lowest levels they are considerably larger. This is because for the inhomogeneous model, the innovation values of the set of pixels which do not receive an innovation, $\bigcup_n \Omega_n$, are excluded from the estimation of the variances. Since these values are relatively small (by the very nature of the splitting decision), their exclusion increases the estimate of the innovation variances.

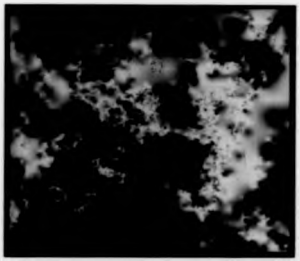


Photo 4.1 - Inhomogeneous Isotropic LIM RBN Model

| level $n$ | variance | level $n$ | variance |
|---|---|---|---|
| 0 | - | 5 | 1266 |
| 1 | 1764 | 6 | 883 |
| 2 | 1670 | 7 | 767 |
| 3 | 2764 | 8 | 652 |
| 4 | 1726 | 9 | 665 |

Table 4.2.1

### 4.3. Multiresolution Isotropic Gauss-Markov Models

The Gauss-Markov source model is derived as an alternative to the scale selection LIM RBN model. Although it is not identical (in particular the blocks are not edge-sharing), it is similar and allows for the derivation of a simple rate distortion function. The reason for developing this model is the difficulty of representing the multiresolution edge-sharing of the LIM RBN model in a rate distortion function.

The basic Gauss-Markov model represents the image as a mosaic of scale selected blocks with a known probability distribution of the blocks over scale as shown in fig 4.3.1. The probability distribution is expressed in terms of the probability that a block at scale $n$ is split into 4 smaller blocks at scale $n+1$. Only intermediate levels of scale are allowed, i.e. no very large blocks or very small blocks are allowed. In this case, only levels $n = 4$ (64 by 64) to $n = 8$ (4 by 4) are allowed.

The pixel intensities within each block in the mosaic are modeled by a stationary first order Gauss-Markov process and the estimated parameters of this process are used to derive a rate distortion function for the source model.

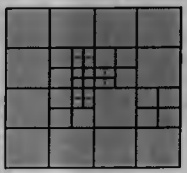


Fig 4.3.1 - Gauss Markov Model Structure

The rate distortion function was originally suggested by Shannon [97], described in detail in the book by Berger [8], and has been investigated by a number of other researchers [3] [7] [92] [93] [10]. It is often expressed parametrically by two coupled functions, $R(\bullet)$ and $D(\bullet)$. $R(\bullet)$ represents the bit rate or entropy of the source, and $D(\bullet)$ the distortion of the source. These variables give the optimal bit rate for a given distortion or alternatively an optimal distortion for a given rate. Ideally a function of the form $R(D)$ or $D(R)$ should be formulated, but it is often difficult to express the function in this form, and it is usual to express it in the parametric form, $R(d)$ and $D(d)$ so that as $d$ varies the variable $R(d)$ gives the rate for the distortion $D(d)$.

The rate function $R(d)$ is expressed in units of bits for this work and $D(d)$ is expressed as a mean squared error or, on graph axes, as a percentage normalised mean squared error with respect to the variance $\sigma^2$ of the image, i.e.

$$D(d) = 100 \times \frac{E[(s(N) - \hat{s}(N))^2]}{E[(s(N) - \mu)^2]} \% \qquad (4.3.1)$$

Where $\tilde{x}(N)$ is the distorted image at a rate of $R(d)$ and $\mu$ is the mean value of the original. The rate distortion function for a block in the Gauss-Markov model is defined from its correlation matrix [8, section 4.5]. For a one dimensional sequence of $2^{N-n}$ values, the correlation matrix for a stationary source is given by equation (4.3.2)

$$\Phi_n = \begin{bmatrix} \phi_0 & \phi_1 & \phi_2 & \cdots & \phi_{2^{N-n}-1} \\ \phi_1 & \phi_0 & \phi_1 & \cdots & \phi_{2^{N-n}-2} \\ \phi_2 & \phi_1 & \phi_0 & \cdots & \phi_{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_{2^{N-n}-1} & \phi_{2^{N-n}-2} & \phi_{2^{N-n}-3} & \cdots & \phi_0 \end{bmatrix} \qquad (4.3.2)$$

where $\phi_k$ is the correlation coefficient between the intensities of pixels which are a distance $k$ apart. It is assumed to be a first order Markov source, which is characterised by the equation

$$\phi_k = \sigma^2 \rho^k \qquad 0 \le \rho \le 1 \qquad 0 \le k < 2^{N-n} \qquad (4.3.3)$$

In other words the correlation decays with distance by a constant factor of $\rho$.

This gives for $\Phi_n$

$$\Phi_n = \phi_0 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{2^{N-n}-1} \\ \rho & 1 & \rho & \cdots & \rho^{2^{N-n}-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho^{2^{N-n}-1} & \rho^{2^{N-n}-2} & \rho^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \qquad 0 \le \rho \le 1 \qquad (4.3.4)$$

For a two dimensional block at scale $n$ of size $2^{N-n}$ by $2^{N-n}$, the correlation function is considered to be separable into two identical functions in orthogonal directions, with the correlation matrix for each function given by equation (4.3.4). This leads to a correlation matrix for the block which is $\phi_0$ times the Kronecker product of $\frac{1}{\phi_0}\Phi_n$ with itself. Using the fact that the eigenvalues of a Kronecker product of two matrices are the product of their eigenvalues [83], and the rate distortion function defined for a Gaussian source given in [8], it is possible to show that for such a model the rate $R_n^I$ and distortion $D_n^I$ functions for the block are

$$R_n^I(d) = \sum_{i=1}^{2^{N-n}} \sum_{j=1}^{2^{N-n}} \left[ \max\left[ 0, \frac{1}{2}\log_2\left[ \phi_0 \frac{\lambda_i \lambda_j}{d} \right] \right] \right] \quad \text{(bits per block)} \quad (4.3.5)$$

$$D_n^I(d) = \sum_{i=1}^{2^{N-n}} \sum_{j=1}^{2^{N-n}} \min(d, \phi_0 \lambda_i \lambda_j) \quad (4.3.6)$$

where $\{\lambda_i \; ; 1 \le i \le 2^{N-n}\}$ are the eigenvalues of the matrix $\frac{1}{\phi_0}\Phi_n$ given in equation (4.3.4).

Equations (4.3.5) and (4.3.6) give the rate distortion function for a fixed size block. For a multiresolution rate distortion function, some means of expressing the scale selection must be found. Considering it within a similar multiresolution structure to the LIM RBN model, where a block is either accepted at level $n$, or split into four new equally sized blocks at level $n+1$, then if the probability of splitting a block at level $n$ is $P(n)$, the multiresolution rate function $R_n^I(d)$ can be defined by the recursive equation (4.3.7).

$$R_n^1(d) = 4P(n)R_{n+1}^1(d) + (1-P(n))R_n^I(d) - P(n)\log_2 P(n) - (1-P(n))\log_2(1-P(n)) \qquad 4 \le n < 8 \quad (4.3.7)$$

where the first term represents the rate of the four blocks at level $n+1$, the second term the rate of one block at level $n$ and the last two terms are the book-keeping information needed to address the splitting decision. The equation has the initial conditions

$$R_8^1(d) = R_8^I(d) \qquad (4.3.8)$$

The final rate per pixel is given by

$$R^1(d) = \frac{R_4^1(d)}{2^{2N-8}} \qquad (4.3.9)$$

Similarly the distortion function can be defined by

$$D_n^1(d) = 4P(n)D_{n+1}^1(d) + (1-P(n))D_n^I(d) \qquad 4 \le n < 8 \quad (4.3.10)$$

$$D_8^1(d) = D_8^I(d) \qquad (4.3.11)$$

and the final per pixel distortion function as

$$D^1(d) = \frac{D_4^1(d)}{2^{2N-8}} \qquad (4.3.12)$$

The correlation coefficient $\rho$ in equation (4.3.4) can be estimated from the image $s(N)$. In order to give comparability with the anisotropic results (section 4.8), the correlation coefficients $\hat{\phi}_k$ were calculated by averaging in directions parallel to and perpendicular to the local feature orientation $\theta_{xy}$ at each point. The estimation of $\theta_{xy}$ is described in chapter 5.

$$\hat{\phi}_k = \frac{1}{2^{2N}} \sum_{x=0}^{2^N-1} \sum_{y=0}^{2^N-1} \frac{1}{4} \Big\{ s_{xy}(N) s_{(x + k\cos(\theta^{\alpha}_{xy}))\ (y + k\sin(\theta^{\alpha}_{xy}))}(N) + \tag{4.3.13}$$

$$s_{xy}(N) s_{(x - k\cos(\theta^{\alpha}_{xy}))\ (y - k\sin(\theta^{\alpha}_{xy}))}(N) +$$

$$s_{xy}(N) s_{(x + k\cos(\theta^{\beta}_{xy}))\ (y + k\sin(\theta^{\beta}_{xy}))}(N) +$$

$$s_{xy}(N) s_{(x - k\cos(\theta^{\beta}_{xy}))\ (y - k\sin(\theta^{\beta}_{xy}))}(N) \Big\} \qquad 0 \le k < K$$

where $\theta^{\alpha}_{xy} = \theta_{xy}$, and $\theta^{\beta}_{xy} = \theta_{xy} + \frac{\pi}{2}$, and $K$ was chosen to be 4.

The value of $\rho$ is determined from $\hat{\phi}_k$ by a least squares fit.

The block splitting probabilities $P(n)$ were estimated from a low rate coding segmentation, since there seems to be no coder-independent way of estimating them. This implies that the probabilities, which are model parameters, are dependent on the distortion. While it may seem paradoxical that a source property is dependent on the distortion criterion, on reflection it is clear that a given model can only fit the image data up to some level of distortion. Hence it is not surprising that the probabilities $P(n)$ vary with the level of distortion. Graph 4.3.1 is a plot of $R(D)$ for the GIRL image from a coding at 0.10 bpp, with the probability distribution given in table 4.3.1.

| level $n$ | $P(n)$ | level $n$ | $P(n)$ |
|---|---|---|---|
| 0 | - | 5 | 0.51 |
| 1 | - | 6 | 0.20 |
| 2 | - | 7 | 0.04 |
| 3 | - | 8 | - |
| 4 | 1.00 | 9 | - |

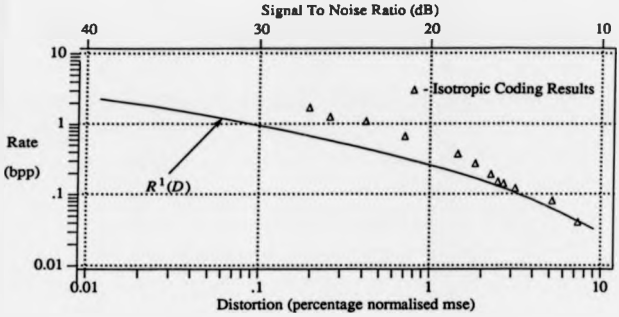
Table 4.3.1

The triangles added into the graph are plots of actual coding results as a comparison to the $R(D)$ curves. The coding results are for the isotropic inhomogeneous RBN coder with arithmetic entropy coding. The difference between the coding results and the $R(D)$ function is caused by keeping the probability parameters of the function fixed, whereas for the coding results these parameters vary. At low rates, where the parameters are the same, the curve fits the results reasonably well. Note that the relationship between the mean squared error for the coding result and the percentage normalised mean squared error given for the graph is determined from equation (4.3.1) with the variance of the GIRL image being $\sigma^2 = 2405.78$, i.e.

$$D = 100 \times \frac{\text{mse}}{2405.78} \tag{4.3.14}$$

and the SNR axis shown in the graph is related to the PSNR by

$$\begin{aligned} \text{SNR} &= \text{PSNR} + 10 \log_{10}\left[\frac{2405.78}{255^2}\right] \\ &= \text{PSNR} - 14.3 \text{ dB} \end{aligned} \tag{4.3.15}$$




Graph 4.3.1 - Isotropic Multiresolution Gauss-Markov R(D) model

### 4.4. Local Anisotropy

Local anisotropy (feature orientation) is a significant property of natural images and hence is important in any application which involves processing of such images. This is particularly true for applications such as coding, where the resulting images are to be viewed by a human observer.

Objects in the 3-d real world are generally discrete and create oriented edges where they overlap in a projected 2-d image. There is a considerable difference in the correlation between pixel intensities measured along this local orientation and the correlations measured across this orientation, and the image source model should reflect this property.

In addition to this source based property, it turns out that local anisotropy has a significant role in vision. The work of Hubel and Wiesel [44] [45] [46] and others [11] has demonstrated the existence of neurons in the primary visual cortex which respond to the local orientation of features within a region of the retinal image. These neurons give a maximum response for a particular orientation. A set of about 30 groups of such neurons, or orientation columns [44], act over a given local area, each responding to a different orientation, and the output of this set provides local feature orientation information. The organisation of these structures within the cortex is surprisingly regular [44]. The outputs of these neural units are transmitted to deeper and more complex neural processing structures, possibly to perform such tasks as boundary definition, object classification, object recognition, and the other abilities which make up vision. They play a significant part in these abilities, as can be demonstrated when the local orientations are distorted by noise.

The importance of local anisotropy is demonstrated by the effect on the perceptual distortion introduced into an image by oriented noise. Noise introduced near locally oriented features has an effect which depends on the alignment of the noise with respect to the local feature orientation. Noise aligned at 0 degrees to the local orientation causes lower perceived distortion than noise at 90 degrees, even if the mean squared error is identical in both cases. This is demonstrated in [56], where two anisotropically filtered white noise images are presented. In one image the filters were aligned with the local orientation of features in an image of a face, and in the other they are orthogonal to these orientations. The structure of the face is clear in the first, but not in the second, noise image. This can be seen as a suggestion that the outputs from the orientation detection circuits of the visual cortex are used in the process of perception or recognition. When the errors align with the local orientation they have

minimum effect on the outputs of these circuits, when they are misaligned they distort the outputs, and therefore also distort any further processing done on these outputs by deeper structures within the visual pathway.

A number of researchers have used this property of vision to develop coding schemes which seek to minimise the perceived distortions (rather than the mean squared error) by aligning the error with the local orientation [61] [112]. The rest of this chapter describes specific LIM RBN models which incorporate local anisotropy and chapter 5 describes codecs based on these models and shows, with results, that they are capable of very high compression ratios, whilst retaining most of the significant perceptual features of the image.

### 4.5. A Homogeneous Anisotropic LIM RBN Model

A number of methods of incorporating the idea of local anisotropy within the LIM RBN model were considered, the most successful being to use it to derive an oriented linear interpolation.

The oriented linear interpolation replaces the bilinear interpolation of equation (2.5.6) at some level within the model. The level at which it replaces the bilinear interpolator will depend on the scale of local features within an image, and will vary across the spatial plane. The basic scheme of the oriented interpolation is shown in fig 4.5.1, with the intensity of the center pixel $c$ being linearly interpolated from the intensities at the boundary points $a$ and $b$, and the intensities at $a$ and $b$ determined from the intensities at the nearest boundary pixels. No attempt is made here to find an explicit definition of $A(n)$ for the oriented interpolation, but the implementation of the interpolator is given in more detail in section 5.5. The first alteration from the homogeneous LIM model to be

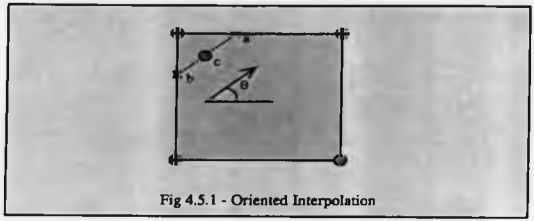



Fig 4.5.1 - Oriented Interpolation

noted is the dependence of $A(n)$ on the local anisotropy at scale $n$, $\theta(n)$ by altering equation (2.2.1) to

$$s(n) = A(n, \theta(n))s(n-1) + B(n)w(n) \qquad (4.5.1)$$

Secondly, note from fig 4.5.1 that all the boundary pixels must be known in order to interpolate the interior of the block. Therefore the innovations sets $\Lambda_n$ at each level of resolution $n$ must include all pixels on these boundaries as shown in fig 4.5.2 rather than simply the corner pixels as in the isotropic LIM RBN models.

Thus for the homogeneous LIM RBN model the definition of the set $\Lambda_n$ for an image of size $2^N+1$ by $2^N+1$ is altered from that of equation (2.4.1) to

$$\Gamma_n = \left\{ \bigcup_{\substack{0 \le i \le 2^N \\ 0 \le j \le 2^n}} (i, j2^{N-n}) \right\} \bigcup \left\{ \bigcup_{\substack{0 \le i \le 2^n \\ 0 \le j \le 2^N}} (i2^{N-n}, j) \right\} \qquad 0 \le n \le N \quad (4.5.2)$$

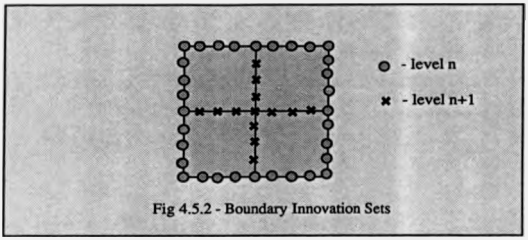



Fig 4.5.2 - Boundary Innovation Sets

and

$$\Lambda_n = \Gamma_n - \bigcup_{m<n} \Lambda_m \qquad 1 \le n \le N \tag{4.5.3}$$

$$\Lambda_n = \Gamma_n \qquad n = 0 \tag{4.5.4}$$

Since there is clearly a considerable degree of correlation between the boundary pixels in the innovation sets at a level, it would not be efficient to consider their innovation values independently. Instead, a one dimensional RBN algorithm is used to reduce the correlation between these values. Thus the innovations operator $B(n)$ is no longer defined by equation (2.5.7). Instead it is defined such that the innovations values $w(n)$ at pixels within the set $\Lambda_n$ are interpolated to other pixels within the set $\Lambda_m$. In particular it corresponds to a one dimensional LIM RBN model for each block edge. Note that this is considered to be a homogeneous model, in that the block splitting is not terminated, so that every pixel in the image has an innovation. This anisotropic model retains all of the homogeneous LIM model properties described in section 2.4.

Photo 4.2 shows $s(N)$ for the new model with the anisotropic parameters of the estimation operator $A(n, \theta(n))$ derived from the GIRL image. Table 4.5.1 shows the variances of the normal white noise $w(n)$ used in the model, which were estimated for each level from the pixels in the innovations sets $A_n$ for the GIRL image. These are substantially lower than these variances for the isotropic model given in table 2.5.1.

Photo 4.2 - Homogeneous Anisotropic LIM RBN Model

| level $n$ | variance | level $n$ | variance |
|---|---|---|---|
| 0 | - | 5 | 146 |
| 1 | 1765 | 6 | 77 |
| 2 | 1670 | 7 | 64 |
| 3 | 2764 | 8 | 39 |
| 4 | 1037 | 9 | 19 |

Table 4.5.1

### 4.6. An Inhomogeneous Anisotropic LIM Model

Instead of splitting down to the smallest scale, particular oriented features within the image may be represented by a block at a given scale, in a similar manner to the inhomogeneous LIM RBN model of section 4.2: if a block at level $n$ meets some acceptance criterion it is not split at the next level. All the pixels within such blocks at level $n$ are represented by the interpolated values produced from the block boundaries by $A(n, \theta(n))$. $\Omega_n$ is defined as in section 4.2 to be the set of all pixels within such blocks at level $n$ excluding those pixels which are members of an innovations set at a previous level. The innovation sets are defined by

$$\Lambda_n^I = \Lambda_n - \bigcup_{m \le n} \Omega_m \qquad 0 \le n \le N \tag{4.6.1}$$

as in equation (4.2.3) with $\Lambda_n$ defined by equations (4.5.2), (4.5.3) and (4.5.4).

The model has the same properties as the inhomogeneous one of section 4.2, and photo 4.3 demonstrates $s(N)$ for the model, with the variances (table 4.6.1), scale selection and local anisotropy parameters estimated from the GIRL image. Note that these variances are larger than the variances in table 4.5.1, since the innovation values for the pixels $\bigcup_n \Omega_n$ are excluded from the estimate. From the very nature of the acceptance criteria the excluded values are small, thus increasing the estimated variances.

Photo 4.3 - Inhomogeneous Anisotropic LIM RBN Model

| level $n$ | variance | level $n$ | [illegible] |
|---|---|---|---|
| 0 | - | 5 | 603 |
| 1 | 1765 | 6 | 254 |
| 2 | 1670 | 7 | 185 |
| 3 | 2764 | 8 | 117 |
| 4 | 1630 | 9 | 68 |

Table 4.6.1

### 4.7. An Inhomogeneous Anisotropic Colour LIM RBN Model

The previous models treated the 2-d signal $s(n)$ as a vector of pixel intensities representing a monochrome image. For colour images each pixel has not only an intensity (luminance) but also colour (chromaticity) components. This colour vector can be represented in a number of ways, the most common being as a 3 component vector representing the red, green and blue (RGB) constituents of the colour. However, in this form the components are highly correlated and so it is inefficient for coding applications.

The YUV representation removes much of this correlation to give 3 new components, the Y or luminance component and two colour components, U and V. The signal model $s(n)$ which is an array of scalar components is therefore extended to an array of vector components $s^c(n)$ in order to represent colour images

$$s^c_{xy}(n) = \begin{bmatrix} s^Y_{xy}(n) \\ s^U_{xy}(n) \\ s^V_{xy}(n) \end{bmatrix} \tag{4.7.1}$$

The structure of the basic anisotropic recursive equation (4.5.1) remains the same except that with $s^c(n)$ now being a vector the linear operators $A(n)$ and $B(n)$ now operate on a two dimensional array of vectors rather than a two dimensional array of scalars and the components of $w^c(n)$ are also vectors as shown in equation (4.7.2).

$$s^c(n) = A(n, \theta(n))\, s^c(n-1) + B(n) w^c(n) \tag{4.7.2}$$

All the properties of the anisotropic inhomogeneous monochrome model of section 4.6 apply (with a suitable transition to vector notation where necessary) to this colour model.

The innovation sets remain identical for all 3 components in this model

$$\Lambda^{TY}_n = \Lambda^{TU}_n = \Lambda^{TV}_n \tag{4.7.3}$$

$$\Omega^Y_n = \Omega^U_n = \Omega^V_n$$

This means that the segmentation structure is identical for all 3 planes. There are two main reasons for keeping the segmentation identical rather than allowing the different planes to be segmented in a different manner. The first is that the linear operator and signal vector models would need to be redefined to allow for an extra dimension (i.e. it would be 3 entirely separate planes rather than one plane with vector components). The second is that in the codecs derived from the model it leads to a greater consistency in the image, i.e. it limits the positions at which gradient discontinuities can occur.

Photo 4.4 shows the inhomogeneous anisotropic colour LIM RBN model with the inhomogeneous and anisotropic parameters derived from the MISS image. Table 4.7.1 shows the variances of the normal white noise $w(n)$ used in the model, which were derived from the GIRL image. The variances are given separately for the Y, U and V planes. The variances generally decrease with scale, and the variances of the Y plane are higher than those of the U and V planes.

Photo 4.4 - Inhomogeneous Anisotropic Colour LIM RBN Model

| level $n$ | variance | | | level $n$ | variance | | |
|---|---|---|---|---|---|---|---|
| | Y | U | V | | Y | U | V |
| 0 | - | - | - | 5 | 61.6 | 20.4 | 4.8 |
| 1 | - | - | - | 6 | 25.8 | 6.7 | 2.5 |
| 2 | 1268.1 | 123.7 | 25.1 | 7 | 22.1 | 1.5 | 1.2 |
| 3 | 487.9 | 343.4 | 18.1 | 8 | 16.6 | 0.7 | 0.7 |
| 4 | 122.6 | 55.48.5 | 8.5 | 9 | 17.4 | 0.6 | 0.6 |

Table 4.7.1

### 4.8. Multiresolution Anisotropic Gauss-Markov Models

The extension of the multiresolution isotropic Gauss-Markov model to include the anisotropic element is made by considering the block correlations to be separable in two orthogonal directions, one aligned with the block orientation $\bar{\theta}$, and one at 90 degrees to the block orientation, as shown in fig 4.8.1, where $\bar{\theta}$ is the local orientation $\theta_{xy}$ averaged over the block, and quantised to one of $N_\theta$ possible values.

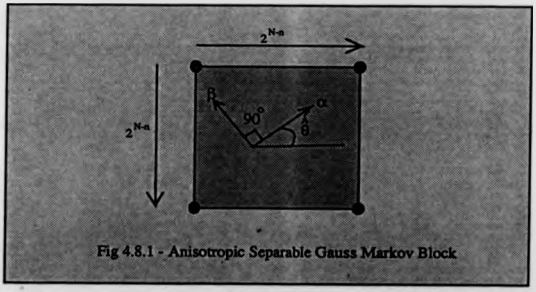



Fig 4.8.1 - Anisotropic Separable Gauss Markov Block

The separate correlation matrices in the α and β axis are then used to derive the

rate distortion function for the block. The stationary first order Gauss-Markov correlation matrices are given by

$$\Phi_n^{\alpha} = \phi_0 \begin{bmatrix} 1 & \rho_\alpha & \rho_\alpha^2 & \cdots & \rho_\alpha^{2^{N-n}-1} \\ \rho_\alpha & 1 & \rho_\alpha & \cdots & \rho_\alpha^{2^{N-n}-2} \\ \rho_\alpha^2 & \rho_\alpha & 1 & \cdots & \rho_\alpha^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_\alpha^{2^{N-n}-1} & \rho_\alpha^{2^{N-n}-2} & \rho_\alpha^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \quad 0 \le \rho_\alpha \le 1 \quad (4.8.1)$$

and

$$\Phi_n^{\beta} = \phi_0 \begin{bmatrix} 1 & \rho_\beta & \rho_\beta^2 & \cdots & \rho_\beta^{2^{N-n}-1} \\ \rho_\beta & 1 & \rho_\beta & \cdots & \rho_\beta^{2^{N-n}-2} \\ \rho_\beta^2 & \rho_\beta & 1 & \cdots & \rho_\beta^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_\beta^{2^{N-n}-1} & \rho_\beta^{2^{N-n}-2} & \rho_\beta^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \quad 0 \le \rho_\beta \le 1 \quad (4.8.2)$$

Where $\rho_\alpha$ and $\rho_\beta$ are the factors between successive correlation terms in the $\alpha$ and $\beta$ axis respectively, noting also that $\phi_0^{\alpha} = \phi_0^{\beta} = \phi_0$.

If the eigenvalues of the matrices $\frac{1}{\phi_0}\Phi_n^{\alpha}$ and $\frac{1}{\phi_0}\Phi_n^{\beta}$ are $\{\lambda_i^{\alpha}\,;\, 1 \le i \le 2^{N-n}\}$ and $\{\lambda_j^{\beta}\,;\, 1 \le i \le 2^{N-n}\}$ respectively then the rate function $R_n^{A}(d)$ for such a block can be defined as

$$R_n^{A}(d) = \sum_{i=1}^{2^{N-n}} \sum_{j=1}^{2^{N-n}} \max\left[0, \frac{1}{2}\log_2\left[\phi_0 \frac{\lambda_i^{\alpha}\lambda_j^{\beta}}{d}\right]\right] + \log_2\left[N_\theta\right] \quad \text{bits per block} \quad (4.8.3)$$

where $N_\theta$ is the number of possible orientations and $\log_2(N_\theta)$ the overhead in representing the orientation. Note that this orientation overhead is treated as fixed, and no distortion of this parameter is allowed. It is also an approximation because of the fixed X-Y alignment of the blocks. The distortion function $D_n^4(d)$ is defined by

$$D_n^4(d) = \sum_{i=1}^{2^{2N-n}} \sum_{j=1}^{2^{2N-n}} \min\left[d,\ \phi_0 \lambda_i^\alpha \lambda_j^\beta\right] \qquad (4.8.4)$$

The multiresolution properties are defined in the same way as in section 4.3 so the anisotropic scale selected stationary first order Gauss-Markov rate distortion function is given by

$$R_n^2(d) = \begin{array}{l} 4P(n)R_{n+1}^2(d) + (1-P(n))R_n^4(d) - \\ P(n)\log_2 P(n) - (1-P(n))\log_2(1-P(n)) \end{array} \qquad 4 \le n < 8 \quad (4.8.5)$$

with initial conditions

$$R_8^2(d) = R_8^4(d) \qquad (4.8.6)$$

and the final function is given by

$$R^2(d) = \frac{R_4^2(d)}{2^{2N-8}} \qquad (4.8.7)$$

The distortion function is similarly defined by

$$D_n^2(d) = 4P(n)D_{n+1}^2(d) + (1-P(n))D_n^4(d) \qquad 4 \le n < 8 \qquad (4.8.8)$$

$$D_{\theta}^{3}(d) = D_{\theta}^{4}(d) \tag{4.8.9}$$

and the final distortion function is given by

$$D^{2}(d) = \frac{D_{4}^{2}(d)}{2^{2N-8}} \tag{4.8.10}$$

The probabilities $\{P(n)\}$ are estimated as in section 4.3 from a coding result of the image. The coder used is the inhomogeneous anisotropic RBN coder of section 5.9. The factors $\rho_{\alpha}$ and $\rho_{\beta}$ are estimated from the image in a similar manner to the isotropic coefficients of section 4.3. The correlation coefficients $\phi_k^{\alpha}$ and $\phi_k^{\beta}$ along the $\alpha$ and $\beta$ axes respectively are estimated by

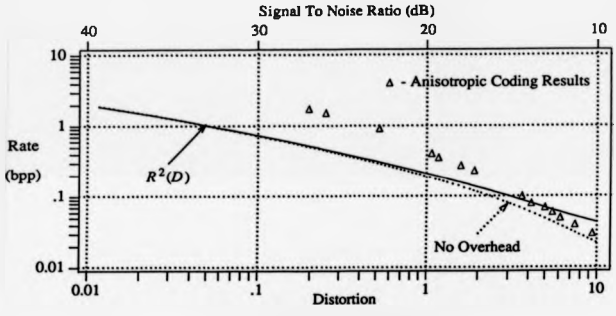
$$\hat{\phi}_k^{\alpha} = \frac{1}{2^{2N}} \sum_{x=0}^{2^N-1} \sum_{y=0}^{2^N-1} \frac{1}{2} \left[ s_{xy}(N) s_{(x+k\cos(\theta_{xy}^{\alpha}))\,(y+k\sin(\theta_{xy}^{\alpha}))}(N) + s_{xy}(N) s_{(x-k\cos(\theta_{xy}^{\alpha}))\,(y-k\sin(\theta_{xy}^{\alpha}))}(N) \right] \quad 0 \le k \le K \tag{4.8.11}$$

$$\hat{\phi}_k^{\beta} = \frac{1}{2^{2N}} \sum_{x=0}^{2^N-1} \sum_{y=0}^{2^N-1} \frac{1}{2} \left[ s_{xy}(N) s_{(x+k\cos(\theta_{xy}^{\beta}))\,(y+k\sin(\theta_{xy}^{\beta}))}(N) + s_{xy}(N) s_{(x-k\cos(\theta_{xy}^{\beta}))\,(y-k\sin(\theta_{xy}^{\beta}))}(N) \right] \quad 0 \le k \le K \tag{4.8.12}$$

where $\theta_{xy}^{\alpha}$ and $\theta_{xy}^{\beta}$ are the orientations of the $\alpha$ and $\beta$ axes, as shown in fig 4.8.1, and $K$ was chosen as 4.

However the points $(x + k\cos\theta,\ y + k\sin\theta)$ will not generally fall exactly on a pixel position so an interpolation is required to estimate the value of $s(\bullet,\bullet)$ at this point. A simple bilinear interpolation from the 4 nearest pixels is used to estimate this intensity.

Graph 4.8.1 shows the rate distortion function for the GIRL image with $\{P(n)\}$ from a coding at 0.10 bpp, as was given in table 4.3.1. The dotted line is the rate distortion function with the overhead of the orientation and block probabilities removed, which is a lower bound on the achievable rate. To obtain a more accurate bound, both the number of orientations and the splitting probabilities need to be varied in order to find the minimum distortion for a given rate. This is discussed further at the end of section 4.9. The triangles on the graph are coding results from the inhomogeneous anisotropic coder described in section 5.9.




Graph 4.8.1 - Anisotropic Multiresolution Gauss-Markov R(D) model

### 4.9. Low-Pass High-Pass Classification

The blocks within an image can be divided into two classes - those with high activity (high-pass blocks) and those with low activity (low-pass blocks) [102] [86] [119] - based on the magnitudes of the orientation vectors within the blocks. The reason for this classification is that the correlation between pixels is significantly higher in the low-pass blocks than in the high-pass blocks, and the division of the block into these two classes allows this property to be used to achieve a higher compression than is possible with only one class. Tasto and Wintz [102] showed that a more efficient rate distortion curve was derived from such a classification, even allowing for the overhead of addressing this classification.

Thus there are now four correlation matrices

$$\frac{1}{\phi_0^l}\Phi_n^{\alpha l} = \begin{bmatrix} 1 & \rho_{\alpha l} & \rho_{\alpha l}^2 & \cdots & \rho_{\alpha l}^{2^{N-n}-1} \\ \rho_{\alpha l} & 1 & \rho_{\alpha l} & \cdots & \rho_{\alpha l}^{2^{N-n}-2} \\ \rho_{\alpha l}^2 & \rho_{\alpha l} & 1 & \cdots & \rho_{\alpha l}^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{\alpha l}^{2^{N-n}-1} & \rho_{\alpha l}^{2^{N-n}-2} & \rho_{\alpha l}^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \qquad 0 \le \rho_{\alpha l} \le 1 \quad (4.9.1)$$

with eigenvalues $\{\lambda_i^{\alpha l};\ 1 \le i \le 2^{N-n}\}$;

$$\frac{1}{\phi_0^h}\Phi_n^{\alpha h} = \begin{bmatrix} 1 & \rho_{\alpha h} & \rho_{\alpha h}^2 & \cdots & \rho_{\alpha h}^{2^{N-n}-1} \\ \rho_{\alpha h} & 1 & \rho_{\alpha h} & \cdots & \rho_{\alpha h}^{2^{N-n}-2} \\ \rho_{\alpha h}^2 & \rho_{\alpha h} & 1 & \cdots & \rho_{\alpha h}^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{\alpha h}^{2^{N-n}-1} & \rho_{\alpha h}^{2^{N-n}-2} & \rho_{\alpha h}^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \qquad 0 \le \rho_{\alpha h} \le 1 \quad (4.9.2)$$

with eigenvalues $\{\lambda_i^{\alpha h}; 1 \le i \le 2^{N-n}\}$;

$$\frac{1}{\phi_0^l}\Phi_n^{\beta l} = \begin{bmatrix} 1 & \rho_{\beta l} & \rho_{\beta l}^2 & \cdots & \rho_{\beta l}^{2^{N-n}-1} \\ \rho_{\beta l} & 1 & \rho_{\beta l} & \cdots & \rho_{\beta l}^{2^{N-n}-2} \\ \rho_{\beta l}^2 & \rho_{\beta l} & 1 & \cdots & \rho_{\beta l}^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{\beta l}^{2^{N-n}-1} & \rho_{\beta l}^{2^{N-n}-2} & \rho_{\beta l}^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \quad 0 \le \rho_{\beta l} \le 1 \qquad (4.9.3)$$

with eigenvalues $\{\lambda_i^{\beta l}; 1 \le i \le 2^{N-n}\}$;

$$\frac{1}{\phi_0^h}\Phi_n^{\beta h} = \begin{bmatrix} 1 & \rho_{\beta h} & \rho_{\beta h}^2 & \cdots & \rho_{\beta h}^{2^{N-n}-1} \\ \rho_{\beta h} & 1 & \rho_{\beta h} & \cdots & \rho_{\beta h}^{2^{N-n}-2} \\ \rho_{\beta h}^2 & \rho_{\beta h} & 1 & \cdots & \rho_{\beta h}^{2^{N-n}-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{\beta h}^{2^{N-n}-1} & \rho_{\beta h}^{2^{N-n}-2} & \rho_{\beta h}^{2^{N-n}-3} & \cdots & 1 \end{bmatrix} \quad 0 \le \rho_{\beta h} \le 1 \qquad (4.9.4)$$

with eigenvalues $\{\lambda_i^{\beta h}; 1 \le i \le 2^{N-n}\}$;

The rate functions for low pass and high pass blocks are given by

$$R_n^{\beta l}(d) = \sum_i^{2^{N-n}} \sum_j^{2^{N-n}} \max\left[0, \frac{1}{2}\log_2\left[\phi_0^l \frac{\lambda_i^{\beta l} \times \lambda_j^{\beta l}}{d}\right]\right] + \qquad (4.9.5)$$

$$\log_2\left[N_\theta\right] \quad \text{bits per block}$$

$$R_n^{\alpha h}(d) = \sum_i^{2^{N-n}} \sum_j^{2^{N-n}} \max\left[0, \frac{1}{2}\log_2\left[\phi_0^h \frac{\lambda_i^{\beta h} \times \lambda_j^{\beta h}}{d}\right]\right] + \qquad (4.9.6)$$

$$\log_2\left[N_\theta\right] \quad \text{bits per block}$$

The distortion functions for low pass and high pass blocks are given by

$$D_n^{AI}(d) = \sum_{i=1}^{2^{N-n}} \sum_{j=1}^{2^{N-n}} \min\left[d,\ \phi_0^2 \lambda_i^{\alpha_I} \lambda_j^{\beta_I}\right] \tag{4.9.7}$$

$$D_n^{AA}(d) = \sum_{i=1}^{2^{N-n}} \sum_{j=1}^{2^{N-n}} \min\left[d,\ \phi_0^2 \lambda_i^{\alpha_A} \lambda_j^{\beta_A}\right] \tag{4.9.8}$$

The average rate and distortion functions for a block of size $2^{N-n}$ by $2^{N-n}$ is given by

$$R_n^{AC}(d) = P(L) R_n^{AI}(d) + (1 - P(L)) R_n^{AA}(d) - P(L)\log_2(P(L)) - (1-P(L))\log_2(1-P(L)) \tag{4.9.9}$$

$$D_n^{AC}(d) = P(L) D_n^{AI}(d) + (1-P(L)) D_n^{AA}(d) \tag{4.9.10}$$

The multiresolution anisotropic classified rate distortion function $R_n^3(d)$ is given by the recursive equation 4.9.11

$$R_n^3(d) = P(n)\, 4\, R_{n+1}^3(d) + (1-P(n))\, R_n^{AC}(d) - P(n)\log_2 P(n) - (1-P(n))\log_2(1-P(n)) \quad 4 \le n < 8 \tag{4.9.11}$$

with initial conditions

$$R_8^3(d) = R_8^{AC}(d) \tag{4.9.12}$$

and the final function given by

$$R^3(d) = \frac{R_4^3(d)}{2^{2N-8}} \tag{4.9.13}$$

The distortion function is similarly defined by

$$D_n^3(d) = P(n)\,4\,D_{n+1}^3(d) + (1-P(n))\,D_n^{AC}(d) \quad 4 \le n < 8 \tag{4.9.14}$$

with initial conditions

$$D_8^3(d) = D_8^{AC}(d) \tag{4.9.15}$$

and the final function by

$$D^3(d) = \frac{D_4^3(d)}{2^{2N-8}} \tag{4.9.16}$$

The low pass and high pass coefficients $\rho_{\alpha l}$, $\rho_{\beta l}$ and $\rho_{\alpha h}$, $\rho_{\beta h}$ are estimated from the original image, as given in equations (4.8.11) and (4.8.12), except that the summations for the low pass estimates are only taken over those pixels which were classified as low pass, and similarly for the high pass estimates. The classification of the pixels is taken from an actual coding result.

Graph 4.9.1 shows the function (solid line), for parameters from a coding of the GIRL image at 0.10 bits per pixel, with $\{P(n)\}$ as in table 4.3.1 and $\{P_n(L)\}$ given in table 4.9.1. The graph also shows $R^3(D)$ (dotted line) and anisotropic

coding results (triangles) as a comparison. From the graph, it is clear that the low pass/high pass classification makes little difference to the mean squared error results. This is in contrast to the work of Tasto and Wintz [102], which shows a clear difference between classified and unclassified rate distortion functions. However, there are a number of differences between this work and that of Tasto and Wintz. Their work was designed around optimising the rate distortion function by varying the classification of blocks into three classes, whereas the classification in this work is into two classes, based simply on an activity measure. More significantly, the work of Tasto and Wintz used a fixed block size segmentation of the image, whereas this work is based on a varying block size segmentation. This segmentation is to some extent a classification of the image, since the low pass areas tend to be congregated into larger blocks, by the very nature of the scale selection criteria, and these larger blocks have a small rate compared with the smaller, generally high pass, blocks.

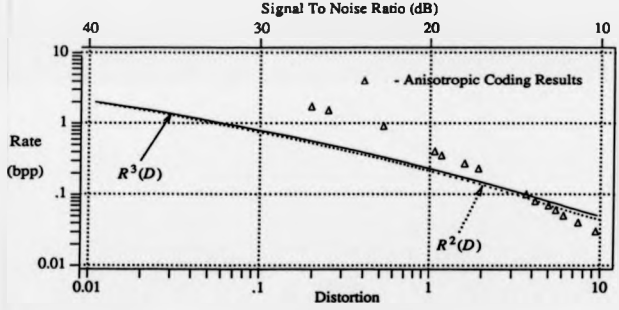
Finally graph 4.9.2 shows all three rate distortion graphs at a larger scale. This does show a noticeable gain at higher rates for the anisotropic model, but at the lowest rates the situation is reversed. While this is due mainly to the orientation coding overhead, it is not an accurate reflection of the subjective comparison between the respective coding results, reflecting the inadequacy of the mean squared error fidelity criterion. Graph 4.9.3 shows the three rate distortion curves without the bookkeeping overheads.

The rate distortion functions reflect the coding results reasonably well around the 0.1 bits per pixel rate at which the parameters for the function were derived. However, at higher rates the function is considerably below that of the coding results, and at lower rates it is above the coding results. This is because the rate distortion function is only realistic in as much as the model on which it is based

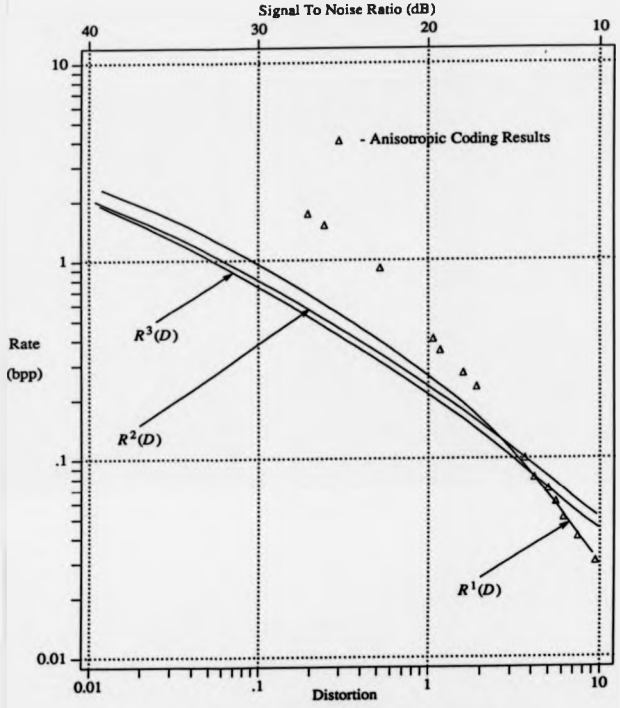
fits the image. Thus while small changes in distortion around the point at which the properties of the model are derived are realistic, for larger changes, the model parameters from this point become less realistic and hence the rate distortion function is less realistic. In other words, the error in the modelling becomes significant with respect to the distortion within the model. Thus as the distortion varies the model parameters must also be varied.

| level $n$ | $P_n(L)$ | level $n$ | $P_n(L)$ |
|---|---|---|---|
| 0 | - | 5 | 0.2640 |
| 1 | - | 6 | 0.1452 |
| 2 | - | 7 | 0.0426 |
| 3 | - | 8 | 0.0000 |
| 4 | 0.0000 | 9 | - |

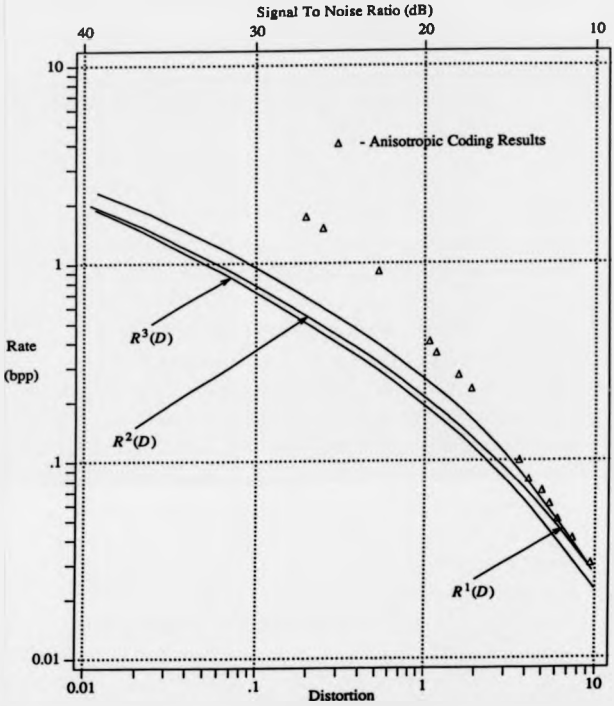
Table 4.9.1




Graph 4.9.1 - Anisotropic Low/High Pass Gauss-Markov R(D) model




**Graph 4.9.2 - Three Different R(D) models**




Graph 4.9.3 - Three Different R(D) models (no overhead)

## 5. Anisotropic Multiresolution Coding

Before describing the coding algorithm and results, it is necessary to consider how to estimate the local orientations from the image data. This is effectively a separate processing step, rather than an integral part of the coder.

### 5.1. Orientation Measure

In this section, the orientation measure necessary for the oriented block interpolation of section 5.6 is considered. The oriented block interpolation is suitable for several different types of feature, including lines, edges and some oriented textures. Indeed, it is suitable for any local feature which is very highly correlated in a given orientation, regardless of its correlation orthogonal to this orientation. Thus the estimation of the orientation measure must be independent of the type of feature in the image. For example, it must give the same orientation for an edge feature with a transition from black to white as it does for an edge feature with a transition from white to black. This implies the existence of a two-fold ($180^0$) rotational symmetry in the orientation. Knutsson [57] has shown that a double angle vector representation is the right approach to this problem. In this representation, the orientation measure is a vector at each pixel, $v_{xy}$, with the magnitude of the vector representing the strength $\beta_{xy}$ or certainty of the local oriented feature, and the angle of the vector representing twice the orientation $\theta_{xy}$ of the feature

$$\theta_{xy} = \frac{1}{2}\arg(v_{xy}) \tag{5.1.1}$$

$$\beta_{xy} = |v_{xy}| \tag{5.1.2}$$

This representation has a secondary advantage in that the averaging of such vectors realistically represents the averaging of the orientation information they represent, which would not be the case, say, for a single angle representation.

The problem of estimating these local orientation vectors was considered in detail by Knutsson [57], who devised an accurate and reasonably efficient algorithm based on a set of four oriented quadrature filters. This estimator is briefly described in section 5.2 and referred to in the rest of the thesis as the full estimator. Unfortunately, this full estimator is computationally expensive in comparison with the other elements of the coders described in this thesis and an alternative estimator requiring less computation was devised. This second estimator is referred to as the fast estimator and is described in section 5.3. While the fast estimator is not as accurate as the full estimator, it was found to be acceptable in practice. The reason for this is that the orientation vectors are averaged over different sized blocks of the image before the orientation information is extracted, and the fast orientation estimate performs adequately within the coder when averaged over these scales.

### 5.2. Full Orientation Estimator

The full orientation estimator consists of four oriented analytic filter pairs [57]. The reason for using analytic filters (quadrature filters) is that they respond equally to line and edge features.

Each filter pair consists of two filters an 'edge' detector $H_e^k$ and a 'line' detector $H_l^k$ which are defined in the polar spatial frequency domain as

$$H_e^k(\rho,\theta) = j\ \mathrm{sgn}\left(\cos(\theta-\theta_k)\right) \exp\left[\frac{-4\ln(2)}{\ln^2(B)}\ln^2\left[\frac{\rho}{\rho_r}\right]\right]\cos^2(\theta-\theta_k) \quad 1 \le k \le 4 \qquad (5.2.1)$$

and

$$H_f^k(\rho,\theta) = \exp\left[\frac{-4\ln(2)}{\ln^2(B)}\ln^2\left[\frac{\rho}{\rho_c}\right]\right]\cos^2(\theta-\theta_k) \qquad 1 \le k \le 4 \qquad (5.2.2)$$

The individual filter pairs are given a specific orientation defined by $\theta_k$ as shown in table 5.2.1,

| k | $\theta_k$ degrees |
|---|---|
| 1 | 0 |
| 2 | 45 |
| 3 | 90 |
| 4 | 135 |

Table 5.2.1

The filter functions of equations (5.2.1) and (5.2.2) and table 5.2.1 have been designed to give an optimal response to an ideal line or edge feature [57].

The angles $\theta_k$ may be offset by 22.5 degrees in order to remove bias introduced by the alignment of the filters with the cartesian axes (the bias is introduced by the cartesian sampling grid imposed on the function) and it also leads to a more efficient implementation in that the four cartesian sampled filter pair kernels are simply rotations and reflections of each other.

If the Fourier transform of the image $s_{xy}$ is denoted by $S_{uv}$ then the filtering operation can be expressed as:-

$$V_{uv}^k = S_{uv}(H_{g\,uv}^k + H_{f\,uv}^k) \qquad 1 \le k \le 4 \qquad (5.2.3)$$

If $V_{uv}^k$ are then inverse transformed to give $v_{xy}^k$ then the final orientation vector is defined by

$$\mathbf{v}_{xy} = \begin{bmatrix} |v_{xy}^1| - |v_{xy}^3| \\ |v_{xy}^2| - |v_{xy}^4| \end{bmatrix} \tag{5.2.4}$$

The theory behind this filtering operation is given in detail in [57]. The result of this orientation estimation is given in photo 5.1 in section 5.3, together with the results of the fast estimator.

The filtering can be implemented in either the spatial frequency or the spatial domain. In the spatial domain, it is performed by convolution with a M by M complex mask [57]. Since there are four such masks, and typically a kernel size of $M = 15$ is required, the convolutions alone require 1800 multiplications per pixel. If it is performed in the spatial frequency domain, it involves 1 fast Fourier transform (FFT) and 4 inverse fast Fourier transforms plus 4 multiplication per pixel for the filter operations and 4 complex magnitude calculations per pixel, giving a total of approximately 150 multiplications per pixel for a 512 by 512 image. It is clearly beneficial to perform the estimation in the spatial frequency domain unless, as in [57], a parallel convolution machine [66] is available. The time complexity of this estimator contrasts with the fast estimator described in the next section which requires only 2 multiplications per pixel.

### 5.3. Fast Orientation Estimator

The fast orientation estimator is a non-linear operator acting on four neighbouring pixels and estimating an orientation for the pixel intensities over the square

defined by the four pixels. Consider the four pixels shown in fig 5.3.1.

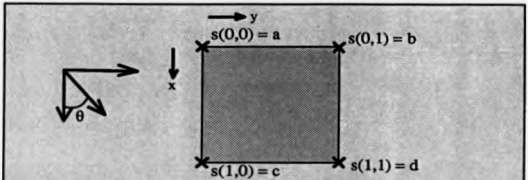



Fig 5.3.1 - Fast Estimator Basis Pixels

The image intensities $s(x,y)$ for the block $(0 \le x,y \le 1)$ are modelled by a simple bilinear interpolation based on these four corner pixels giving the familiar function of equation (5.3.1)

$$s(x,y) = a + (c-a)\,x + (b-a)\,y + (a+d-c-b)\,xy \qquad 0 \le x,y \le 1 \qquad (5.3.1)$$

Define the orientation vector $\mathbf{v}_\theta$ which is to be derived from equation (5.3.1) as

$$\mathbf{v}_\theta = r\begin{bmatrix} v_\theta^x \\ v_\theta^y \end{bmatrix} = r\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \qquad (5.3.2)$$

The value of r is set to 1, since it is the orientation $\theta$ that is of interest. The value of $\theta$ which minimises the integral over the block of the squared value of

the gradient of $s(x,y)$ in the direction $v_\theta$ is the estimate for the block orientation. This can be written as

$$\min_{\theta} \int_0^1 \int_0^1 \left[ \text{grad}(s(x,y)) \cdot v_\theta \right]^2 dx\, dy \qquad (5.3.3)$$

where "$\cdot$" denotes inner product and

$$\text{grad}(s(x,y)) = \begin{bmatrix} \frac{\partial s}{\partial x} \\ \frac{\partial s}{\partial y} \end{bmatrix} \qquad (5.3.4)$$

so the problem can be expressed as

$$\min_{\theta} \int_0^1 \int_0^1 \left[ \frac{\partial s}{\partial x}\cos\theta + \frac{\partial s}{\partial y}\sin\theta \right]^2 dx\, dy \qquad (5.3.5)$$

Solving equation (5.3.5) [Appendix B] gives a solution for $\theta_{min}$ in the form of a vector

$$v_{2\theta_{min}} = m \begin{bmatrix} \cos 2\theta_{min} \\ \sin 2\theta_{min} \end{bmatrix} = \begin{bmatrix} 2(b-c)(d-a) \\ ((b-c)-(d-a))((b-c)+(d-a)) \end{bmatrix} \qquad (5.3.6)$$

with

$$m = (d-a)^2 + (b-c)^2 \qquad (5.3.7)$$

Note that the solution is in the double angle representation of equation (5.1.1) and (5.1.2). The magnitude or normalisation factor $(d-a)^2 + (b-c)^2$ is a measure of the strength of the feature. Since the solution is automatically in the double angle representation and the values of the intensities of the pixels are assumed to be integers, the estimator process requires only two integer multiplications per vector. Note that the estimate should be regarded as the orientation for the point at the centre of the block. However, for this application the vectors are averaged over larger blocks before being used, and thus the vector can be assigned to the top left pixel without any serious bias at the scale at which it is used. This allows each pixel in the image to be regarded as the top left pixel of such a block of four pixels and thus the estimate produces an orientation vector at each pixel in the image at a cost of two multiplications per pixel. It is also worth noting that the model used in deriving this estimator is essentially an edge model, since the size of neighbourhood and the use of a bilinear interpolation precludes features such as lines. Nonetheless at such small scales, this is a reasonable assumption, as is borne out by the results obtained with this method.

The estimated orientation vectors for the GIRL image are shown in photo 5.1. The orientation vectors are displayed as colour images, with the brightness representing the vector magnitude and the colour representing the orientation. The bottom left shows the orientation vectors estimated by the full orientation estimation on a 256 by 256 version of the GIRL image. The top right quadrant shows the orientation vectors estimated by the fast orientation estimation on the same image. Comparing the two, it is clear that although they generally have

the same colour (ie. orientation) there is a considerable difference in the dynamic range of the magnitudes of the two estimates. The coders described in this thesis average these vectors over a given block and then extract the orientation information form this average vector to use as the basis for an oriented interpolation used to represent the block (section 5.6). Thus it is important that, even in regions where the magnitude is low, the orientations are correctly estimated. In order to display this information, a hierarchical procedure [9] which performs some normalisation on the vector magnitudes, but which does not alter the orientations, is employed. The result of this is shown in the bottom right hand quadrant of photo 5.1. Clearly these orientations are comparable to those of the full estimator. The question arises as to whether the normalised or unnormalised estimate should be used in the coder, but since the unnormalised estimate performed adequately, this question has not been addressed, and the results presented use the unnormalised estimate. In section 5.10 two similar coding results (photo 5.14 and photo 5.15) are presented based on the different estimators in order to compare their performance.

Photo 5.1 Orientation Estimates (GIRL)

### 5.4. Orientation Consistency

The orientation vectors derived from either estimator are used to determine whether a particular block within the RBN splitting algorithm has a single consistent orientation. Blocks which are found to have a consistent orientation are then represented by the oriented interpolation scheme described in section 5.5, whereas blocks which do not have this single orientation must be split into four subblocks in the usual manner of the RBN algorithm.

Only the intermediate levels of resolution ($n = 4$ to $n = 8$ in a 512 by 512 image) are considered for this representation, it being considered unsuitable to represent the other levels as a single oriented block. Larger blocks ($n < 4$) are unlikely to contain such single features and even if they do, the extra cost in the bit rate to split it into blocks at level $n = 4$ is insignificant, whereas the computation involved in processing large blocks is significant. Thus the algorithm automatically splits down to level $n = 4$ before checking for consistent blocks. At level $n = 9$ (in a 512 by 512 image), the oriented interpolation of section 5.5 is not a significant improvement on the bilinear interpolation, but the cost of the orientation overhead for such a block is significant, and so the orientation is not used at this level.

The criterion for determining whether a block contains a single orientation depends on the orientation vectors $\mathbf{v}_{ij}$ over the block $x \le i \le x+sx$ and $y \le j \le y+sy$. Defining $\hat{\mathbf{v}}^m$ as the sum of the vector magnitudes

$$\hat{\mathbf{v}}^m = \sum_i \sum_j |\mathbf{v}(i,j)| \qquad (5.4.1)$$

and $\vec{v}$ as the vector sum of the orientation vectors

$$\vec{v} = \sum_i \sum_j v(i,j) \tag{5.4.2}$$

A consistency measure [56] for the block can be defined by

$$c = \frac{|\vec{v}|}{\vec{v}^m} \tag{5.4.3}$$

i.e. $c$ is the length of the sum of the vectors divided by the sum of the lengths of the vectors. Clearly $c$ is in the range of 0 to 1 , being close to 0 when the vectors are randomly oriented and 1 when the vectors have a single orientation. In addition, $\vec{v}^m$ is a measure of the activity within the block and can be used to classify the blocks into low pass and high pass blocks.

It is possible, however, to get a feature set within a block which has a single orientation, but is not consistent along the direction of that orientation. This occurs at the ends of features such as lines or edges which do not intersect other features, and in textures. In order to detect this problem, the oriented interpolation given in section 5.6 is performed on any block which satisfies the single orientation measures $c$ and $\vec{v}^m$, and the maximum error $m$ between the interpolated intensities of the pixels within this block and the corresponding intensities in the original is calculated. A decision variable $d_{ij}(n)$ on whether to split a block into four quadrants can be determined by comparing these three measures to set thresholds $t_n^1$, $t_n^2$ and $t_n^3$ (which will vary with scale) as given in equation 5.4.4

$$d_{ij}(n)=\begin{cases}1 \text{ iff } ((c<t_n^1)\wedge(\bar{v}^m>t_n^2))\vee(m>t_n^3)\\ 0 \text{ else}\end{cases} \tag{5.4.4}$$

where $d_{ij}(n) = 1$ denotes a splitting of the block. Thus a block is acceptable at scale n if the orientation is sufficiently consistent or if it is sufficiently low pass.

Note that a block at a high level may satisfy $d_{ij}(n) = 0$, but had that block been split into subblocks, some of the subblocks may not have satisfied the criterion. This can occur where a large feature encompassing most of the block dominates a feature confined to one small area of the block, such as a corner. Thus the coder examines all possible subblocks of each block, and the block fails to meet the consistency criterion if any of its subblocks fail.

In order to code the orientation overhead required by the decoder, the average block orientation $-\frac{\pi}{2} \le \hat{\theta} \le \frac{\pi}{2}$ derived from the average orientation vector $\bar{v}$ for such a block is quantised to give $\hat{\theta}_q$, the quantising function being scaled by a scale dependent factor $\rho_n$.

$$\hat{\theta}_q = Q_\theta\left[\frac{\hat{\theta}}{\rho_n}\right] \qquad \rho_n \ge 1 \tag{5.4.5}$$

Based on the fact that Hubel and Wiesel found about 30 different orientation detectors in their work [44], the maximum number of orientations was set at 31, i.e. the possible indices are $[0 \cdots 30]$. The index 31 was used to denote an inconsistent block. This means that the coding of the orientation values can incorporate the addressing information. Since the orientation requires less precision at smaller scales, the orientation vector $\hat{\theta}$ is scaled by a scale dependent variable $\rho_n$, which gets larger at smaller scales.

## 5.5. Prefiltering

The aim of the prefiltering is to reduce the noise in the image without substantially distorting the significant features of the image. This smoothing of the image generally has the effect of reducing the bit rates achievable at the cost of some loss of sharpness (or even some loss of detail) within the image. It is necessary because the new coder, in common with most predictive coders, is sensitive to noise in the original image. In the anisotropic RBN coder, the prefiltering has the effect of removing 'brush stroke' artefacts from the image at lower bit rates. These artefacts occur because of the spatial connectivity of the edge sharing blocks and the oriented interpolation. At lower rates the quantisation errors can lead to smoothly varying features at the edges of blocks being accentuated into sharp features. This in itself is not too significant, but the oriented interpolation from these edges 'brushes' these accentuated features out into a larger linear features as shown in fig 5.5.1

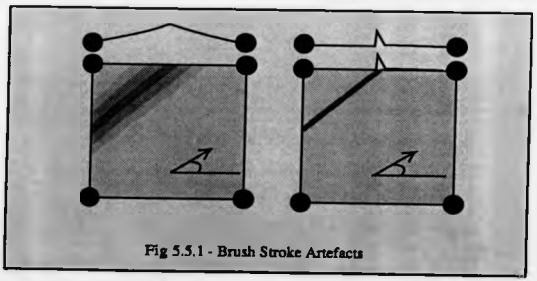


Fig 5.5.1 - Brush Stroke Artefacts

This leads to the distinct brush stroke artefacts seen in the image at lower bit rates. These artefacts appear more visible in low pass regions than in high pass regions, presumably because of edge masking effects, as shown in photo 5.2, which is a coding result for the GIRL image at 0.32 bpp, and with a PSNR of 33.44 dB.

Photo 5.2 - Anisotropic RBN Result at 0.32 bpp (GIRL)

Because of the local orientation dependence, the artefacts generally align with the orientation of nearby features, hence giving the image a 'painted' feel to it, as if it had been smeared with a wide brush. The use of a prefilter eliminates much of the noise which causes these artefacts and allows for improved results at lower rates. Note that the prefilter is required to eliminate noise in the low pass regions, but in high pass regions such as lines or edges it must not smooth the features, since this could cause a large visual distortion in the image. Therefore the filters used should be adaptive to local orientation, with a larger degree of smoothing in low pass regions, a reasonable degree of smoothing along local features, but little if any smoothing across such features. There are a number of

possible implementations of such a filter, but perhaps the best solution is the work by Knutsson Wilson and Granlund [56]. Alternatively, the quadtree based restoration scheme of Clippingdale [21] could be used.

In this work a simple scheme was used. It consists of filtering spatially with a 9 by 9 filter kernel which is adapted across the image, with a filter kernel being selected for each pixel in the image. This adaptation implies that the scheme cannot be performed in the Fourier domain, and thus it required $9 \times 9 = 81$ multiplications per pixel, and is therefore expensive compared to the rest of the codec (particularly if the fast orientation estimator is used). The schemes given in [21] may be more suitable, but were unavailable until comparatively late in the project. The filter for each pixel is derived from the orientation vectors in the local region around the pixel, and fall into one of three classes:

1) These filters are selected in areas of low pass activity, determined as those areas in which $\hat{\mathbf{v}}^m$ of equation (5.4.1) is less than some threshold. The filter used in these regions is a circular Gaussian filter, with a relatively large variance.

2) These filters apply to the regions of high activity but where this activity has no consistent orientation, e.g. corners. This is determined from the coefficient $c$ of equation (5.4.3) and the value of $\hat{\mathbf{v}}^m$ of equation (5.4.1). The filter for this class consists of a circular Gaussian function with a relatively small variance.

3) These filters are selected in areas where there is a high pass features such as an edge, in a consistent orientation. These are determined from the coefficient $c$ of equation (5.4.3) and the value of $\hat{\mathbf{v}}^m$ of equation (5.4.1).

The filters are an elliptical Gaussian function, with the major axis of the ellipse aligned along the local feature orientation (as defined by $\hat{\theta}$ in equation (5.4.5) ) and the minor axis orthogonal to it. This ensures that it smoothes more along the feature than across it.

The three classes of filter are shown in fig 5.5.2.

For the colour images, the Y, U and V planes are prefiltered, using the same filter for all planes at each pixel.

The results of the prefiltering are shown in photo 5.3 for the GIRL image, and photo 5.4 shows a sample (at 10 by 10 pixel spacing) of the filters used to produce photo 5.3.

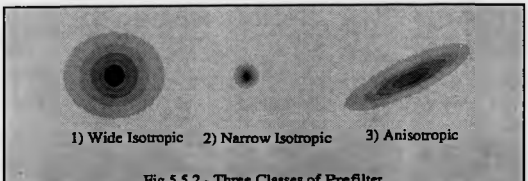


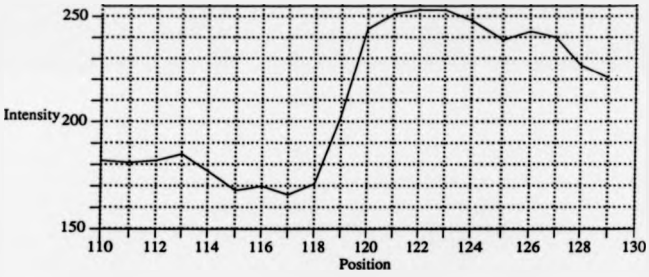

Fig 5.5.2 - Three Classes of Prefilter

In order to demonstrate the effect of the filters on an edge, a one dimensional profile of the left hand edge of the hat in the GIRL image was taken along the horizontal axis. The actual profile is for the line 150 and pixels 110 to 130

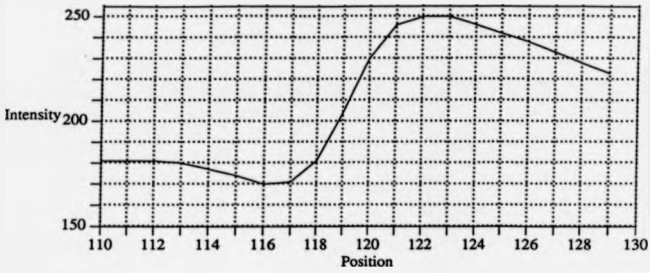
Photo 5.3 Anisotropic Prefiltering (GIRL)

Photo 5.4 Sample Anisotropic Filters (GIRL)

along that line. Graph 5.5.1 shows the profile of the edge in the original image and fig 5.5.2 shows the profile of the prefiltered image.




Graph 5.5.1 - Edge Profile for Original GIRL Image




Graph 5.5.2 - Edge Profile for Prefiltered GIRL Image

It is clear that there is some reduction in the bandwidth of the image, since the prefiltering increases the width of the transition region by about one pixel. Since this must by definition reduce the rate required for a given image, table 5.4.1 shows the entropies of the prefiltered images for the test images GIRL, BOATS and LAKE, as a comparison with table 1.2.1, and table 5.4.2 shows the isotropic noiseless (with respect to the prefiltered image) coding results in comparison with those of table 3.1.3, which are repeated here for convenience, along with the mean squared error due to filtering. A comparison for a low rate coding result is 28.67 dB PSNR at 0.10 bpp for the prefiltered result, and, for the same set of parameters, 30.71 dB PSNR at 0.18 bpp for the unprefiltered result. Performing a linear interpolation between the nearest prefiltered results given in table 5.9.1 gives '30.43' dB at '0.18' bpp, very close to the unprefiltered result. Clearly the prefiltering has little effect on the PSNR results and, from the subjective appearance of the prefiltered images, and the mean

squared error figures, it is clear that the main effect of the filtering is the removal of noise. A comparison for a high rate coding result is 30.99 dB PSNR at 0.25 bpp for the prefiltered result, and, for the same set of parameters, 35.58 dB PSNR at 0.59 bpp for the unprefiltered result.

| Entropy | Image | | |
|---|---|---|---|
| | GIRL | LAKE | BOATS |
| $E^0$ | 7.20 | 7.02 | 7.59 |
| $E^1$ | 3.63 | 3.38 | 4.04 |
| $E^D$ | 3.91 | 3.64 | 4.48 |

Table 5.4.1

| Image | Isotropic Noiseless RBN Bit Rate (bpp) | | Prefiltering Error | |
|---|---|---|---|---|
| | prefiltered image | unfiltered image | mse | PSNR |
| GIRL | 2.46 | 4.29 | 35.37 | 32.64 |
| BOATS | 2.86 | 4.61 | 60.49 | 30.31 |
| LAKE | 3.48 | 5.68 | 155.68 | 26.20 |

Table 5.4.2

### 5.6. Oriented Interpolation

The average orientation vector $\bar{\mathbf{v}}$ over a block at scale $n$ is used to derive the orientation for that block when it has satisfied the consistency measures. The orientation $\hat{\theta}$ is simply given by

$$\hat{\theta} = \frac{1}{2}\arg[\bar{\mathbf{v}}] \tag{5.6.1}$$

The block orientation $\hat{\theta}$ is used to derive a simple linear interpolator which interpolates all the interior pixels from the pixels on the block boundary as

shown in fig 5.6.1.

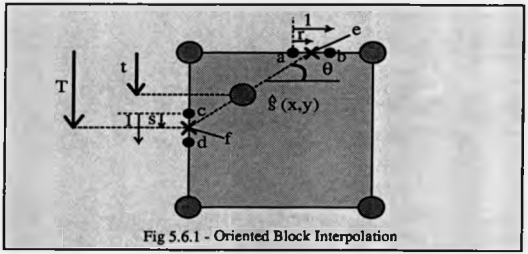



Fig 5.6.1 - Oriented Block Interpolation

The value of the pixel intensity $\hat{s}(x,y)$ is simply a bilinear interpolation from the four pixels {a,b,c,d} shown in fig 5.6.1

$$\hat{s}(x,y) = \frac{t}{T}\,\hat{s}(f) + \frac{T-t}{T}\,\hat{s}(e) \qquad (5.6.2)$$

where

$$\hat{s}(e) = r\,b + (1-r)\,a \qquad (5.6.3)$$

and

$$\hat{s}(f) = s\,d + (1-s)\,c \qquad (5.6.3)$$

These three interpolations are performed for each pixel in the interior (excluding the boundaries) of the block.

The intersection points for all pixels in the block can be determined efficiently, since the orientation is constant for the block. This means that once the line intersection points have been calculated for a given pixel in the block the intersection points for all other pixels can be determined by simple additions and decisions (to determine when the intersection point traverses a block corner) as can the values of $r$, $s$, $t$, and $T$. Since each linear interpolation costs 1 multiplication and there are three such interpolations to derive the intensity of an interior pixel, the complexity of the process is approximately three multiplications per pixel.

### 5.7. Boundary Pixels

The interpolator in section 5.6 was based on the boundary pixels of a given block. This means that, unlike the standard RBN algorithm; where only the corner pixels of the blocks are required, the entire boundary set must be available as described in section 4.5 on the anisotropic model. These boundary pixels must be coded in some form.

Various schemes were investigated for coding the boundary pixels including a one dimensional transform (DCT), however the most suitable was found to be a one dimensional version of the RBN algorithm both because of its simplicity and efficiency, and its natural connection with the rest of the codec (i.e. its multiresolution structure and spatial consistency). The anisotropic RBN codec is implemented by coding the corner pixels of the block in the same manner as the isotropic codec at each split, until a block meets the orientation consistency criteria. The corner pixels of this block are now known and so for each edge of the block the two end pixels of the edge are known, and a one dimensional RBN starting with these two pixels can be performed. The one dimensional RBN

algorithm is illustrated in fig 5.7.1 Each new pixel is coded as a quantised prediction error, with the predictor being simply the average of its two neighbours at the previous resolution.

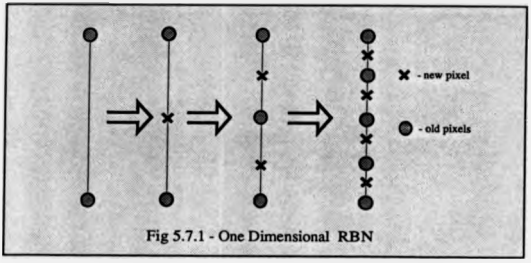



Fig 5.7.1 - One Dimensional RBN

In the inhomogeneous case the splitting of the block is terminated at this stage, and the block is represented by the oriented interpolation from these coded boundary pixels, so that the interior pixels are not coded by a prediction error.

In the homogeneous case, the block is split down to the lowest levels, such that every pixel is coded by a prediction error. This is to allow for high rate and noiseless coding. This is achieved by continuing the RBN splitting process to the lowest level, even after the block meets the orientation criterion. At each split subsequent to a block meeting the criteria, all the boundary pixels of the new blocks are coded using the oriented interpolation, based on the boundary pixels of the block which is being split, as the predictor for the new boundary pixels. This is illustrated in fig 5.7.2.

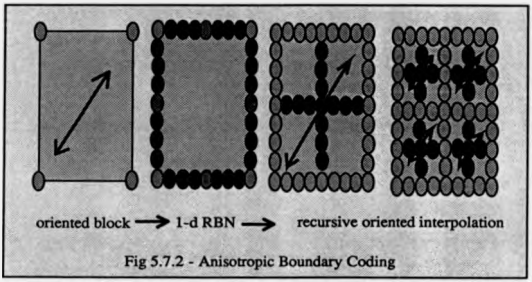



Fig 5.7.2 - Anisotropic Boundary Coding

### 5.8. A Homogeneous Anisotropic Codec

The homogeneous anisotropic codec corresponds to the homogeneous anisotropic model of section 4.5. There is no scale selection on this coder, i.e. the RBN framework continues down to a single pixel level everywhere in the image and every pixel in the image is coded. However, the orientation information is still scale selected (in this sense the model and coder are, strictly speaking, inhomogeneous), leading to a segmentation of the image into edge sharing blocks, each with its own quantised orientation (except in those areas where there is no consistent orientation, which are treated as isotropic). The prediction errors are therefore variously based on either the bilinear interpolation (before an orientation block is scale selected), the one dimensional RBN algorithm (the edges of a scale selected block), or the oriented interpolation (the interior of a scale selected block), as described in section 5.6. This codec allows for noiseless coding (by removing the quantisation of the prediction errors) the results of

which are given in table 5.8.1. These figures are the actual bit rates of noiseless coding results rather than entropies. Note that the quantised block orientation information and the addressing of these orientation blocks is included in the bit rates (though it only accounts for approximately 0.5% of the total rate). Note also that these results are almost identical to the isotropic noiseless coding results, which suggests that either the anisotropic properties are not a significant source property (with respect to noiseless coding), or that the anisotropic model used is not the best way to model anisotropy. Since many regions of these images clearly contain oriented features, it might be suspected that the latter is the case. A closer examination of the images shows, however, that while the 'piecewise linear' model of features is a reasonable approximation it is far from perfect. Complexity of boundary shape - curvature and irregularities - make the model an approximation to the image. This seems the likelier reason for these noiseless coding figures.

| Image | Bit Rate (bpp) |
|---|---|
| GIRL | 4.25 |
| BOATS | 4.46 |
| LAKE | 5.76 |
| MISS | 8.81 |
| TREV | 7.56 |
| SPLIT | 8.35 |

Table 5.8.1

### 5.9. An Inhomogeneous Anisotropic Codec

The inhomogeneous anisotropic codec corresponds to the inhomogeneous anisotropic model of section 4.6. All the photographic results shown in this section have been prefiltered with the anisotropic prefilter described in section 5.4. The codec is initialised by the coding of the four corner pixels of the

images. The image is then split in the normal RBN manner, coding the quantised prediction errors of the corner pixels at each stage. For levels 4 to 8 (on a 512 by 512 image), each block is tested against the orientation criterion of equation (5.4.4). If the criterion is not met, then this is coded by an 'inconsistent orientation' symbol, and the block is split in the normal RBN manner. If the block satisfies the criterion, then the quantised average block orientation is coded. Then the four edges of the block are coded by the one dimensional RBN algorithm given in section 5.6. All the interior pixels of the block are now represented by the oriented interpolation (using the quantised block orientation) from the edge pixels and the algorithm moves on to the next block. If a block at level 8 is not accepted, then it is split all the way down to the single pixel level (i.e. blocks at levels less than 8 are not considered efficient). Note that the 'inconsistent orientation' code must be distinguishable from the quantised orientations, i.e. the decoder must be able to read the symbol and use it to determine if a) it is the inconsistency symbol and therefore the block has been split, or b) it is a quantised orientation, therefore it has not been split. Thus the addressing is intimately linked to the coding of the orientation values. Table 5.9.1 gives a list of bit rate vs mean squared error and peak signal to noise ratio for the results of this codec, with the higher rates (>0.27 bpp) derived from unprefiltered images.

Photos 5.5 to 5.7 show the results of the codec at low bit rates for the three test images. Note that in general these images have a painted look to them, and that although they have considerable differences from the originals they still look like consistent images, and display no blocking artefacts. At a first glance, the main features of the images stand out, giving a clear impression of the image, yet closer study reveals considerable differences from the originals. Note that the particularly low PSNR for the LAKE image arises from the clash of the

| Image | Bit Rate (bpp) | mse | PSNR (dB) |
|---|---|---|---|
| GIRL | 1.71 | 4.72 | 41.39 |
| | 1.50 | 5.94 | 40.39 |
| | 0.91 | 12.60 | 37.13 |
| | 0.40 | 25.72 | 34.03 |
| | 0.35 | 28.18 | 33.63 |
| | 0.32 | 29.44 | 33.44 |
| | 0.27 | 38.15 | 32.32 |
| | 0.23 | 45.75 | 31.53 |
| | 0.10 | 88.23 | 28.67 |
| | 0.08 | 99.79 | 28.14 |
| | 0.06 | 132.73 | 26.90 |
| | 0.04 | 179.28 | 25.60 |
| | 0.02 | 305.22 | 23.28 |
| | 0.01 | 524.16 | 20.94 |
| BOATS | 0.14 | 135.54 | 26.81 |
| | 0.44 | 78.18 | 29.20 |
| LAKE | 0.25 | 279.94 | 23.66 |
| | 0.82 | 175.82 | 25.68 |

Table 5.9.1

trees (black) with the sky and lake (white), leading to large errors in those areas where the codec (mainly in the prefiltering) has merged or overlapped the two, even though these errors are not particularly significant to the human observer. Photos 5.8 to 5.10 show the results of the codec at high bit rates for the three test images.

Photos 5.11, 5.12 and 5.13 show the block segmentations, coding error magnitude, and result for a coding of the GIRL image at 0.08 bits per pixel and a peak signal to noise ratio of 28.14dB. The block segmentation shows the distribution of the blocks across the image, as well as the orientation of each block, which is represented by a line in the direction of the orientation, passing through the centre of the block. The error magnitude image has been scaled by a non-linear function for display purposes. Note that a close inspection of the curved edges

(eg. the shoulder) in the reconstructed image shows them to be piecewise linear.

Photo 5.5 - 0.10 BPP, 28.67 PSNR

(GIRL)

Photo 5.6 - 0.14 BPP, 26.81 PSNR

(BOATS)

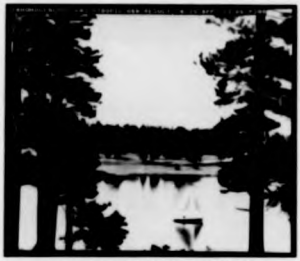


Photo 5.7 - 0.25 BPP, 23.66 PSNR

(LAKE)

Photo 5.8 - 0.25 BPP, 30.99 PSNR

(GIRL)

Photo 5.9 - 0.44 BPP, 29.20 PSNR

(BOATS)

Photo 5.10 - Result 0.82 BPP, 25.68 PSNR

(LAKE)

Photo 5.11 - 0.08 BPP, 28.14 PSNR

(GIRL)

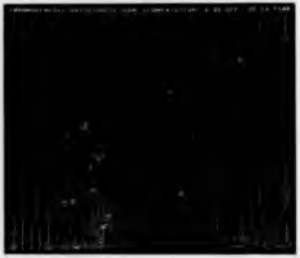


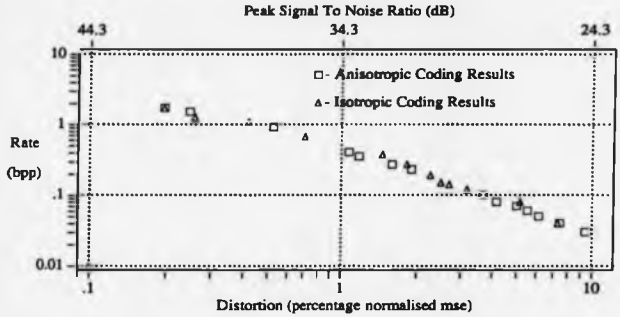
Photo 5.12 - Block Segmentation at 0.08 bpp

(GIRL)

Photo 5.13 - Coding Error at 0.08 bpp

(GIRL)

Finally, Graph 5.8.1 plots the anisotropic (triangles) and isotropic (squares) RBN codec results on the same graph, for a comparison.




Graph 5.9.1 - Isotropic vs Anisotropic Coding Results

It is clear from the graph that the isotropic and anisotropic coders perform almost identically in terms of mean squared error, and yet there is a considerable difference in the visual quality of the two. This suggests that either the significance of orientation with respect to coding is as a property of the human visual system rather than a source property, or that the anisotropic source model used is not particularly good (with respect to mean squared error). Although it is clear from the work of Hubel and Wiesel [44] and others that orientation is a significant property of the human visual system, it is also true that the anisotropic source model is relatively simple, and it is possible that a more complex

anisotropic model may lead to reduced mean squared error results (although of course this does not guarantee better visual quality). It is hard to assess the likelihood of these explanations without considerable further work. In all probability, some combination of the two is near the truth: the distortion criteria should be modified to take account of orientation selectivity and a refinement of the source model is required.

### 5.10. An Inhomogeneous Anisotropic Colour Codec

A selection of the results for the colour codec is given in table 5.10.1.

| Image | Bit Rate (bpp) | | | | | PSNR (dB) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Y | U | V | O | Total Bit Rate | Y | U | V | R | G | B | Average YUV | Average RGB |
| MISS | 0.09 | 0.04 | 0.03 | 0.04 | 0.20 | 33.03 | 35.41 | 35.83 | 28.45 | 33.59 | 26.26 | 34.54 | 28.48 |
| | 0.14 | 0.07 | 0.05 | 0.16 | 0.43 | 34.21 | 36.76 | 37.46 | 29.40 | 34.86 | 27.26 | 35.91 | 29.52 |
| | 0.27 | 0.08 | 0.08 | 0.16 | 0.60 | 36.58 | 40.05 | 39.77 | 32.93 | 36.04 | 30.79 | 38.50 | 32.75 |
| TREV | 0.16 | 0.03 | 0.03 | 0.06 | 0.29 | 31.63 | 40.35 | 38.77 | 29.27 | 31.83 | 27.15 | 35.17 | 29.01 |
| | 0.22 | 0.03 | 0.04 | 0.12 | 0.41 | 32.47 | 40.33 | 38.79 | 29.85 | 32.68 | 27.40 | 35.20 | 29.46 |
| | 0.37 | 0.03 | 0.04 | 0.13 | 0.57 | 35.21 | 42.77 | 41.55 | 33.03 | 34.87 | 31.12 | 38.50 | 32.74 |
| SPLIT | 0.30 | 0.07 | 0.07 | 0.11 | 0.56 | 29.75 | 38.01 | 38.21 | 27.63 | 30.00 | 26.01 | 33.41 | 27.58 |
| | 0.38 | 0.09 | 0.08 | 0.16 | 0.71 | 30.22 | 38.44 | 38.66 | 28.13 | 30.40 | 26.54 | 33.87 | 28.08 |
| | 0.58 | 0.12 | 0.10 | 0.18 | 0.99 | 35.05 | 40.55 | 40.98 | 32.46 | 34.63 | 30.59 | 37.94 | 32.26 |

Table 5.10.1

The peak signal to noise ratios for the Y, U, V, R, G and B planes are quoted as defined in equation (1.2.5), with respect to a maximum of 255. The average RGB and average YUV PSNR's are calculated by taking the average of the three mean squared errors for the planes, and then quoted as a PSNR with respect to a peak value of 255. Photo 5.14 shows two results for the 256 by 256 MISS image, at 0.20 bits per pixel. The difference between the two results is

Photo 5.14 - Inhomogeneous Anisotropic RBN Result (MISS)

due to the estimator used to derive the orientation vectors. The image on the left uses the vectors derived from the fast orientation estimator, the one on the right uses the vectors from the full orientation estimator. Both images are prefiltered with approximately the same filter set, with the slight difference coming from the fact that the prefiltering on the left image is controlled by the fast orientation estimate, and that on the right image by the full orientation estimate. The block segmentation shown below each image corresponds to the RBN splitting of the image, and the lines within each block represent the block orientation. Note that these orientations are normalised since no orientation magnitudes are coded for the blocks.

Photo 5.15 shows two results for the 256 by 256 TREV image, at 0.30, and 0.29 bits per pixel. Again the result on the left and the block segmentation are for the fast orientation estimator and on the right for the full orientation estimator.

Photo 5.16 shows results for the 256 by 256 SPLIT image, at 8.42, 0.99, and 0.29 bits per pixel. Note that the noiseless result is not the optimal result

Photo 5.15 - Inhomogeneous Anisotropic RBN Result (TREV)

Photo 5.16 - Inhomogeneous Anisotropic RBN Result (SPLIT)

(section 5.8 gives a slightly lower rate noiseless result). The image at the top right is coded at 0.99 bpp and has no prefiltering, while the image at the bottom left is at 0.56 bpp and is prefiltered. The segmentation image shown in the bottom right is displayed as a colour map, where each colour within a block corresponds to the particular orientation of that block. The reason for showing the orientation segmentation in this form, rather than with the line drawings

previously used is to show up the black areas within the image, which represent areas with no consistent orientation, which are split isotropically (i.e. with the predictor being bilinear) down to single pixel scale.

### 5.11. Comparison of Results

In this section, the quality of the results of the anisotropic coders described in this thesis are compared with published work on other coding schemes. Two different measures are used: 1) the mean squared error based PSNR and 2) visual quality. Although the second gives a better measure of the coders performance, the first is included since it gives an exactly defined quantity, whereas the visual quality is somewhat open to interpretation, particularly when comparing published work, where there is a heavy dependence on the quality of the printing.

Table 5.11.1 gives a comparison between the PSNR results for this work taken from table 5.9.1 and other published work. All the figures quoted are for comparable 512 by 512 8-bit images. Note that, for the GIRL image, the results presented in this thesis are better than all except the transform-predictive competitive coder of Wilson [110]. For the BOATS and LAKE images, the PSNR performance is not as good as the GIRL image, possibly because of the greater degree of texture in the images.

The anisotropic coders presented in this thesis perform particularly well in terms of visual quality. The results at 0.25 and 0.1 bpp for the GIRL image show no blocking on the edges, although in the 0.1 bpp result some distortions are starting to be noticeable, particularly around the mouth and eyes (which are particularly sensitive in terms of visual quality). The BOATS and LAKE

images also performed well in terms of visual quality, showing none of the blocking around edges often seen with other block coders, although, not surprisingly, at the lowest rates there is some loss of detail.

| Image | Anisotropic Coder Results | | Published Results | | |
|---|---|---|---|---|---|
| | Rate | PSNR | Rate | PSNR | Authors |
| GIRL | 4.25 | noiseless | 5.07 | noiseless | Ho & Gersho * [43] |
| | 0.91 | 37.13 | 1.06 | 37.0 | Wilson [111] |
| | | | 1.0 | 31.55 | Ramamurthi & Gersho [86] |
| | | | 1.0 | 34.8 | Cohen and Woods [23] |
| | 0.40 | 34.03 | 0.50 | 28.55 | Ramamurthi & Gersho [86] |
| | | | 0.50 | 30 | Wilson et al [112] |
| | | | 0.45 | 32.5 | Cohen and Woods [23] |
| | 0.27 | 32.32 | 0.28 | 30.61 | Ho & Gersho [43] |
| | | | 0.27 | 27 | Wilson et al [112] |
| | | | 0.25 | 32.8 | Wilson [110] |
| | | | 0.25 | 30 | Kim et al [53] |
| | 0.23 | 31.53 | 0.19 | 29.58 | Feng and Nasrabadi † [31] |
| | 0.10 | 28.67 | | | |
| | 0.06 | 26.90 | 0.06 | 24.70 | Ho & Gersho [43] |
| LAKE | 0.25 | 23.66 | 0.37 | 25.8 | Wilson [110] |
| | 0.82 | 25.68 | | | |
| BOATS | 4.46 | noiseless | 5.26 | noiseless | Ho and Gersho * [43] |
| | 0.44 | 29.20 | 0.38 | 30.40 | Ho and Gersho [43] |

Table 5.11.1

† Feng and Nasrabadi quote figures for the green plane of the GIRL image (whereas the image used in this work is believed to be the Y plane).

* Ho and Gersho use a progressive transmission technique and therefore the noiseless figures are not a very good comparison.

# 6. Conclusions And Further Work

## 6.1. Thesis Summary

The work in this thesis is concerned with the issues involved in image data compression, based around multiresolution linear signal models, and the development of codecs utilising the fast predictive framework of such models.

In chapter 2 a Linear Interpolative Multiresolution (LIM) model was defined and developed into the LIM RBN model, which is based on the Recursive Binary Nesting algorithm developed at BTRL [107]. The model is non-causal in the spatial dimensions, but causal in the scale dimension. The correlation properties of the model were considered and an example image derived from such a model was presented.

In chapter 3, the basic RBN codec corresponding to the LIM RBN model of chapter 2 was considered. The segmentation, prediction, scalar quantisation and entropy coding schemes used in the homogeneous RBN codec were described, as well as the termination scheme used in the inhomogeneous RBN codec and a vector quantisation scheme considered as an extension to the scalar quantisation. Results were presented for noiseless coding as well as for the homogeneous and inhomogeneous RBN codecs.

In chapter 4, the LIM RBN model was expanded into various adaptive forms by the introduction of scale selection (termination) and anisotropy in the form of a locally oriented interpolation. Various multiresolution approximations to the rate distortion function were formulated and the results of these functions presented for parameters derived from the GIRL image. The anisotropic LIM

RBN model was expanded to encompass colour images.

In chapter 5, the various techniques required to implement the codecs derived from the models of chapter 4 were described and the results of these codecs presented. The double angle vector representation of the orientation information was described and two techniques for estimating this information from an image were described. The first, the estimator described by Knutsson [57], produces a better estimate of the orientation, but is computationally complex. The second is a new estimator which is computationally simple, and which, although not as accurate as the first, produces an adequate estimate of the local orientation at the range of scales over which it is utilised within the codec. The use of the estimated orientation vector as a means of scale selection, based on a 'single consistent orientation' criterion was described, as well as its use in determining a set of anisotropic filters used in an adaptive anisotropic prefiltering technique. The oriented interpolation used within the scale selected blocks was described, as well a technique for performing the required coding of the boundary pixels of such blocks. Results were presented for noiseless coding, and finally the results of the most efficient codec implemented were presented for monochrome and colour images.

### 6.2. Conclusions

The results of the anisotropic, as opposed to the isotropic, codecs demonstrate the significance of preserving the local anisotropy properties of images in coding applications where the images are to be viewed by a human observer. This applies not only to the preservation of existing local feature orientations but also to ensuring that no artificial oriented local features, such as blocking artefacts, are created by the distribution of the quantisation noise as a result of

the structure used by the coding algorithm. The anisotropic LIM RBN models and codecs perform well in this respect, since they share some of the properties which have been demonstrated to be significant in vision [44]. The performance is particularly noticeable at low bit rates, where the reconstructed images take on a painted look. It is the edge-sharing oriented interpolation which imparts these properties to the codecs, by maintaining a locally oriented spatial consistency despite the presence of quantisation noise.

The RBN algorithm has been demonstrated to be a useful framework for predictive coding. It provides a fast and efficient method of addressing scale selection, allows for some non-causality within the spatial dimensions whilst retaining the efficiency of a causal representation, and it has some spatial consistency in the presence of quantisation noise, which can lead to visually acceptable results at low bit rates. In the isotropic model, the edge-sharing across scale ensures that the blocks within the reconstructed image do not suffer from pixel intensity discontinuities at their boundaries, an issue which is often the cause of significant blocking artefacts with other block coding techniques. In addition, the introduction of the oriented interpolation from these shared edges in the anisotropic model reduces discontinuities in the gradients of the pixel intensities at the block boundaries. This results from the fact that neighbouring blocks tend to have similar orientations, and the gradients along these orientations tend to be consistent.

The PSNR fidelity measure has been clearly shown to be a poor measure of image quality for applications where the reconstructed image will be viewed by a human observer. Although no alternative to the mse measure is suggested in this work, it is interesting to note that the model in some respects compensates for this by itself sharing some of the properties of the visual system which

would go into such a measure. Perhaps an orientation weighted mean squared error measure, such as that suggested in [56] might provide a partial solution to the problem. In order to be of general utility, however, a significant amount of non-trivial psychophysical measurement would be required to produce a standard for such an error measure.

### 6.3. Limitations

The adaptive LIM RBN models assume that the classes of image to be modelled consist of locally oriented features at varying scales. Although this is true for some classes of images there are other classes with features such as random textures which have no orientation, fractal features which have a constantly varying orientation, and higher order features which may have more than one orientation, for which this model is not particularly suitable. Additionally, features such as curves are represented in the model by a series of piecewise linear segments, a feature which is noticeable in the coding results at the lowest bit rates.

This work takes no account of channel coding: the codec implementations are particularly susceptible to any distortions in the coded data transmitted across the channel. This is mainly due to the arithmetic coder, which has virtually no noise immunity. Once a single bit is altered in the binary output stream of the arithmetic coder, all the subsequent coded data symbols may be lost. The basic RBN algorithm is also susceptible to channel noise, and no consideration is given in the work to the re-synchronisation of the algorithm in the case of channel distortions.

The models for which the rate distortion functions are derived lack the essential multiresolution edge sharing nature of the LIM RBN model and its derived

codecs. Since this is a significant factor in the performance of such codecs, this limits the usefulness of the rate distortion functions. The functions are also derived with respect to mean squared error as a fidelity measure rather than a more meaningful visual quality measure, which again limits their usefulness. In particular the work suggests no means by which a codec can automatically operate to a fixed rate or fixed distortion, both properties which would be related to the appropriate rate distortion work. The question of variation of the model parameters such as block splitting probabilities to minimise the distortion for a given rate, rather than fixing them, also needs more careful examination. In summary, the rate distortion work is at a very early stage and requires much more work before it can be considered complete.

### 6.4. Improvements

The models and codecs presented could be extended to encompass higher order features such as curvature and higher order interpolations such as splines. Some attempts were made to incorporate curvature into the codecs but none was considered particularly successful, the most significant problem being to estimate curvature from the image. One technique which would reduce the problem of curvature estimation to that of local orientation estimation (at which the current codecs are successful), and which would additionally retain the block boundary consistency, which is such a feature of the codecs, would be to store the orientations not for the blocks but for the block corners (or edges) and perform some form of interpolation on these corner (or edge) orientations to get a (spatially consistent) varying orientation field for the block. The problem was considered, but the solution necessary for an efficient implementation (i.e. to determine for any pixel in the interior of a block its two boundary intersection points) was not found, nor was it clear how areas of inconsistent orientation

should be handled, and the few preliminary results suggested that even if solved, the technique may lead to only minor improvements in efficiency. A more fruitful extension might be to consider the formulation of the interpolation functions in the form of two dimensional splines, still incorporating the orientation information (and possible curvature information) and retaining the block boundary consistency. However, even such an approach would not cope well with statistical textures. One possibility to cope with such features would be to incorporate a statistical texture model into the current model with some form of selection between the LIM and the statistical models. This could remove the boundary consistency and therefore make blocking artefacts visible at the estimated boundaries between regions represented by the different models.

For practical implementations, where the channel coding is not immune to noise, the codecs require improvements to both the structure of the RBN algorithm (to ensure synchronisation) and to the entropy coding (to minimise the distortion caused by channel noise). The alteration to the RBN algorithm could simply be to split the image into blocks of 64 by 64 pixels and apply the algorithm separately to each block in a raster sequence, with signalling (certain arithmetic coder output sequences can never occur and could be used to identify such signals) to enforce synchronisation between the coder and decoder at each new block. This would have an insignificant cost in terms of bit rate, since in any practical application there will be few acceptable larger blocks, but it still leaves the problem of large areas of distortion if channel distortions force the coder and decoder out of step. The problem is perhaps better faced in the entropy coding, where, if the arithmetic coder cannot be formulated with better synchronisation, a reversion to some form of self-synchronising Huffman coder may be preferable. This may correspond to limiting the number of events incorporated into each arithmetic codeword.

For many practical applications the codecs require some automatic mechanism for producing fixed rate or fixed distortion coded output. This could be based on negative feedback mechanisms adjusting the thresholds as the rate or distortion builds up across the image, or on a 'split and merge' process which codes at a high rate and then relaxes the thresholds from the lowest scale upwards until it reaches the fixed rate or distortion. However, perhaps the optimal solution could come from a correctly formulated rate distortion function for the LIM RBN model. This could provide the basis for optimal or sub-optimal coding at fixed rate or distortions based on a two pass algorithm [17]: one pass to 'assess' the image and one pass to code it using the thresholds determined in the first pass. It would obviously be useful if such a rate distortion function was based on a more realistic distortion measure than mean squared error [56].

One obvious extension is to incorporate time into the models and codecs in order to code video sequences. The basic RBN algorithm has been studied for time sequences by Cordell and Clarke [24] [25] [26]. The anisotropic LIM RBN models and codecs could simply be applied to the frame differences between successive images, or they could be applied across several successive frames. This leads to 3-d models and the estimation of both local orientation and motion. This estimation problem has been investigated by Knutsson [58].

The problem of extracting and representing more general features, such as texture, is encountered by a number of other multiresolution representations [67] [2]. In order to overcome this, more complex models, such as those of [114] [99], and the more general image representation of [14] have been suggested. The question of the complexity necessary to extract such features arises, and must be weighed against the benefit such information provides, particularly in coding applications. Another more recent development in extracting

appropriate features for coding - the so-called 'model-based' methods [32][28] - have shown some promise for restricted application areas. Nonetheless, it would appear that they have major problems, both in terms of modelling and estimation, which must be overcome before they can be considered to represent a viable alternative to 'conventional' signal-based methods such as those described in the present work.

## APPENDIX A

By definition the predictor which produces an error which is orthogonal to the data is the minimum variance predictor [81]. For the model, this orthogonality principle is satisfied if $\rho_{xypq}$ defined by equation (A1) is equal to zero for all *xypq*.

$$\rho_{xypq} = E\left[\left[s_{xy}(n) - \hat{s}_{xy}(n)\right] s_{pq}(n-1)\right] \tag{A1}$$

Expanding this for the predictor $A(n)$

$$\rho_{xypq} = E\left[\left[s_{xy}(n) - \sum_{pq} A_{xypq}(n) s_{pq}(n-1)\right] s_{pq}(n-1)\right] \tag{A2}$$

but from equation (2.3.3)

$$s_{xy}(n) - \sum_{pq} A_{xypq}(n) s_{pq}(n-1) = \sum_{rs} B_{xyrs}(n) w_{rs}(n) \tag{A3}$$

giving

$$\rho_{xypq} = E\left[\left[\sum_{rs} B_{xyrs}(n) w_{rs}(n)\right] s_{pq}(n-1)\right] \tag{A4}$$

or

$$\rho_{xypq} = \sum_{rs} B_{xyrs}(n) E\left[w_{rs}(n) s_{pq}(n-1)\right] \tag{A5}$$

Since from equation (2.3.5) $w_{rz}(n)$ is orthogonal to $s_{pq}(n-1)$, every term in equation (A5) is zero, and hence

$$E\left[\left[s_{xy}(n)-\hat{s}_{xy}(n)\right]s_{pq}(n-1)\right]=0 \qquad \text{(A6)}$$

for all $xypq$, for the predictor $A(n)$. $A(n)$ is therefore the minimum variance predictor for $s(n)$ in terms of $s(n-1)$.

**APPENDIX B**

To determine the orientation of the feature, find the value $\theta_{min}$ which minimises

$$r = \int_0^1\int_0^1 \left[ \frac{\partial s}{\partial x}\cos\theta + \frac{\partial s}{\partial y}\sin\theta \right]^2 dx\, dy \qquad \text{(B1)}$$

Differentiating with respect to $\theta$

$$\frac{dr}{d\theta} = \frac{\partial}{\partial\theta}\int_0^1\int_0^1 \left[ \frac{\partial s}{\partial x}\cos\theta + \frac{\partial s}{\partial y}\sin\theta \right]^2 dx\, dy \qquad \text{(B2)}$$

$$= \int_0^1\int_0^1 2\left[ \frac{\partial s}{\partial x}\cos\theta + \frac{\partial s}{\partial y}\sin\theta \right]\left[ \frac{\partial s}{\partial y}\cos\theta - \frac{\partial s}{\partial x}\sin\theta \right] dx\, dy \qquad \text{(B3)}$$

$$= 2\int_0^1\int_0^1 \left[ \frac{\partial s}{\partial x}\frac{\partial s}{\partial y}\left(\cos^2\theta - \sin^2\theta\right) + \left[ \left(\frac{\partial s}{\partial y}\right)^2 - \left(\frac{\partial s}{\partial x}\right)^2 \right]\cos\theta\sin\theta \right] dx\, dy \qquad \text{(B4)}$$

$$= 2\left[ \cos 2\theta \int_0^1\int_0^1 \frac{\partial s}{\partial x}\frac{\partial s}{\partial y}\, dx\, dy + \frac{1}{2}\sin 2\theta \int_0^1\int_0^1 \left[ \left(\frac{\partial s}{\partial y}\right)^2 - \left(\frac{\partial s}{\partial x}\right)^2 \right] dx\, dy \right] \qquad \text{(B5)}$$

From equation (5.3.1)

$$s(x,y) = a + \alpha x + \gamma y + \beta xy \qquad \text{(B6)}$$

$$\frac{\partial s}{\partial x} = \alpha + \beta y \qquad \text{(B7)}$$

$$\frac{\partial s}{\partial y} = \gamma + \beta x \qquad \text{(B8)}$$

Substituting these into equation (B5) gives

$$\frac{dr}{d\theta} = 2\cos 2\theta \int_0^1 \int_0^1 (\alpha + \beta y)(\gamma + \beta x)\, dx\, dy + \qquad \text{(B9)}$$

$$\sin 2\theta \int_0^1 \int_0^1 \left[ (\gamma + \beta x)^2 - (\alpha + \beta y)^2 \right] dx\, dy$$

$$= 2\cos 2\theta \left[ \alpha\gamma + \frac{1}{2}\alpha\beta + \frac{1}{2}\gamma\beta + \frac{1}{4}\beta^2 \right] + \sin 2\theta \left[ \gamma^2 - \alpha^2 + \gamma\beta - \alpha\beta \right] \qquad \text{(B10)}$$

$$= \frac{1}{2}\cos 2\theta (2\alpha + \beta)(2\gamma + \beta) + \sin 2\theta (\gamma - \alpha)(\gamma + \alpha + \beta) \qquad \text{(B11)}$$

Replacing $\alpha$, $\beta$ and $\gamma$ from equations (5.3.1) and (B6)

$$\frac{dr}{d\theta} = \frac{1}{2}\cos 2\theta (2c - 2a + a + d - c - b)(2b - 2a + a + d - c - b) +$$

$$\sin 2\theta (b - a - c + a)(b - a + c - a + a + d - c - b) \qquad \text{(B12)}$$

$$= \frac{1}{2}\cos 2\theta(c - a + d - b)(b - a + d - c) + \sin 2\theta(b - c)(d - a) \quad \text{(B13)}$$

$$= \frac{1}{2}\cos 2\theta\left[(d - a)^2 - (b - c)^2\right] + \sin 2\theta(b - c)(d - a) \quad \text{(B14)}$$

Setting the $\frac{dr}{d\theta} = 0$ to find the critical points $\theta_p$ and rearranging gives

$$\cos 2\theta_p\left[(d - a)^2 - (b - c)^2\right] = \sin 2\theta_p 2(b - c)(d - a) \quad \text{(B15)}$$

Thus the solution to this is a vector of the form

$$\begin{bmatrix} \cos 2\theta_p \\ \sin 2\theta_p \end{bmatrix} = \pm t \begin{bmatrix} 2(b - c)(d - a) \\ (b - c)^2 - (d - a)^2 \end{bmatrix} \quad \text{(B16)}$$

$$= \pm t \begin{bmatrix} 2(b - c)(d - a) \\ ((d - a) - (b - c))((b - c) + (d - a)) \end{bmatrix} \quad \text{(B17)}$$

where $t \neq 0$.

Assigning $e = b - c$ and $f = d - a$ and deriving the second differential of equation $r$ with respect to $\theta$ from equation (B14)

$$\frac{d^2r}{d\theta^2} = \frac{d}{d\theta}\left[\frac{1}{2}\cos 2\theta\left[(f)^2 - (e)^2\right] + \sin 2\theta(ef)\right] \quad \text{(B18)}$$

$$= -\sin 2\theta(f^2 - e^2) + 2\cos 2\theta(fe) \tag{B19}$$

At the critical point $\begin{bmatrix}\cos 2\theta_p \\ \sin 2\theta_p\end{bmatrix} = +t\begin{bmatrix}2ef \\ e^2 - f^2\end{bmatrix}$ the value of the second derivative is given by

$$\frac{d^2r}{d\theta^2} = t\left[\frac{\left(f^2 - e^2\right)^2}{e^2 + f^2} + \frac{2^2e^2f^2}{e^2 + f^2}\right] \tag{B20}$$

$$= \frac{t\left(e^2 + f^2\right)^2}{e^2f^2} = t\left(e^2 + f^2\right) \tag{B21}$$

Therefore this point is the minimum value with respect to $\theta$, i.e.

$$v_{2\theta_{min}} = \begin{bmatrix}2(b-c)(d-a) \\ (b-c)^2 - (d-a)^2\end{bmatrix} \tag{B22}$$

with

$$m = (d-a)^2 + (b-c)^2 \tag{B23}$$

where $m$ is the magnitude of the orientation vector.

Considering the other critical point, $\begin{bmatrix}\cos 2\theta_p \\ \sin 2\theta_p\end{bmatrix} = -t \times \begin{bmatrix}2ef \\ e^2 - f^2\end{bmatrix}$. The value of the second derivative at this point is given by

$$\frac{d^2r}{d\theta^2} = -t\left[\frac{\left[f^2 - e^2\right]^2}{e^2 + f^2}\right] \tag{B24}$$

$$= -t\left[e^2 + f^2\right] \tag{B25}$$

So this point is a maximum.

Note that the values of $2\theta_p$ for two critical points are exactly $\pi$ apart, so the values of $\theta_p$ are $\frac{\pi}{2}$ apart. This means that, not surprisingly, the maximum and minimum estimate for the orientation are orthogonal.

# An Anisotropic Multi-Resolution Image Data Compression Algorithm.

*M. Todd, R. Wilson.*

Department of Computer Science, University of Warwick
Coventry, CV4 7AL.

## ABSTRACT

This paper describes a new image data compression algorithm, based on multiresolution interpolation controlled by local feature orientation. The new coder is one of a class of pyramidal coders employing what may be seen as non-causal predictive methods. It is shown to be effective at rates below 1 bit per pixel and to be capable of producing decoded images with no gross distortions at compression ratios of up to 100. The coder is described in the context of a new class of image models - pyramid models - and its performance on a number of test images presented and compared with other methods. The paper is concluded with a discussion of the implications of this work for image modelling and data compression theory.

## Introduction

In recent years, there have been a number of developments in image data compression which seek to exploit known properties of the human visual system [1-5]. Among these properties, two have been found particularly significant : the use of multiple scales of image representation is known to occur in the visual system and has found many applications in computer vision ; the orientation selective properties of simple cells in the primary visual cortex [6] are well documented and this idea has also been exploited in a range of image processing applications including data compression [1,3]. While such methods have been shown to be effective in practice, they still lack the rigorous mathematical basis of more traditional methods, such as predictive or transform coding [7].

The aim of this paper is therefore twofold. First, a general predictive framework is presented for multiresolution image coders, based on a new class of image models which are causal not in the image plane, but in the scale dimension. This represents a generalization of the model discussed in [8] and used for image restoration. Then a new coder which falls within this general class will be described and its performance evaluated. In addition to its use of multiresolution methods, the new coder is distinguished by its application of the local orientation estimation procedure used successfully in [1]. The results achieved with this coder serve to illustrate the potential of methods derived from the new models and to suggest ways in which classical linear signal models can usefully be supplemented by models which are perhaps more closely related back to the content of natural images and to the perceptual machinery humans use to process them.

## Multiresolution Image Modelling

The linear multiresolution models which form the subject of this paper can all be described by means of the simple recursive equation (1)

$$S(n) = A(n)\,S(n-1) + B(n)\,W(n) \qquad 0 < n \le N \tag{1}$$

in which $S(n)$ and $W(n)$ are 2-d vectors

$$S(n) = \left\{ s_{x,y}(n) \right\} \qquad 0 \le x, y < M \tag{2}$$

and $A(n)$, $B(n)$ are linear operators, i.e.

$$s_{x,y}(n) = \sum_{p,q} a_{x,y,p,q}(n)\, s_{p,q}(n-1) + \sum_{p,q} b_{x,y,p,q}(n)\, w_{p,q}(n) \tag{3}$$

The components $w_{p,q}(n)$ are taken from a zero-mean normal white noise process, independent over $n$, giving

$$E\left[ w_{p,q}(n)\, w_{r,s}(m) \right] = \delta_{nm}\delta_{pr}\delta_{qs} \tag{4}$$

$$E\left[ w_{p,q}(n) \right] = 0 \quad \forall p,q,n \tag{5}$$

The initial conditions for the recursion of equation (1) are

$$A(0) = 0 \qquad S(0) = B(0)W(0) \tag{6}$$

and the image is simply the $N^{th}$ level of the process, $S(N)$. To this extent the model is identical in form to the causal models which have received much attention in the literature on coding. The major distinction between the multiresolution model and the causal ones is that for a causal model, an image of dimension $M \times M$ pixels requires $N = M^2$, or, if each line of the image is a "state vector" $N = M$, whereas a multiresolution model typically has for some $n$, $N = \log_n M$, where $n$ is the scale constant of the representation. Often the choice $n = 2$ is made on computational grounds.

*Linear Interpolative Multiresolution* (LIM) models form a subset of the general class of multiresolution models distinguished by the following conditions. For each level $s$ of the recursion there is a set of points $\Lambda_s = \{(x_{1,s}, y_{1,s}), (x_{2,s}, y_{2,s}), \cdots\}$ such that if $\Lambda$ is the set of all points in the image,

$$\Lambda = \bigcup_{0 \le x,y < M} (x,y) \tag{7}$$

$$\Lambda = \bigcup_{0 \le s \le N} \Lambda_s \tag{8}$$

$$\Lambda_s \cap \Lambda_n = \emptyset, \quad s \ne n \tag{9}$$

$$B_{xyst}(n) = 0, \quad \text{if } (x,y) \in \Lambda_s \tag{10}$$

$$A_{xyst}(n) = \delta_{xs}\delta_{yt}, \quad \text{if } (x,y) \in \bigcup_{n<s} \Lambda_n \tag{11}$$

The index-limiting operator $I(\Lambda_s)$ associated with the set $\Lambda_s$ is defined by

$$(I(\Lambda_s))_{xyst} = \begin{cases} \delta_{xs}\delta_{yt} & (x,y) \in \Lambda_s \\ 0 & \text{else} \end{cases} \tag{12}$$

Then from equation (8), it follows that the identity operator is

$$I = \sum_{n=0}^{N} I(\Lambda_n) \tag{13}$$

It follows from equation (11) that

$$I(\Lambda_n)\,S(n) = I(\Lambda_n)\,S(m) \quad m > n \tag{14}$$

Now suppose there is a linear operator $C(n,m)$, $m>n$, such that

$$C(n,m)\,S(m) = S(n) \tag{15}$$

Then from equation (1), if $m > n+1$

$$S(n+1) = A(n+1)\,C(n,m)\,S(m) + B(n+1)\,W(n+1) \tag{16}$$

or from equations (9),(14)

$$S(n+1) = (I - I(\Lambda_{n+1}))\,A(n+1)\,C(n,m)\,S(m) + I(\Lambda_{n+1})\,S(m) \tag{17}$$

so that $S(n+1)$ can be obtained from $S(m)$, $m>n$, using the operator $C(n+1,m)$ in equation (15)

$$C(n+1,m) = (I - I(\Lambda_{n+1}))\,A(n+1)\,C(n,m) + I(\Lambda_{n+1}) \tag{18}$$

But from equations (6),(14)

$$S(0) = I(\Lambda_0)\,S(n) \tag{19}$$

i.e. $\quad C(0,m) = I(\Lambda_0)$ $\tag{20}$

It follows that every level $n$ of the signal can be reconstructed exactly from any level $m>n$. In particular, each signal $S(n)$ can be obtained from the image $S(N)$ in this way. The ability to reconstruct every level from the image is one of the most useful features of the LDM models in the data compression application, for it underlies the predictive coding strategy described below. Note that each point $(x,y) \in \Lambda_k$ in the final image can be written in the form

$$S_{xy}(N) = \sum_{(\nu,\eta) \in \bigcup_{k>n} \Lambda_k} a_{xy\nu\eta} S_{\nu\eta}(N) + v_{xy}, \quad (x,y) \in \Lambda_n \tag{21}$$

where $v_{xy}$ is independent of the variables $S_{\nu\eta}(N)$

$$E\left[v_{xy} S_{\nu\eta}(N)\right] = 0, \quad (x,y) \in \Lambda_n,\ (\nu,\eta) \in \Lambda_k,\ \ k \le n \tag{22}$$

It is this linear interpolative structure which gives the model its name. The multiresolution component is expressed in the structure of the sets $\Lambda_n$. For the basic *Recursive Binary Nesting* (RBN) algorithm[5] the union of $\Lambda_k$ for $k \le n$ defines the corners of edge-sharing blocks which segment the image and whose size decreases with $n$. In the application discussed below the union of $\Lambda_k$ for $k \le n$ represents a square lattice of block boundaries, whose density increases with $n$ (Fig 1). Another noteworthy feature of the LDM model is the ease with which non-stationarities, representing *local* anisotropies of various scales, for example, may be incorporated in the model, giving the modified recursion equation

$$S_{xy}(n) = \sum_{(\nu,\eta)} A_{xy\nu\eta}(\theta_{xy}(n))\,S_{\nu\eta}(n-1) + \sum_{(\nu,\eta)} B_{xy\nu\eta}(\theta_{xy}(n))\,W(n) \tag{23}$$

where $\theta_{xy}(n)$ is in general a vector field of model parameters, such as orientation [1],[3]. The problem of estimating such parameters from the image $S(n)$ is discussed in some detail in [14].

In effect, the images $S(n)$ are *successive approximations* to the final image $S(N)$, based on linear interpolation within edge-sharing blocks of dimension $(2^{N-n}+1)$ by $(2^{N-n}+1)$. The parametric model which has proved most successful in applications is defined in terms of a vector $\theta(n)$ of dimension $2^n \times 2^n$ of block orientations: each component $\theta_{xy}(n)$ is an angle in the range $(0,\pi)$, specifying the orientation which the interpolation $A(n)$ should take within the block (Fig 2). This model therefore expresses both the multiresolution and orientation selective properties of the visual cortex.

## A Predictive Coder

The general recursive structure of these models naturally suggests the use of a predictive coder, just as it does in the 1-d case. Indeed it follows from equations (1)-(6) that the minimum m.s.e. predictor for $S(n)$ in terms of $\{S(n-k), 0 < k \le n\}$ is just (e.g. [7] ),

$$\hat{S}(n \mid n-1) = A(n)\,S(n-1) \tag{24}$$

The addition of a quantizer into the loop results in operational equations of the form

$$\hat{S}_q(n \mid n-1) = A(n)\,S_q(n-1) \quad 1 \le n \le N \tag{25}$$

$$E(n) = S(n) - \hat{S}_q(n \mid n-1) \tag{26}$$

$$S_q(n) = Q(E(n)) + \hat{S}_q(n \mid n-1) = S(n) + V_q(n) \tag{27}$$

Where $Q(\cdot)$ is a quantization function, which may be a scalar quantizer or a vector quantizer (both types have been investigated) and $V_q(n)$ is the vector of quantized noise components. The similarity of these equations to those for the standard predictive coder should be immediately apparent. Indeed there is no formal difference save that $N = \log_2 M$ in the present case. It should also be noted that only those error components corresponding to the innovations at level $n$ are transmitted, that is

$$(Q(E(n)))_{xy} = 0 \qquad \text{if} \quad \sum_{k=0}^{n-1} |B_{xy\nu\eta}(n)| = 0 \tag{28}$$

The orientation dependence of the coder can be emphasized by writing, in place of equation (25)

$$\hat{S}_q(n \mid n-1) = A(n, \hat{\theta}_q(n))\,S_q(n-1) \tag{29}$$

Where $\hat{\theta}_q(n)$ is the vector of quantized block orientations. Finally, it may be noted that the successive approximation process in the coder may be terminated at a level $n < N$ if, for example some error criterion for the $(i,j)^{th}$ block at level $n$ is satisfied. In the tests described below, a maximum error criterion was used, viz

$$d_{ij}(n) = \max_{\substack{i2^{N-n} \le x \le (i+1)2^{N-n} \\ j2^{N-n} \le y \le (j+1)2^{N-n}}} |S_{xy}(N) - S_{xy}(n)| \le d \tag{30}$$

Where $d \ge 0$ is a threshold. In other words, these coders allow a simple form of threshold coding to be applied : any block for which the error criteria is satisfied is not subdivided. The address information which this produces must of course be transmitted along with the prediction errors.

## Test Results

The test images were pre-filtered with an anisotropic filter based on the orientation estimate $\theta(N)$. A number of coders based on the above models have been tested. The simplest was a variation on the original RBN coder [5] in which the block interpolation $A(n)$ was just a bilinear interpolation from the four corner points. The prediction errors were quantized using a vector quantizer and a maximum error threshold criteria used. The results from this coder were satisfactory, but did not represent a significant improvement on the basic RBN scheme. The orientation selective coder, on the other hand has produced significantly better results on a number of 8-bit (512x512) pixel test images. The block orientations were estimated using the unbiased estimator described in [1] and the orientation quantized to one of 31 angles

$$\theta_q = i\frac{\pi}{31} \quad 0 \le i \le 30 \tag{31}$$

Arithmetic coding with a dynamic probability estimation was employed for all transmitted components and addresses [13]. For this coder, the interpolator $A(n)$ requires the luminances at all points on the block perimeter. This involves 1-d coding of the prediction errors for each block edge.

A variety of 1-d coders were tested, including a discrete cosine transform. It was found however that a 1-d version of the RBN algorithm performs as well as other methods. This was the method selected for the tests whose results are shown in Figs 3-6 and Tables 1,2. Figs 5 and 6 show the results of coding the test images "GIRL" and "BOATS" respectively at rates of 0.08bpp and 0.26bpp. Although some errors are visible at these rates, there are none of the gross distortions which are typical of most block coders. Table 1 and 2 show that acceptable results can be obtained with these images at rates below 0.5 bpp, comparing favourably with the results reported for either the traditional transform or predictive coders or the more esoteric systems described in e.g. [3].

## Conclusions

It has been shown that a common framework exists for multiresolution image processing, based on a new class of the recursive linear model of equation (1). Some properties of these models have been discused (cf [11]) and a form suited to image coding - the linear interpolative form - introduced. Parametric forms giving varying degrees of global and local non-uniformity were also presented and applied successfully to the problem of image data compression. By establishing a firmer mathematical basis for these methods, it becomes possible to exploit much of the existing mathematical apparatus of signal theory within the multiresolution framework. This will have important practical consequences for the development of effective procedures for signal estimation and analysis as well as for prediction (see also [8][9]). New applications of rate-distortion theory may also be anticipated.


## Acknowledgements

This work has been supported by BTRL Martlesham and SERC. Thanks are due both to Dr C Nightingale of BTRL and Dr H. Knutsson of Linköping University whose help has been invaluable.


## References


[1] H. Knutsson, R. Wilson, G. Granlund, "Anisotropic Nonstationary Image Estimation and its Applications: Part I-Restoration of Noisy Images, Part II-Predictive Image Coding.", IEEE Trans. COM-31, Nr 3, pp. 388-397, pp. 398-406 : March 1983.

[2] P. Burt, E. Adelson, "The Laplacian Pyramid as a Compact Image Code", IEEE Trans. COM-31, Nr 4, pp. 532-540 : April 1983.

[3] M. Kunt, A. Ikonomopoulos, M. Kocher, "Second-Generation Image-Coding Techniques" Proc. IEEE Vol 73, Nr 4, pp. 549-573 : April 1985.

[4] R. Wilson, "Quad-Tree Predictive Coding: A New Class Of Image Data Compression Algorithms", Proc. IEEE Conf. on A.S.S.P. San Diego, CA : March 1984.

[5] J. M. Beaumont, "The RIBENA Algorithm - Recursive Predictive Still Picture Coding", British Telecom Memo RT4343/88/14 : March 1988.

[6] D. H. Hubel, T. N. Wiesel, "Brain Mechanisms Of Vision", Sci. American pp. 130-144 : September 1979.

[7] A.K. Jain, "Image Data Compression: A Review", Proc. IEEE Vol 68, pp. 366-406 : March 1980.

[8] S.C. Clippingdale, R.G. Wilson, "Quad-Tree Image Estimation: A New Image Model and its Application to Minimum Mean Squared Error Image Restoration", Proc. 5th Scand. Conf. on Image Anal. pp. 699-706 : 1987.

[9] A. Calway, R. Wilson, "A Multiresolution Descriptor for Nonstationary Image Processing", Proc. Conf. IMA Mathematics in Signal Processing : December 1988.

[10] A.K. Jain, "Advances in Mathematical Models for Image Processing", Proc. IEEE pp. 502-528 : 1981.

[11] R. Wilson, S.C. Clippingdale, "A Class Of Non-Stationary Image Models and their Applications", Proc. Conf. Mathematics in Signal Processing IMA : December 1988.

[12] M. Todd R.Wilson, "Image Data Compression Combining Multiresolution, Feature Orientation and Arithmetic Entropy Coding", Proc. Conf. Mathematics in Signal Processing IMA : December 1988.

[13] J.L. Mitchell, W.B. Pennebaker, "Software Implementation Of The Q-coder", IBM Research Report, RC 12660 : April 1987.

[14] S.C. Clippingdale, "Multiresolution Image Modelling and Estimation", Univ. of Warwick, PhD Thesis : 1988.

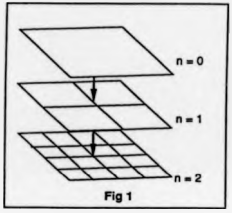



Fig 1

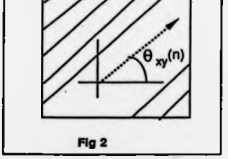


Fig 2

Fig 3 Original

Fig 4 Original

Fig 5 0.08 bpp

Fig 6 0.26 bpp

| GIRL | | |
|---|---|---|
| BPP | MSE | PSNR |
| 0.08 | 167.78 | +25.88 |
| 0.12 | 119.16 | +27.37 |
| 0.20 | 82.33 | +28.98 |
| 0.29 | 64.34 | +30.05 |

BPP - Bits Per Pixel
MSE - Mean Squared Error
PSNR - Peak Signal To Noise Ratio

Table 1

| BOATS | | |
|---|---|---|
| BPP | MSE | PSNR |
| 0.06 | 398.61 | +22.13 |
| 0.13 | 273.22 | +23.72 |
| 0.26 | 102.93 | +28.01 |
| 0.31 | 87.50 | +28.71 |
| 0.43 | 70.90 | +29.62 |

Table 2

# REFERENCES


[1] N. Abramson, *Information Theory And Coding*, McGraw-Hill Book Co. Inc. New York (1963).

[2] E. H. Adelson and E. Simoncelli, *Orthogonal Pyramid Transforms for Image Coding*, Proc SPIE Cambridge M.A. (1987).

[3] D. Anastassiou and D. J. Sakrison, *New Bounds to R(D) for Additive Sources and Applications to Image Encoding*, IEEE Trans IT-25 Nr 2 pp. 145-155 (March 1979).

[4] D. Anastassiou, W. B. Pennebaker and J. L. Mitchell, *Grey-Scale Image Coding For Freeze-Frame Videoconferencing*, IEEE Trans COM-35 Nr 4 pp.382 (April 1986).

[5] H. C. Andrews and W. K. Pratt, *Fourier Transform Coding of Images*, Proc. Int Conf. System Sciences, Hawaii, pp. 677-679 (January 1968).

[6] J. M. Beaumont, *The RIBENA Algorithm - Recursive Predictive Still Picture Coding*, British Telecom Memorandum Nr RT4343/88/14 (March 1988).

[7] T. Berger, *Rate Distortion Theory for Sources with Abstract Alphabets and Memory*, Information and Control Vol 13 pp. 254-273 (1968).

[8] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*, Englewood Cliffs, N.J., Prentice-Hall (1971).

[9] A. Bhalerao and R. Wilson, *Multiresolution Image Segmentation*, Research Report RR149 (September 1989). Department of Computer Science, Warwick University, United Kingdom.

[10] J. Binia , M. Zakai and J. Ziv, *On the ε-Entropy and the Rate Distortion Functions of Certain Non-Gaussian Processes*, IEEE Trans IT-20 Nr 4 pp. 517-524 (July 1974).

[11] C. Blakemore and F. W. Campbell, *On the Existence of Neurones in the Human Visual System Selectively Sensitive to the Orientation and Size of Retinal Images* J. Physiol. Vol 203 pp. 203-260 (1969).

[12] P. Boucher and M. Goldberg, *Color Image Compression By Adaptive Vector Quantization*, IEEE CH1945-5/84/0000-0315

[13] P. J. Burt, E. H. Adelson, *The Laplacian Pyramid as a Compact Image Code*, IEEE Trans COM-31 Nr 4 pp.532-540 (April 1983).

[14] A. Calway, *The Multiresolution Fourier Transform: A General Purpose Tool for Image Analysis*, PhD Thesis, University of Warwick (September 1989).

[15] J. C. Candy and R. H. Bosworth, *Methods for Designing Differential Quantisers Based on Subjective Evaluations of Edge Busyness*, Bell Sys Tech Journal Vol 51 Nr 7 pp. 1495-1516 (1972).

[16] S. Carlsson, *Sketch Based Coding of Grey Level Images*, Signal Processing Vol 15 pp. 57-83 (1988).

[17] W. H. Chen and W. K. Pratt, *Scene Adaptive Coder*, IEEE Trans. COM-32 pp.225-232 (March 1984).

[18] W. H. Chen and C. Harrison-Smith, *Adaptive Coding of Monochrome and Color Images*, IEEE Trans COM-25 Nr 11 pp.1004-1012 (November 1977).

[19] R. J. Clarke, *Transform Coding of Images*, Academic Press (1985).

[20] S. C. Clippingdale and R. G. Wilson, *Quad-Tree Image Estimation:A New Image Model and its Application to MMSE Image Restoration*, Proc. 5th Scandinavian Conf. Image Analysis, Stockholm, pp. 699-706 (1987).

[21] S. C. Clippingdale, *Multiresolution Image Modelling and Estimation*, PhD Thesis, University of Warwick (September 1988).

[22] S. C. Clippingdale and R. G. Wilson, *Least-Squares Image Estimation on a Multiresolution Pyramid*, Proc. ICASSP 89, Glasgow pp. 1409-1412 (May 1989).

[23] R. A. Cohen and J. W. Woods, *Sliding Entropy Coding of Images*, Proc ICASSP 89, Glasgow pp. 1731-1734 (May 1989).

[24] P. J. Cordell and R. J. Clarke, *Recursive Binary Nesting for Moving Pictures*, Proc. Int. Picture Coding Symposium, Turin, nr. 13.11 (September 1988).

[25] P. J. Cordell and R. J. Clarke, *Moving Image Sequence Coding using Recursive Binary Nesting*, Melecon 89, Lisbon (April 1989).

[26] P. J. Cordell and R. J. Clarke, *An Interpolative Spatial Domain Technique for Coding Image Sequences*, Proc. ICASSP 89, Glasgow pp. 1917-1920 (May 1989).

[27] T. M. Cover, *Enumerative Source Coding*, IEEE Trans. IT-19 pp.73-77 (January 1973).

[28] N. D. Duffy and J. F. S. Yau, *A Knowledge Based Approach to Image Coding*, Proc EUSIPCO-88 pp.1125-1128 (September 1988).

[29] M. Eden, M. Unser and R. Leonardi, *Polynomial Representation of Pictures*, Signal Processing Vol 10 pp. 385-393 (1986).

[30] P. Elias, *Predictive Coding - Part I and Part II*, IRE Trans IT-1 pp. 16-33 (March 1953).

[31] Y. Feng and N. Nasrabadi, *A Dynamic Address-Vector Quantisation Algorithm Bases on Inter-Block and Inter-Colour Correlation for Color Image Coding*, Proc ICASSP 89, Glasgow, pp. 1755-1758 (May 1989).

[32] R. Forchheimer and O. Fahlander, *Low Bit Rate Coding through Animation*, PCS-83 (1983).

[33] A. Gersho, *Asymptotically Optimal Block Quantisation*, IEEE Trans. IT-25 pp. 373-380 (July 1979).

[34] A. Gersho, *On the Structure of Vector Quantizers*, IEEE Trans. IT-28 pp. 157-166 (March 1982).

[35] R. M. Gray, *Vector Quantisation*, IEEE ASSP Magazine (April 1984)

[36] C. A. Gonzales, W. B. Pennebaker and J. L. Mitchell, *Adaptive Still-Frame Graylevel Compression System using an Arithmetic Coder*, IBM Computer Science RC 12690 (#57096) (April 1987).

[37] S. K. Goyal and J. B. O'Neal Jr, *Entropy Coded Differential Pulse-Code Modulation Systems for Television*, IEEE Trans COM-23 pp. 660-665 (June 1975).

[38] R. N. Haber and M. Hershenson, *The Psychology of Visual Perception*, Holt, Rinehart, Winston, NY (1980).

[39] A. Habibi, *Hybrid Coding of Pictorial Data*, IEEE Trans COM-22 Nr. 5 pp. 614-624 (1974).

[40] A. Habibi and R. S. Hershal, *A Unified Representation of Differential Pulse-Code Modulation (DPCM) and Transform Coding Systems*, IEEE Trans COM-22 Nr. 5 pp. 692-696 (1974).

[41] Ernest L. Hall, *Computer Image Processing and Recognition*, Academic Press, (1979).

[42] H. M. Hang and B. G. Haskell, *Interpolative Vector Quantisation of Color Images*, IEEE Trans. COM-36 Nr.4 pp.465-470 (April 1988).

[43] Y. S. Ho and A. Gersho, *Classified Transform Coding of Images Using Vector Quantisation*, Proc ICASSP 89 Glasgow (May 1989).

[44] D. H. Hubel and T. N. Wiesel, *Brain Mechanisms Of Vision*, Sci. American pp.130-144 (September 1979).

[45] D. H. Hubel and T. N. Wiesel, *Receptive Fields of Single Neurones in the Cat's Striate Cortex*, J. Physiol. Vol 148 pp. 574-591 (1959).

[46] D. H. Hubel and T. N. Wiesel, *Receptive Fields, Binocular Interaction and Functional Architecture in the Cats's Visual Cortex*, J. Physiol. Vol 160 pp. 106-154 (1962).

[47] D. A. Huffman, *A Method for the Construction of Minimum-Redundancy Codes*, Proc. IRE Vol 40 pp. 1098-1101 (September 1952). Key Papers in The Development of Information Theory IEEE Press (Re-Print) (1974).

[48] G. M. Hunter and K. Steiglitz, *Operations on Images using Quad Trees*, IEEE Trans PAMI-1 Nr 2 pp. 145-153 (April 1979).

[49] A. Ikonomopoulos and M. Kunt, *High Compression Image Coding Via Directional Filtering*, Signal Processing Vol 8 pp. 179-203 (1985).

[50] A. K. Jain, *Image Data Compression:A Review*, Proc. IEEE Vol 69 Nr 3 (March 1981).

[51] A. K. Jain, *Advances in Mathematical Models for Image Processing*, Proc. IEEE Vol 69 pp. 502-528 (1981).

[52] F. Jelinek, *Probabilistic Information Theory*, McGraw-Hill Book Co. Inc. New York (1968).

[53] C. S. Kim, M. J. T. Smith and R. M. Mersereau, *An Improved SBC/VQ Scheme for Color Image Coding*, Proc ICASPP 89, Glasgow, pp. 1941-1944 (May 1989).

[54] N. M. Nasrabadi and R. A. King, *Image Coding using vector quantisation*, IEEE Trans COM-36 Vol 8 pp. 957-971 (1988).

[55] N. M. Nasrabadi, Shikuan E. Lin and Yushu Feng, *Interframe Hierarchical Vector Quantisation*, Proc ICASSP 89 Glasgow (May 1989).

[56] H. E. Knutsson, R. Wilson and G. H. Granlund, *Anisotropic Nonstationary Image Estimation and its Applications: Part I-Restoration of Noisy Images*, IEEE Trans. COM-31 Nr.3 pp.388-397 (March 1983).

[57] H. E. Knutsson, *Filtering And Reconstruction In Image Processing*, PhD Thesis, Linkoping University, Sweden (1982).

[58] H. Knutsson, *A Tensor Representation of 3-d Structure*, IEEE ASSP Workshop on Multidimensional Signal Processing, Noordwijkerout, Holland (1987).

[59] M. Kocher and R. Leonardi, *Adaptive Region Growing Technique using Polynomial Functions for Image Approximation*, Signal Processing Vol 11 pp. 47-60 (1986).

[60] T. Kronander, *Motion Compensated 3-Dimensional Wave-Form Image Coding*, Proc ICASSP 89 Glasgow (May 1989).

[61] M. Kunt, A. Ikonomopoulos and M. Kocher, *Second-Generation Image-Coding Techniques*, Proc. IEEE Vol 73 Nr 4 (April 1985).

[62] G. G. Langdon, J. J. Rissanen, *A Simple General Binary Source Code*, IEEE Trans IT-28 Nr 5 pp.800-803 (September 1982).

[63] G. G. Langdon, *An Introduction To Arithmetic Coding*, IBM J.Res.Develop. 28 pp.135-149 (March 1984).

[64] G. G. Langdon and J. J. Rissanen, *Compression of Black-White Images With Arithmetic Coding*, IEEE Trans. COM-29 pp.858-867 (June 1981).

[65] A. Lempel, J. Ziv, *Compression of Two-Dimensional Data*, IEEE Trans IT-32 Nr 1 pp.2-8 (January 1986).

[66] K. Lundgren, D. Antonsson and G. H. Granlund, *GOP, A Different Architecture for Fast and Flexible Image Processing*, Proc. Nat. Comput. Conf., Chicago, IL, 1981.

[67] S. G. Mallat, *A Theory For Multiresolution Signal Decomposition: The Wavelet Representation*, IEEE Trans. PAMI-11 Nr 8 pp. 674-693 (July 1989).

[68] J. L. Mannos and D. J. Sakrison, *The Effects of a Visual Fidelity Criterion on the Encoding of Images*, IEEE Trans IT-20 Nr 4 pp. 525-536 (July 1974).

[69] D. Marr, *Vision*, Freeman, San Francisco, CA (1982).

[70] J. Max, *Quantizing for Minimum Distortion*, IRE Trans. IT-6 pp.7 (1960).

[71] R. W. McColl and G. R. Martin, *Quantisation of Digitised Colour Images*, Research Report RR112, Department Computer Science, Warwick University, United Kingdom (November 1987).

[72] P. Meer, E. S. Baugher and A. Rosenfeld, *Frequency Domain Analysis And Synthesis Of Image Pyramid Generating Kernels*, IEEE Trans. PAMI-9 Nr 4 pp.512-522 (July 1987).

[73] J. L. Mitchell, W. B. Pennebaker, *Software Implementation Of The Q-coder*, IBM Research Report,RC 12660 (April 1987).

[74] J. L. Mitchell, W. B. Pennebaker, *Probability Estimation For the Q-coder*, IBM Research Report RC 12659 (April 1987).

[75] J. L. Mitchell and W. B. Pennebaker, *Optimal Hardware and Software Arithmetic Coding Procedures for the Q-Coder*, IBM Communications RC 12658 (#57028) (April 1987).

[76] H. Mostafavi and D. J. Sakrison, *Structure and Properties of a Single Channel in the Human Visual System*, Vision Res Vol 16 pp 957-968 (1976).

[77] A. N. Netravali and J. O. Limb, *Picture Coding:A Review*, Proc IEEE Vol 68 pp.366-406 (March 1980).

[78] J. B. O'Neal Jr and T. R. Natarajan, *Coding Isotropic Images*, IEEE Trans IT-23 Nr 6 pp. 697-707 (November 1977).

[79] J. B. O'Neal Jr, *Differential Pulse-Code Modulation (PCM) with Entropy Coding*, IEEE Trans IT-22 Nr 2 pp. 169-174 (March 1976).

[80] B. N. Oliver, *Efficient Coding*, Bell Syst Tech J. Vol 31 pp. 724-750 (July 1952)

[81] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Series in Electrical Engineering (International Student Edition) (1984)

[82] G. Peano, *Selected Works of Giuseppe Peano*, Translated and Edited by Hubert C. Kennedy, Allen and Unwin Ltd, London (1972).

[83] M. C. Pease, *Methods of Matrix Algebra*, Academic Press (1965).

[84] W. K. Pratt, J. Kane and H. C. Andrews, *Hadamard Transform Image Coding*, Proc. IEEE Vol 57 Nr 1 pp.58-68 (January 1969).

[85] W. K. Pratt, *Digital Image Processing*, Wiley, New York, (1978).

[86] B. Ramamurthi and A. Gersho, *Classified Vector Quantization Of Images*, IEEE Trans. COM-34 pp.1105-1115 (November 1986).

[87] J. J. Rissanen, *Generalised Kraft Inequality And Arithmetic Coding*, IBM J.Res.Dev. vol 20 pp.198-203 (May 1976).

[88] J. J. Rissanen, G. G. Langdon, *Universal Modeling And Coding*, IEEE Trans IT-27 Nr 1 pp.12-23 (January 1981).

[89] J. J. Rissanen and G. G. Langdon, *Arithmetic Coding*, IBM J. Res. Dev. vol 23 pp.149-162 (1979)

[90] J. A. Roese and W. K. Pratt, *Interframe Cosine Transform Image Coding*, IEEE Trans. COM-25 pp.1329-1338 (November 1977).

[91] D. J. Sakrison, *On The Role of the Observer and a Distortion Measure in Image Transmission*, IEEE Trans COM-25 Nr 11 pp. 1251-1267 (November 1977).

[92] D. J. Sakrison, *The Rate Distortion Function For A Class Of Sources*, Information and Control Vol 15 pp. 165-195 (1969).

[93] D. J. Sakrison, *The Rate of a Class of Random Processes*, IEEE Trans IT-16 Nr 1 pp. 10-16 (January 1970).

[94] J. L. Salinas and R. L. Baker, *Laplacian Pyramid Encoding: Optimum Rate and Distortion Allocations*, Proc ICASSP 89 Glasgow (May 1989).

[95] A. Sandman and B. Sapir, *Third Order Polynomial - Its use in Data Compression*, Signal Processing Vol 15 pp. 405-418 (1988).

[96] C. E. Shannon, *A Mathematical Theory of Communication*, Bell Syst Tech J Vol 27 pp. 379-423 Part 1,2 (July 1948). Bell Syst Tech J Vol 27 pp. 623-656 Part 3,4,5 (October 1948). Key Papers in The Development of Information Theory IEEE Press (Re-Print) (1974).

[97] C. E. Shannon, *Coding Theorems for a Discrete Source with a Fidelity Criterion*, IRE Nat. Conv. Rec. pt 4, pp. 142-163 (March 1959) Key Papers in The Development of Information Theory IEEE Press (Re-Print) (1974).

[98] M. J. T. Smith and T. P. Barnwell, *A Procedure for Exact Reconstruction Filter Banks For Tree-Structured Subband Coders*, CH1945-5/84/0000-0285 1984 IEEE.

[99] M. Spann, *Texture Description and Segmentation in Image Processing*, PhD Thesis, University of Aston (September 1985).

[100] T. G. Stockham, *Image Processing in the Context of a Visual Model*, Proc. IEEE Vol 60 Nr 7 pp. 828-842 (July 1972).

[101] S. L. Tanimoto and T. Pavlidis, *A Hierarchical Data Structure for Picture Processing*, Comp. Graph. Im. Proc. Vol 4 pp. 104-119 (1975).

[102] M. Tasto and P. A. Wintz, *A Bound on the Rate-Distortion Function and Application to Images*, IEEE Trans IT-18 Nr 1 pp. 150-159 (January 1972).

[103] M. Tasto and P. A. Wintz, *Image Coding by Adaptive Block Quantisation*, IEEE Trans COM-19 Nr 6 pp. 957-972 (December 1971).

[104] M. Todd and R. Wilson, *Image Data Compression*, Research Report RR128, Department of Computer Science, Warwick University, United Kingdom (October 1987).

[105] M. Todd and R. Wilson, *Image Data Compression Combining Multiresolution, Feature Orientation and Arithmetic Entropy Coding*, Proc. IMA Conf. Math. in Signal Processing, Warwick (December 1988).

[106] M. Todd and R. Wilson, *An Anisotropic Multiresolution Image Data Compression Algorithm*, Proc ICASSP 89 Glasgow (May 1989).

[107] D. Tricker et al, *Real Time Decoding of Hybrid Recursive Binary Nesting*, Proc. Int. Picture Coding Symposium , Stockholm, p. 52 (June 1987).

[108] L. Wang and M. Goldberg, *Pyramid Transform Coding using Vector Quantization*, IEEE Conf ICASSP 88 (M2.19) pp. 812-815 (1988).

[109] P. H. Westerink, D. E. Boekee, J. Biemond and J. W. Woods, *Subband Coding of Images using Vector Quantization*, IEEE Trans COM-36 Nr 6 pp. 713-719 (June 1988).

[110] R. Wilson, *Quad-Tree Predictive Coding*, Proc. ICASSP 84, San Diego 29.3 (March 1984).

[111] R. Wilson, *Quad-Tree Predictive Coding: A New Class of Image Data Compression Algorithms*, Internal Report (LITH-ISY-I-0609), Dept. of Elec. Eng., Linköping University, Linköping, Sweden, (June 1983).

[112] R. Wilson, H. E. Knutsson and G. H. Granlund, *Anisotropic Nonstationary Image Estimation and its Applications: Part II-Predictive Image Coding*, IEEE Trans. COM-31 Nr.3 pp.398-406 (March 1983).

[113] R. Wilson and G. H. Granlund, *The Uncertainty Principle in Image Processing*, IEEE Trans PAMI-6 Nr 6 pp. 758-766 (November 1984).

[114] R. Wilson and M. Spann, *Finite Prolate Spheroidal Sequences and their Applications: Part II: Image Feature Description and Segmentation*, IEEE Trans PAMI-10 pp. 193-203 (1988).

[115] P. A. Wintz, *Transform Picture Coding*, Proc. IEEE Vol 60 Nr 7 pp.809-820 (1972).

[116] I. H. Witten, R. M. Neal and J. G. Cleary, *Arithmetic Coding For Data Compression*, Comms. of the ACM Vol 30 Nr 6 pp.520 (June 1987).

[117] I. H. Witten and R. M. Neal, *Using Peano Curves for Bilevel Display of Continuous-Tone Images*, IEEE CG&A pp. 47-52 (May 1982).

[118] J. W. Woods and S. D. O'Neil, *Subband Coding of Images*, IEEE ASSP-34 pp. 1278-1288 (October 1986).

[119] J. K. Yan and D. J. Sakrison, *Encoding of Images Based on a Two-Component Source Model*, IEEE Trans COM-25 Nr 11 pp. 1315-1322 (November 1977).

[120] J. Ziv, *Coding of Sources with Unknown Statistics, Part 1 : Probability of Encoding Error, Part 2 : Distortion Relative to a Fidelity Criterion*, IEEE Trans IT-18 Nr 3 pp. 384-389 , pp. 389-394 (May 1972).

TITLE Image Data Compression Based On a Multiresolution Signal Model

AUTHOR *Martin Peter Todd*

INSTITUTION and DATE The University of Warwick 1989