LOW BIT-RATE IMAGE SEQUENCE CODING

Christopher Cubiss

A thesis submitted for the degree of Doctor of Philosophy.

Department of Electrical Engineering
The University of Edinburgh.
1994

Declaration of originality

This thesis and the work herein was originated and composed entirely by the author.

The work described in chapter 3 was carried out as a joint project with John A. Elliott, formerly a fellow member of the Signal Processing Group, Department of Electrical Engineering, The University of Edinburgh. All results quoted in chapter 3 for **occam** simulations are entirely due to John Elliott whereas the results using parallel 'C' were produced by myself entirely.

Christopher Cubiss

Acknowledgements

I would like to thank the following people for their invaluable assistance during the course of PhD:

- Peter Grant and Edward McDonnell, my supervisors, for their continous support and guidance. Also for reading and checking this manuscript.
- John Elliott and the other members of Signal Processing Group for their support at the start of my PhD.
- Northern Telecom (NI) Ltd and the SERC for providing financial support.

Contents

List of Figures

List of Tables

Glossary

Blocking	Coding artifact caused by quantization of transform coefficients.
Codec	Coder decoder.
Load Balancing	A technique for distributing the computational load evenly amongst the processors of a parallel computer.
Pel	Picture element.
Pixel	Picture element.
Portable	The traditional definition of *portability* is that the program should be independent of the chosen computing platform. For parallel computing this definition must be extended to include independence of processor topology.
Scalable	A parallel program is described to be scalable if when extra processors are added the execution time of the program is proportionally reduced.

List of Symbols

A	Number of pixels in root block.
B_f	Number of bits required for failed data.
B_s	Number of bits for state data.
B_t	Number of bits required for trees.
B_v	Number of bits required for vector data.
$d_{\max}$	Maximum displacement.
$d_x(x, y, t)$	x-component of a displacement function.
$d_y(x, y, t)$	y-component of a displacement function.
$\hat{d}_x(x, y, t)$	x-component of estimated displacement function.
$\hat{d}_y(x, y, t)$	y-component of estimated displacement function.
$\Delta d_x(x, y, t)$	x-component of estimated displacement error.
$\Delta d_y(x, y, t)$	y-component of estimated displacement error.
$D(\phi)$	Parametric distortion function.
$e(x, y, t)$	Discrete prediction error function.
$\mathrm{E}[\cdot]$	Expectation operator.
$f(x, y, t)$	Continous image intensity function.
$F(\omega_x, \omega_y, \omega_t)$	Fourier transform of continous image intensity function.
$\mathcal{E}$	Mean absolute error or mean square error.
$H(X)$	Entropy of X.
$i(x, y, t)$	Discrete image intensity function.

$I(\cdot)$	Self-information of occurence of $\cdot$.
$I(\omega_x, \omega_y, \omega_t)$	Fourier transform of discrete image intensity function.
$l(x, y, t)$	Sampling lattice.
$L(\omega_x, \omega_y, \omega_t)$	Fourier transform of sampling lattice.
N_i	Number of leaves at depth i in a tree.
N_i^f	Number of failed leaves at depth i in a tree.
N_t	Number of root blocks in regular decomposition of an image.
$p(\cdot)$	Probability of $\cdot$ occuring.
$P(\cdot)$	Uniformity predicate.
r_v	vector rate.
r_d	data rate.
$R(D)$	Rate-distortion function.
$R(\phi)$	Parametric rate function.
$\mathbf{R}_{xx}$	Covariance matrix of x.
$S_{ee}(\omega_x, \omega_y, \omega_t)$	Power spectrum of prediction error.
$S_{ii}(\omega_x, \omega_y, \omega_t)$	Power spectrum of image.
ω	Angular frequency.
ω_x	Angular frequency of x-component of a signal.
ω_y	Angular frequency of y-component of a signal.
ω_t	Angular frequency of time component of a signal.
λ_i	Eigenvalues of covariance matrix.
μ	Mean value of a signal.
ρ	Correlation coefficient.
σ^2	Variance.
$\mathbb{R}$	The set of real numbers.
$\mathbb{Z}$	The set of integers.
$\in$	Is a member of the set.

List of Abbreviations

AR	Autoregressive.
BMA	Block matching algorithm.
BTC	Block truncation coding.
CCITT	Comité Consultatif Internationale de Télégraphique et Téléphonique.
CD	Compact disc.
CIF	Common intermediate format.
CPU	Central processing unit.
CSP	Communicating sequential processes.
DAT	Digital audio cassette.
DCT	Discrete cosine transform.
DFD	Displaced frame difference.
DPCM	Differential pulse coded modulation.
DSP	Digital signal processor or digital signal processing.
ECS	Edinburgh concurrent supercomputer.
GOB	Group of blocks.
HDTV	High definition television.
HVS	Human visual system.
I/O	Input-output.
ISDN	Integrated services digital network.
JPEG	Joint photographic experts group.

KLT	Karhunen-Loève transform.
MB	Macroblock.
MC	Motion compensation.
MCI	Motion-compensated interpolation.
MCP	Motion-compensated prediction.
MIMD	Multiple instruction multiple data.
MISD	Multiple instruction single data.
MOP	Million operations per second.
MPEG	Moving picture expert group.
MSE	mean square error.
PCM	Pulse coded modulation.
PE	Processing element.
PSNR	Peak signal-to-noise ratio.
PSTN	Public switched telephone network.
pdf	Probability density function.
QCIF	Quarter common intermediate format.
RAM	Random access memory.
RGB	Red-green-blue colour coordinate system.
RISC	Reduced instruction set computer.
SIMD	Single instruction multiple data.
SISD	Single instruction single data.
SFM	Spectral flatness measure.
SNR	Signal-to-noise ratio.
VBCR	Variable block size conditional replenishment.
VDU	Visual display unit.
VLC	Variable length code.
VQ	Vector quantisation.
VSBMA	Variable size block matching algorithm.
YC_rC_b	Luminance and colour difference colour coordinate system.

Chapter 1

Introduction

Digital images are increasingly being employed in a diverse range of applications for the storage and transmission of picture information. The sheer bulk of data contained within an digital image has prompted the development of compression algorithms to reduce the storage and transmission requirements of these images. Currently, image compression is recognised as an *enabling technology* since it is crucial to the development of many applications. Typical applications which employ image compression include: facsimile transmission (fax), medical diagnosis, videoconferencing and multimedia.

Over the past three decades much research has been dedicated to developing image compression algorithms [1–5] for the storage and transmission of image data. A diverse range of algorithms have been generated which range from simple algorithms based upon linear prediction [1](chapter 6) to complex algorithms which employ wireframe models of the head and shoulders [6]. This diversity is due to researchers constantly seeking to increase the compression ratio whilst still maintaining the image integrity.

To enable the exchange of compressed picture information several standards have been developed. The CCITT H.261 [7] and the motion picture experts group (MPEG) standards [8] both target full motion video. They are based upon a hybrid coding architecture which combines motion-compensated prediction with transform coding. Although motion-compensated transform codecs have been found to produce reasonable quality reconstructed images, the quality of the reconstructed image rapidly degrades after a critical value of compression ratio is reached [9]. In general, it is difficult to give an absolute value for this critical value of compression ratio since the tolerable distortion threshold varies from person to person and indeed may vary according to application.

However, for small head and shoulder type images (typically 176 × 144 pixels) which are used for videotelephony applications, 64 kbits/s appears to be the lower limit of the rate at which images can be encoded whilst maintaining *acceptable* quality. Algorithms which can encode images at rates below 64 kbits/s are currently of great interest since they will allow video to be transmitted on conventional PSTN telephone lines and on mobile communication channels.

The degradation in the integrity of the reconstructed images generated from a motion-compensated transform architecture can be attributed to two major factors:

1. The transform encoder is optimal for encoding real-world images not prediction errors.
2. Both the transform encoder and the motion-estimation algorithm are block-based algorithms and as such do not describe the semantic detail of a image. As the compression ratio is progressively increased the regular block structure becomes apparent and eventually dominates the structure of the image.

Thus to further increase the compression ratio whilst still maintaining the integrity of the decoded image, the spatial encoder must be optimised for compressing prediction errors and the semantic detail of the image must be also be encoded. Strobach [9] has recently introduced such an encoder which is based upon a quadtree segmentation of a prediction error. However, this scheme fails to reach its full potential since it still employs a block-based motion-estimation algorithm.

1.1 Synopsis of thesis

The research recorded in this thesis investigates methods for efficiently encoding image sequences. The objective of the research is to characterise the prediction error signal and then, based upon these results, to develop a new algorithm for encoding the prediction error.

Chapter 2 introduces image compression. It includes a description of the theoretical basis for image compression, a description of some typical algorithms and a discussion of the various standards for the compression of greyscale and colour images.

To further increase compression ratios increasingly complex and sophisticated compression algorithms will be required. Both H.261 and MPEG already require considerable computational power to achieve real-time encoding/decoding. As compression algorithms become more complex, progressively more sophisticated dedicated hardware implementations will be required. Chapter 3 addresses this problem by investigating parallel implementations of image compression algorithms on a reconfigurable, multiple instruction multiple data computer. The aim of the work is twofold: firstly, to produce a real-time simulation tool which allows video compression algorithms to be tested and optimised, and secondly, to investigate the inherent parallelism in image coding algorithms to determine scalable and portable solutions for mapping this class of algorithm onto a parallel architecture.

Motion estimation for image coding is examined in chapter 4. Motion-compensated prediction, which is used in many state-of-the-art video codecs, is discussed in some detail. Theoretical models of the prediction error are developed and compared to experimental results. It is demonstrated that motion-compensated prediction can be considered to be non-ideal spatial high-pass filtering of the original image.

Based upon the analysis of the optimum hybrid motion-compensated prediction coder by Girod [10] it is demonstrated that, for small distortions, the signal-to-noise coding gain of this encoder is asymptotically bounded by the inverse spectral flatness measure of the prediction error. Theoretical and empirical results show that the spectral flatness measure of typical prediction errors approach unity, and thus, only small gains can be expected from applying a spatial encoding algorithm. Finally, some serious flaws in the motion-compensated transform encoder are highlighted.

Based upon the results of chapter 4, a new algorithm is developed for the compression of an image sequence in chapter 5. The new algorithm employs a variable size block matching motion estimation algorithm to segment images so that they can be encoded using conditional replenishment. The resulting segmentation partitions the image into regions which can be copied between frames and regions which must be replenished. The complex block structure is compactly described using a tree and the regular nature of the decomposition allows simple inequalities to be formulated so that compression is always ensured. A novel method of controlling rate and distortion is also introduced which is based upon adaptively parsing the tree structure so that leaf nodes can be merged. Finally, to reduce the computational overhead of the variable block size motion estimation algorithm, a stationary background segmentation algorithm is employed.

Chapter 6 summarises the results presented in chapters 3, 4 and 5. Conclusions of the work are discussed with suggestions for extensions to this work.

The results presented in this thesis are based upon applying encoding algorithms to three different image sequences: Miss America, Clare, and Salesman. The Miss America and Clare sequences both contain images with a plain background, which contain very little frame to frame motion. These two sequences are almost ‘ideal’ for videotelephony. The Salesman sequence still contains head and shoulder type images but the images contain a complex background and the frame to frame motion is quite complex. In comparison to the Miss America and Clare sequences the Salesman sequence is, generally, more difficult to encode.

Appendix A contains a list of the original publications which are associated with this thesis. Appendix B contains a proof of the fact that the Karhunen-Loève transform approaches the rate-distortion bound for Gaussian sources. Some of the intermediate results from this appendix are also used in chapter 4 to derive a figure of merit for the signal-to-noise gain of the discrete motion-compensated transform coder. Appendix C contains a description of the software contained on the disk attached to the back cover of this thesis.

Chapter 2

Image coding

2.1 Introduction

Compression algorithms can be broadly divided into two classes: information-lossless or information-lossy. The former is able to reconstruct an exact copy of the original image whereas the latter produces a reconstructed image which is an approximation to the original. Information-lossless coding algorithms are used in applications where the quality of the reconstructed image is critical, *e.g.* medical images. Information-lossy algorithms give the greatest compression and are used in applications in which the storage or transmission bandwidth is limited and a *good* quality reproduction is sufficient, *e.g.* videotelephony.

Compression is achieved by exploiting naturally occurring redundancy in an image or image sequence. In general, three different types of redundancy can be identified and exploited: pyschovisual redundancy which is based upon limitations of the human visual system; coding redundancy which is caused by inefficient assignment of code words; and spatial and temporal redundancy which allows one part of an image to be predicted from other parts of the image.

This chapter reviews image compression for greyscale and colour images. The different types of redundancy which can be exploited to obtain compression are described. Information theory and its derivative rate-distortion theory are introduced. Popular algorithms for the compression of images are also discussed and then the JPEG, H.261 and MPEG standards are described.

2.2 Psychovisual redundancy

The human visual system (HVS), which comprises of the eye, optic nerve and finally the brain for interpreting images, is exceptionally powerful but nevertheless, many images contain detail which the HVS is simply not capable of registering. This represents a rich source of redundancy which if correctly exploited results in significant compression without introducing noticeable degradation in the image fidelity.

Clearly, to control the distortion which is introduced into an image a model is required which faithfully mimics the HVS. Unfortunately, the complexity of the human visual system precludes the development of such a model, and since the perceived quality of an image is extremely subjective it is doubtful that such models could ever be developed. Instead, images are quantised based upon empirical rules developed from simple experiments to measure the response of the HVS to various stimuli.

2.2.1 Brightness sensitivity

The human visual system (HVS) is able to adapt to an enormous range of intensities; although at any one time it is only able to simultaneously adapt to a small subset of its full range.

Experiments to measure the response of the HVS to intensity perturbations consider two cases: the response of the HVS to intensity perturbations presented against a uniform background and the response of HVS to intensity perturbations presented against a non-uniform background. The response of the HVS to the former is well known and is given by the Weber law [11], that is, the just noticeable intensity perturbation, ΔI, is given by,

$$\Delta I = kI \tag{2.1}$$

where I is the intensity of the uniform background and k is a constant which is experimentally measured to be approximately 2% of the background intensity.

If the perturbation is presented against a non-uniform background then the response of the human visual system is found to be considerably modified. No comprehensive model exists to explain the response of the eye to non-uniform stimuli. But, in general, a complex background masks luminance perturbations, although in some cases the perturbation becomes more noticeable [6]. If a luminance perturbation is presented close to an image edge then it is known that the just-noticeable threshold increases either side of the edge [11]. Girod [12] has shown that this masking effect is greater on the dark side of the edge.

For images displayed on VDUs the above phenomena are considerably modified by the non-linear response [11, 12] of the cathode ray tube and light falling upon the screen.

2.2.2 Spatial response

The response of the HVS to spatial phenomena has been measured using spatial sine-wave patterns. These experiments show that the eye has an approximately bandpass response to spatial frequency [12]. The degradation in spatial response at high frequencies is far greater than that at low frequencies and as such is the most important for image coding.

2.2.3 Colour sensitivity

The HVS is well known to be more sensitive to changes of intensity than changes in colour. Indeed, greater compression ratios can be achieved in colour images since the colour information masks coding artifacts [2].

Many encoding schemes exploit this by using a tri-stimulus colour coordinate system which uses luminance and two colour difference components (YC_rC_b) instead of the red-green-blue (RGB) colour coordinate system which is frequently used for computer graphics. The advantage of the YC_rC_b colour coordinate system is that luminance (Y) component is the greyscale image and the two colour difference signals (C_rC_b) are relatively uncorrelated compared to their RGB counterparts. In many applications the colour difference signals are subsampled by a factor of two without introducing a noticeable degradation in image fidelity.

2.2.4 Temporal response

The visibility of spatial detail falls as the velocity of an object increases, which results in short term temporal masking. The duration of this masking effect is generally so small that it is difficult to exploit in practical coding schemes [12].

2.3 Information theory

Information theory [13] can be used to quantify the redundancy contained within a waveform and to derive the lower bound to the rate at which a signal can be encoded. The fundamental concept of information theory is uncertainty, that is, the more uncertain an event is the more information it contains.

The self information, $I(x_i)$, of a discrete-time, discrete-amplitude source $X \in \{x_0, x_1, x_2 \ldots x_{N-1}\}$ in which a symbol $\{X = x_i\}$ has a probability $p(x_i)$ of occurring is defined to be,

$$I(x_i) = \log\left(\frac{1}{p(x_i)}\right) = -\log p(x_i) \tag{2.2}$$

Thus, if $p(x_i) = 1$, then this symbol contains no information, *i.e.* the symbol is completely predictable and as such contains no self-information. The base of the logarithm in equation (2.2) determines the units in which self-information is measured. Generally logarithms to the base 2 are used which produces units of bits/symbol, although in certain cases natural logarithms are favoured for mathematical simplicity.

The average self-information of a signal is known as the entropy, $H(X)$, of the signal,

$$H(X) = -\frac{1}{N}\sum_{i=0}^{N-1} I(x_i) = -\sum_{i=0}^{N-1} p(x_i)\log_2 p(x_i) \quad \text{bits/symbol} \tag{2.3}$$

where $H(X) \in [0, \log_2 N]$.

For a noiseless channel a signal can be encoded at rate, $R(X)$, which is arbitrarily close to the entropy of the signal , *i.e.*

$$R(X) = H(X) + \epsilon \tag{2.4}$$

where ϵ is an arbitrarily small real number. This result is known as the *noiseless coding theorem* [13](Chapter 4). Huffman codes [14] and arithmetic codes are both variable length codes which attempt to code a signal at a rate which is close to its entropy bound.

2.3.1 Rate-distortion theory

The rate-distortion function, $R(D)$, determines the rate, R, at which a signal can be encoded for a given average distortion measure, D. The distortion measure is a non-negative cost function which, ideally, measures the perceived quality of an encoded image. In practice, formulating a distortion measure which measures perceived image quality is very difficult (see section 2.2) and instead distortion measures such as mean square error (MSE) are used. The mean square error, $\mathcal{E}$, between an image, $i(x, y)$, and an encoded version of the image, $\hat{i}(x, y)$, is defined to be,

$$\mathcal{E} = \mathrm{E}\left[(i(x,y) - \hat{i}(x,y))^2\right] \tag{2.5}$$

where $\mathrm{E}[\cdot]$ denotes the expectation operator.

Other distortion measures which are frequently quoted include signal-to-noise ratio (SNR) and peak signal-to-noise ratio (PSNR) which are defined to be,

$$\begin{aligned} \text{SNR} &= 10\log_{10}\frac{\sigma_x^2}{\sigma_e^2} && (2.6) \\ \text{PSNR} &= 10\log_{10}\frac{\text{maxval}^2}{\sigma_e^2} && (2.7) \end{aligned}$$

where σ_x^2, σ_e^2 and maxval^2 are the variance of the image $i(x, y, t)$, the variance of the coding error, and the maximum amplitude of the signal respectively. If the coding error is a zero mean signal then σ_e^2 and the mean square error are equivalent.

In practice, the rate-distortion function can only be solved for a few simple examples in which it is assumed that the images have Gaussian statistics. Images rarely have Gaussian statistics but the rate-distortion function of a Gaussian source can be usefully interpreted as the upper bound to the actual performance [15] (see also appendix B).

2.4 Spatial and temporal redundancy

Spatial and temporal redundancy allows pixels or regions of an image to be *predicted* from other pixels. Compression is achieved by *transforming* the data so that redundancy is removed. Frequently the removal of spatial or temporal redundancy does not result in compression and can in fact result in expansion. However, the resulting data is frequently easily compressed and these algorithms are often the basis of many very successful coding algorithms [1,3,4,11].

2.5 Coding techniques

The various types of redundancy which can be exploited to obtain compression have been discussed. This section discusses a number of compression algorithms which have been developed to compress greyscale and colour images.

2.5.1 DPCM

Differential pulse coded modulation (DPCM) [1](Chapter 6) reduces or removes naturally occurring correlation by predicting the signal based on previous values of the signal. The signal is then represented by the difference, $e(n)$, between the actual value, $y(n)$, and the predicted value, $\hat{y}(n)$.

$$e(n) = y(n) - \hat{y}(n) \tag{2.8}$$

The predictor complexity depends upon the source model. The lowest complexity predictors use short-term memory of the signal to make a linear prediction. More

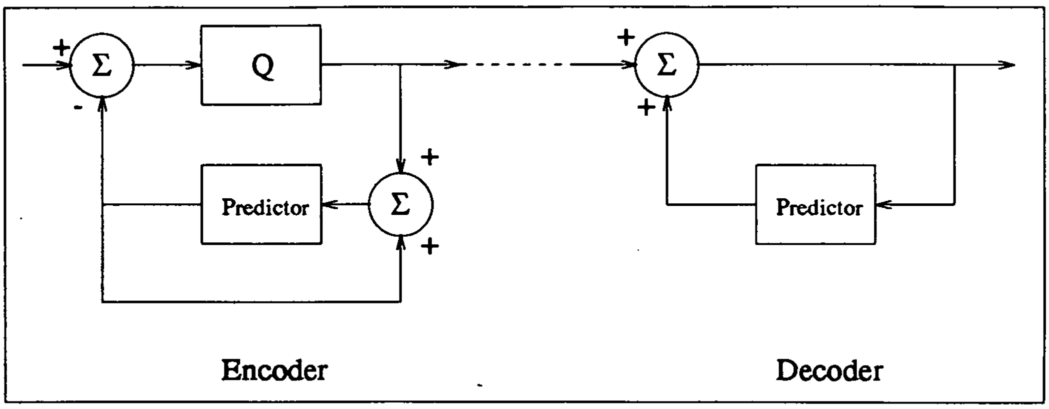


Figure 2.1: *Schematic diagram of a DCPM codec.*

complicated predictors either use longer memory or use short-term memory to make adaptive predictions.

2.5.2 Transform coding

Transform coding is yet another algorithm for reducing or removing naturally occurring correlation from images. Transform coding reduces spatial redundancy in an image by applying a linear orthogonal transform to the image or subblocks of the image so that the resulting transform coefficients contain little or no correlation. In general, the transform is not applied to a whole image but to smaller subblocks to take advantage of strong local correlations. The transform operation is defined by,

$$\theta = \mathbf{A}\mathbf{x} \tag{2.9}$$

where θ, $\mathbf{x}$ and $\mathbf{A}$ are the tensors containing the transformed samples, pixel intensities and the basis matrix respectively. Transform encoding can be applied to data sets of any dimension but in practice one or two-dimensional transforms are invariably used.

The one-dimensional transform is the easiest to understand and as such it will be used to illustrate the principle of transform coding. A sequence of N points is to be transformed, this sequence can be viewed as an N-dimensional vector, $\mathbf{x}$. By analogy with vector geometry the transform operation is a rotation, scaling and translation of

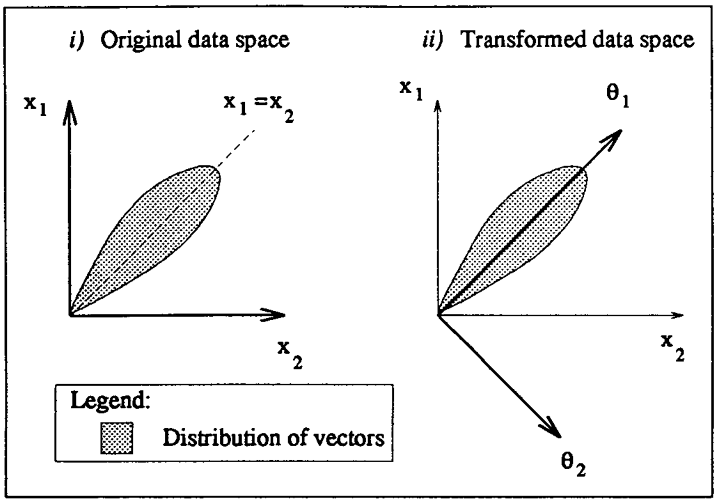


Figure 2.2: *Example of transform coding with two-dimensional vectors.*

the axes (or vector). The transform decorrelates the vector by rotating the axes so that the projection of the vector onto the new axes is small for all but a few (ideally one) of the axes. This can alternatively be envisaged as *energy packing*, in which the transform packs the majority of the vector energy into M of the N transform coefficients.

For example, consider an image in which pairs of neighbouring pixels are used to form the data vectors, $\mathrm{x} = (x_1,x_2)^\mathrm{T}$. Because neighbouring pixels are used we would expect for real-world images that the majority of the data vectors would have approximately equal components, *i.e.* $x_1 \simeq x_2$. Hence the majority of the vectors will be distributed about the line $x_1 = x_2$ in the vector space. Figure 2.2(*i*) shows a two-dimensional space in which the vectors are contained within the shaded region and figure 2.2(*ii*) shows the transformed axes. Clearly, the projection of the vectors onto the θ_1 axis will be large and the projection onto the θ_2 axis will be small.

The Karhunen-Loève transform [2](chapter 3) (KLT) is an optimal transform for the coding of images (see appendix B). The KLT requires the eigenvectors and eigenvalues of the image covariance matrix to be calculated and as such is generally not used because of its high computational complexity.

For data which can be modelled as having autoregressive statistics (AR) with a correlation coefficient approaching unity the discrete cosine transform (DCT) has basis vectors which are almost identical to those of the KLT [1](Chapter 12). The 1-d forward and inverse N-point transforms are defined as,

$$\theta(k) = \sqrt{\frac{2}{N}}\alpha(k)\sum_{n=0}^{N-1} x(n)\cos\frac{(2n+1)k\pi}{2N} \quad k = 0,1,2,\ldots,N-1 \tag{2.10}$$

$$x(n) = \sqrt{\frac{2}{N}}\sum_{n=0}^{N-1} \alpha(k)\theta(k)\cos\frac{(2n+1)k\pi}{2N} \quad n = 0,1,2,\ldots,N-1 \tag{2.11}$$

where $\alpha(0) = 1/\sqrt{2}$ and $\alpha(k) = 1$ if $k \neq 0$ and $x(n)$ are the original data values.

Images are frequently modelled as having AR statistics and as such the DCT is described as being close to optimal. The small computational overhead of the DCT in comparison to that of KLT also makes it very attractive.

2.5.3 Motion compensation

Image sequences frequently contain more redundancy in the temporal dimension than they contain in the spatial dimension. A powerful technique for image compression is to estimate and then compensate for the motion of objects between frames [3]. Compression is achieved by applying one of two techniques: motion-compensated prediction in which the motion estimates are used in a temporal prediction algorithm (section 2.5.1) and motion-compensated interpolation in which the motion vectors are used to reconstruct skipped frames.

The most popular algorithms for motion estimation are block matching and pel-recursive algorithms. The former make motion estimates by dividing an image into a regular array of blocks and then search for the *best* match between blocks in the previous frame. The latter use a steepest gradient algorithm to make motion estimates for individual pixels.

2.5.4 Vector quantization

Vector quantizers encode an image by classifying vectors (subblocks) of the image as vectors from a code book. The formal definition of vector quantization (VQ) is a mapping, Q, of a k-dimensional Euclidian space, $\mathbb{R}^k$, onto a finite subset of $\mathbb{R}^k$, that is,

$$Q : \mathbb{R}^k \rightarrow Y \tag{2.12}$$

Compression is achieved by creating a code book which is a small subset of all possible input data vectors. The classical algorithm for choosing an optimum code book from input data is the Linde, Buzo and Gray algorithm (LBG) [16]. Nasrabadi [17] reviews a range of VQ algorithms.

2.5.5 Fractal coding of images

Fractal image compression was originally introduced by Barnsley [18] and was heralded by the popular literature as a method of achieving enormous compression ratios whilst still maintaining high image quality. This has been subsequently shown to be untrue and it has also been shown that fractal encoding performs no better than any other image coding technique.

Fractal image compression is based upon iteratively applying a transform to a subblock of an image [19]. The transform is chosen to be contractive so that after a finite number of iterations the subblock forms a unique fixed point. The image is reconstructed by iteratively applying the transform.

2.5.6 Object-oriented algorithms

Object-oriented algorithms attempt to encode the semantic detail of an image rather than reduce statistical redundancy [6]. Object-oriented algorithms, in general, give higher compression ratios but at the price of reducing the generality of the encoder. For example, model-based coding algorithms fit a wire frame to the face [20–22] and then transmit the changes to the vertices of the wire frame which requires only a few tens of bits. However, if the object to be encoded differs significantly from the model the encoder will fail.

2.6 Standards for image coding

To enable the interchange of compressed picture information, image coding standards have been introduced. Three standards have been introduced for compressing greyscale and colour images: the joint photographic experts group (JPEG) standard for still greyscale and colour images, and the CCITT H.261 and the motion picture expert group (MPEG) for full motion video.

2.6.1 JPEG

The joint photographic experts group (JPEG) [23] proposed standard is for greyscale or colour still pictures of natural, real-world scenes. It comprises two basic compression algorithms, an information-lossy coding scheme employing the DCT (see section 2.5.2) and an information-lossless coding scheme employing a predictive coding algorithm (see section 2.5.1). The lossy coding algorithm is known as the *baseline* method.

Figure 2.3 shows a schematic diagram of the baseline encoder. Non-overlapping 8×8 pixel blocks of the image are transformed using the DCT. The resulting transform coefficients are then quantized using a user-specified quantization table. Quantization of the d.c. coefficient leads to visually disturbing results and as such it is not quantized. The quantized DCT coefficients are then re-ordered into a one-dimensional array by zig-zag scanning the coefficients starting at the d.c. coefficient and finishing at the highest frequency coefficient (see figure 2.4). The re-ordered transform coefficients are then grouped into run-amplitude pairs and variable length encoded. The d.c. coefficients are encoded separately using a predictive coding algorithm and a variable length code.

2.6.2 H.261

The CCITT H.261 [7] standard is a hybrid of motion-compensated prediction and DCT encoding for transmitting moving images at $p \times 64$ kbits/s (where p is an integer). A schematic diagram of the H.261 encoder is shown is shown in figure 2.5. The encoder consists of motion-compensated prediction, DCT, a quantizer, run length and variable

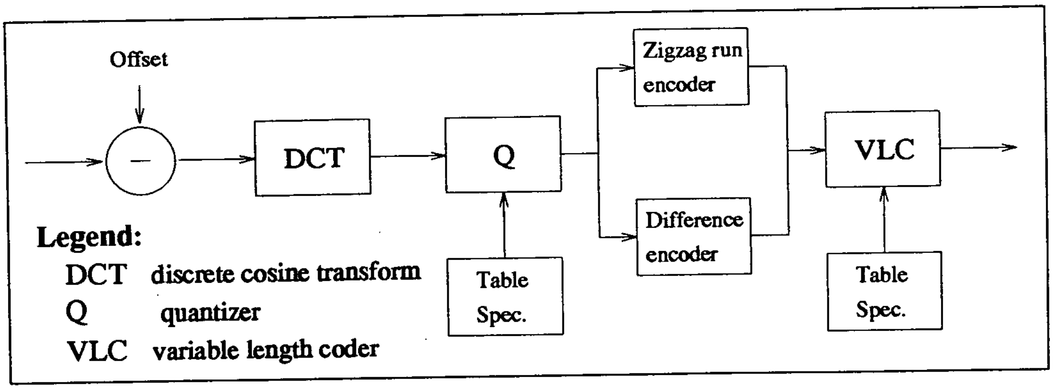


Figure 2.3: *Schematic diagram of the JPEG encoder.*

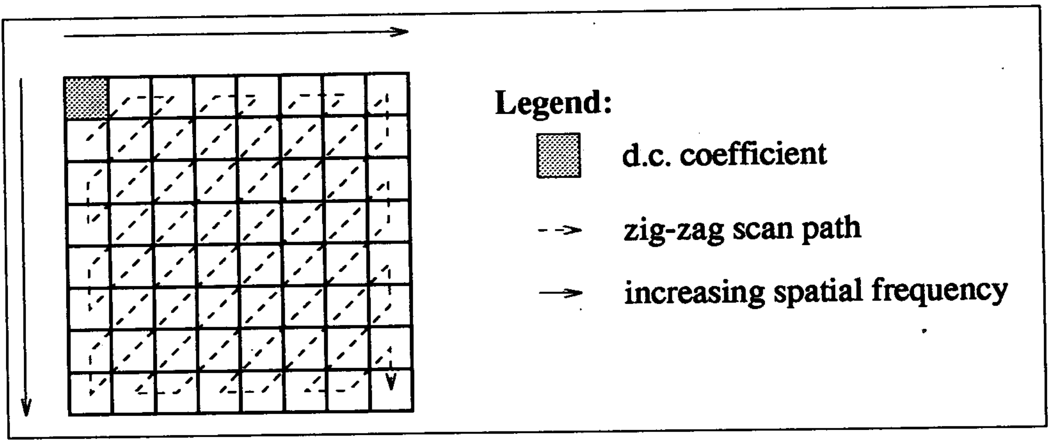


Figure 2.4: *Zig-zag scan path for DCT coefficient ordering.*

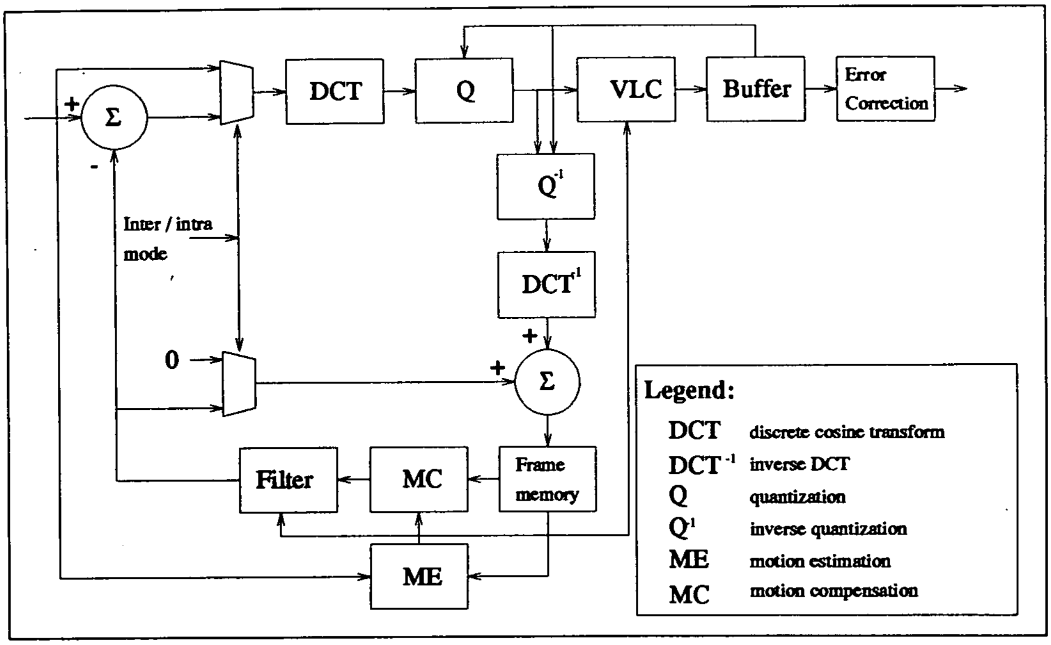


Figure 2.5: *Schematic diagram of the H.261 encoder.*

length encoding.

The encoder may operate in two modes: inter-frame or intra-frame. The intra-frame mode codes spatial detail and is similar to the JPEG algorithm. The inter-frame mode uses a prediction algorithm which can be optionally augmented by motion estimation.

To allow the transmission of video across international boundaries a frame format known as common intermediate format (CIF) has been defined. The CIF format consists of non-interlaced frames occurring 30000/10001 (approximately 29.97) times a second. A frame is coded as luminance (Y) and two colour difference frames (C_B and C_R) as defined by CCIRR recommendation 601. Two different sized CIF formats have been defined: quarter-CIF (QCIF) and CIF. The bigger format, CIF, has a luminance signal which contains 288 lines × 352 pixels. The colour difference signals are sub-sampled by factor of two in each dimension. The smaller format, QCIF, has half the number of lines and half the number of pixels. The standard defines that all decoders should decode QCIF and the CIF format is optional.

Each CIF or QCIF frame is organised hierarchically. At the lowest level the frame is divided into 8 × 8 blocks. Four 8 × 8 luminance blocks and two 8 × 8 chrominance blocks are called a *macroblock* (MB). The macroblocks are also grouped together to form a *group of blocks* which contains three rows of 11 macroblocks.

2.6.3 MPEG

The motion picture expert group (MPEG) standard is a generic standard for the compression of video and audio. The standard is split into three parts: MPEG-1 for audio and video compressed to be within a bandwidth of 1.5 Mbits/s, which is the bandwidth of uncompressed compact disc (CD) and digital audio cassette (DAT); MPEG-2 for higher quality video and audio compressed to be within the range 2-15 Mbits/s; and MPEG-4 will address video and audio compressed to less than 64kbits/s. MPEG-3 was intended for compressing HDTV pictures but it was subsequently found that the MPEG-2 standard could be used for this purpose,, and as such it was superceded before it was even designed!

Although MPEG is designed to be a generic standard in the sense that is independent of any particular application; it has been designed with certain application specific features in mind. These features include random access, fast forward/reverse searches, reverse playback, editibility and format flexibility.

MPEG-1

MPEG-1 [8] and H.261 share many similar features since the MPEG committee strived as much as possible to make the two standards compatible. The two standards differ in that H.261 is designed for video communications at bandwidths as low as 64 kbits/s and as such it has a much more tightly constrained bitstream than MPEG-1.

MPEG defines three different frame types: Intra-frame (I), predicted frame (P) and interpolated or bi-directional frame (B). The intra-frames give the lowest compression but provide random access points. The predicted frames are coded with reference to a past 'I' frame or a previous 'P' frame and are the same as the H.261 prediction errors. The interpolated frames provide the highest compression by transmitting the sequence at a reduced temporal frequency and then reconstructing the missing frames using interpolation. Interpolated frames are coded with reference to the two past and future 'I' or 'P' frames. This is done by testing the forward vector, or the previous vector or by averaging the future and past blocks. If none of these works well the block is intra-coded. The frequency of the 'B' and 'P' frames is application dependent but the 'I' frames are separated by 12 frames.

MPEG-2

The MPEG-2 standard is for high quality video and audio and at the time of writing was not completely specified. It aims to be a compatible extension to the completely specified MPEG-1 standard and also supports interlaced video formats and features to support HDTV.

MPEG-4

MPEG-4 is aimed at very-low-bit-rate encoding of video (4,800-64,000 kbits/s). MPEG-4 will allow video to be transmitted on conventional PSTN telephone lines and mobile communication channels. At the time of writing the standard was still at the proposal stage, although the following techniques were being considered: morphology, subband coding and model-based coding.

2.7 Summary

This chapter introduced various aspects of image compression. Initially, the redundancies which are typically exhibited by image data were discussed. This was followed by descriptions of popular coding algorithms. In conclusion the international standards for the compression of greyscale and colour images were introduced.

Chapter 3

Parallel implementations of image coding algorithms

3.1 Introduction

For real-time applications, *i.e.* encoding/decoding at full frame rate, many algorithms for video coding require considerable computational power. As the algorithms for the encoding/decoding of images become increasingly more complex progressively more sophisticated hardware solutions are required to obtain real-time performance. For example, Whybray [24] estimates the worst case computational load for the CCITT H.261 standard to be 9000 million operations per second (MOPS). Current DSP microprocessors are only able to operate at a few tens of MOPS and as such H.261 is usually implemented on dedicated hardware or as a custom chip set.

The computational requirements of image coding algorithms can also be a hindrance to development. Many parameters such as quantizer step size are perceptually optimised by simulating the algorithm. Since a single frame can take several minutes to process on a conventional workstation this can be a time-consuming process.

Parallel architectures are a natural way of obtaining the required performance for simulating and implementing these algorithms. Indeed, it is common in hardware implementations to exploit simple parallel architectures such as pipelines or systolic arrays. Many modern microprocessors also contain parallelism, which is frequently hidden from the user but is nevertheless present. In this chapter strategies for explicitly exploiting the inherent parallelism in image coding algorithms are considered. Using H.261 as an example, results will be presented based upon work aimed at producing a high performance simulation tool for image coding algorithms on a transputer surface.

3.2 Parallel architectures

Flynn's taxonomy [25] broadly divides computer architectures into four classes:

SISD: Single Instruction Single Data machines are simple single processor sequential machines. This is the classical Von Neumann architecture.

SIMD: Single Instruction Multiple Data machines have many processors which apply the same instruction to many different data objects.

MISD: Multiple Instruction Single Data apply multiple operations to each piece of data fetched from memory.

MIMD: Multiple Instruction Multiple Data machines contain several independent processors which execute independent programmes. MIMD architectures can be further sub-divided into two classes:

- Shared memory or tightly coupled architectures in which processors share the same global memory. These architectures are simple to program but the shared memory creates a bottleneck which limits the number of processors which can be usefully connected together.
- Distributed memory or loosely coupled architectures in which the processors each have their own local memory and they communicate by message passing.

The current trend in parallel computing is towards either SIMD or MIMD architectures. For the purposes of creating a tool for simulating image coding algorithms or indeed for creating an actual codec, the MIMD architecture is the most natural to use since extra processors may be added at will to increase the system performance.

3.3 Transputers and multicomputers

Although many of the methods described in this chapter are independent of the computer architecture it is essential to describe the system used to obtain the simulation results.

The inmos transputer [26] combines a central processing unit, four bi-directional serial communications links, and memory in a single RISC device. Both internal RAM and external memory can be addressed. Serial communication between transputers is provided by four 20 Mbit/s links. The links and CPU run autonomously and concurrently, thus allowing communications and computation to be overlapped. The IMS T800 is a transputer with a floating point processing unit which provides efficient hardware for floating point operations.

The Edinburgh Concurrent Supercomputer (ECS) [27] is a Meiko Computing Surface which houses 425 T800s. It is a distributed memory MIMD or message passing machine. These are shared amongst single-user *domains* of various sizes and resources — *e.g.* graphics board, frame grabber, *etc.* Each T800 has 4 Mbytes of external RAM. The domain topology is configured by the programmer prior to run-time. This allows for easy experimentation with various topologies.

The ECS has proved to be an ideal computing engine for many image processing problems. However, simulating H.261 in real-time presents some special input/output (I/O) bandwidth problems. The simulations were performed upon standard image sequences such as "Clare" and "Miss America" stored on disk. Obviously, disk access times would prohibit real-time simulations. Instead, a sequence of up to 20 frames was read from disk into the RAM of the master processing element (PE) before simulation commenced. Once in RAM the image data could be accessed as quickly as possible.

Displaying results in real-time proved more difficult. Actually displaying 10 frame/s RGB colour images was not a problem on the ECS. However, the H.261

algorithm operates on YC_rC_b colour data which must be transformed pixel by pixel to RGB format before display on conventional monitors is possible. This amount of extra processing dramatically reduces the performance [28]. However, dedicated hardware is available which can do this conversion extremely quickly and so could remove the problem at a very small cost[1]. Alternatively the results can be stored on a the disk in RGB format and then read back later for real-time viewing.

3.4 Parallel considerations

Processing elements in distributed memory machines have distinct address spaces and hence must exchange information by message passing. This act of communication introduces an overhead which decreases the performance of the system. The transputer allows communication to be overlapped with computation so that this overhead can be minimised. For small networks of transputers in which the transputers are fully connected, the communication overheads can be negligible, but as the network size increases they can become significant. Communication delays can be attributed to:

Through-routing: Routing messages through intermediate processors and so 'stealing' CPU time in making a decision as to where next to send the message.

Link contention: Messages may need to travel along the same link at the same time resulting in one or more of them being delayed. This can be alleviated by adding more link bandwidth or employing an intelligent routing algorithm which utilises alternative routes.

For maximum efficiency, optimal use must be made of the resources available; there is little use in having many processors if only a few of them are doing useful work on the data at any given time. The work should be distributed among them so as to keep them all busy most of the time. This is called *load balancing*.

In order to parallelise a program, the problem must be divided into separate parts. Selecting the size of these portions which are allocated to each processor is critical to

---

[1] British Telecom Laboratories use such boxes

the efficiency of any parallel program. This is known as the implementation *grain size*. Intuitively, one expects fine-grained concurrency to deliver faster results while using more processors. However, for message passing machines there is a point at which the additional communications outweigh the expected benefits.

The choice of grain size effects the scalability of the program: too small a grain and the many messages saturate the limited bandwidth between the many processors. A parallel machine can also be thought of as having a grain size. In this case it is some function relating communication power to computation power, but it is not obvious how to evaluate it [29]. Thus it can be seen that the selection of grain size depends on both the application and the target machine. In practice an educated guess at the best granularity can be made and then empirical experiments will provide the best result.

3.4.1 Parallel languages and development tools

The Communicating Sequential Processes (CSP) model developed by Hoare [30] describes a parallel program as a set of communicating sequential processes. Occam [31] and the transputer are both based on the CSP model. The channel between processes is only used for communications between those processes. Each act of communication forces the processes involved to synchronise, preventing both from doing useful processing. Processes are not allowed to share data even if they reside on the same processor. This can lead to unnecessary copying of data.

Since the transputer has only four links, it may not be possible for processes to be directly linked via channels. Instead, communication protocols must be devised and implemented by the programmer. When the number of communicating processes becomes large, the handling of channels can become a daunting task.

For these practical reasons a better communication model is one which addresses messages by destination process rather than channel name [32]. *Tiny* [33] is a message passing system which supports such a model of communication and was developed at the University of Edinburgh on the ECS. It functions on any processor array topology allowing the user to easily experiment with various transputer configurations and process mappings. Message routing is optimised for small messages so as to allow high granularity to be exploited in applications thus leading to better speedup and load balancing. The speedup of a parallel program is defined as:

$$Speedup = \frac{T(1)}{T(n)} \leq n \tag{3.1}$$

where $T(1)$ is the time for the program to execute on one PE and $T(n)$ the time for concurrent execution on n PEs. *Tiny* is supplied as a compiled library. User processes are coupled to *Tiny* by means of an **occam** harness. It can currently be used to link processes written in **occam**, Fortran, or C. The important features of a good message router are:

- Communication times grow slowly with the number of processors, n, *i.e.* $O(\log n)$.
- The routing is parallel, *i.e.* alternative routes and adaptive strategies are used under conditions of high network loading.
- Non-local communications are fast so that the program can be ported to different topologies and still run efficiently.
- Communications are not forced to be synchronous so that unnecessary delays are removed.

The principal advantage of a message router is that programs are made more portable. Unconventional topologies can be experimented with without having to worry about how complicated the communication protocol will be.

There are now several parallel program development environments available. They support single processor or multiprocessor applications using familiar development environments and standard languages such as C and Fortran and comprise: compilers, configuration tools, and high level communication services. Their aim is to hide most of the difficulties of parallel programming from the user while still allowing the user access to the underlying parallelism of the hardware. Unfortunately, the natural consequence of such a generalised system is increased overheads which can result in reduced efficiencies.

Meiko's CS Tools [34] (Communicating Sequential Tools) is such a program development toolset. Its communications services provide a set of hardware independent high level communication functions. CS Tools provides similar functionality to *Tiny* but less efficiently.

3.4.2 Problem decomposition and mapping

The problem must be mapped to the network of processors so as to minimise the time for the set of tasks to complete execution. This 'mapping problem' can be tackled by models which assign execution times and communication patterns to tasks [35, 36]. The proposed mapping models do not match reality very closely and are also difficult to apply. Instead programmers have come to use stereotyped solutions which they then try to optimise heuristically [37, 38]. Experience has shown that if the *data space* is mapped instead of the *algorithm*, greater efficiency can be achieved and expensive, complex and inelegant solutions [39] (see section 3.5) can be avoided.

The decomposition of the problem is critical to achieving good performance and scalability. The decomposition can be classify into one of two techniques:

Functional decomposition: In an algorithmic or functional decomposition the algorithm to be implemented is partitioned into functional blocks. Processes to implement these blocks are distributed across the processor network which is configured so as to mimic the *data flow* between the functional blocks. Algorithmic decompositions do not scale well and it is difficult to balance their load. Thus they do not achieve very high efficiencies. Figure 3.1 shows a functional decomposition used by Sexton [40] to simulate H.261 on a Meiko Computing Surface with up to 14 transputers at British Telecom Labs (BTL) (see section 3.5).

Data decompositions: Rather than splitting the *problem* into functional blocks the *data set* can be decomposed so that each processor executes the same process but only operates on a part of the data set. Data decompositions can be further subdivided into two groups:

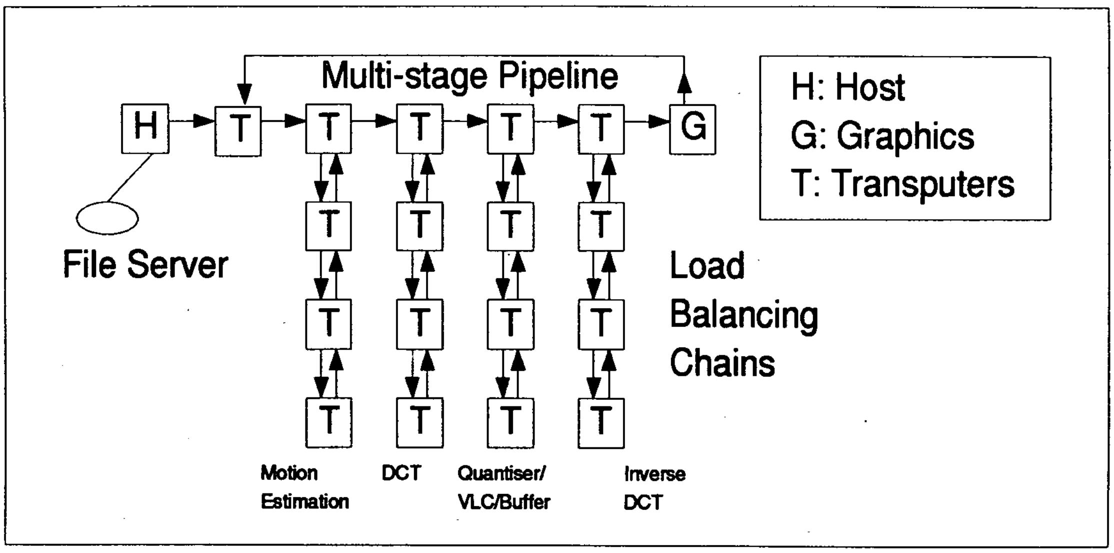


Figure 3.1: *Functional decomposition of H.261 by Sexton at BTL.*

Geometric decomposition: The data set is divided equally amongst the processors. This can lead to uneven load balancing due to localised computationally intense regions in the data set being distributed to a small fraction of the processors.

Task farm: The task farm is a well known dynamic load-balancing technique which uses the classic master-worker control structure. The master processor distributes grains of data to the workers. When each worker completes a task it requests more work from master.

Data decomposition can provide automatic load balancing by dynamic task scheduling and, with the correct choice of processor topology, can also scale extremely well.

3.5 Related parallel implementations

Parallel image processing problems have been traditionally tackled as a multi-staged pipeline [36]. This was the approach used by Sexton *et al.* [40]. They produced a functional decomposition of the H.261 standard based upon chains (or load-balancing pipes), figure 3.1. At the time no message routers were available and so they were restricted to using simple topologies such as chains, despite their obvious inefficiencies.

Figure 3.1 shows the data flow through the pipelined processor graph. Load balancing is a problem if multi-stage pipelines are used; considerable effort must be made to ensure that each stage of the pipeline is equally computationally intense [41]. Multi-stage pipelines have a communications problem; if each load balancing pipe is the correct length then, on average, each job travels half way down the pipe and back up again. This usually makes for excessive communications overheads. However, fundamental problems prevented the simulation from running near to real-time.

Sexton's original serial implementation was coded in C on a VAX 750 and ran at approximately 15 minutes per QCIF frame, parallelisation of the C code with a custom written **occam** harness on a 14 transputer pipelined topology produced speeds of around 10 seconds per QCIF frame. This is two orders of magnitude away from the requirements for real-time coder simulation.

A fine-grained functional decomposition has been implemented using transputers by Ichikawa and Shamura [39]. This uses 100 transputers in a hybrid topology tailored specifically to the requirements of H.261. The result is a massive amount of non-reusable engineering which would far better have been achieved in hardware since changes in the algorithm would almost certainly require changes in the topology.

Rudberg *et al.* [42] have implemented a variation of recommendation H.261 for a local area network (LAN). Their variation on H.261 operates on monochrome images, does not include variable length coding of the DCT coefficients, and only *intra-frame* coding is implemented. In this way the most computationally intense part of the algorithm, *viz.* motion compensation, is ignored. Their mapping strategy is essentially the same as the farming model adopted in this paper whereby all the H.261 functions are implemented on each transputer. By contrast however, they do not use a message passing system and all their programs are written in **occam**. The results published are for a tree topology using 16 transputers. Given all of the above constraints, they report execution speeds of less than two QCIF frame/s.

Function	*% Time*	*Absolute time (ms)*	
	occam	occam	*C*
DPCM	1.3	1.9	3.2
DCT	19.0	30.0	12.0
Scanning	5.5	8.3	0.4
Quantisation	1.1	1.7	0.4
Inv. quantisation	1.0	1.4	0.4
Inv. scanning	2.2	4.9	0.5
Inv. DCT	19.0	29.0	11.4
Inv. DPCM	1.2	1.8	3.1
VLC	1.0	1.1	0.5
Low pass filter	4.4	7.3	9.6
Motion estimation	47.0	70.0	31.0

Table 3.1: *Breakdown of H.261 function execution times on a single macroblock.*

3.6 Multiprocessor implementations

Initially code was developed on a minimal network of two transputers: one for the simulations and the other for displaying the graphical output[2]. In this way, complications due to communications were removed. Also the timing performance measured for one processor is necessary in order to calculate the efficiencies of larger processor networks. Initially all programs were written in occam since it was the only language available on the ECS at the time. The uniformity of occam makes it very easy to write processes without needing to know whether they are running on the same or different processors.

Table 3.1 shows the proportion of time taken to execute the various processes for the functional blocks of H.261. These values are averages for standard videophone test image sequences. Clearly, motion estimation consumes far more computational power than any of the other functional blocks.

---

[2] It is a feature of the ECS graphics domains that the graphics processor is never the processor connected to the filestore (via the host). Thus *two* is the smallest number of transputers that could be employed.

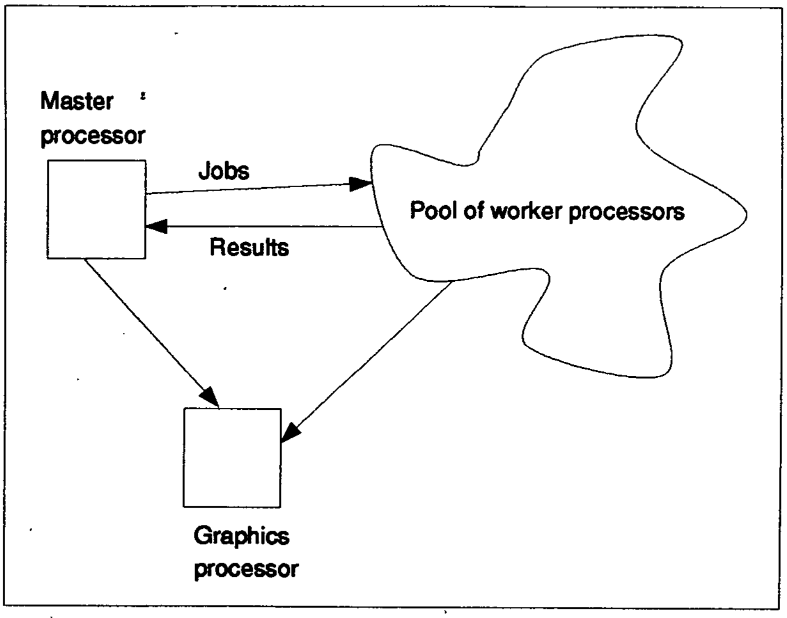


Figure 3.2: *The chosen process structure for the simulations.*

3.6.1 Mapping employed

A hybrid of the task farm and algorithmic paradigms of concurrent program decomposition were used to simulate H.261. The implementation contained three distinct processes as shown in figure 3.2:

Master: Distributes tasks, collates results and maintains global structures.

Worker: Accepts tasks and performs the main coding functions.

Graphics: Displays colour blocks received from the master or any worker PE.

The communication pattern is complex. The master passes packets of data to each of the workers which in turn pass packets of data back to the master and on to the graphics processor. Without a topology independent message routing harness, it would be extremely difficult to program this efficiently, taking advantage of alternative routes to increase overall throughput. With all the main H.261 functions contained in each worker process, it was easy to experiment with various topologies with little reprogramming. The load balancing scheme used in these simulations is dynamic since it is determined at run-time and adaptive since the tasks are sent to whichever worker processors are ready for more work.

3.6.2 Choosing the granularity

For this particular algorithm the data is already broken down into a hierarchical structure (section 2.6.2). From this, the smallest element upon which all the functions (DPCM, DCT, quantisation, variable length coding, *etc*) can be performed independently is the MB. The MB was chosen as the grain size. Any smaller would not be practical to operate upon independently; any larger would not utilise the number of PEs available and allow good load balancing. There are 99 MBs in a single QCIF image frame.

Communications were overlapped with computation by ensuring that each worker processor is initially sent *two* MBs. Once received, the first is processed while the second is buffered. On completing the processing of the first MB, the worker sends the results to the master and *immediately* starts work on the buffered MB. On receiving a result from a worker, the master sends that worker another MB to process.

Experiments with more than one MB buffered at the worker showed that communications were already maximally overlapped with computation for the chosen grain size.

3.6.3 Processor array topology

If the communication patterns of the mapping are simple then a simple topology such as a grid or a tree may be appropriate. For more complicated communication patterns, determining optimal processor topologies can be difficult. Tree and chain topologies have often been used to implement processor farms [43] (see [44] for analyses) regardless of whether they mimic the actual communication patterns, simply because these reduce the complexity of communication protocols. Topology independent routing harnesses remove the need for the programmer to be concerned with the complexity of the communication protocols. Indeed trees and chains may not be good topologies to use with routing harnesses which provide an adaptive routing strategy; in these topologies, there is only one path to each of the nodes and so the routing harness cannot re-route messages to avoid congested links.

For a task farm we require that the master can deliver data to the workers as fast as possible. This is further complicated when the graphics processor is included because the workers must deliver the processed data to the graphics processor as fast as possible. To this end, the best topologies are *compact* ones in which all the processors are as densely connected (or as "close") as possible to each other. The processor topology must be further constrained so that no "hotspots" (or bottle-necks) exist, *i.e.* no paths should exist through which a large proportion of the communications traffic *must* pass.

Ideally, some form of automated system is required to generate *good* processor network topologies. Thus the essential communications characteristics of a processor network can be modelled using techniques from graph theory. Processor graphs have a set of attributes such as diameter and inter-processor distance which can be optimised so as to minimise communication delays. The terms topology and graph will be used interchangeably throughout the following discussion.

Graph theory

Some, but not all, graph theoretic terms used are defined here. The reader unfamiliar with the field is referred to the recent paper by Ghafoor [45] where more rigourous definitions are provided. In using graph theory to describe a processor network, the processors are represented as *nodes* or *vertices*, and their interconnecting links by *edges*. The *degree* of a node is the number of edges with which it is incident. The *distance* between two nodes is the smallest number of edges traversed in travelling between them and the *diameter* is the largest of all the shortest paths between nodes.

A wiring file for an ECS domain describes which processors are directly connected. From this information, a table of distances between all processors is easily deduced. In the graph theory literature this is termed the "all-pairs shortest path" problem for the solution of which there are several algorithms (*e.g.* Floyd-Warshall described by Cormen [46]). The adopted solution was similar to these but simpler since all edges in a processor graph are unweighted (equal communications load since all worker PEs perform identical tasks) and undirected (bi-directional links). Thus the table of shortest paths is symmetric and so only half of its values need be computed.

Graph generation

The application graphs used for these simulations were subjected to several constraints. All nodes are of degree four since transputers have four links. The master processor is linked to the host machine and so only has three remaining links for inclusion in networks being created. The graphics processor is unique and so perhaps should be carefully positioned in the network. This explains why the mean inter-PE distance measure is important rather than just the mean master-to-worker distance. Also this is another reason why the tree topology is not good for these simulations: there is no obvious place for the graphics processor.

The hypercube (or binary k-cube) has been a very popular choice of topology for parallel machines of fixed architectures. The reason for this popularity is the high interconnectivity provided for a small number of connections at each node. For transputer-based machines however it is limiting. With four links at each node, at most sixteen nodes can be interconnected in this way. Thus for this project, hypercubes are not suitable. Additionally, analysis shows that smaller diameters and average inter-node distances can be obtained from irregular graphs. Experiments were carried out with three different types of networks, one of which is 'regular' *i.e.* the PE connections have some kind of symmetry, and the other two are 'irregular' [47]. The network generation algorithms are all restricted to forming a Hamiltonian[3] ring initially so that there are no isolated nodes. Brief descriptions of each of the algorithms are given below (refer to figure 3.3):

Chordal rings These regular graphs are formed by connecting chords across a ring of processors [48]. Trial and error can be used on the chord length to generate optimum graphs of this kind. Chordal rings have been popular in the past because their regular structure makes routing algorithms easier to implement [49].

'Greedy' graphs These graphs are produced by iteratively placing links so as to *locally* optimise the partial solution. Thus a 'greedy' graph might be generated by progressively connecting links which will most reduce the diameter of the

---

[3] A Hamiltonian path is one which visits all nodes in the graph exactly once.

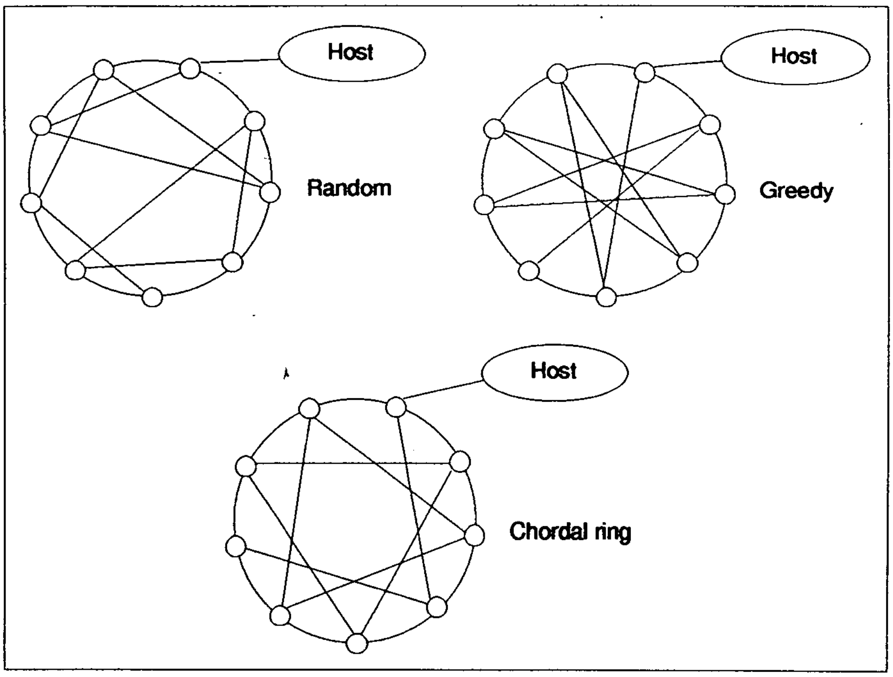


Figure 3.3: *Examples of the three graph types for a total of nine PEs.*

graph. The resulting graph may not have minimal diameter for the number of processors but it will have a *low* diameter. 'Greedy' algorithms [50] are often used in optimisation problems.

Random graphs A random graph is easily configured by first wiring the PEs in a ring and then randomly connecting up remaining pairs of links. The only constraint on the graphs is that they contain no self-links. Such a linking scheme could allow bottle-necks to appear. Several random graphs were generated and the one with the lowest mean inter-PE distance was selected.

Having generated the processor graphs according to these various schemes, it was of interest to see how the mean inter-node distances varied with the number of nodes. Figure 3.4 shows this result plotted with the number of processors on a logarithmic scale. It is clear that the 'greedy' graph scales the best, although the random graphs scale almost as well. Since the plots appear to be almost linear for both 'greedy' and random graphs, it can be deduced that their rate of increase is O(log n), where n is the number of processors. The chordal graph plot increases much more quickly, and so it is concluded that chordal rings do not scale so well. Browne and Hodgson [49] give

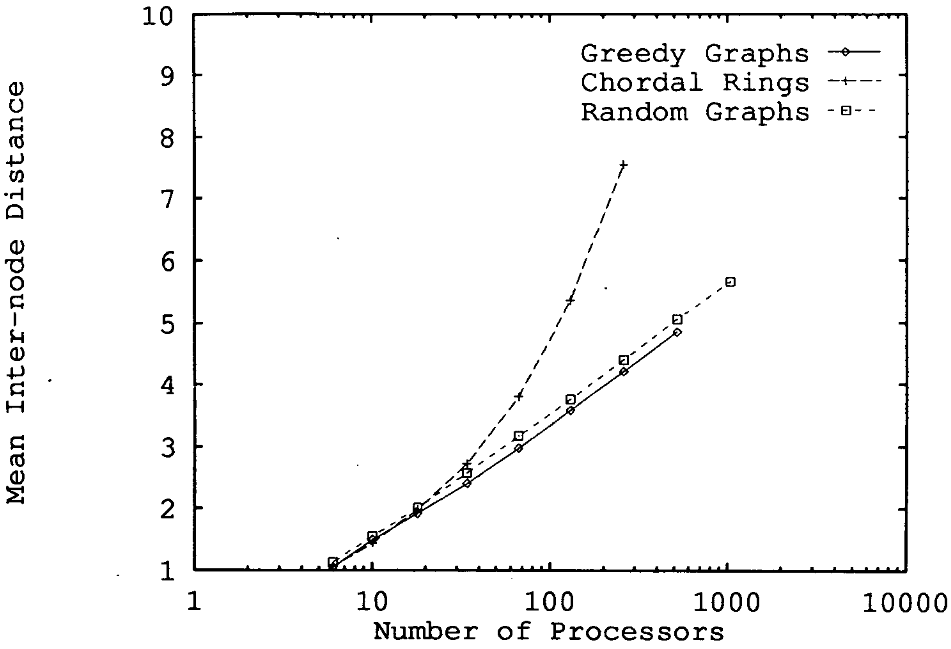


Figure 3.4: *Mean inter-processor distance for various processor graphs.*

the chordal ring mean inter-PE distance as being at best $O(\sqrt{n})$. Since the advent of topology independent routers, the programmer is no longer concerned with keeping the processor topology simple and regular. This result shows that regular graphs do not scale as well as irregular ones. Figure 3.5 shows how the H.261 simulation execution times for a QCIF frame varied for these three topologies for up to 64 worker PEs. The variation is small as would be expected from the results shown in figure 3.4; for small numbers of transputers (*i.e.* less than twenty) the mean inter-PE distances are almost the same.

The random graphs were optimised for low inter-PE distances and no account was taken of potential bottle-necks. This can be seen in figure 3.5 by the fact that the execution time for the random topology *increases* when moving from 16 up to 33 workers, and then the execution time decreases again for 64 workers. Hence this indicates that the 33 worker random graph is not a good one and may contain bottle-necks. The symmetry of chordal rings guarantees no bottle-necks provided that the computation is evenly distributed. It is also unlikely that the 'greedy' graphs will contain bottle-necks since at each stage, the link placement strategy aims to minimise the graph diameter; after each link is placed, the diameter must shift to another area of

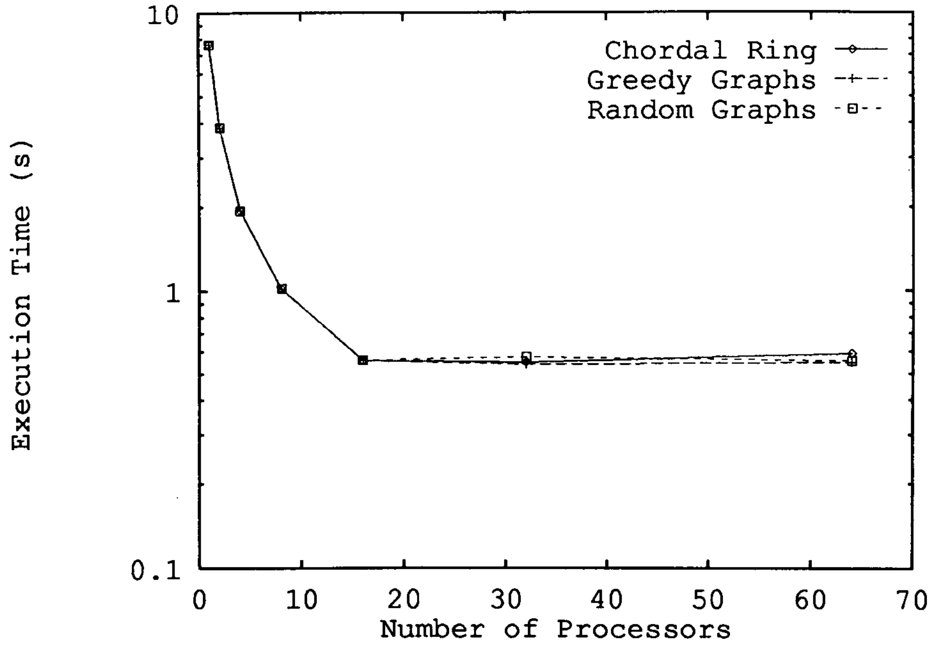


Figure 3.5: *Timings for H.261 (without motion compensation) simulation on a QCIF frame for 'greedy', chordal and random topologies using* **occam** *and tiny.*

the graph which is not as well connected. In this way the communications bandwidth is fairly evenly distributed throughout the graph.

In order to verify that the chain topology performs as poorly as predicted, the DCT/inverse DCT coding loop was implemented on a balanced chain. Figure 3.6 shows the timing results for the chain topology compared to a random graph which confirms the theoretical prediction. The disparity is apparent from upwards of two worker PEs. Above 16 workers the simulation using the chain topology actually slows down whereas the simulation using the random graph topology continues to make small gains. Thus the chain topology was abandoned.

3.6.4 Choice of languages

Initially these simulations were written entirely in **occam** under the **inmos occam** programming system (OPS). The sequential processes themselves, however, are far more easily written in a more semantically rich language such as C [51] which allows the use of pointers, recursion, memory allocation and for which useful, optimised libraries have been written. Development in C is further aided by the existence of familiar envir-

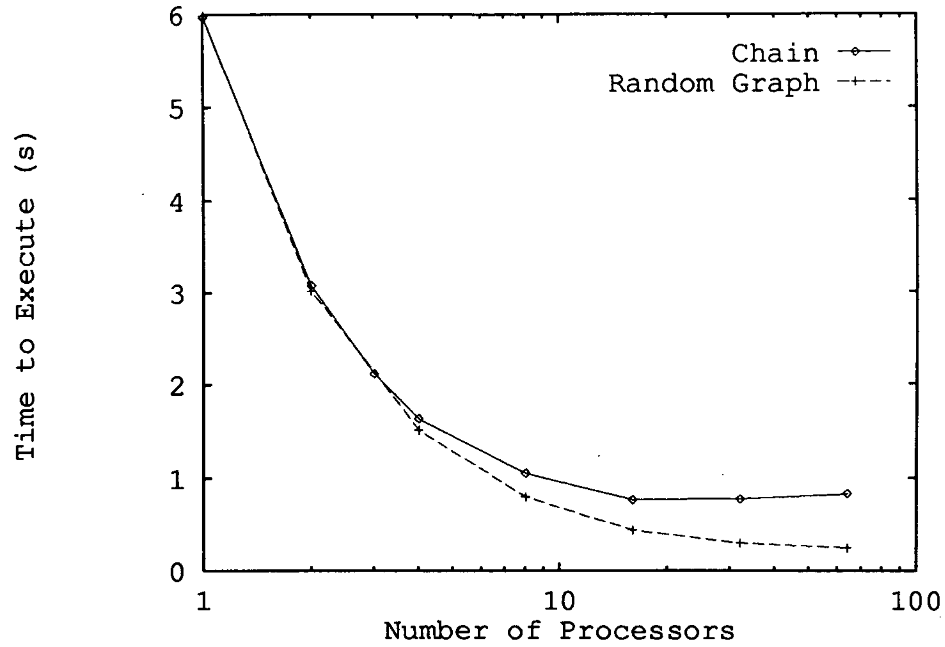


Figure 3.6: *Timings for DCT/IDCT simulation on a QCIF frame for chain and random Hamiltonian topologies.*

onments with powerful debugging tools. It was found that translating processes from occam into C could easily achieve up to a factor of two speedup in sequential execution time (refer to table 3.1). The reasons for this are mainly due to matters of memory management. When programming in C, it is easy to determine where critical pieces of code reside in memory. Thus for maximum speeds, the transputer on-chip memory is employed as much as possible. With occam compilers, this is far from straightforward. Also the availability of pointers in C aids the writing of efficient implementations of algorithms. A similar reduction in development time was observed. For these reasons the simulation serial programs are now developed in C. The sequential processes were parallelised using CS Tools (section 3.4.1).

3.7 Visualisation

One of the primary aims of these simulations was that the user should be able to view the output from *any* stage of the algorithm. This was one of the main reasons for choosing the compact topologies in which the master and all of the worker PEs

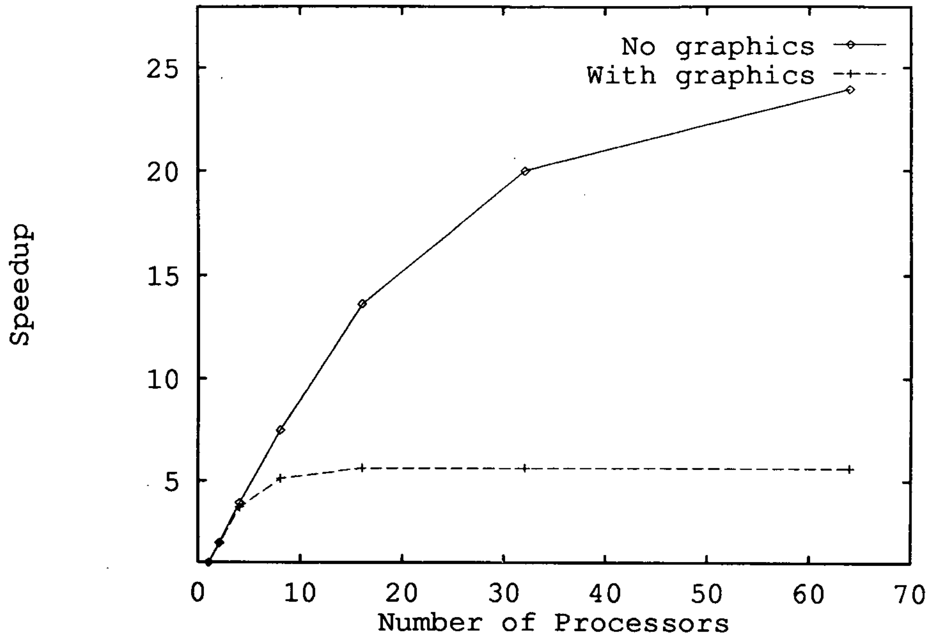


Figure 3.7: *The effect on speedup of all worker PEs sending results to both the graphics and master PEs.*

are as close to the graphics PE as possible. Also this is why the mean inter-PE distance was minimised in the graph optimisations. Unfortunately the ECS graphics domains contain a single graphics PE to service all graphics requests. Thus a severe bottle-neck is present. Figure 3.7 shows how displaying graphical output saturates the communications bandwidth and prevents the simulations benefiting from additional worker PEs. The solution adopted was to allow the user of the simulation to turn the graphics on or off as desired. The results can be stored on disk and displayed off-line in real-time.

3.8 Performance evaluation

Figure 3.8 shows plots of execution time for the H.261 algorithm without MC against increasing numbers of transputers. Results are presented for simulations using occam with the *Tiny* routing harness and C with CS Tools. These simulations used random Hamiltonian processor topologies but, as explained in section 3.6.3, the performance of ‘greedy’ graphs and chordal rings is virtually identical for these numbers of PEs.

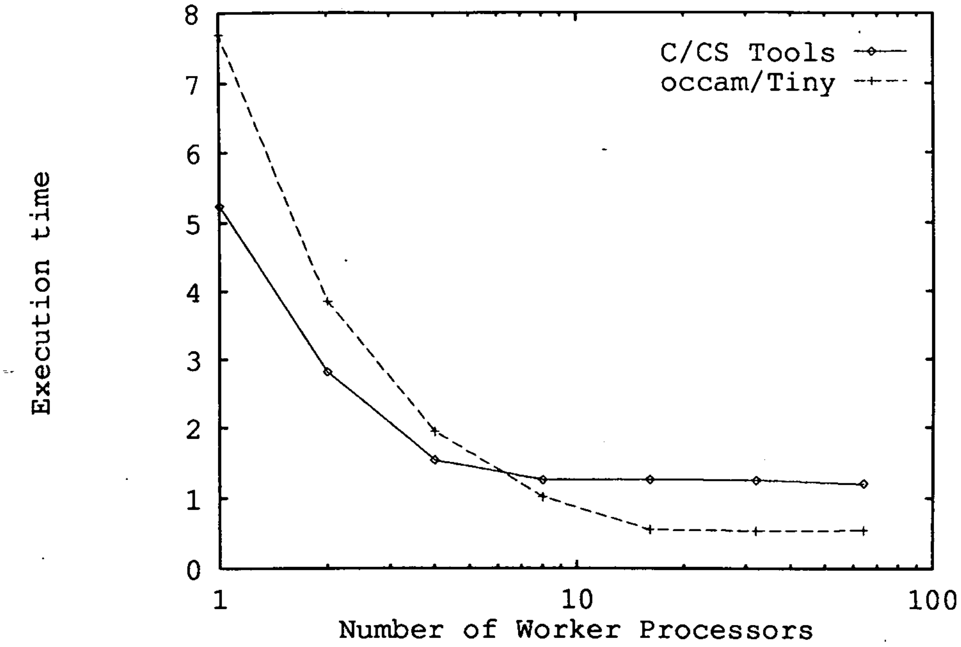


Figure 3.8: *Timings for H.261 simulation (without motion compensation) using a) C & CS Tools and b) occam & Tiny.*

Though the C serial programs are considerably faster than the occam, *Tiny* is more efficient than CS Tools. The CS Tools simulation seems to saturate at about four workers. Whereas, with *Tiny*, gains continue to be made up to 16 workers. Approximately two frames can be processed in one second using 16 workers. This is a fifth of real-time. The simulation is clearly communications bound at this point and so faster execution speed could only be achieved by adding more communications bandwidth.

The efficiency of a parallel program can be defined as follows

$$Efficiency = \frac{T_{calc}}{T_{calc} + T_{comms}} \leq 1 \tag{3.2}$$

where T_{comms} is the non-overlapped communication time and T_{calc} is the total computation time, reflecting that any *inefficiency* is due to the communication overheads. Thus it is desirable to minimise non-overlapped communication and maximise the overlap of communication with computation which the transputer can provide. Section 3.6 described how the implementations achieved these aims. An equivalent definition of efficiency which does not require communication and computation times to be

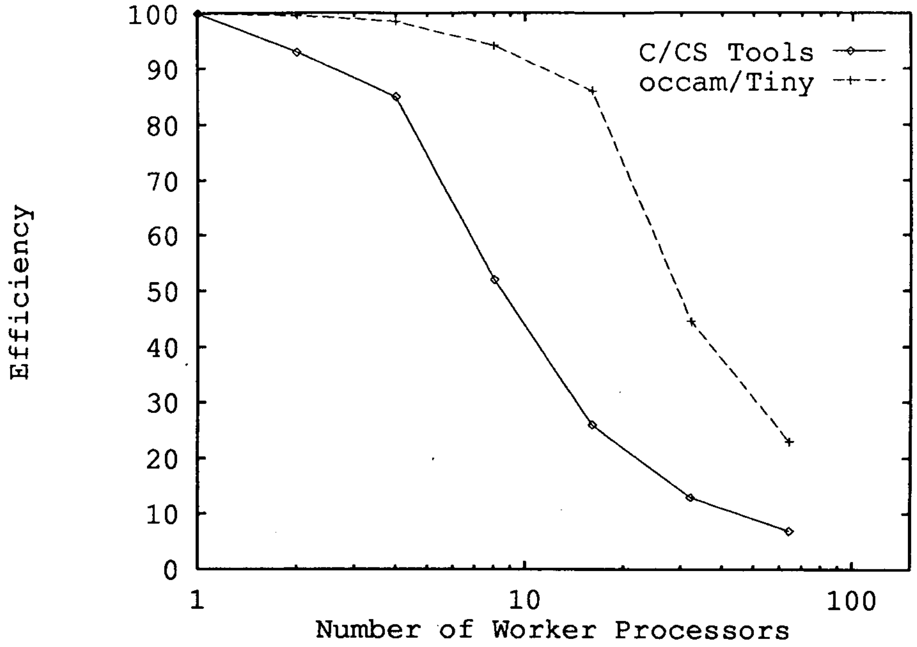


Figure 3.9: *Efficiencies for H.261 simulations (without motion compensation) using a) C & CS Tools and b) occam & Tiny.*

calculated is

$$Efficiency = \frac{T(1)}{nT(n)} \equiv \frac{Speedup(n)}{n} \leq 1 \qquad (3.3)$$

where $T(n)$ is the total execution time for n processors and $T(1)$ is the execution time for a single processor. Figure 3.9 shows efficiency curves for the simulation timings shown in figure 3.8. It is clear how much more efficient *Tiny* is than CS Tools. The former maintains 50% efficiency for up to 30 worker PEs, whereas the latter can only manage that for 10 workers PEs.

3.9 Summary and conclusions

Difficulties in simulating a complex image compression algorithm in real-time have been discussed. The evolution of the implementation of the simulations has been discussed from the use of occam with the *Tiny* routing harness, through C with Meiko's CS Tools. Clearly there is great demand for tools which will automatically and efficiently parallelise sequential algorithms and existing serial programs.

The advent of topology independent message routers means that programmers are no longer tied to simple processor topologies with simple message passing protocols. Nor do they have to worry about rewriting communications software for each new application. Instead a highly optimised software routing harness can be written once and for all. The router can be topology independent and hence the programmer is free to experiment with any processor network topologies. The next phase will be to relinquish the handling of communications completely to a hardware packet switching network.

The results presented in this chapter demonstrate that although close-to-real-time simulation can be achieved using 20 or more transputers the execution speed of the simulations is bounded by communications overheads. However, using compact graph topologies it was found possible to maintain 50% efficiency for up to 30 worker transputers. The simulations performed approximately an order of magnitude faster than the preceding work by Sexton using chain topologies.

Perhaps the most important discovery is that for small numbers of transputers (≤ 20), provided all available link bandwidth is used, one topology performs as well as any other. It is only when larger numbers of PEs are utilised that the differing scaling properties come into play. In the past this fact has been hidden for two main reasons:

- Efficient topology independent routing harnesses have not been available and so programmers have restricted themselves to simple topologies such as chains and trees which do not use all the available link bandwidth.

- Most transputer research has been conducted using either small numbers of transputers, or else larger numbers in a fixed topology (see, for example, the conference proceedings of Transputer Applications '91 [52]). The work presented here has concentrated on parallelism with *large* numbers of PEs.

All three of the compact topologies experimented with are very easy to generate for any number of PEs; they provide a practical means of interconnecting processors having four interprocessor links and are incrementally extensible. For large numbers of PEs both 'greedy' and random graphs perform well. The chordal ring does not scale well, and since routing algorithms are no longer a problem there is no reason to favour *regular* processor topologies. Irregular graphs have the lowest inter-node distances and scale well, but for up to 100 PEs, all the graphs perform virtually as well as each other. Thus it can be concluded that for machines with numbers of PEs of this order, *topology* is no longer such a burning issue and efforts should be concentrated on automating the *mapping* of the problem to the PEs.

Finally we conclude that although the results presented here are for an implementation of H.261 on a transputer surface many of the strategies presented apply equally to any problem and any MIMD machine. In particular, these methods will apply to new generations of DSP processors which include high speed communication links such as the Texas Instruments C40.

Chapter 4

Motion estimation and compensation for image coding

4.1 Introduction

Motion estimation is a powerful tool for the compression of image sequences. Indeed, many image sequences contain much more temporal redundancy than spatial redundancy. For example, if an object in a scene is moving with constant uniform velocity, parallel to the focal plane of the camera, once the velocity vector of the object is known no further information need be transmitted. Clearly, this an over-simplification and in many 'real' image sequences the motion of objects within the scene is far more complex. Nevertheless, considerable temporal redundancy is present in many image sequences.

This chapter reviews different algorithms for estimating motion in image sequences. Motion-compensated prediction is then discussed in detail. Theoretical models of the prediction error are developed and compared to empirical results. The theoretical models are then used to estimate the coding gain of an optimum hybrid motion-compensated codec. Finally, various hybrid algorithms are discussed.

4.2 Motion estimation

Motion within an image sequence can be divided into two categories: *global* motion caused by camera motion such as zoom and pan and *local* motion caused by the motion of individual objects within the scene. Estimation of motion is, in general, difficult since both local and global motion may occur simultaneously; all the objects within the scene may move independently, undergo deformations and occlude each other. In general, no *a priori* information concerning the structure of the objects within a scene is available, so that motion estimation is a problem of estimating the unknown displacement of unknown objects.

To enable the estimation of motion within an image simplifying assumptions are usually made about the type of motion which is present. The most common assumptions are that motion is restricted to be parallel to the focal plane of the camera and there is no change of ambient illumination. If no occlusion occurs then a pixel at position $\mathrm{x} = (x, y)^{\mathrm{T}}$ is displaced by $\mathrm{d}(\mathrm{x}) = (d_x(x, y), d_y(x, y))^{\mathrm{T}}$. Thus a point in an image at time t is related to point in the image at time $(t - \Delta t)$ by,

$$f(\mathrm{x}, t - \Delta t) = f(\mathrm{x} - \mathrm{d}(\mathrm{x}), t) \qquad (4.1)$$

In practice, occlusion will occur in a scene and as such equation (4.1) will only apply to disjoint subsets of the image.

4.2.1 Pel-recursive motion estimation

Pel-recursive algorithms estimate the motion of each pixel (pel) independently using a steepest gradient algorithm to minimise the displaced frame difference. The first recursive motion estimation algorithm was introduced by Netravali and Robbins [53] in 1979. The algorithm assumes that the image contains only translational motion so that a pixel in the t^{th} frame is equivalent to one in the $(t - 1)^{\mathrm{th}}$ frame offset by a displacement vector,

$$i(\mathrm{x}, t) = i(\mathrm{x} + \mathrm{d}, t - 1) \qquad (4.2)$$

The displaced frame difference (DFD) is defined to be,

$$\mathrm{DFD}(\mathrm{x}, \hat{\mathrm{d}}_i) = i(\mathrm{x}, t) - i(\mathrm{x} - \hat{\mathrm{d}}_i, t - 1) \tag{4.3}$$

It measures the intensity difference between two pels in the current and previous frame located at x and $\mathrm{x} - \hat{\mathrm{d}}_i$ respectively. The choice of the next vector is determined by minimising the square of the displaced frame difference (DFD) using a steepest gradient algorithm.

The algorithm uses an initial displacement estimate $\hat{\mathrm{d}}_i$ to produce a better displacement estimate $\hat{\mathrm{d}}_{i+1}$,

$$\hat{\mathrm{d}}_{i+1} = \hat{\mathrm{d}}_i + \mathrm{u}_i \tag{4.4}$$

where u_i is the iteration update term. If equation (4.3) is expanded as a Taylor series and only the first order term retained it can be shown that the iterative steepest gradient algorithm is given by,

$$\hat{\mathrm{d}}_{i+1} = \hat{\mathrm{d}}_i - \frac{1}{2}\epsilon\nabla_{\hat{\mathrm{d}}_i}[\mathrm{DFD}(\mathrm{x}, \hat{\mathrm{d}}_i)]^2 \tag{4.5}$$

where $\nabla_{\hat{\mathrm{d}}_i}$ is the gradient operator with respect to $\hat{\mathrm{d}}_i$ and ϵ is a positive constant which determines the rate of convergence. Several adaptations of the basic algorithm have been published which generally aim to decrease the time for convergence [54–56].

Pel-recursive algorithms give poor motion estimates at the boundaries of moving objects since the previous motion estimate is not a good estimate of the actual motion. Furthermore, motion estimates can easily be corrupted by noise. Kleihorst [57] has recently introduced a pel-recursive algorithm which is less sensitive to noise based upon the triple correlation of three images.

4.2.2 Block matching motion estimation

Block matching algorithms (BMAs) estimate motion by matching blocks of pixels between frames. The simplest BMAs arbitrarily choose blocks by dividing an image into a regular array of fixed sized blocks in which it is assumed that all pixels have uniform motion.

Blocks of pixels are matched by calculating a distortion measure between blocks. Common distortion measures include normalised cross-correlation $R(i,j,t)$ (equation (4.6)), mean squared error $\mathcal{E}(i,k,t)$ ($\alpha = 2$ in equ. (4.7)) and mean absolute error ($\alpha = 1$ in equ. (4.7)).

$$R(i,j,t) = \frac{\sum_{m=0}^{N-1}\sum_{m=0}^{N-1} i(x,y,t)i(x+i,y+j,t-1)}{\left[\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} i^2(x,y,t)\right]^{1/2}\left[\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} i^2(x+i,y+j,t-1)\right]^{1/2}} \tag{4.6}$$

$$\mathcal{E}(j,k,t) = \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} |i(x,y,t) - i(x+j,y+k,t-1)|^{\alpha} \quad \alpha \in \{1,2\} \tag{4.7}$$

The estimated motion vector for a given block is found by searching for the maximum value of cross-correlation or the minimum mean square error or minimum mean absolute error between blocks. The mean squared error and mean absolute error are generally used in preference to the normalised cross-correlation simply because they require less computation.

Reduced computation search algorithms

An exhaustive search of an image is guaranteed to find the best match between a pair of blocks but is computationally very expensive. If the search for the best fit between a pair of blocks is limited to a distance of $\pm d_{\max}$ pixels from the origin, a complete search of all possible displacements in this area requires $(2d_{\max} + 1)^2$ calculations of a distortion measure. Thus a search of the whole image is impractical and as such $d_{\max}$ is usually not greater than 15 pixels.

The number of search positions needed to find the best match between a pair of blocks can be significantly decreased by calculating a distortion measure for a small subset of the total $(2d_{\max} + 1)^2$ search positions and then using this information to predict the position of the absolute minimum.

To decrease the total computation in making a displacement estimate Jain and

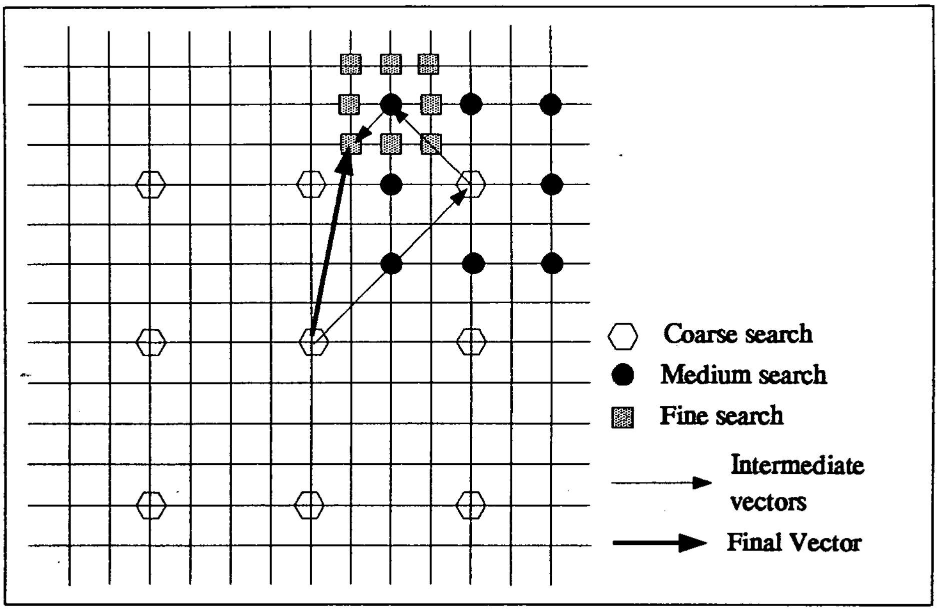


Figure 4.1: *2-D logarithmic search procedure.*

Jain [58] use a two-dimensional logarithmic search. This algorithm uses a three stage iterative search in which the resolution of the search is increased at each stage until it finally uses a full resolution search in a 3 × 3 pixel window (see figure 4.1).

It can be shown [58] that the if the error surface is monotonically increasing away from the absolute minimum then a two-dimensional logarithmic search will converge to this value. If this condition is not satisfied, the algorithm may converge to a local minima on the error surface.

The conjugate direction search is another important reduced computation search [59] in which motion estimates are made first along the x-axis and then along y-axis, until a minimum/maximum of the distortion measure is found in both directions.

Vector accuracy and block size

The accuracy of the estimated displacement vector depends upon the size of block used. Generally, larger blocks give more reliable displacement estimates since it is unlikely that there will be another similar block within the image. But as the block size becomes progressively larger it is unlikely that all pixels within the block will be uniformly displaced, and as such the derived displacement vector will only be accurate for some of the pixels within the block.

To overcome this problem variable resolution algorithms can be employed which make motion estimates for varying block sizes. Bierling [60] proposes a hierarchical block matching scheme in which large blocks are used to make a coarse displacement estimate and then smaller blocks are used to refine the initial estimate. The final motion estimate is given by summing the vector estimates for each level of the hierarchy.

An alternative algorithm suggested by Chan [61] adaptively varies the block size by initially making motion estimates with a large block. If the mean square error is greater than a threshold value the block is split into two smaller blocks and the search is continued. This splitting of the block is repeated until either an adequate motion estimate has been made or a minimum block size is reached. Chan's algorithm has the advantage of reducing the overhead information required to transmit the vectors by only using small blocks when they are required.

Sub-pixel motion estimation

The block matching algorithms described previously make motion estimates which are restricted to an integer pixel grid, in reality the displacement of the objects within an image will be over non-integer pixel distances. Motion estimates to a fractional pixel accuracy can be made by using interpolation to estimate the pixel intensities at non-integer grid points (see section 4.4.1). Motion estimation to fractional pixel displacements further increases the computational load required to make a motion estimate.

4.2.3 Image pyramids for motion estimation

Wang and Clarke [62, 63] describe a hierarchical motion estimation algorithm using "image pyramids" to overcome some of the deficiencies of pel-recursive algorithms. A pyramid is constructed from successively down-sampled versions of the original image. The down-sampling is such that a pixel on any level (except the base level) is an average of an $n \times n$ block of pixels on the previous level (see figure 4.2). A pixel on the l^{th} level of the pyramid at time t has an intensity of,

$$i_l(j_l, k_l, t) = \frac{1}{n^2} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} i_{l-1}(j_{l-1} + x, k_{l-1} + y, t) \tag{4.8}$$

Motion estimates are made by constructing a pyramid for both the current and previous image. Pixels are then matched on each level of the pyramid using the displacement estimate from the previous level as an initial displacement estimate for the current level. Wang and Clarke create a pyramid by averaging 4 × 4 pixel blocks to produce a pyramid with 3 levels. Thus the base level of a pyramid is a full resolution image and the higher levels are successively reduced resolution copies of the original image. Pixels are matched within a 3 × 3 pixel area on the top level and within a 5 × 5 pixel area on the middle and bottom levels of the pyramid allowing a maximum motion estimate of 26 pixels.

The image pyramid has superior motion compensation performance compared to current pel-recursive algorithms since it is robust to noise and can compensate for displacements of up to 26 pixels/frame. Unfortunately the averaging of pixels produces ambiguities in the motion estimates which if not carefully tracked can result in poor motion estimates.

Image Pyramids for block matching

The image pyramid can also be used for block matching; as described in the previous section the pyramid is constructed from successive averaging of pixels. Motion estimates are then made by making a displacement estimate at each level of the pyramid. In contrast to Wang and Clarke's scheme the size of the block to be matched is varied from level to level so that it is equivalent to a $N \times N$ block on the base level of the pyramid.

Incorrect motion estimates at the higher levels of the pyramid propagate through to the lower levels, which can result in mean absolute errors and mean square errors which are worse than those given by frame differencing. This problem can be overcome by comparing the minimum MSE for each level against the frame difference MSE and

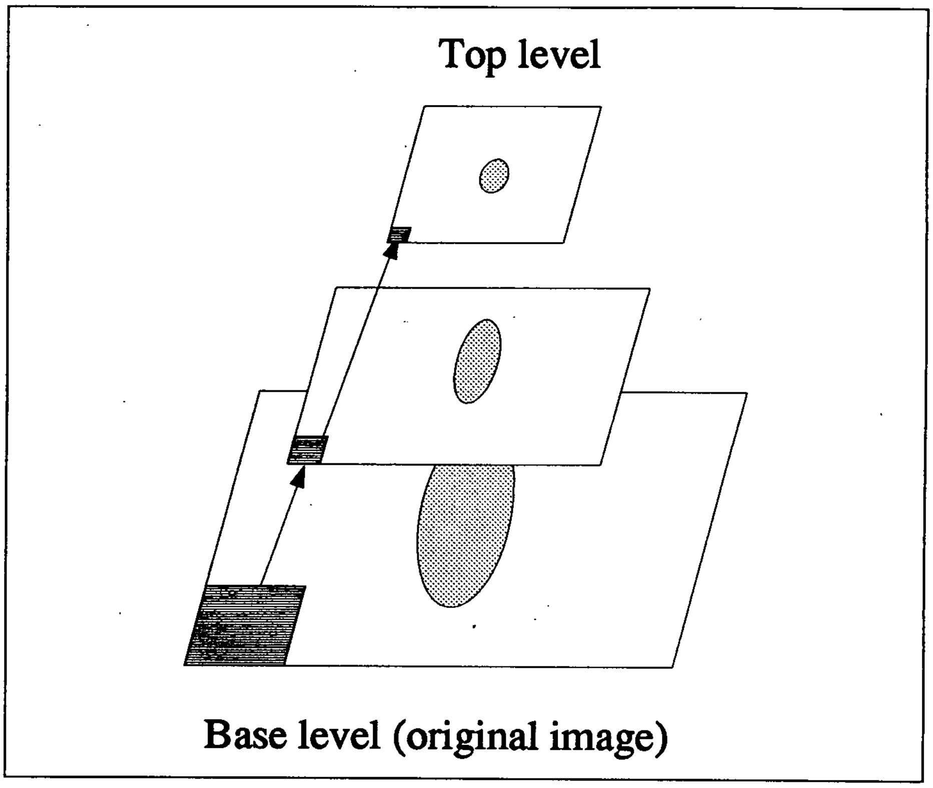


Figure 4.2: *An image pyramid for motion compensation.*

then restarting the search every time the frame difference error is smaller. This ensures that the pyramid algorithm will always produce mean square errors which are less than or equal to those given by frame differencing.

Further improvement can be achieved by comparing the MSE of the best fit on the previous level with the MSE of the current best fit. If the MSE on the current level is less than to equal the MSE on the previous level the search is continued otherwise it is reset on the current level. This procedure ensures reasonable displacement estimates, but as with Wang's original scheme averaging pixels results in ambiguities in the displacement estimates which cannot be easily resolved.

4.2.4 Phase-correlation

An alternative method of calculating block motion is to use phase-correlation [9,64–67]. If two images or sub-images are related by pure displacement their Fourier transforms are related by a phase shift. Thus for a displacement d = (d_x, d_y),

$$\frac{I(\omega_x, \omega_y, t)I(\omega_x, \omega_y, t-1)^*}{|I(\omega_x, \omega_y, t)I(\omega_x, \omega_y, t-1)|} = \exp[-j(\omega_x d_x + \omega_y d_y)] \qquad (4.9)$$

where $I(\omega_x, \omega_y, t)$ and $I(\omega_x, \omega_y, t-1)$ are the two-dimensional Fourier transforms of $i(x, y, t)$ and $i(x, y, t-1)$ respectively. The inverse Fourier transform of equation (4.9) produces a finite impulse at the displacement. If the block contains objects with different displacements then phase correlation will produce a set of candidate vectors which must be must matched to objects in the block.

4.2.5 Other motion estimation algorithms

Block matching and pel-recursive motion estimation algorithms are only able to compensate for translatory motion. Some recently introduced algorithms attempt to overcome this deficiency by compensating for rotation and non-rigid deformations.

Li [68] and Seferidis [69] both employ motion estimation schemes based upon decomposing the image into polygons. The vertices of the polygons are then manipulated to estimate the motion. Li's scheme is based upon using triangles whereas Seferidis employs quadrilaterals.

Papadopoulos and Clarkson [70–72] employ a second order geometric transform to compensate for motion. The algorithm performs geometric transforms to stretch and warp the image.

The computational load imposed by these algorithms is far greater than that of block matching and pel-recursive algorithms, and as such they are unsuitable for real-time applications. However, these algorithms can be usefully employed for encoding pre-recorded material for television or video.

4.2.6 Estimation of global motion

In scenes which contain global motion, such as zoom and pan, estimating local motion is of limited use. Combining global motion and local motion estimates can increase compression ratios.

Accurately estimating local rotation and zoom local is very difficult, since the image must be accurately segmented into the regions which correspond to each type of motion. However, if rotation and zoom are present as global parameters all the pixels in the image undergo this motion and there is no need for segmentation. This allows zoom, rotation and pan to be estimated.

For example, Wu [73] has introduced an algorithm based upon a pel-recursive motion estimation using a second order Taylor expansion which is able to simultaneously estimate change of scale, pan and rotation. Other algorithms for global motion estimation include several which are based upon the Hough transform, see *e.g.* [74,75]

4.3 Motion-compensated interpolation

Motion-compensated interpolation (MCI) algorithms transmit only one frame in every two or three. The missing frames can then be reconstructed by simple frame repetition or linear interpolation, both these methods have been shown [76] to introduce visual impairment in the reconstructed image sequence. MCI attempts to minimise such distortion by using estimated displacement vectors to reconstruct the missing frame(s).

The accuracy of the estimated motion vectors is critical to the quality of the reproduced frames, since inaccurate motion vectors can produce a visually disturbing rendition of the frame-to-frame motion.

Motion compensated interpolation algorithms are invariably complex to implement, requiring that the frame be segmented into moving areas, stationary areas, uncovered and covered background. Thoma [77] implements such a coder using Bierling's hierarchical block matching algorithm (see section 4.2.2).

4.4 Motion-compensated prediction

Motion-compensated prediction (MCP) is adaptive DPCM in the temporal dimension. Figure 4.3 shows a schematic diagram of a typical motion-compensated predictor. The encoder predicts the contents of the current frame based on the contents of a previously encoded frame or frames using motion estimates. The difference between the current frame, $i(x,y,t)$ and the prediction frame, $\hat{\imath}(x,y,t)$, is known as the prediction error or the motion-compensated difference frame, *i.e.*

$$e(x,y,t) = i(x,y,t) - \hat{\imath}(x,y,t) \tag{4.10}$$

The prediction error and optionally the estimated motion vectors are transmitted to the receiver. If the motion estimation algorithm is working *well* little or no error information need be transmitted to the receiver.

In contrast to MCI the accuracy of the motion estimate is not important; rather, the sole purpose of the motion estimator is to minimise the magnitude of the prediction error. The resulting prediction error has a much smaller variance than the original image and as such it can be encoded at a lower rate for a given value of distortion. For example, for memoryless encoding of a signal with a Gaussian statistics the rate distortion function, $R(D)_G$, is given by,

$$R(D)_G = \max\left\{0, \frac{1}{2}\log_2 \frac{\sigma^2}{D}\right\} \tag{4.11}$$

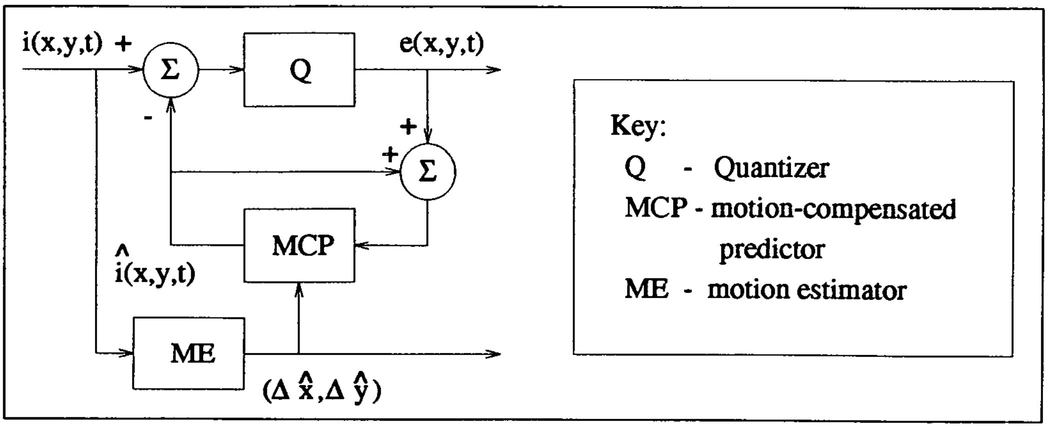


Figure 4.3: *Schematic diagram of MCP coder.*

where R is the rate, D is mean square error and σ^2 is the variance of the image. Clearly, a reduction in variance results in a reduction in rate. Prediction errors do not, in general, have a Gaussian pdf in which case equation (4.11) can be beneficially interpreted as the upper bound to the minimum rate at which the data can be encoded, *i.e.* $R(D) \leq R(D)_G$.

The energy of an image, J, is defined to be the sum of the square of the pixel intensities,

$$J = \sum_{x,y} i(x,y,t)^2 \tag{4.12}$$

Mean energy and variance are equivalent for zero mean signals and as such energy can be used to predict the potential gains from motion-compensated prediction.

In the following sections energy and entropy will be used to demonstrate the potential gains given by motion-compensated prediction. Block matching algorithms will be used throughout to demonstrate these results, although many of the results derived apply equally to other motion estimation algorithms. Comparisons between pel-recursive and block matching algorithms can be found in [63].

4.4.1 Block matching motion-compensated prediction

Figure 4.4 shows the energy of the original uncoded frames of the Miss America standard image sequence, the energy of the difference frames and the energy of the motion compensated difference frames using the motion estimates from an exhaustive

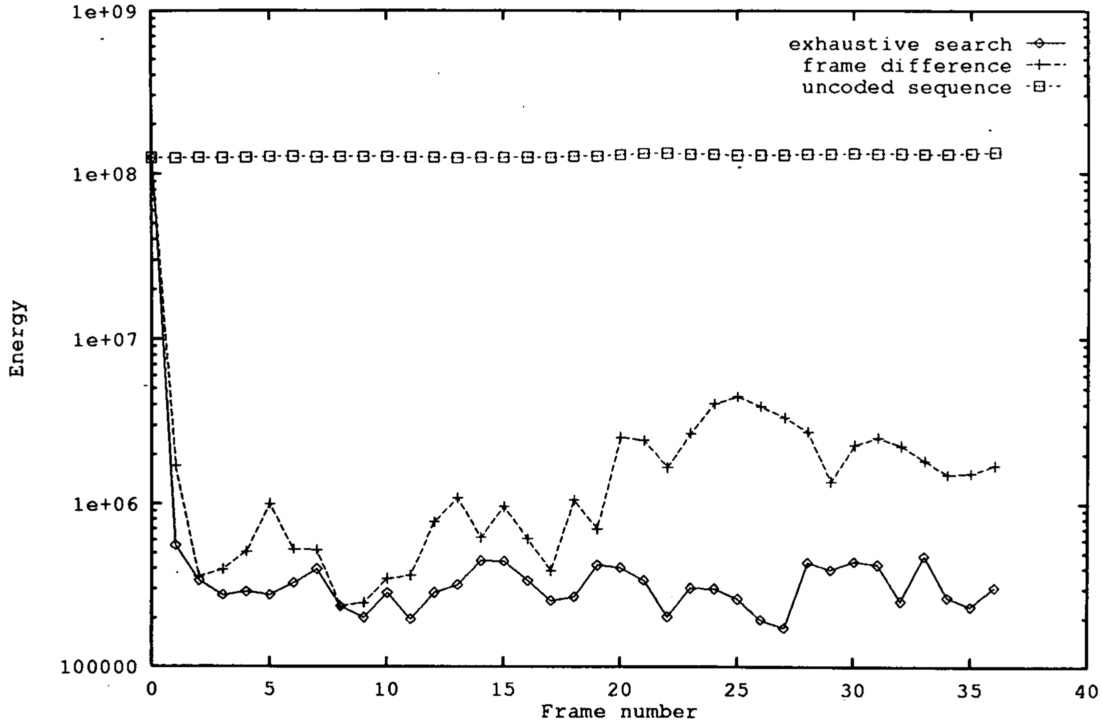


Figure 4.4: *Energies of the frame differences, motion-compensated frame differences and the original images for the Miss America standard image sequence (block size* 16 × 16 *pixels, max. displacement = 15 pixels).*

search for a block size of 16 × 16 pixels. The frame difference results are included to demonstrate the efficiency of the motion compensation algorithm. Clearly, both the difference frames and the motion compensated difference frames show a remarkable decrease in energy compared to the original uncoded sequence. In figure 4.4 the motion compensated difference frames contains on average less than 0.6% of the original image energy.

The frame to frame motion in the Miss America sequence increases as the sequence progresses. This trend is reflected in the frame difference energies which also increase with frame number. In contrast, the motion-compensated difference frame is relatively flat showing that the block matching algorithm compensates for the motion between frames.

Figure 4.5 shows the distribution of pixel intensities in the motion compensated frame differences of the Miss America standard image sequence. The distinctive dominant central peak of the distribution is typical for all motion prediction errors. In

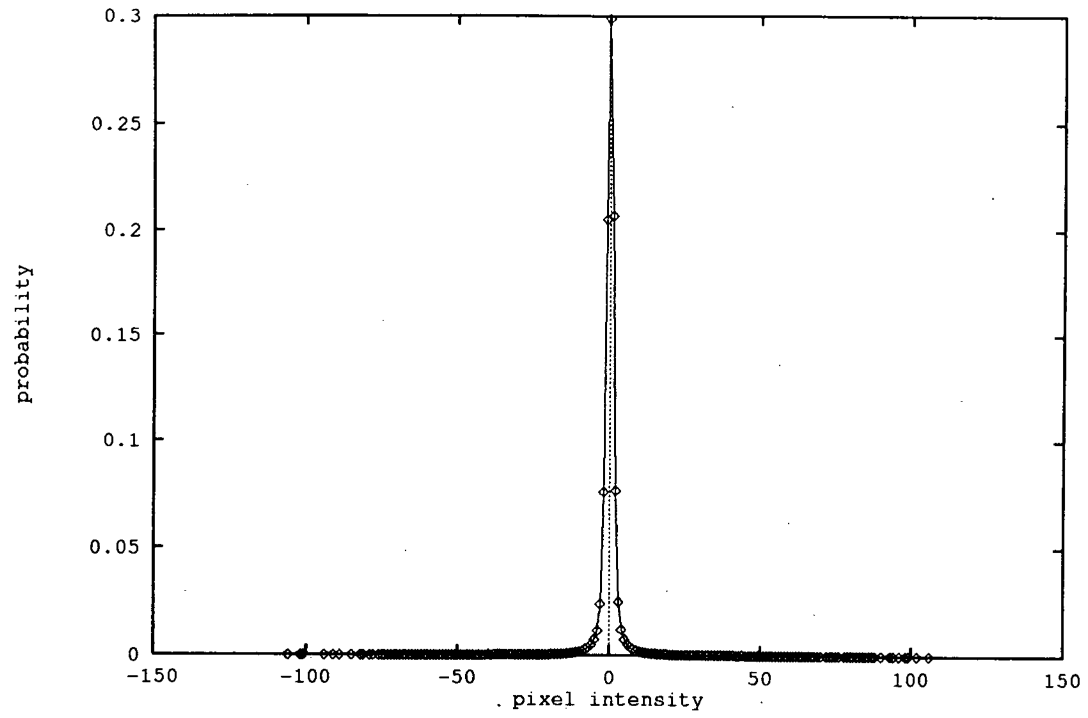


Figure 4.5: *Distribution of pixel intensities in the motion compensated frame differences of the Miss America standard image sequence (block size = 16 × 16, max. displacement = 15 pixels).*

figure 4.5 more than 90% of the pixels have an intensity in the range -10 to 10.

The rate at which an image can be encoded without distortion is given by its entropy (see section 2.3). Figure 4.6 compares the entropy of the original image sequence, the frame difference and the motion compensated frame differences using exhaustive search for the Miss America standard image sequence. The motion compensated frame differences show a reduction in entropy of approximately 40-50%, compared to a reduction in entropy of only 30% for the ordinary difference frame. As predicted, reducing the energy of a signal results in a decrease in rate. Also note that the entropy data shows similar trends to the energy data but the reduction in entropy is considerably less since the rate required to encode an image is proportional to the logarithm of the variance (see equation 4.11).

Variable length codes, such as Huffman codes, would in general not be generated for each frame since this would require a code book to be transmitted for each image. Instead, a code would be generated for the average statistics of a sequence and as such

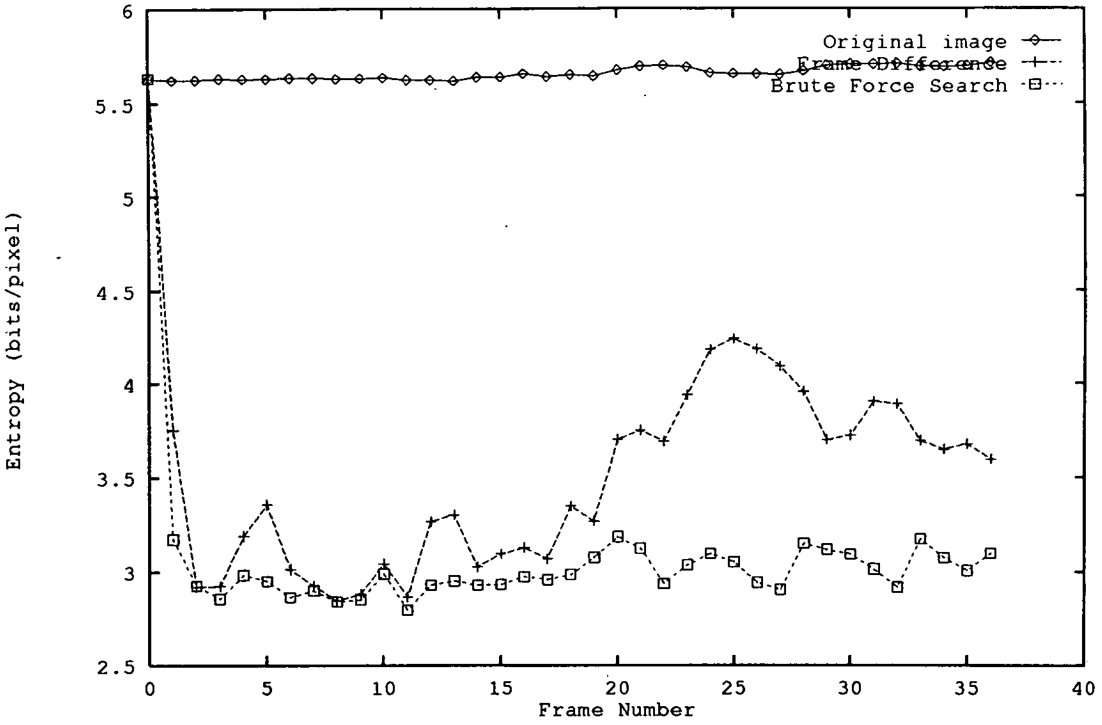


Figure 4.6: *Entropies of the frame differences, motion-compensated frame differences (block size* 16 × 16 *max. displacement = 15 pixels) and the original images for the Miss America standard image sequence.*

entropy results will be quoted for entire sequences rather than single frames in the following sections.

Block size

MCP simply aims to minimise the energy of the error signal and as such the accuracy of the displacement estimate is not deemed to be important. A small block size gives greater energy reduction than could be obtained from using a large block size, but reducing the block size also increases the number of vectors which need to be transmitted. The choice of block size is thus a compromise between the compression achieved by MCP versus the extra bandwidth required to transmit the motion vectors.

For a maximum displacement $\pm d_{\text{max}}$ pixels there are $(2d_{\text{max}}+1)^2$ search positions. Hence, if the motion vectors are PCM encoded, an upper bound to the minimum rate required to transmit the motion vectors, r_v, for a block size of $N \times N$ pixels is given by,

$$R\text{vec} = \left\lceil \frac{2}{N^2} \log_2(2d_{\max} + 1) \right\rceil \quad \text{bits/pixel} \tag{4.13}$$

where $\lceil \cdot \rceil$ denotes the nearest integer which is greater than $\cdot$.

Table 4.1 compares the entropies of the motion prediction errors of three standard image sequences, Miss America (Miss A), Clare and Salesman (Sale) for various block sizes with the maximum bandwidth required to transmit the motion vectors for that block size. Clearly, reducing the block size reduces the entropy of the motion prediction errors but for the 4 × 4 and 8 × 8 blocks the increase in rate required to transmit the motion vectors is greater than the reduction of entropy in the prediction errors. Note, that the derived rates for the transmission of the motion vectors assume the vectors are PCM coded, this rate can be further reduced by exploiting statistical redundancies in the vector information.

Block size	Max. motion vector rate (bits/pixel)	Entropy (bits/pixel)		
		Miss A	Clare	Sale
4 × 4	0.625	2.616	2.444	3.283
8 × 8	0.156	2.893	2.673	3.358
16 × 16	0.039	3.015	2.764	3.383

Table 4.1: *Entropies of the motion prediction errors of three standard image sequences, Miss America, Clare and Salesman (Sale) for various block sizes compared to maximum bandwidth required to transmit the vectors for that block size.*

Reduced computation searches

In section 4.2.2 reduced computation searches were introduced which substantially decrease the computation needed to make a motion estimate compared to that of an exhaustive search. These searches are prone to finding local minima on the error surface and as such are not guaranteed to give the maximum energy minimisation.

Figure 4.7 shows the energies of the prediction errors given by a logarithmic search and a exhaustive search. The superiority of the exhaustive search can be seen from the higher frame numbers which contain larger motion. Table 4.2 compares the entropies of the motion prediction errors of an exhaustive search and logarithmic search for

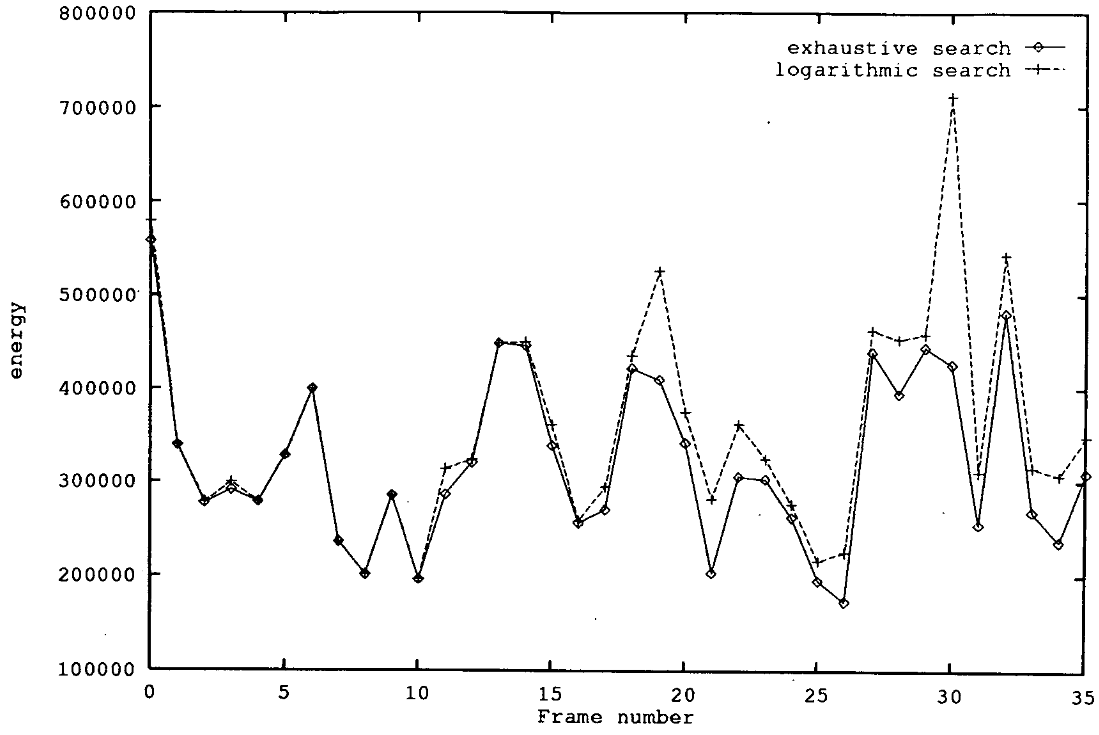


Figure 4.7: *Energies of the motion prediction errors of the Miss America image sequence for a logarithmic and exhaustive search. The results for the exhaustive search are the same as those shown in figure 4.4.*

Search	Entropy (bits/pixel)		
	Miss A	Clare	Sale
Exhaustive	3.015	2.764	3.383
Logarithmic	3.058	2.770	3.387

Table 4.2: *Comparison of entropies of the motion prediction errors of standard image sequences (block size* 16 × 16 *pixels).*

three standard image sequences. The exhaustive search gives entropies which are 1-2% lower than those given by logarithmic search for an increase in computation of approximately 270%. Although this appears to make the exhaustive search of little practical significance it is used in hardware implementations [78] since it does not require that intermediate search positions be stored.

Sub-pixel motion estimation

Sub-pixel motion estimation was previously introduced as method of refining the accuracy of the motion estimate. In the context of MCP sub-pixel motion estimation is

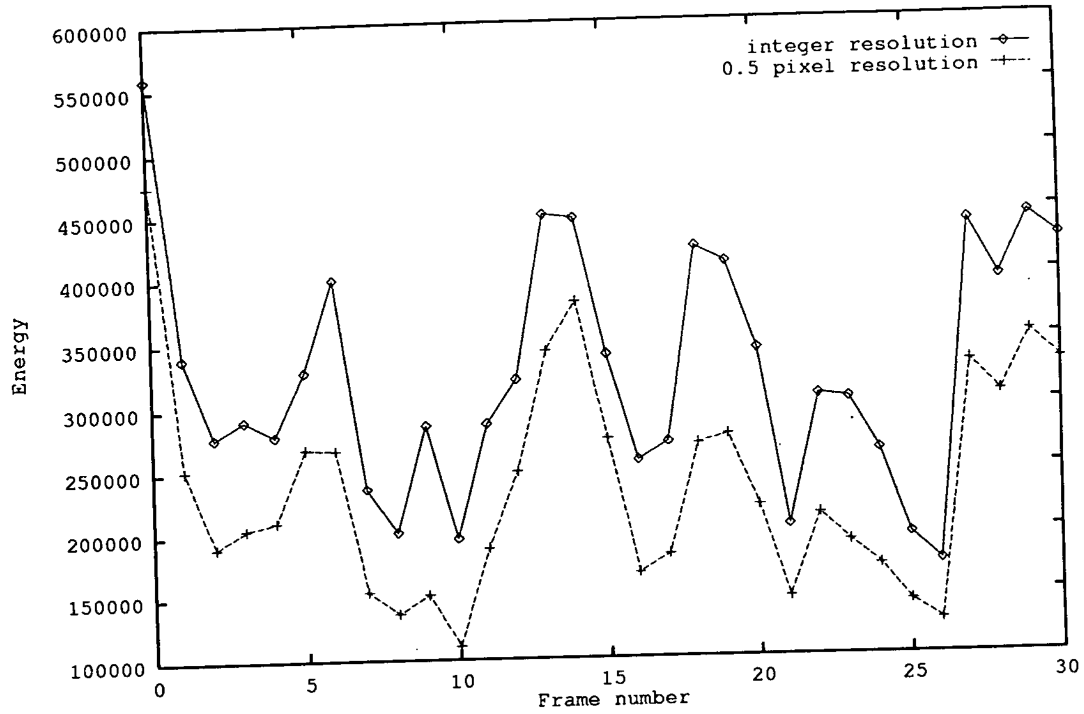


Figure 4.8: *Energy of the Miss America sequence for various search resolutions.*

Resolution	Entropy (bits/pixel)		
	Miss A	Clare	Sale
1	3.015	2.764	3.383
0.5	2.836	2.626	3.358
0.25	2.769	2.259	3.383

Table 4.3: *Comparison of data rates for sub-pixel motion estimation (block size* 16×16 *pixels).*

an effective tool for energy minimisation. Figure 4.8 shows the energy of the prediction error for two search resolutions.

Table 4.3 shows the entropies of the three standard image sequences over a wider range of search resolutions. The Miss America (Miss A) and Clare image sequences show a reduction in entropy of approximately 0.1-0.2 bit/pixel on progressing to 1/4 pixel resolution.

Motion compensation to non-integer pixel accuracy requires that extra bandwidth be used to transmit the motion vectors. For a resolution of 0.5 pixel, 2 bits per vector extra are required and for a search resolution of 0.25 pixel, an extra 4 bits per vector are required. This would imply an increase of bandwidth of $4/N^2$ bits/pixel and $8/N^2$ bits/pixel for a resolution of 0.5 and 0.25 pixels respectively. Thus for a block size of 16×16 pixels the increase in rates would be 0.016 bits/pixel and 0.031 bits/pixel which is below the decrease in entropy of the prediction errors implying that subpixel interpolation reduces the rate required to encode an image.

4.4.2 The prediction error

In the previous sections the performance of the motion compensation algorithms were quantified in terms of the energy and entropy of the prediction errors. Although both are useful for comparing the performance of various motion-compensated prediction algorithms they do not give any information about the spatial and spectral properties of the prediction errors, which are essential for developing an efficient intra-frame coder to exploit any remaining redundancy in the prediction error. In the following sections the prediction error is studied in detail.

Spatial properties

The spatial properties of the prediction error signal can be approximated using a simple motion model in which it is assumed that the motion of any pixel can be described by a displacement vector $(d_x(x,y,t), d_y(x,y,t))^{\mathrm{T}}$ and that there are no changes in ambient illumination. The displacement functions $d_x(x,y,t)$ and $d_y(x,y,t)$ are continuous functions which deform the image. Note, that this assumption is somewhat unrealistic since it does not include the possibility of objects being (un)covered or objects moving in or out of the scene. It is more accurate to describe the displacement functions, $d_x(x,y,t)$ and $d_y(x,y,t)$, as piecewise continuous functions over disjoint subsets of the image, where each subset corresponds to an object in the image.

A digital image, $i(a,b,n)$, is assumed to be formed from sampling a continuous image function[1], $f(x,y,t)$, with a lattice of unit impulse functions, $l(x,y,t)$, given by

$$l(x,y,t) = \sum_{a=0}^{X-1}\sum_{b=0}^{Y-1}\sum_{n=-\infty}^{\infty} \delta(x-a\Delta x)\delta(y-b\Delta y)\delta(t-n\Delta t) \tag{4.14}$$

where Δx, Δy and Δt are the sampling intervals in the x, y and t dimensions respectively. Thus the discrete image is given by,

$$i(a,b,n) = f(x,y,t)l(x,y,t) \tag{4.15}$$

Using this model the prediction error is attributed to errors in the displacement estimates, $(\Delta d_x(x,y,t), \Delta d_y(x,y,t))^{\mathrm{T}}$ defined by,

$$\Delta d_x(x,y,t) = d_x(x,y,t) - \hat{d}_x(x,y,t) \tag{4.16}$$

$$\Delta d_y(x,y,t) = d_y(x,y,t) - \hat{d}_y(x,y,t) \tag{4.17}$$

where $(\hat{d}_x(x,y,t), \hat{d}_y(x,y,t))$ is the estimated displacement vector. Note, that the displacement estimate is made in the discrete image and as such it should be discrete in both time and space but for mathematical simplicity it is assumed to be interpolated so that it is continuous.

The compensated image can be written in terms of the current image offset by the displacement error defined in equations (4.16) and (4.17), *i.e.*

$$f(x+\hat{d}_x(x,y,t), y+\hat{d}_y(x,y,t), t-\Delta t) = f(x-\Delta d_x(x,y,t), y-\Delta d_y(x,y,t), t) \tag{4.18}$$

If the image $f(x,y,t)$ has continuous partial derivatives then $f(x-\Delta d_x(x,y,t), y-\Delta d_y(x,y,t), t)$ can be expanded as a Taylor series to give,

$$f(x-\Delta d_x(x,y,t), y-\Delta d_y(x,y,t), t) = f(x,y,t) - \sum_{N=1}^{\infty}\frac{1}{N!}\left(\Delta d_x(x,y,t)\frac{\partial}{\partial x} + \Delta d_y(x,y,t)\frac{\partial}{\partial y}\right)^N f(x,y,t) \tag{4.19}$$

---

[1] Assuming Nyquist conditions

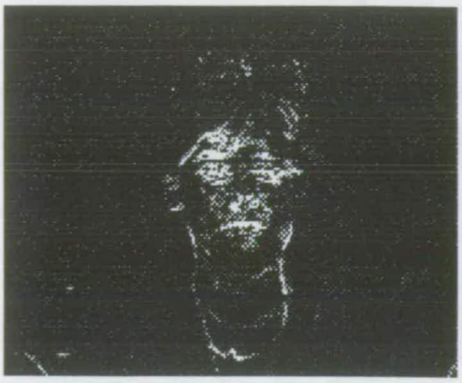


Figure 4.9: *A typical prediction error frame from the Miss America sequence.*

Thus the prediction error is given by,

$$e(a,b,n) = -l(x,y,t)\sum_{N=1}^{\infty}\frac{1}{N!}\left(\Delta d_x(x,y,t)\frac{\partial}{\partial x}+\Delta d_y(x,y,t)\frac{\partial}{\partial y}\right)^N f(x,y,t) \qquad (4.20)$$

From equation (4.20) it can be seen that the prediction error will contain high magnitude regions where the partial derivatives of the image are large, such as in textured regions or at image edges. Figure 4.9 shows a typical prediction error generated from the Miss America standard image sequence. The prediction error has high magnitudes in blocks which correspond to object edges which is consistent with the argument presented above.

Although equation (4.20) has been derived for a single displacement estimate per pixel it applies equally to block matching algorithms. The only difference being that a single motion estimate, (k_x, k_y), is made per $N \times N$ block of pixels and, as such, the definition of displacement estimate needs to be adjusted accordingly.

Equation (4.20) can be used to model the mean square error surface generated in a block matching search. The estimated displacement is defined to be,

$$\hat{d}_x(x,y,t) = k_x\Delta x \qquad (4.21)$$

$$\hat{d}_y(x,y,t) = k_y\Delta y \qquad (4.22)$$

where $k_x, k_y \in \{\mathbb{Z} | -d_{\max} \le k_x, k_y \le d_{\max}\}$.

The displacement errors are thus defined to be,

$$\Delta\hat{d}_x(x,y,t) = d_x(x,y,t) - k_x\Delta x \tag{4.23}$$

$$\Delta\hat{d}_y(x,y,t) = d_y(x,y,t) - k_y\Delta y \tag{4.24}$$

For a block with origin at (x', y') and which has dimensions of $X \times X$ pixels the mean square error surface is given by,

$$\mathcal{E}(k_x,k_y,n) = \frac{1}{X^2}\sum_{l=x'}^{X-1}\sum_{n=y'}^{X-1}\left(l(x,y,t)\sum_{N=1}^{\infty}\frac{1}{N!}\left(\Delta d_x(x,y,t)\frac{\partial}{\partial x} + \Delta d_y(x,y,t)\frac{\partial}{\partial y}\right)^N f(x,y,t)\right)^2 \tag{4.25}$$

The optimum estimated displacement for a block is given by the absolute minimum of the error surface.

Although equation (4.25) would not be used to generate an actual error surface it can be used to make some general statements about block matching. Firstly, if the partial derivatives of the image tend to zero, *i.e.* the image is flat, the error surface will also be flat. Under these circumstances the displacement estimate is easily corrupted by noise. Secondly, if two moving objects are separated by an edge the motion estimate will be more accurate for the region around the edge than the flat regions, even though conversely the magnitude of the prediction error is greater in the edge region. Physically this is expected since in these regions a small change in displacement produces a large contribution to the total error.

Spectral properties

A model of the prediction error was previously generated based upon a Taylor expansion of a simple image model. The spectral properties of the prediction error can be derived from the Fourier transform of this equation but, in general, the complexity of the model hinders analysis. In the following, the model used previously is further simplified and a model of the spectral properties of the prediction error is generated.

The model developed in the previous section assumes that the prediction error is generated from a failure to correctly estimate displacement vectors. The model allows for a displacement estimate per pixel and allows any pixel to move independently of any other pixel. To allow a model of the spectral properties of prediction error to be developed the displacement errors will be assumed to be constant across the whole image. Thus the prediction error is now given by,

$$e(a,b,n) = (f(x,y,t) - f(x - \Delta d_x(t), y - \Delta d_y(t), t))l(x,y,t) \qquad (4.26)$$

where $\Delta d_x(t)$ and $\Delta d_y(t)$ are the constant displacement errors.

If we assume that the images are wide sense stationary a closed form solution (see [79]) for the prediction error spectrum can be derived. Note, that the displacement errors vary from frame to frame and as such they are a function of time, in the following these displacement errors will be assumed to be stationary random variables. The two-dimensional Fourier transform of equation (4.26) is given by,

$$E(\omega_x, \omega_y, n) = F(\omega_x, \omega_y, t)\,(1 - \exp(j(\Delta d_x(t)\omega_x + \Delta d_y(t)\omega_y))) * L(\omega_x, \omega_y, t) \qquad (4.27)$$

where $F(\omega_x, \omega_y, t)$ is the two-dimensional transform of the continuous image function, $f(x,y,t)$, and $L(\omega_x, \omega_y, t)$ is the Fourier transform of the sampling lattice $l(x,y,t)$, and '$*$' is the convolution operator. Note, that $f(x,y,t)$ is assumed to be bandlimited. The function $L(\omega_x, \omega_y, t)$ leads to the well known result of baseband replications and the Gibbs phenomena [80].

The ultimate aim of developing a spectral model of the prediction error is to apply rate-distortion theory to estimate the coding gain for different scenarios. The rate-distortion function of a finite discrete image is asymptotically bounded by the rate-distortion function of a discrete image with infinite samples. In this case the Gibbs phenomena does not occur and only the baseband replications exist. Thus without loss of generality only the baseband need be considered.

From equation (4.27) the time dependent two-dimensional power spectrum of the prediction error, $S_{ee}(\omega_x, \omega_y, n)$, is thus given by,

$$S_{ee}(\omega_x, \omega_y, n) = S_{ii}(\omega_x, \omega_y, n)\sin^2\left(\frac{\Delta d_x\omega_x + \Delta d_y\omega_y}{2}\right) \qquad (4.28)$$

where $S_{ii}(\omega_x, \omega_y, n)$ is the two-dimensional power spectrum of the uncoded images.

Since the images are assumed to be related by uniform displacement with no (un)covered background, an image at time n has the same power spectrum as one at time $(n-n')$. That is, the power spectrum of the image sequences is independent of time. If the displacement errors are such that $\Delta d_x, \Delta d_y \in [-0.5, 0.5]$ and all displacement errors are equi-probable then the average two-dimensional prediction error spectrum is given by,

$$S_{ee}(\omega_x, \omega_y) = \frac{1}{2} S_{ii}(\omega_x, \omega_y) \left(1 - \frac{\sin(\omega_x/2)}{\omega_x/2} \frac{\sin(\omega_y/2)}{\omega_y/2}\right) \tag{4.29}$$

The average two-dimensional power spectrum of the prediction error signal is thus a high-pass filtered version of the two-dimensional spectrum of the original image sequence.

Equation (4.29) can be used to estimate the average energy gain due to motion compensation for various model spectra. The isotropic correlation function, $R_{ii}(x, y)$, given below is frequently used to model the spectrum of an image [81],

$$R_{ii}(x, y) = e^{-\alpha\sqrt{(x^2+y^2)}} \tag{4.30}$$

where $\alpha = -\log \rho$ and ρ is the correlation coefficient. The power spectrum of the above isotropic correlation function is given by,

$$S_{ii}(\omega_x, \omega_y) = \left(\frac{2\pi\alpha}{(\alpha^2 + \omega_x^2 + \omega_y^2)^{3/2}}\right) \tag{4.31}$$

Figure 4.10 shows the theoretical energy gain for an isotropic correlation function (solved using numerical methods). Real-world images are frequently modelled as having a correlation coefficient $\rho \simeq 0.9$, which gives an energy gain of approximately 0.7%. This agrees favourably with the experimentally measured gain in section 4.4.1 of 0.6%.

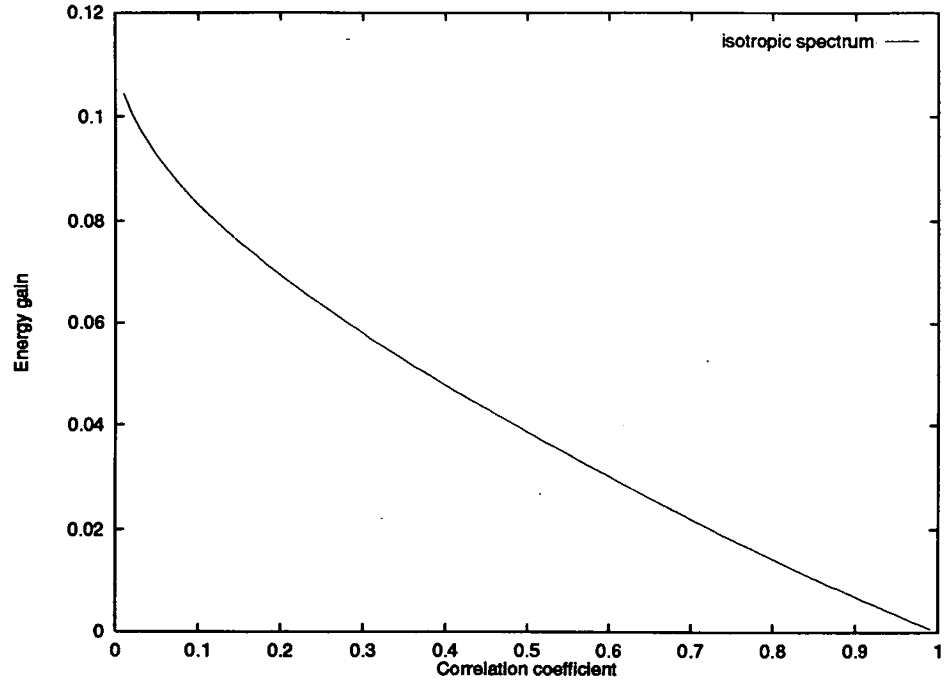


Figure 4.10: *Theoretical energy gain using an isotropic correlation function for various values of correlation (see equation 4.30).*

4.4.3 Hybrid-coding of the prediction error

Motion-compensated prediction is frequently combined with a spatial coding algorithm to remove any remaining redundancy from the prediction error. To provide the maximum compression for a given image fidelity the spatial encoder must exploit redundancy which is specific to the prediction error. A common problem, in many hybrid encoders, is that the spatial encoder is optimised for compressing real-world images and as such produces poor results when applied to the prediction error. In the following sections the benefits of spatial encoding the prediction error will be assessed.

The signal-to-noise coding gain of an optimum spatial encoder is asymptotically bounded by the inverse spectral flatness measure (SFM) of the signal, *e.g.* see [1](chapters 6 and 12). The spectral flatness measure, γ_x^2, of a signal is defined by,

$$\gamma_x^2 = \frac{\exp\left[\frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\log(S_{xx}(\omega_x,\omega_y))\,d\omega_x\,d\omega_y\right]}{\frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}S_{xx}(\omega_x,\omega_y)\,d\omega_x\,d\omega_y} \tag{4.32}$$

where $S_{xx}(\omega_x, \omega_y)$ is the power spectrum of the signal and $\gamma_x^2 \in [0, 1]$. SFM can be interpreted as a measure of the redundancy in a signal, for example, if $\gamma_x^2 = 1$ then the signal has a white spectrum and as such it is uncorrelated and contains no redundancy. The architecture of a hybrid motion-compensated encoder differs considerably from a standard spatial encoder and as such it is necessary to further investigate this to demonstrate that the above result is still true.

The parametric rate, $R(\phi)$, and distortion, $D(\phi)$, functions of a Gaussian source with a power spectrum, $S_{xx}(\omega_x, \omega_y)$, are given by,

$$D(\phi) = \frac{1}{(2\pi^2)} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \min\{\phi, S_{xx}(\omega_x, \omega_y)\}\, d\omega_x\, d\omega_y \tag{4.33}$$

$$R(\phi) = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \min\left\{0, \frac{1}{2}\log_2 \frac{S_{xx}(\omega_x, \omega_y)}{\phi}\right\} d\omega_x\, d\omega_y \tag{4.34}$$

The optimal mapping [1](Appendix D) which achieves the rate-distortion bound of equations (4.33) and (4.34) is given by the application of a non-ideal filter (equation (4.35)) followed by the addition of additive bandlimited noise (equation (4.36)).

$$H(\omega_x, \omega_y, \phi) = \max\left\{0, 1 - \frac{\phi}{S_{ee}(\omega_x, \omega_y)}\right\} \tag{4.35}$$

$$S_{nn}(\omega_x, \omega_y, \phi) = \max\left\{0, \phi\left[1 - \frac{\phi}{S_{ee}(\omega_x, \omega_y)}\right]\right\} \tag{4.36}$$

Figure 4.11 shows a schematic diagram of an optimum motion-compensated hybrid architecture. The optimum spatial coder is given by the optimum forward channel and $G(\omega_x, \omega_y, t)$ is simply a phase shift (section 4.4.2), *i.e.*

$$G(\omega_x, \omega_y, t) = e^{-j(\omega_x d_x(t) + \omega_y d_y(t))} \tag{4.37}$$

It can easily be shown that the power spectrum of the prediction error, $S_{ee}(\omega_x, \omega_y)$, is given by,

$$S_{ee}(\omega_x, \omega_y) = \mathrm{E}\left[\left|\frac{1 - G(\omega_x, \omega_y, t)}{1 - H(\omega_x, \omega_y, \phi) + H(\omega_x, \omega_y, \phi)G(\omega_x, \omega_y, t)}\right|^2\right] S_{ii}(\omega_x, \omega_y)$$

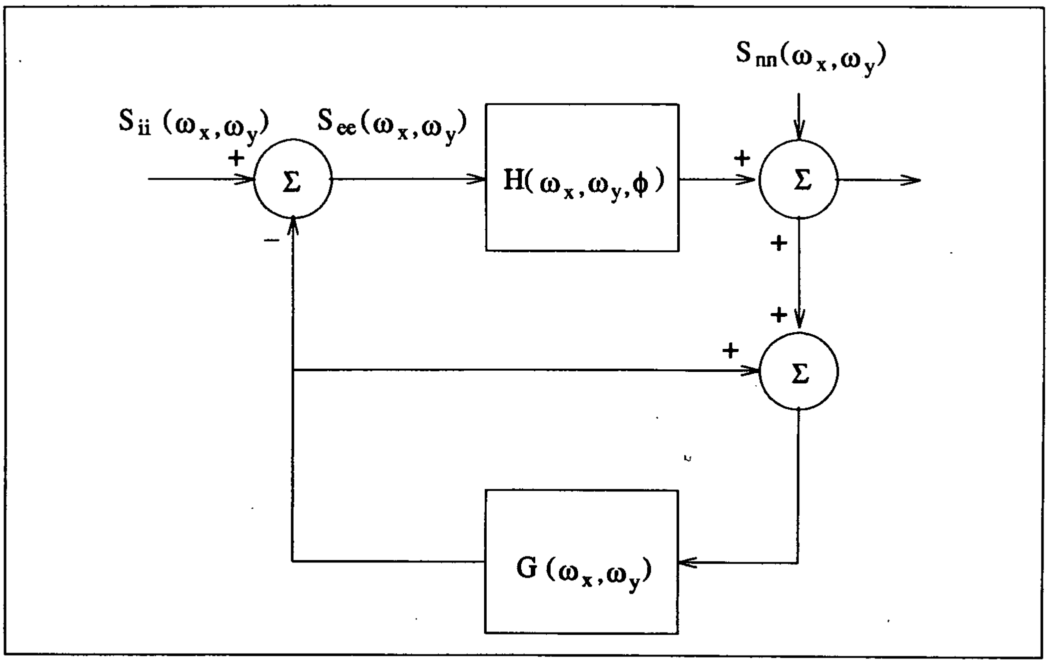

$S_{nn}(\omega_x,\omega_y)$
$S_{ii}(\omega_x,\omega_y)$
$S_{ee}(\omega_x,\omega_y)$
+
Σ
−
$H(\omega_x,\omega_y,\phi)$
+
Σ
+
+
+
Σ
$G(\omega_x,\omega_y)$

Figure 4.11: *Schematic diagram of motion-compensated prediction algorithm with an optimum spatial encoder.*

$$-\mathrm{E}\left[\left|\frac{G(\omega_x,\omega_y,t)}{1-H(\omega_x,\omega_y,\phi)+H(\omega_x,\omega_y,\phi)G(\omega_x,\omega_y,t)}\right|^2\right]S_{nn}(\omega_x,\omega_y) \quad (4.38)$$

where E[·] denotes the expectation operator with respect to the displacement errors.

If equations (4.35) and (4.36) are substituted into equation (4.38) it can be seen that the left-hand side of equation (4.38) depends upon the prediction error power spectrum and as such is very difficult to solve. Girod [10] solves this problem by considering two cases, $\phi \ll S_{ee}(\omega_x,\omega_y)$ and $\phi \geq S_{ee}(\omega_x,\omega_y)$, for which equation (4.38) can be solved and then argues that the intermediate region smoothly approaches these two solutions.

For the purposes of this chapter, the case when $\phi \ll S_{ee}(\omega_x,\omega_y)$ is sufficient. If $\phi \ll S_{ee}(\omega_x,\omega_y)$ equation (4.38) simplifies to,

$$S_{ee}(\omega_x,\omega_y) = \mathrm{E}\left[|1-G(\omega_x,\omega_y,t)|^2\right]S_{ii}(\omega_x,\omega_y) + \phi\mathrm{E}\left[|G(\omega_x,\omega_y,t|^2\right] \quad (4.39)$$

Substituting $G(\omega_x, \omega_y, t)$ into equation (4.39) gives,

$$S_{ee}(\omega_x, \omega_y) = \frac{1}{2} S_{ii}(\omega_x, \omega_y) \left(1 - \frac{\sin(\omega_x/2)}{\omega_x/2} \frac{\sin(\omega_y/2)}{\omega_y/2} \right) + \phi \tag{4.40}$$

Finally, note that $\phi \ll S_{ee}(\omega_x, \omega_y)$ and as such equation (4.40) is equivalent to equation (4.29). The rate-distortion function is hence given by,

$$R = \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{\log[S_{ee}(\omega_x, \omega_y)]}{D} \, d\omega_x \, d\omega_y \tag{4.41}$$

For zero mean signals the variance of the signal, σ_x^2, is given by,

$$\sigma_x^2 = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} S_{xx}(\omega_x, \omega_y) \, d\omega_x \, d\omega_y \tag{4.42}$$

Substituting equations (4.42) and (4.32) into equation 4.41 gives,

$$R = \frac{\gamma_e^2 \sigma_e^2}{D} \tag{4.43}$$

where γ_e^2 is the spectral flatness measure of the prediction error and σ_e^2 is the variance of the prediction error.

The rate-distortion function for memoryless encoding of a Gaussian source is given by,

$$R(D)_G = \frac{1}{2} \frac{\log(\sigma_e^2)}{D} \tag{4.44}$$

Hence, the signal-to-noise coding gain when using an optimum spatial coder is given by,

$$G = (\gamma_e^2)^{-1} \tag{4.45}$$

It can thus be demonstrated that the signal-to-noise coding gain of an optimum hybrid motion-compensated coder, for small distortions, is asymptotically bounded by the inverse SFM of the prediction error. This is an important result since, as will be demonstrated in the following sections, typical prediction errors have an SFM which is close to unity.

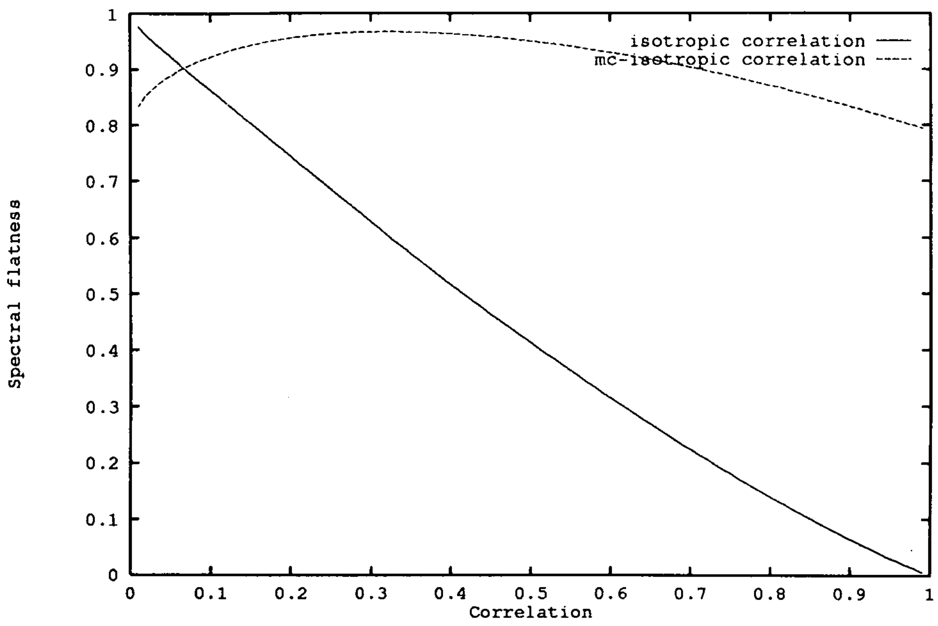


Figure 4.12: *Spectral flatness measures of an isotropic correlation function and motion-compensated isotropic correlation function for various values of correlation.*

Spectral flatness of model correlation functions

The coding gain of an optimum hybrid motion-compensated prediction encoder can be assessed for various model correlation functions using equations (4.29) and (4.45).

Figure 4.12 compares the spectral flatness measure of a isotropic correlation model (equation (4.31)) and a motion-compensated isotropic model for various values of correlation in the input image. As expected, for high values of correlation the prediction error power spectrum has a significantly whiter spectrum than the original. But as the correlation coefficient is decreased, the spectrum of the image becomes progressively whiter and effect of the motion-compensated prediction filter is to decrease the SFM of the prediction error. Since many real-world images are approximated to contain predominantly low frequency information their prediction errors will certainly show an increased SFM.

Figure 4.12 shows that the minimum value of spectral flatness measure for the motion-compensated isotropic signal is approximately 0.8. This implies that the maximum coding gain is $G \simeq 1.25$.

Sequence	SFM of image
Miss A	0.019
Clare	0.017
Sale	0.144

Table 4.4: *Spectral flatness of different image sequences.*

Spectral flatness measure of real data

Previously the spectral flatness measure of a model correlation function was evaluated and it was demonstrated that for images which can be modelled as containing predominantly low frequency information, motion compensation results in an increase in the spectral flatness measure. In this section the SFM of actual images is measured and compared to the results shown previously.

The coding gain of a discrete source is given by the SFM of the eigenspectrum of the image covariance matrix (see equation B.10 in appendix B) which is given by the ratio of geometric mean and arithmetic mean of the eigenvalues of the covariance matrix. Thus for an $N \times N$ pixel block the coding gain, ${}^{N\times N}G$ is given by,

$$ {}^{N\times N}G = \frac{\left(\prod_{i=1}^{N} \lambda_i\right)^{1/N^2}}{\frac{1}{N^2}\sum_{i=1}^{N^2} \lambda_i} \tag{4.46} $$

Table 4.4 and table 4.5 shows the spectral flatness measures of the original images and their corresponding prediction errors respectively. All the covariance matrices are calculated using 8×8 blocks which are row stacked to make a vector [2]. The prediction errors are formed using exhaustive search block matching in which block sizes of 4×4, 8×8 and 16×16 were used.

Clearly, the prediction errors have a much greater SFM than the original images as expected from theory. However, the spectral flatness of the actual prediction errors is not as great as is predicted by the theoretical results. This is because the assumptions made to generate the theoretical models are an over simplification of what occurs in the real-world images. Note, also that the spectral flatness of the prediction errors

MCP block size	SFM of PE		
	Miss A	Clare	Sale
4×4	0.689	0.779	0.807
8×8	0.524	0.658	0.540
16×16	0.430	0.597	0.505

Table 4.5: *Spectral flatness measure of the prediction errors of three different image sequences. The prediction error is formed using block matching motion-compensation with three different block sizes.*

increases as the block size is decreased. This occurs because decreasing the blocks size increases the accuracy of the motion estimate.

Motion-compensated transform coding

The analysis of the hybrid motion-compensated architecture discussed in section 4.4.3 is for an *optimum* spatial encoder. Since transform coding is frequently combined with motion-compensated prediction it is worth making some comments on the expected performance of this particular combination.

The KLT is an optimum algorithm for encoding images and it can be shown to approach the rate-distortion bound (appendix B). It is generally not used for encoding images because of its computational complexity and instead, suboptimal transforms such as the DCT are used.

The DCT has a performance which is close to optimal for images which can be approximated to be AR with correlation approaching unity. Since in many real-world images this assumption is frequently true the DCT performs well. However, the properties of a typical prediction error deviate significantly from those required for the DCT to have optimal performance and as such it is expected that transform coding of the prediction error will give particularly poor results. In particular, the high intensity lines which correspond to the edges of objects in the image and which are vital to maintaining the fidelity of the encoded images, cannot easily be encoded by a transform operation [9]. However, since block matching algorithms do no correctly estimate all types of motion, the prediction error contains significant structure which allows the transform encoder to continue producing gains.

A problem which may occur with non-adaptive transforms is that the transform operation may increase the correlation between pixels rather than decrease the correlation. This would manifest itself as a decrease in the spectral flatness measure of the transform coefficients compared to the spectral flatness measure of the original data. Clarke [82] has measured the spectral flatness measure of transform coefficients, his results indicate that the SFM of the transformed data is around 0.9 indicating that the transform coding is working as required.

The result of applying suboptimal non-adaptive block-based transforms to the prediction error is well known to be a loss of high frequency information which results in poor edge definition and, in extreme cases, the block structure which the transform is based upon becomes apparent.

Tree-structured encoding

To conclude this section Strobach's tree-structured scene adaptive coder [9] is discussed. This algorithm encodes the prediction error by performing a regular decomposition. The image is first split into arbitrary disjoint blocks and then each block is recursively split until a subblock contains pixels with an almost homogeneous intensity. The image is then encoded by describing the block structure with a tree and assigning each block a mean intensity value. The prediction error is easily decoded by adding the mean intensity value to the corresponding region in the previous image after motion compensation has been applied.

In comparison to transform coding this technique is optimised for encoding prediction errors and Strobach has reported good results at rates of approximately 61 kbits/s.

4.5 Summary and conclusions

This chapter discussed and presented results for various aspects of motion estimation for image compression. In particular, motion-compensated prediction and the prediction error were discussed in detail.

Theoretical models were developed to describe the spatial and spectral properties of the prediction error. It was shown from these models that the prediction error could be characterised as a zero mean signal in which the majority of the pixels have values which are close to zero. It was also shown that high intensity edges would be found in regions where there are large changes of pixel intensity in the original image.

This model was then used to determine the nature of an error surface generated using a BMA. The accuracy of the motion estimate was shown to depend upon the contents of a block:

- motion estimates made with blocks in which the pixels have approximately uniform intensities are easily corrupted by noise.
- motion estimates made with blocks in which there are large changes in pixel intensities are robust to noise.

This implies that the high intensity lines in the prediction error may have ‘accurate’ displacement estimates associated with them.

The model which was used to develop the spatial description of the prediction error was further simplified to produce a model of the spectral properties of the prediction error. For sufficiently accurate motion estimates motion-compensated prediction was demonstrated to be equivalent to non-ideal spatial high-pass filtering of the original image.

Based upon the analysis by Girod [10] of the optimum hybrid motion-compensated codec it was demonstrated that the signal-to-noise coding gain is asymptotically bounded by the inverse of the spectral flatness measure of the prediction error. Theoretical analysis of the SFM showed that values of 0.9 could be expected for an image which has an isotropic correlation function with correlation coefficient greater than 0.9. The coding gain for discrete images is given by the ratio of the geometric mean and arithmetic mean of the eigenvalues of the image covariance matrix. The SFM of actual prediction errors was shown to be greater than the SFM of the original images as predicted by the theoretical results. However, the SFM of the actual prediction error was found to be less than the SFM predicted by the theoretical model. This was attributed to the simplicity of the theoretical model.

The chapter concluded by discussing two different hybrid coding schemes: the motion-compensated transform coder and a tree-structured scene adaptive coder. The motion-compensated transform coder, which is used in many state-of-the-art codecs, was shown to have a poor performance since the transform encoder is not optimised for compressing prediction errors. In contrast, Strobach's tree-structured scene adaptive coder has been optimised for compressing prediction errors and as such has a superior performance.

Chapter 5

Variable block size conditional replenishment

5.1 Introduction

Many hybrid motion-compensated codecs have a poor performance since the spatial coding algorithm is not optimised for encoding prediction errors. In particular, motion-compensated transform coding has been observed to produce very poor results. It is the aim of this chapter to develop a coding scheme which exploits the properties of the prediction error so that the quality of the reconstructed image is preserved.

5.2 Region-based image coding

Block-based coding algorithms, although being relatively simple to implement, can result in visually disturbing artifacts as the compression ratio is progressively increased. Two reasons for this were highlighted at the end of chapter 4: firstly, the blocks are chosen without considering the contents of a particular block, and secondly, and as the distortion is progressively increased the regular block structure becomes apparent.

Region-based techniques overcome these problems by first segmenting the image according to some property of the encoder. The resulting segmentation frequently corresponds to actual objects in the image and, as such, even with very poor image fidelity, the structure of the image is still apparent, rather than the chequerboard pattern of the block structure.

Segmentation algorithms can be broadly divided into two classes: region-based methods which depend upon pixels statistics over localised areas of the image and edge-based methods which detect discontinuities between regions. Region-based segmentation may be defined as follows [83]. Let S be a set which represents the entire image. The segmentation algorithm then partitions S into n subregions (subsets), $S_1, S_2, \ldots, S_n$ such that

1. $\bigcup_{i=1}^{n} S_i = S$,
2. S_i is a connected region, $i = 1, 2, \ldots n$,
3. $S_i \cap S_j = \emptyset$ for all i and j, $i \neq j$,
4. $P(S_i) =$ TRUE for $i = 1, 2, \ldots, n$,
5. $P(S_i \cup S_j) =$ FALSE for $i \neq j$,

where $P(S_i)$ is a uniformity predicate over the set S_i.

The uniformity predicate assigns a point to be TRUE or FALSE depending upon some property of the image. The choice of uniformity predicate is very much problem dependent. For example, if the image is to be segmented into regions of approximately homogeneous greyscale, mean and variance are a good choice of uniformity predicate. If the image is to be segmented into regions corresponding to objects then textural information may be more useful. The accuracy of a segmentation can be greatly enhanced by segmenting using multiple image properties, *e.g.* textural and edge information can be combined to improve a segmentation.

5.2.1 Region growing

Region growing algorithms segment an image by starting with a set of *seed* points which are grown into larger regions by appending neighbouring pixels if the uniformity predicate is true and the pixels are connected to this region [84](Chapter 7). Region growing is terminated when no further pixels can be appended to the current region.

5.2.2 Region splitting and merging

In contrast to region growing, split-and-merge algorithms [84](chapter 7) are top-down algorithms in which arbitrary disjoint subregions are subdivided into smaller subregions in an attempt to satisfy the uniformity predicate. Blocks with identical properties are then merged to form irregular regions. A common split and merge algorithm is to successively subdivide a square image into quadrants until the uniformity predicate is true.

5.2.3 Regular decompositions and trees

A regular decomposition is a simplified version of the split and merge algorithm which produces regular shaped regions as the final output of the segmentation. The resulting segmentation is a hierarchical data structure which describes regions of the image and can be compactly encoded using a tree. Regular decompositions have been shown to be able to describe complex structures [85] and are particularly good at describing sparse line structures; a fact which Strobach [9] exploits in his tree-structured scene adaptive encoder. Another advantage of the regular decomposition is that it can be easily manipulated with a digital computer [86].

There is no restriction on how a block may be split but, in general, decompositions which result in square or rectangular blocks are favoured. Quadtrees are commonly used in regular decompositions images because of their inherent symmetry. Figure 5.1 shows an example of a quadtree decomposition and its associated tree. Each node in the tree is associated with an area of the image.

A *tree* is defined to be a graph which is connected, acyclic and undirected [46](chapter

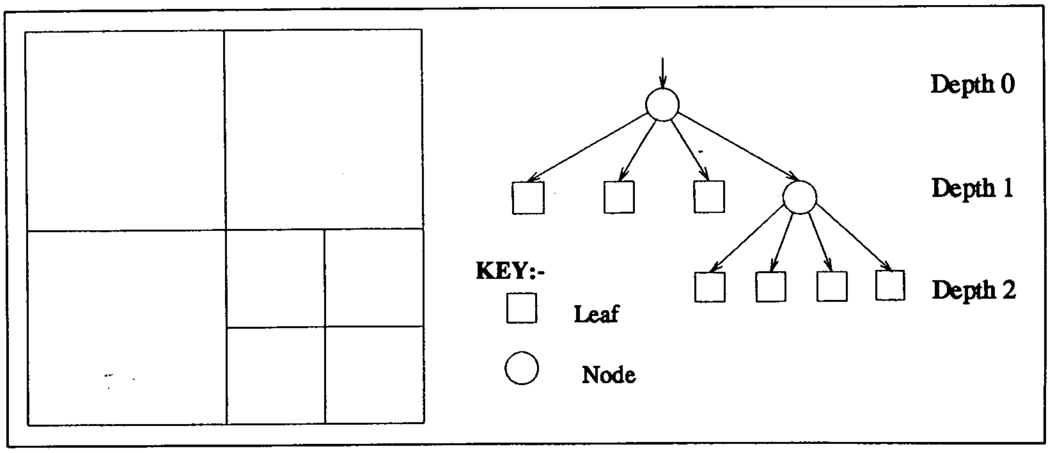


Figure 5.1: *A regular decomposition of a block and its associated tree structure. The tree has a height of two and each node is degree four.*

5). The tree shown in figure 5.1 is a *rooted tree.* A rooted tree is a type of tree which has one node which is distinguished from the others. This node is described as the *root node.* For a regular decomposition this node is always associated with an entire image or a subblock of the image. Throughout this chapter the area of the image which corresponds to a root node is described as a *root block.*

Any node which has no children is described as a *leaf.* The *depth* of a node, x, is defined to be length of the path from the root node to the node, x, where *length* is defined to be the smallest number of edges which must be traversed to go from the root node to x. Each leaf is associated with an area of the image and the depth of a leaf indicates the size of this area. For example, if the root block in figure 5.1 has a dimension of 8×8 pixels, a leaf at depth one corresponds to a 4×4 pixel block.

The maximum depth of any node is the *height* of the tree. Finally, the number of children at a node is the *degree* of a node. For a regular decomposition the degree of a node is generally always fixed to a constant value, k.

A rooted tree can be described by a variable length binary string. For example, if a leaf is denoted by a binary zero and a node by a binary one, the tree in figure 5.1 can be described by the binary string 1/0001/0000. To allow this code to be uniquely decipherable some convention is need to describe the ordering of the binary digits. For example, the binary string 1/0001/0000 was generated by starting at the top left hand block and reading to the bottom right hand block by scanning rows.

5.3 Conditional replenishment

Prediction errors are characterised as having an approximately zero mean with the majority of pixels having values which are close to zero. High magnitude edges are found in regions where the original image has a high gradient, such as at object edges. In regions where the displacement estimate is sufficiently accurate the prediction error can be approximated to be white noise.

It was shown in chapter 4 that transform coding of the prediction error is not optimal and to ensure maximum coding gain a different spatial encoding scheme should be employed. Strobach addresses this problem by using a tree-structured scene adaptive coder [9] which encodes the prediction error using a quadtree decomposition in which the prediction error in a particular block is approximated by its mean value. An alternative scheme which can be employed is conditional replenishment, which is a simple but effective method of encoding images based upon only transmitting the changed part of an image. The image is then decoded by copying pixels from the previous frame to the current frame. This is similar to Strobach's encoder in that copied regions correspond to blocks which have a zero mean and the replenished regions correspond to blocks which have a finite mean.

Using conditional replenishment to encode a prediction error does not significantly degrade the fidelity of the image, since the prediction error in the copied regions can be approximated to be low power white noise. However, if conditional replenishment is used to encode an image which is a low-quality reproduction of the original image, no increase in image fidelity will be seen, therefore, throughout this chapter it is assumed that both the transmitter and receiver start with high fidelity images.

Much of the early work on conditional replenishment was completed in the late 1960's by Mounts [87]. He described a conditional replenishment system in which the current image is subtracted from a reference picture stored in memory and only picture elements which have changed significantly are updated. The update threshold is made a function of the output buffer occupancy so that a constant rate is maintained. This results in a serious degradation of the image quality during periods of large motion.

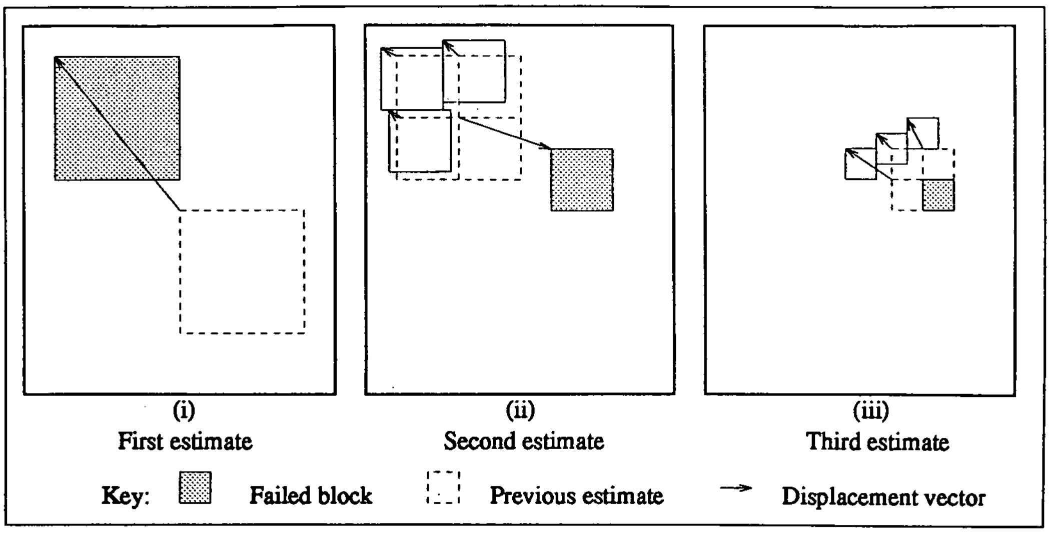
Hein [88] describes a conditional replenishment algorithm based on a block matching motion predictor in which the residual blocks are encoded using a Walsh transform. Ghanbari [89] describes a conditional replenishment encoder using a block matching motion estimator. In this scheme the image is replenished in strips to decrease the overhead information required to transmit the address of regions to be replenished.

To maintain the image fidelity accurate displacement estimates are required. BMAs assume that all pixels within a block are homogeneously displaced. To ensure this is true small blocks should be used. However, using a small block size requires a large overhead for the vector data and as such larger blocks are usually employed which are not suitable for conditional replenishment.

This problem can be circumvented since in real-world scenes entire objects move coherently and as such small blocks are only required for a small subset of the image. Wang [90] has suggested a solution to this problem based upon the grouping of vectors into irregular regions. Chan's variable block size motion estimation algorithm [61] performs a regular decomposition so that small blocks are only used where necessary. A variable size block matching algorithm (VSBMA) will be used in the following sections to generate a segmented image for conditional replenishment.

5.4 Variable block size conditional replenishment

Chan's variable block size motion estimation algorithm [61] attempts to segment an image into regions of approximately homogenous motion using a regular decomposition. The algorithm makes motion estimates by initially dividing the image into a regular array of blocks. A motion estimate is made for each block; if a motion estimate for a particular block is not sufficiently accurate, this block is subdivided and a new motion estimate is made for each new block. This process is continued until either a sufficiently accurate motion estimate is found or a minimum block size is reached. Neighbouring blocks with the same displacement vector are then merged.



(i) First estimate

(ii) Second estimate

(iii) Third estimate

Figure 5.2: *An example of a variable block size motion estimate.*

Figure 5.2 shows a simple example of this type of motion estimate, in which only the *active* blocks are shown. The minimum block size is set so that after three iterations the search terminated. Initially, a motion estimate is made for the large block (fig. 5.2(i)). The accuracy of this estimate is not deemed sufficient so this block is split into four smaller blocks and a new motion estimate is made for each of these blocks using the previous best match as a starting point (fig. 5.2(ii)). Three of the four blocks pass in the second stage. Finally, the failed block is subdivided and a motion estimate is made for each of the smaller blocks after which the search is terminated (fig. 5.2(iii)). The block structure in figure 5.2 is equivalent to that shown in figure 5.1 and as such can easily be described by this tree.

The VSBMA algorithm will be used in the following to segment an image using a regular decomposition so that conditional replenishment can be used. The encoder can be considered to be similar to Strobach's tree-structured scene adaptive coder except the conventional BMA is replaced with an enhanced BMA which endeavours to generate zero mean blocks with a white power spectrum. In contrast to Strobach's encoder no attempt will be made to describe the prediction error. If a block is labelled as requiring replenishment the corresponding block in the original image will be encoded.

The choice of threshold used to determine whether a block should be copied or replenished is critical to maintaining the integrity of the encoded images. An optimum threshold is one which mimics the HVS, unfortunately the complexity of the HVS prohibits the formulation of such a threshold. After some experimentation the ratio of mean square error to mean block energy was found to be a reasonable choice of the threshold. That is, a block of pixels, S, is labelled pass or fail according to the following,

$$S = \begin{cases} \text{copy} & \text{iff } \dfrac{\sum_{x,y \in S} e(x,y,t)^2}{\sum_{x,y \in S} i(x,y,t)^2} < \mathrm{T} \\ \text{replenish} & \text{otherwise} \end{cases} \tag{5.1}$$

where $T \in [0, 1]$ is the motion estimation threshold value.

The success of this threshold can be attributed to its similarity to Weber's law (section 2.2) since more distortion is permitted in high intensity regions than low intensity regions. Mean absolute error could also be used and has the advantage of requiring less computation but it was found to be too insensitive to small isolated regions of high intensity pixels and as such 'incorrectly' assigned them as pass blocks.

Figure 5.3 shows a typical segmentation of an image generated with a variable block size motion estimation algorithm using the above threshold. White regions correspond to areas which can be encoded by copying and shaded regions correspond to fail regions. Note, that the eyes and mouth cannot be encoded by copying since the distortion of the lips or the closing of the eyes cannot be described by the displacement vectors.

The output of the VSBMA is a hierarchical data structure in which blocks are labelled as either pass (copy region) or fail (replenish region). To encode an image the hierarchical data structure is described by a tree, the pass blocks with a vector and the fail blocks with a code which corresponds to the block of pixels in the original image. The tree is described with a variable length binary string and the vector data must be encoded with an information-lossless algorithm. The failed blocks may be encoded using an information-lossy algorithm such as VQ or transform coding. However, it should be noted that the algorithm used to encode the failed blocks is critical to the efficient operation of the replenishment coder, since if too much coding distortion is introduced these blocks may be unnecessarily replenished at later stages of coding.

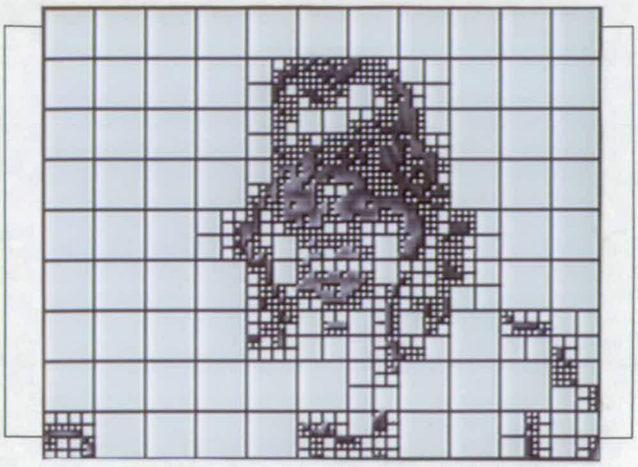


Figure 5.3: *A typical segmentation of Miss America generated with a threshold* T = 0.001.

The regular nature of the decomposition allows the rate required to encode an image to be calculated based upon the number of leaf nodes in a particular decomposition.

5.4.1 Rate required for tree data

The total number of bits required to encode a tree is easily calculated since each leaf and node requires one bit. This can be calculated from the number of leaf nodes, N_i at depth i in the trees and the degree of the node, k. For example, assume that the maximum height of the trees is L and there are N_L leaves at this depth. Clearly, N_L bits are required for these leaves. At depth $L-1$ there are N_{L-1} leaves and N_L/k nodes. By repeated application of this rule the total number of bits required to encode the trees can be calculated.

Before deriving the general expression a simple example is considered. Figure 5.1 shows a three level quadtree decomposition ($k = 4$) of a block in which $N_0 = 0$, $N_1 = 3$ and $N_2 = 4$. This decomposition can be described by the binary string 1/0001/0000 and hence requires 9 bits to encode. Applying the previous rule the number of bits required to encode this is 4 bits at depth two, $3+4/4 = 4$ bits for depth one and $0+(3+1)/4 = 1$ bits for depth zero. Hence the total number of bits required is 9.

For the general case of a node which has degree k the number of bits required to describe the structure of the trees, B_t, is,

$$B_t = N_L + \frac{N_L}{k} + N_{L-1} + \frac{N_{L-1}}{k} + \frac{N_L}{k^2} + N_{L-2} + \cdots + N_0 \tag{5.2}$$

Equation (5.2) can be simplified by grouping terms which include the blocks from the same level so that the series becomes the summation of a set series, *i.e.*

$$B_t = \sum_{i=0}^{L} \sum_{m=0}^{L-i} \frac{N_{L-i}}{k^m} \tag{5.3}$$

The inner summation in equation (5.3) is a geometric series and as such can be simplified using,

$$\sum_{i=0}^{n-1} r^i = \frac{1 - r^n}{1 - r} \tag{5.4}$$

to give

$$B_t = \frac{k}{k-1} \sum_{n=0}^{L} (1 - k^{i-L-1}) N_{L-n} \tag{5.5}$$

5.4.2 PCM encoding of the vector data.

Each leaf which has been labelled ‘passed’ is assigned a vector, thus if N_i^f leaves fail at depth i the total number of bits required for the vectors, B_v, is given by,

$$B_v = \sum_{i=0}^{L} (N_i - N_i^f) r_v \tag{5.6}$$

where r_v is the rate required to encode the vectors. If a maximum displacement of $d_{\max}$ pixels is allowed then r_v is given by,

$$r_v = \log_2 \left((2d_{\max} + 1)^2 \right) \quad \text{bits/vector} \tag{5.7}$$

5.4.3 PCM encoding of failed blocks

Initially, analysis and simulation results will be presented for PCM encoding of the failed leaf data such that zero distortion is introduced. PCM encoding of the ‘failed’ data is generally unsatisfactory since if large areas of the image are labelled failed there will be a dramatic increase in rate. However, this scenario allows some useful mathematical analysis of the encoder to be performed and the simulation results may be beneficially interpreted as the upper bound to the rate and the lower bound to the distortion.

The rate required to encode a failed block is proportional to the total number of pixels within a block. For example, if the root blocks contain A pixels then a block at depth i in the tree will contain Ak^{-i} pixels. The number of bits required to encode the failed regions is simply given by,

$$B_f = \sum_{i=0}^{L} N_i^f A k^{-i} r_d \tag{5.8}$$

where r_d is the rate required to encode each pixel.

As yet no information concerning the state of the node has been included. Thus to differentiate between a pass and fail leaf an extra binary digit will be included per leaf node of the tree. The total number of bits required to describe the state information is given by,

$$B_s = \sum_{i=0}^{L} N_i \tag{5.9}$$

Total rate

The total number of bits required to encode an image, B_{tot} is given by,

$$B_{\text{tot}} = B_t + B_s + B_v + B_f \tag{5.10}$$

where B_t, B_s, B_v and B_f are the number of bits required for the tree, state data, vector data and failed leaf data respectively.

Thus substituting equations (5.5), (5.9), (5.6)and (5.8) into equation (5.10) gives,

$$B_{\text{tot}} = \sum_{i=0}^{L} \left(\frac{k}{k-1}(1 - k^{i-L-1})N_{L-i} + N_i + (N_i - N_i^f)r_v + N_i^f A k^{-i} r_d \right) \tag{5.11}$$

The rate required to encode an image which has dimensions $X \times Y$ pixels is given by,

$$R = \frac{1}{XY} B_{\text{tot}} \tag{5.12}$$

Analysis of the coder

Equation (5.11) allows the rate required to encode an image to be calculated if the number of leaf nodes and number of failed nodes are known. This equation can also be used to determine simple expressions which ensure compression is always achieved.

As a given decomposition becomes progressively more complex the overhead required to describe the tree and vector data increases correspondingly. It is conceivable for a sufficiently complex decomposition that compression will no longer occur. The tree and vector codes require a maximum number of bits when all the leaves of the trees are at a depth which is equal to the height of the tree, *i.e.*

$$N_i = \begin{cases} 0 & \text{if } i \neq L \\ N_L & \text{otherwise} \end{cases} \tag{5.13}$$

Hence, this represents a limiting case and will be used to calculate the conditions necessary to ensure that compression is always obtained.

For an image which has dimensions $X \times Y$ pixels compression is given when,

$$B_{\text{tot}} < XYr_d \tag{5.14}$$

From equation (5.11) this condition is satisfied when,

$$\frac{k(1 - k^{-(L+1)})}{k - 1} N_L + N_L + (N_L - N_L^f) r_v < (XY - N_L^f A k^{-L}) r_d \tag{5.15}$$

Since there are no blocks on any level of the tree except the L^{th} level, the area of the image, XY, is equal to,

$$XY = N_L A k^{-L} \tag{5.16}$$

Height of tree (L)	$\frac{N_L^f}{N_L}$
0	0.999
1	0.995
2	0.893
3	2.165

Table 5.1: *Ratio of failed blocks to maximum number of blocks so that compression is just achieved when the height of the tree is L (see equation (5.19)). (k = 4, r_d = 8 bits/pixel, r_v = 10 bits/pixel and A = 16 × 16 pixels).*

Substituting equation (5.16) into equation (5.15) gives,

$$\frac{N_L^f}{N_L} < 1 - \frac{\frac{k}{k-1}\left(1 - k^{-(L+1)}\right) + 1}{Ak^{-L}r_d - r_v} \tag{5.17}$$

Table 5.1 shows the ratio of failed blocks to the total number of blocks at depth L required to just achieve compression using a quadtree decomposition ($k = 4$), PCM encoding the failed pixels and vectors at 8 bits/pixel and 10 bits/vector respectively and using root blocks of dimension 16 × 16 pixels. As expected the ratio of failed blocks to total number of blocks is seen to progressively decrease as the total number of levels increases. Note, for $L = 3$ the ratio of failed blocks to total number of blocks is greater than unity and since $N_L^f \leq N_L$ this is not possible and as such compression cannot occur. Clearly, this is a trivial example since when $L = 3$ a block corresponds to a single pixel and since $r_v > r_d$ compression will never occur.

Although equation (5.17) provides a method of determining the conditions necessary for compression for a given maximum number of levels, it provides no information on the optimum maximum number of levels which should be used. That is, although a tree with a maximum of L levels may provide compression it may be more efficient to use less levels.

The optimum number of levels which should be employed to obtain maximum compression is easily calculated from the change in rate, ΔB, when an extra level is added to a tree with height l. Assume that there N_l^f failed leaves at this level so that when the extra level is added kN_l^f new leaves are created of which N_{l+1}^f fail. Since the

Height of tree (l)	$\frac{N^f_{l+1}}{N_{l+1}}$
0	0.997
1	0.985
2	0.920
3	1.875

Table 5.2: *Ratio of failed blocks so that compression just occurs when the height of the tree is increased from l to $l+1$ (see equation (5.19)). ($k = 4$, $r_d = 8$ bits/pixel, $r_v = 10$ bits/pixel and $A = 16 \times 16$ pixels).*

tree with height $l+1$ is simply the tree with height l extended by an extra level the increase in rate to describe the tree is trivially kN^f_l. The increase in the rate due to the state information is $(k-1)N^f_l$ and the increase in rate from the vector information is $(kN^f_l - N^f_{l+1})r_v$. Finally, the increase in rate from the failed regions is given by $(N^f_{l+1}Ak^{-(l+1)} - N_lAk^l)$. Thus, ΔB is given by,

$$\Delta B = N^f_l(2k-1) + (kN^f_l - N^f_{l+1})r_v + (N^f_{l+1} - kN^f_l)Ak^{-(l+1)}r_d \tag{5.18}$$

and as long $\Delta B < 0$ adding an extra level will result in compression. Using $N_{l+1} = kN^f_l$ it can be shown that compression is given when the following is true,

$$\frac{N^f_{l+1}}{N_{l+1}} < 1 - \frac{2k-1}{Ak^{-l}r_d - kr_v} \tag{5.19}$$

In comparison to equation (5.1) this a much stronger condition since not only does it ensure compression but it also determines the maximum number of levels which can be usefully be employed.

Table 5.2 shows the ratio of failed blocks to total number of blocks which are required so that compression just occurs when increasing the height of a tree from l to $l+1$. Once again a quadtree decomposition (k=4) is used in which the failed regions are encoded at, $r_d = 8$ bits/pixel, and the vectors at $r_v = 10$ bits/vector and the root blocks have dimensions of 16×16 pixels.

The results in tables 5.2 must be interpreted with care since equation (5.18) does not include the possibility of blocks merging. These results are best interpreted for the case of an extra level being added to a single failed node. For example, if an extra level is added to a node in a quadtree, four new blocks are generated. When l = 0, 1 or 2 the ratio of failed blocks to total number of blocks must be greater than 0.9 for an increase in rate to occur. Even if three out the four new blocks need to be replenished, compression will still occur. Indeed, even if all four blocks fail, no increase in rate will occur since these blocks will be merged back to their previous state. If the ratios in table 5.2 were interpreted globally it would be tempting to think that adding an extra level which contained a small number of pass blocks would result an increase in rate, but the merging of some of the fail blocks would ensure that this was never so.

This has important implications for implementing such an encoder since if the ratio of failed blocks to total number of blocks to just achieve compression is greater than 0.75 no control mechanisms need be installed. However, if the ratio is less than 0.75 some care must be taken to ensure that the maximum compression always occurs. As predicted from table 5.1 adding an extra level when $l = 3$ results in expansion and as such $l = 2$ represents the maximum number of levels which can be used.

Figure 5.4 shows the rate required to encode Miss America for a threshold T = 0.001. Results are shown for a minimum block size of 8×8 pixels ($L = 1$), 4×4 pixels ($L = 2$) and 2×2 pixels ($L = 3$). As predicted from the above a decrease in rate occurs each time the minimum block size is decreased.

Rate and distortion versus threshold

The number of bits required to encode an image is a function of the motion estimation threshold, *i.e.* as the threshold is increased the complexity of decomposition decreases and a greater number of blocks are encoded by copying, which results in a decrease in rate. Likewise, the total distortion which is introduced is a function of the motion estimation threshold, except in this case the distortion increases as the threshold is increased. Results will be presented in this section to demonstrate these trends. The results quoted for both rate and distortion are average results and it should be noted that considerable variation of these quantities may occur between the individual images of the sequence (*e.g.* see figure 5.4).

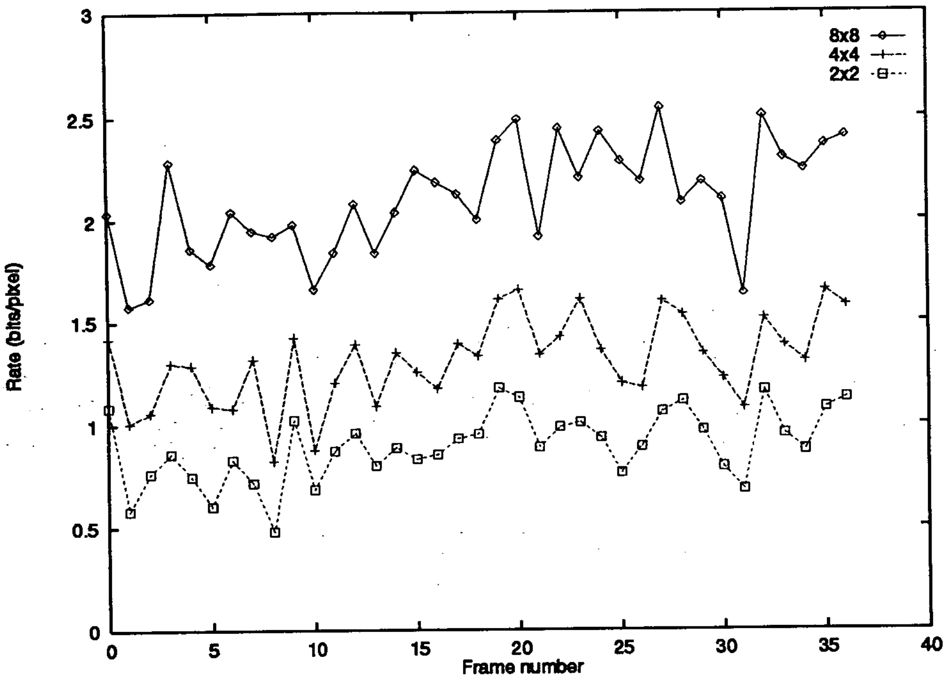


Figure 5.4: *Rate required to PCM encode Miss America using a threshold of 0.001. Vectors are PCM encoded at 10 bits/vector and the failed regions are PCM encoded at 8 bits/pixel.*

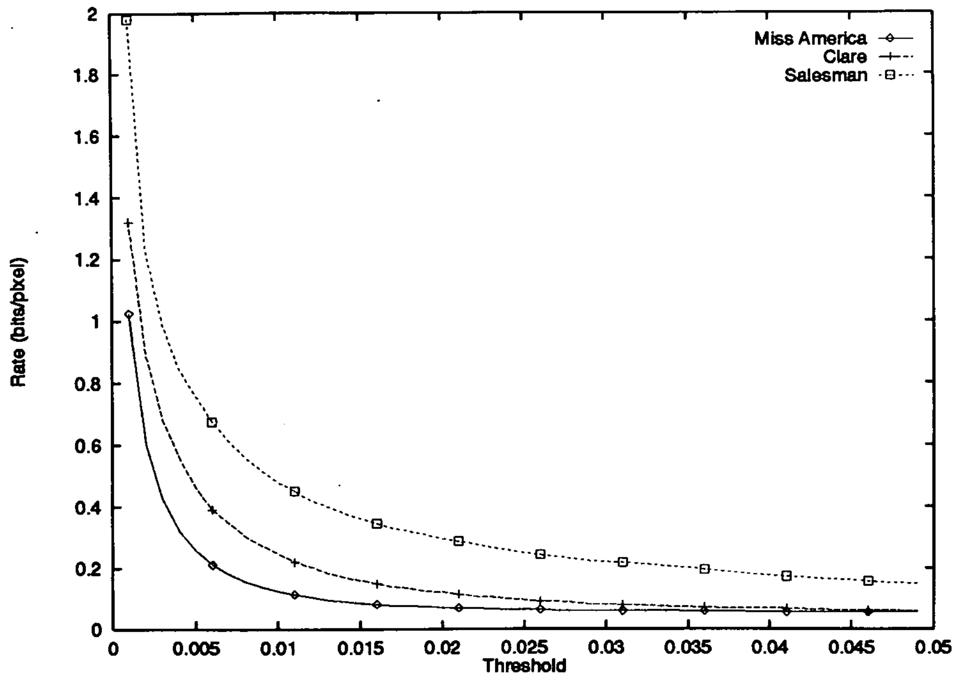


Figure 5.5: *Average rate versus threshold for three different image sequences.*

Figure 5.5 and figure 5.6 show the average rate and average PSNR for three different image sequences for various values of threshold. As expected, increasing the threshold results in a decrease in the average rate and a decrease in the PSNR (increase in distortion). These results are the upper bound to the rate and the lower bound to the distortion. If a more sophisticated spatial coding algorithm were to be employed, a decrease in rate and an increase in distortion would be observed.

Unfortunately, using the motion estimation threshold as a method of controlling rate and distortion is, in general, not satisfactory. This is because the threshold controls the motion estimation algorithm and as the threshold is increased, the motion estimation algorithm terminates for progressively larger blocks. This results in poor displacement estimates which produce an unacceptable degradation in the image fidelity. A better method of controlling rate and distortion is to make motion estimates with a small threshold and adaptively merge blocks as required. This is discussed in greater detail in section 5.4.7.

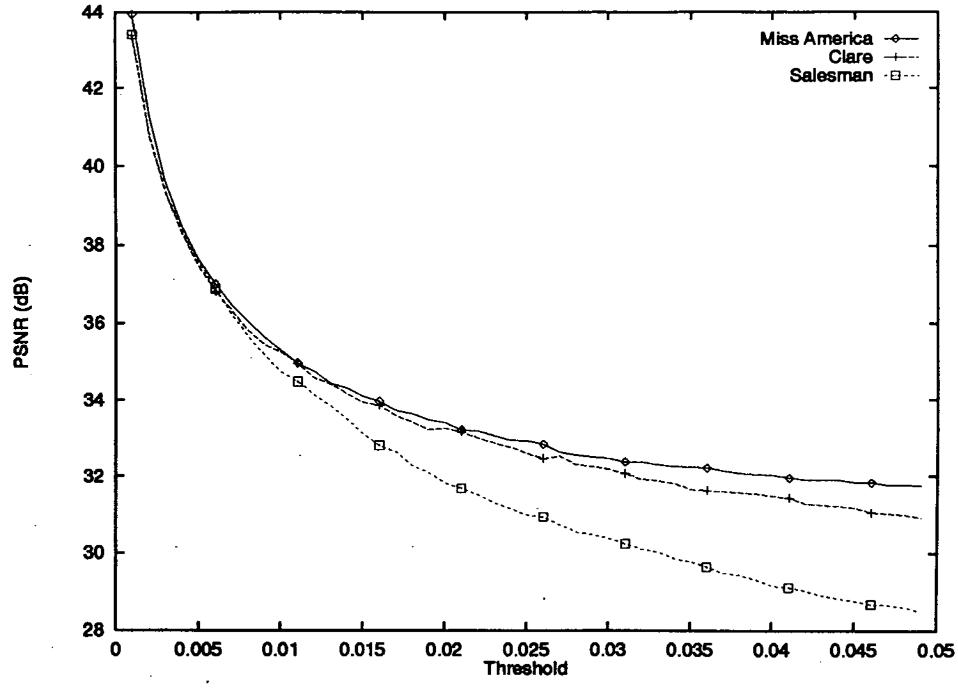


Figure 5.6: *Peak SNR for various thresholds.*

5.4.4 Lower bound to rate

The upper bound to the rate required to encode a sequence of images using the VBCR encoder was found by PCM encoding the failed regions. A good approximation to the lower bound can easily be calculated by simply encoding all the blocks with vectors. The lower bound, like the upper bound, is dependent upon the minimum block size and the motion estimation threshold.

Figure 5.7 compares the upper and lower bound to the rate at which the Miss America standard image sequence can be encoded for various values of threshold using a minimum block size of 2 × 2 pixels. For small values of threshold there is a large difference between the lower and upper bound (approximately 0.35 bit/pixel) but as the threshold is steadily increased the two curves converge to the same value. This indicates there are no failed blocks within the images and as such this point represents the maximum value of threshold which can be usefully employed. For the Miss America thresholds of less than 0.025 should be employed.

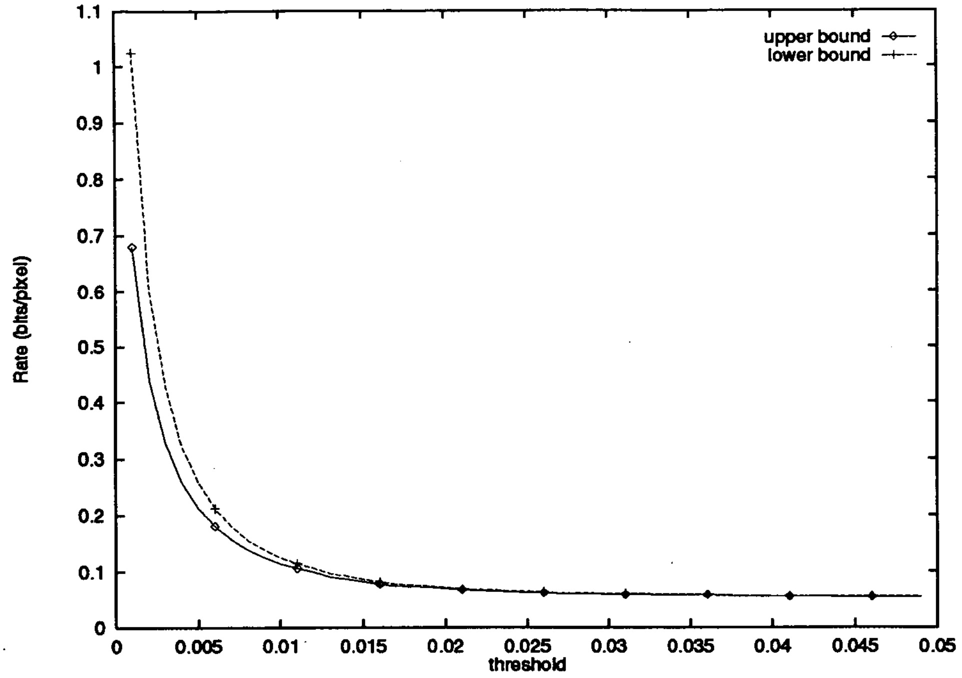


Figure 5.7: *Upper and lower bound to rate required to encode the Miss America sequence of images.*

5.4.5 Tree complexity

Equation (5.11) allows the number of bits required to encode an image to be calculated and allows simple rules to be derived so that compression is always ensured (equations (5.17) and (5.19)). Unfortunately, these equations cannot be used for determining the number of blocks which will be generated for any particular decomposition. In practice, it is doubtful that this can be accomplished except for a few simple cases. The best that can be hoped for is some qualitative understanding of the process so that simple rules can be established which help determine complexity.

Clearly, the complexity of any given decomposition depends upon the size, shape and number of objects within the image. Also the complexity of the decomposition depends upon the type of motion which is present within the scene. That is, an object which moves with translatory motion will produce a relatively simple decomposition whereas an object which is rotating will produce a more complex decomposition.

The complexity of a tree produced from a conventional split and merge segmentation of an image is well known to be dependent upon the position of the objects within the image [9]. Likewise, the position of an object also affects the complexity of a segmentation when using a variable block size motion estimation algorithm.

For example, figure 5.8 shows two images in which a moving object is represented by a single square block which is assumed to undergo constant translatory motion. The large grid squares denote the initial decomposition of the image for motion estimation.

In figure 5.8(i) the object exactly coincides with the grid squares and as such the variable block size motion estimation algorithm is able to make a motion estimate without splitting blocks. In figure 5.8(ii) the square is offset from the grid. In contrast to figure 5.8(i) this decomposition is much more complex requiring a block structure given by $N_0 = 7$, $N_1 = 4$, $N_2 = 8$ and $N_3 = 32$ as opposed to $N_0 = 9$ in figure 5.8(i). From equation (5.5) it can be seen that the tree in figure 5.8(i) requires 9 bits to describe whereas the tree in figure 5.8(ii) requires 65 bits.

This sensitivity to the position of an object is caused by the fact that the initial decomposition into the large grid squares represents a naive segmentation of the image which is subsequently refined. The complexity of the tree can be greatly reduced by choosing a better initial decomposition. Strobach overcomes this problem in his tree-structured scene adaptive coder by re-calculating the segmentation for several different initial block placements. However, for the variable block size motion estimation system this is not practicable since the computational overhead required to do this is excessive. Similar arguments can be invoked to show that the size of the object also effects complexity.

5.4.6 Alternative methods of encoding failed regions

PCM encoding of the failed regions is, in general, unsatisfactory since if large areas of the image are labelled failed the rate required to encode an image increases dramatically. Almost any algorithm can be employed to encode the failed blocks as long as the coding distortion is restricted to be small, to avoid unnecessary replenishment at later stages of encoding.

The performance of many spatial coding algorithms is dependent upon the area of

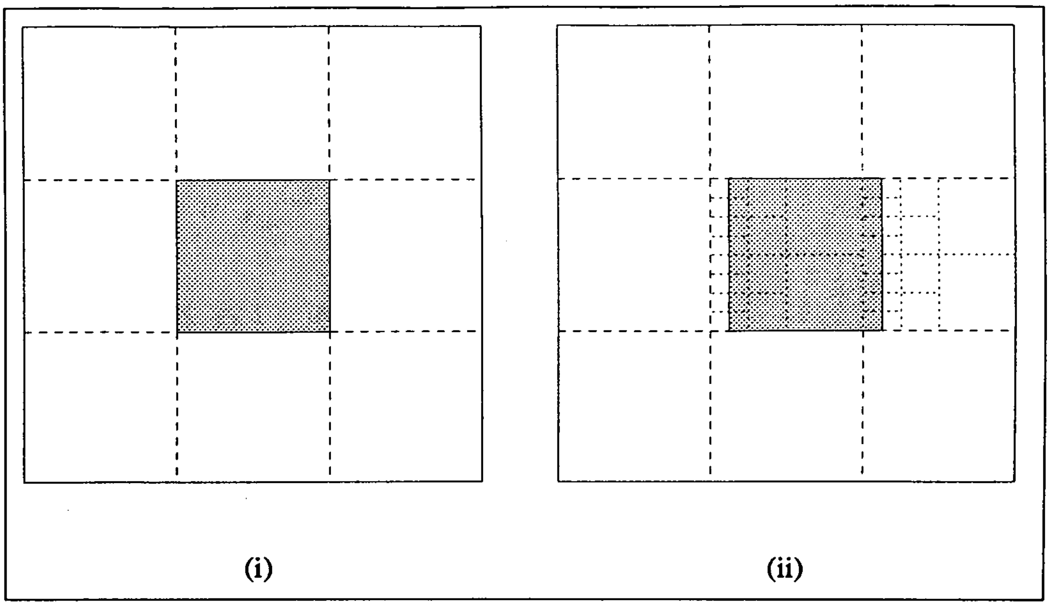


Figure 5.8: *Example of quadtree complexity versus position.*

the image which is encoded. This can have important implications for the design of the encoder, since as the block size is progressively decreased it may be more efficient to prematurely terminate the motion estimation algorithm and employ the spatial encoder. To illustrate this problem a block truncation coder will be employed to encode the failed regions.

Block truncation coding

Block truncation coding [91–99] is a moment preserving non-parametric quantizer which is adaptive over local areas of the image. The basic algorithm, introduced by Delp [91], subdivides an image into $N \times N$ blocks which are encoded independently by quantizing each block to a two level signal. The level for each block is chosen so that the first moments are preserved.

If the intensities of the pixels in a block are given by, $X_1, X_2, \ldots, X_n$ a pixel with intensity X_i is quantized according to the following.

$$X_i = \begin{cases} b & \text{if } X_i \geq \mu \\ a & \text{otherwise} \end{cases} \tag{5.20}$$

where μ is the mean of the pixel values. The values of a and b are determined by the following,

$$a = \mu - \sigma\sqrt{\frac{q}{N^2 - q}} \tag{5.21}$$

$$b = \mu + \sigma\sqrt{\frac{N^2 - q}{q}} \tag{5.22}$$

where σ and q are the standard deviation of the pixels and number of pixels greater than the mean respectively.

Each block is then described by its mean and standard deviation and a $N \times N$ bit plane indicating whether the pixels are above or below the threshold. If the mean and standard deviation of a block are encoded at 8 bits each, the rate required to encode a block is given by,

$$r_d = 1 + \frac{16}{N^2} \tag{5.23}$$

Equation (5.18) was used to predict the ratio of failed leaves to total number of leaves which are required for a decrease in rate to occur when an extra level is added to the tree when PCM encoding of the failed leaves. The conditions necessary to ensure compression when a BTC encoder is employed are given by substituting equation (5.23) into equation (5.18) to give,

$$\frac{N^f_{l+1}}{N_{l+1}} < 1 - \frac{2k - 1}{Ak^l + 16 - kr_v} \tag{5.24}$$

Table 5.3 shows the ratio of failed blocks to total number of blocks at depth l required to just decrease the rate required to encode an image using a quadtree decomposition in which the vectors are PCM encoded at r_v = 10 bits/vector and the root blocks have dimensions of 16 × 16 pixels. As with table 5.2 compression is guaranteed for trees with depthes of $l = 0$ and $l = 1$ but for a depth of two adding an extra level increases the rate required to describe a tree. Thus, the smallest block size which can be employed is 4 × 4 pixels.

In general, a BTC encoder cannot be employed to encode blocks of arbitrary size since the blocks may contain significant activity which cannot be encoded with a bi-

Depth l	$\frac{N_l^f}{N_l}$
0	0.97
1	0.82
2	1.29

Table 5.3: *Ratio of fail blocks to total number of blocks to just allow compression.*

level signal. Roy [99] has introduced a hierarchical block truncation scheme which employs a quadtree decomposition which could be integrated with the VBCR encoder. But for purposes of simplicity, simulation results will be quoted for a fixed block size of 4 × 4 pixels. This results in a simplified VBCR encoder in which failed blocks no longer need to be merged, and since the failed blocks only occur at a depth equal to the height of the tree, no state information need be included.

Figure 5.9 compares the rates to encode the Miss America sequence when the failed blocks are encoded using BTC and PCM for various values of threshold. For small values of threshold many blocks in the image are labelled failed and as such the BTC encoder provides large gains (approximately 0.4 bit/pixel). However, as the threshold is progressively increased less blocks are labelled failed and the BTC encoder provides very little gain.

The distortion results which correspond to the rate results in figure 5.9 are shown in figure 5.10. For small values of threshold there is a large decrease in the PSNR, *e.g.* for a threshold of 0.001 there is an 8 dB decrease. This large decrease in PSNR is caused by the signal being quantised to a two level signal. Figure 5.11 shows two images from the Miss America sequence separated by 25 frames, some degradation in quality has occurred although the image quality is still acceptable.

BTC is of limited use for maintaining image integrity since it quantizes the signal to a bi-level signal. It was used in the above example to demonstrate the potential gain from encoding the failed blocks. In practice, an algorithm such as transform coding or VQ should be used. However, in the following section it will be demonstrated that, in many cases, spatial coding is not required, except when large scene changes occur.

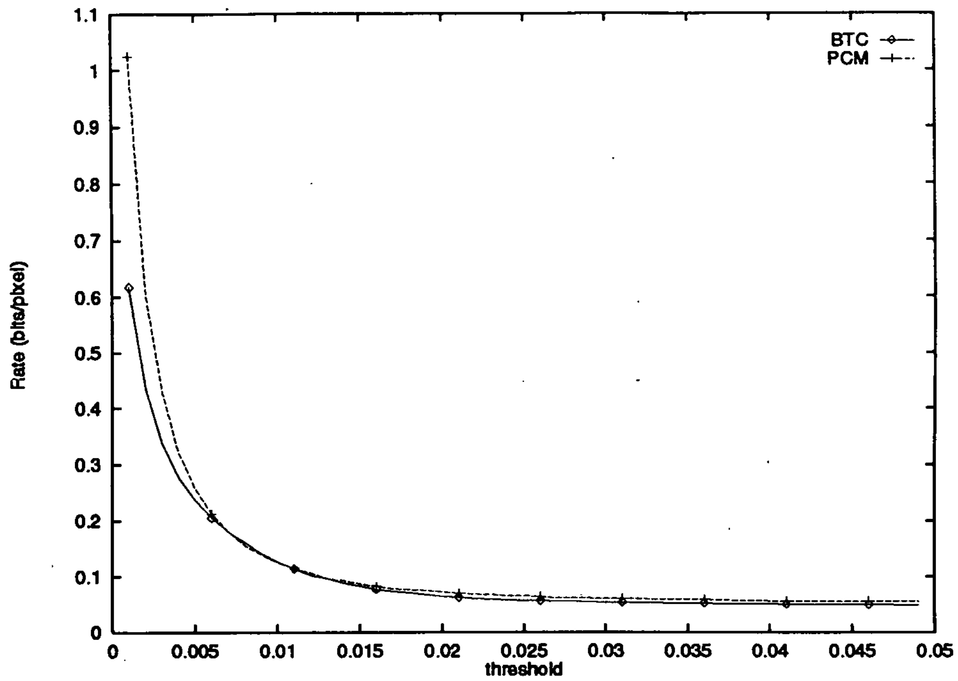


Figure 5.9: *Comparison of rates to encode the Miss America sequence using block truncation coding and PCM of the failed regions.*

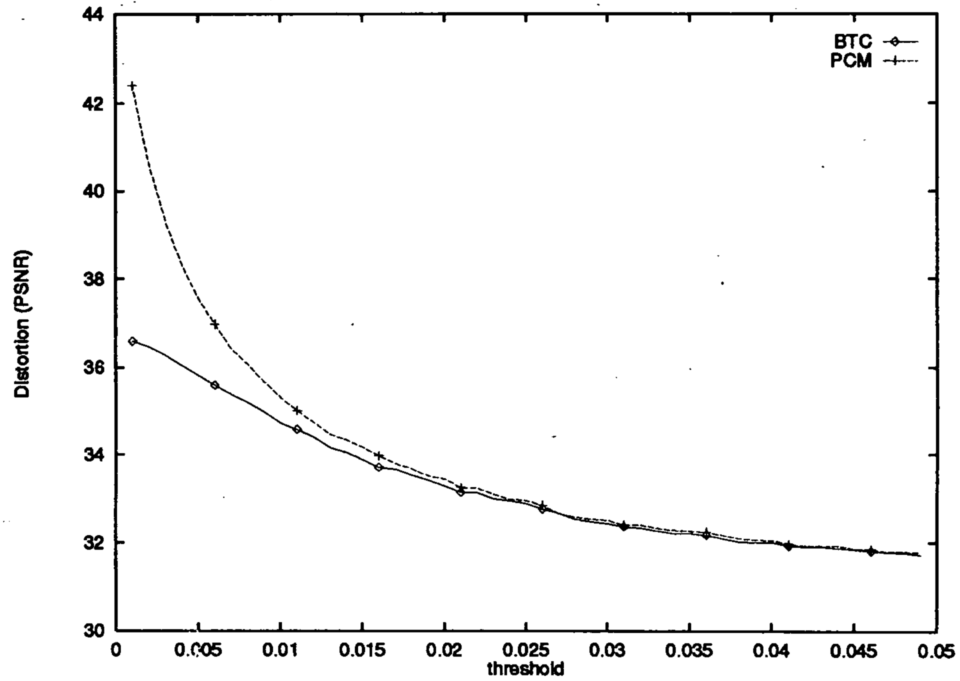
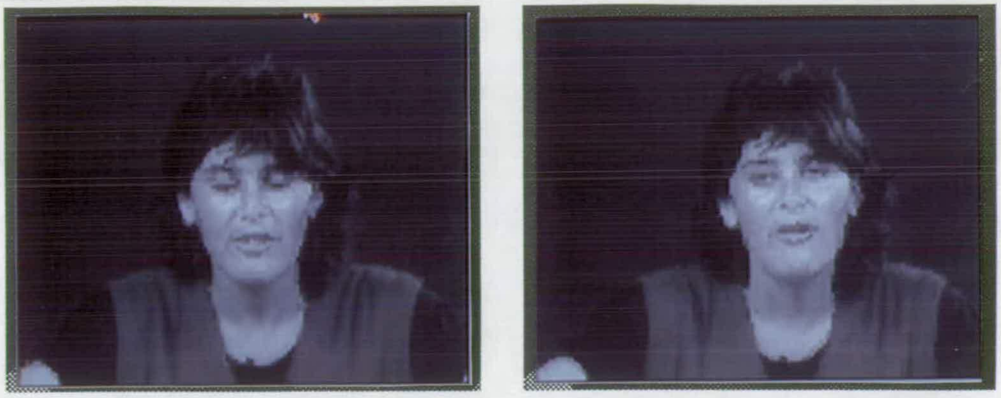


Figure 5.10: *Comparison of PSNR for block truncation coding and PCM encoding of failed regions.*



(i) Rate = 0.45 bits/pixel
PSNR = 37

(ii) Rate = 0.66 bits/pixel
PSNR = 36 dB

Figure 5.11: *Two images from the Miss America sequence encoded using variable block size conditional replenishment and block truncation coding.*

5.4.7 Rate control

The difficulties of controlling the rate and distortion via the motion estimation threshold were discussed in section 5.4.3. In this section an alternative approach is formulated based upon adaptive relabelling and merging of leaf nodes.

The motion estimation threshold should not be used to control the rate at which an image is encoded since it is responsible for maintaining the integrity of the displacement estimates. For example, figure 5.12 shows the block structures generated from a motion estimate using two different motion estimation threshold values. The tree in figure 5.12(ii) is generated using a larger threshold value than for the tree in figure 5.12(i). Clearly, the tree in figure 5.12(ii) is less complex than the tree in figure 5.12(i). As the motion estimation threshold is progressively increased the complexity of the motion estimates will be further reduced, until eventually all motion estimates terminate for the root blocks. Under these conditions the decoded images will contain large mismatches at the block boundaries which are perceptually very disturbing and, as such, conditional replenishment should not be employed.

Clearly, to preserve the integrity of the decoded images the accuracy of the displacement estimates need to be preserved. This can be achieved by using a small

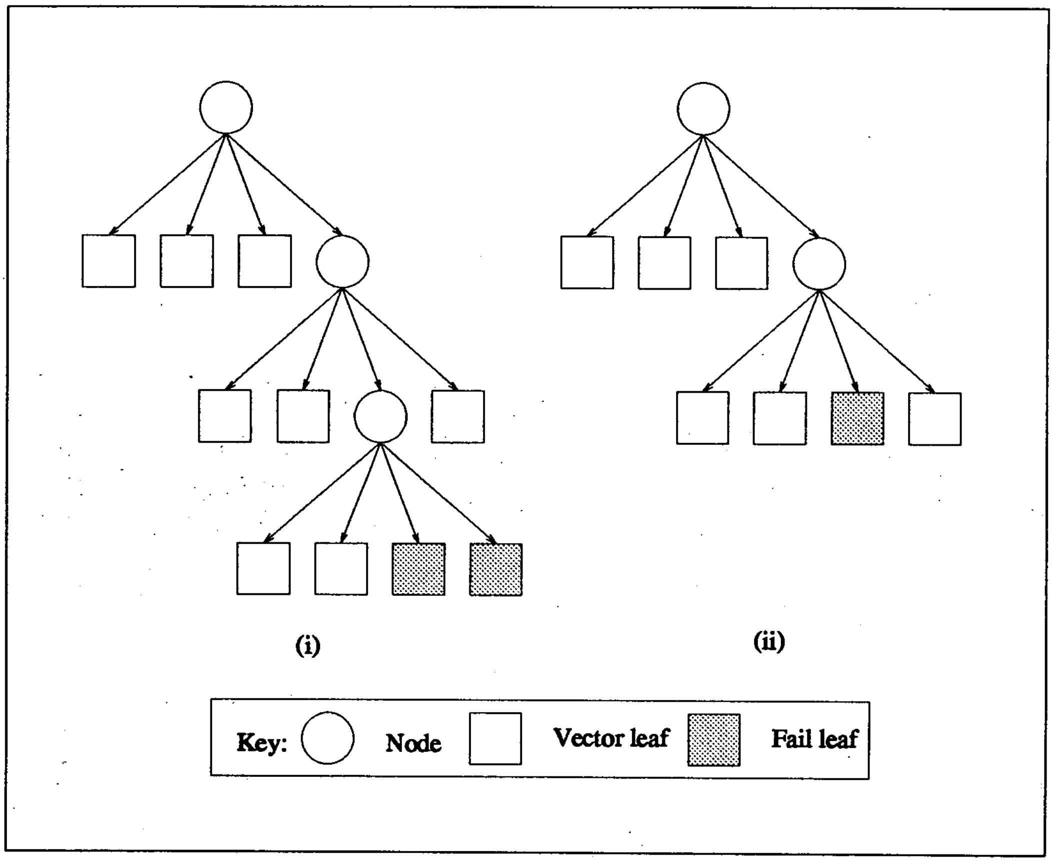


Figure 5.12: *Two trees generated for different motion estimation thresholds.*

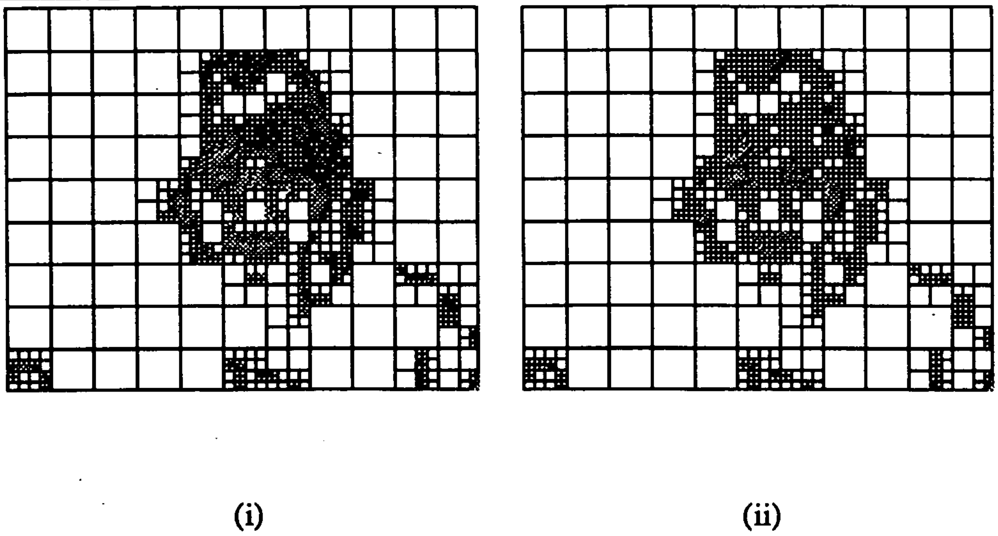


Figure 5.13: *(i) Block structure after motion estimation (ii) block structure after adpative relabelling of (i).*

motion estimation threshold to generate a complex decomposition; leaves and nodes are then merged and relabelled to achieve the desired rate. For example, if the two fail nodes in figure 5.12(i) were to be changed to pass nodes a reduction in rate would occur and the accuracy of the motion estimate would be preserved.

The simplest method of removing any remaining redundancy from the tree is to parse the tree structure and compare the magnitude of the prediction error at each failed leaf node to a new threshold. A failed leaf is then relabelled as passed if the magnitude of the prediction error is less than the new threshold, the relabelled leaves are then merged with any neighbouring leaves. This operation increases the computational overhead by a very small amount since the magnitude of the prediction error and motion vector for each node can be stored in a 'pointer' structure in the processor memory.

Figure 5.13(i) shows a segmentation generated for a frame of the Miss America sequence using a motion estimation threshold of 0.001. As with figure 5.2 white regions denote blocks which can be copied and coloured regions denote blocks which cannot be copied. Figure 5.13(ii) shows the result of parsing the tree and adaptively relabelling fail blocks as pass blocks using a 'relabel' threshold of 0.01. This results in a large reduction in the number of failed blocks, and as such there is a corresponding

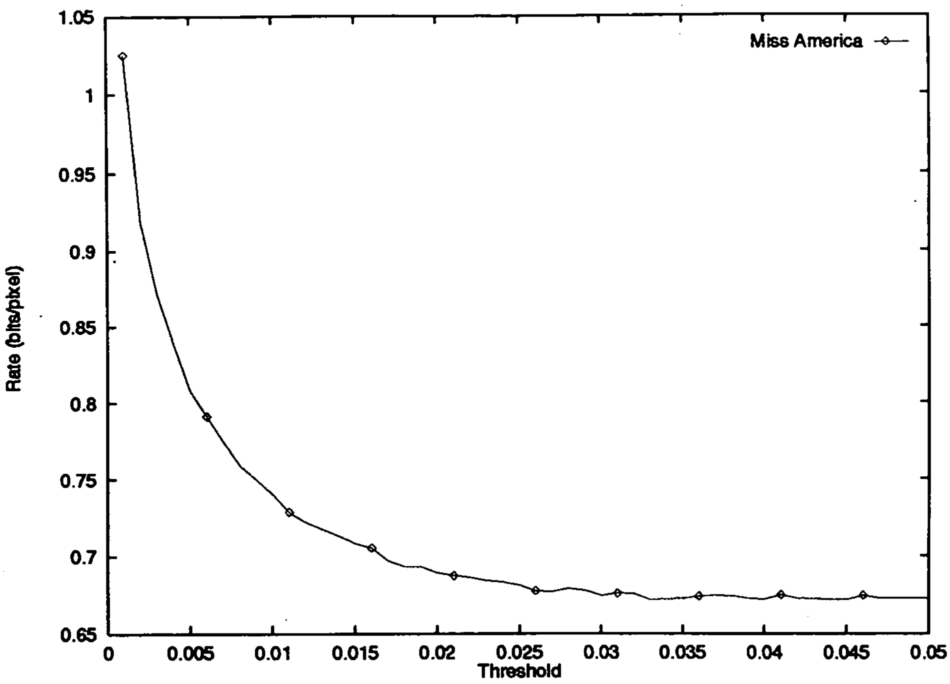


Figure 5.14: *Rate required to encode the Miss America sequence for various values of 'relabel' threshold, using a motion estimation threshold of 0.001.*

reduction in rate (in this case 0.3 bit/pixel).

Relabelling fail leaves as pass leaves results in a substantial decrease in the number of fail blocks in an image. The total number of fail blocks which remain after parsing the trees depends upon the relabel threshold, experiments with the Miss America sequence show that less than 10 fail blocks per image are found when the relabel threshold is greater than 0.03. This supports the claim in section 5.4.6 that the failed blocks only need to be encoded when large changes occur.

Figure 5.14 shows the average rate at which the Miss America sequence can be encoded for various values of 'relabel' threshold, using a motion estimation threshold of 0.001. The minimum rate achieved for the range of thresholds shown is 0.68 bit/pixel. The picture quality at this rate is excellent. For a greyscale image which has dimensions of 176 × 144 pixels encoded at a frame rate of 10 Hz this corresponds to an average rate of 172 kbits/s, which is much greater than the desired rate of 64 kbits/s.

A further reduction in rate can be achieved by increasing the motion estimation threshold so that the initial decomposition is less complex. However, manipulating two thresholds to achieve the desired rate is unnecessarily complex and the same result

Image	Entropy (bits/vector)
Miss America	6.8
Clare	6.2
Salesman	5.4

Table 5.4: *Entropy of vectors and failed data. Results generated from using a threshold of 0.001*

can be achieved by improving the tree parsing algorithm. For example, the algorithm used to generate the above results, relabels fail leaves as pass leaves and then merges any redundant leaves. This algorithm can be extended by adding a further stage which reduces the complexity of the trees by adaptively merging nodes to achieve the desired rate.

The estimate of the minimum rate at which an image can be encoded, which was made in section 5.4.4, can be usefully employed with the above problem. For example, it can be seen from figure 5.14 that the minimum rate achievable for a motion estimation threshold of 0.001 is 0.62 bit/pixel. If rates of less than 0.62 bit/pixel are required then the tree structure must be simplified by merging some leaf nodes.

5.4.8 Vector entropy

In the previous sections the displacement vectors were PCM encoded at 10 bits/vector to allow a displacement of ± 15 pixels in both dimensions. A further reduction in rate can be achieved by entropy encoding the vectors. Table 5.4 shows the entropies of the vectors for the Miss America, Clare, and Salesman standard image sequences generated for a motion estimation threshold of 0.001 and relabel threshold of 0.05.

Clearly, entropy encoding of the displacement vectors will result in a reasonable reduction in the rate. For example, if the vectors for the Miss America sequence were encoded at rate equal to the entropy of the vectors the Miss America sequence could be encoded at an average rate of approximately 0.49 bit/pixel. However, in practice it is not always possible to achieve the entropy bound and a slight increase in rate will occur. As yet, variable length coding algorithms have not been included in the simulations and the above results represent an approximation to the actual results.

5.5 Stationary background prediction

The computational load imposed by the variable block size motion estimation far exceeds that of normal block matching algorithms. In chapter 3 heuristics and paradigms were described which could be used to port image coding algorithms to a parallel computer. The variable block size motion estimation could also be implemented on a parallel computer using the methods described in chapter 3 to achieve the desired performance. However, it is still desirable to reduce the computational overhead required to make a motion estimate. This will be achieved using a stationary background prediction algorithm.

Conventional algorithms for the identification of stationary background are based upon change detectors which compare the magnitude of the frame difference to a threshold value [77,100–104]. In general, segmented images contain isolated regions or isolated single pixels which are caused by incorrect segmentation of the image. Clearly, the number and size of these isolated regions depends upon the chosen threshold. These small isolated regions require too much information to encode and as such should be removed to improve compression efficiency. Various algorithms have been suggested for the removal of isolated regions. For example, Bierling [77] uses a 5 × 5 median filter to remove isolated single pixels. More sophisticated algorithms [102–104] have been introduced which compare the size of a region to a threshold to decide whether a region should be merged into the surrounding region. The removal of isolated regions can be combined with an adaptive threshold estimate by assuming that all regions marked as stationary background contain noise.

For the purposes of variable block size motion estimation a simplified stationary background segmentation is used, based upon a quadtree decomposition of the frame difference image. The quadtree decomposition is terminated at a block size of 8 × 8 pixels so that isolated regions are not formed. Figure 5.15 shows a typical stationary background segmentation from the Miss America standard image sequence. The

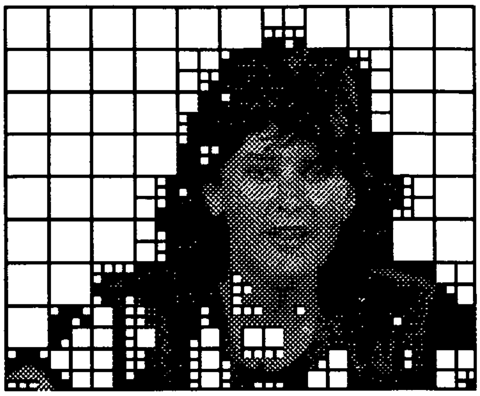


Figure 5.15: *A typical stationary background segmentation from the Miss America standard sequence using a threshold,* $T = 0.001$.

threshold used to determine whether a block is part of the stationary background is the same as that used for the variable block size motion estimation algorithm.

Although, the segmentation has correctly identified the majority of the stationary background there are some regions which are incorrectly segmented. For instance, there are small clusters of blocks in the hair and on the throat of Miss America which are labelled as stationary background and which inreality are part of the moving region. These regions are only a small part of the total image area and as such no attempt has been made to remove them. They do not significantly degrade the quality of the reconstructed image but they do result in a slight increase in rate.

Figure 5.16 compares the rate required to encode the Miss America sequence[1] using a stationary background prediction followed by motion estimation. As predicted, employing stationary background prediction results in a slight increase in rate for low values of threshold. However, as the threshold is increased the deviation between the results for the stationary background prediction and the full search becomes greater. This is because the magnitude of the prediction error is so large that increasing the threshold does not decrease the complexity of the stationary background prediction.

The aim of employing a stationary background prediction is to reduce the computational overhead required to make a motion estimate. Clearly, figure 5.15 shows that

---

[1] PCM encoding the replenished blocks.

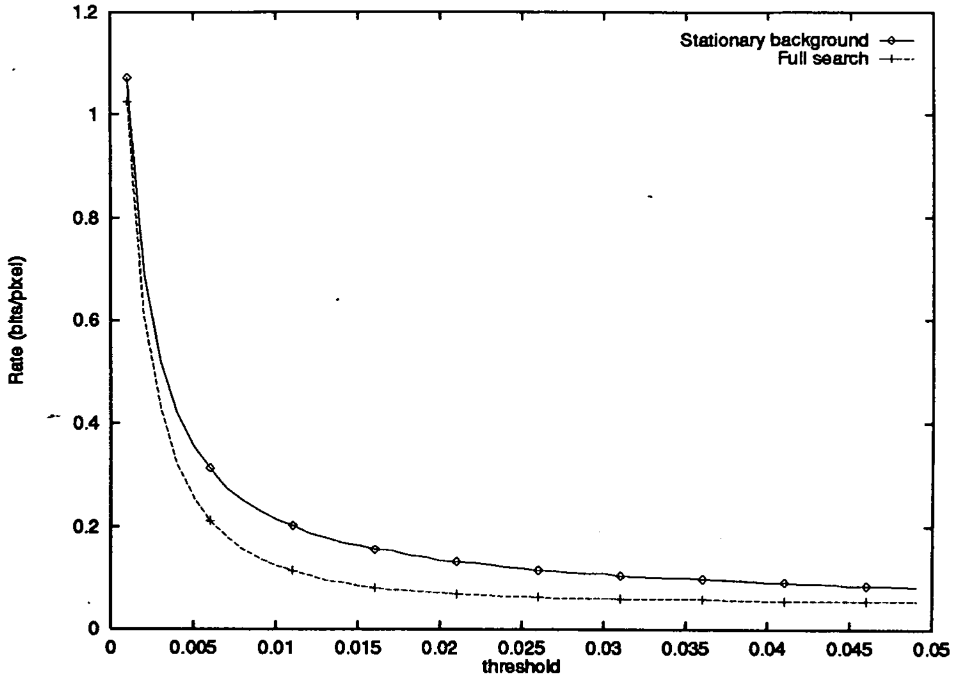


Figure 5.16: *Comparison of rates to encode Miss America using stationary background prediction and full search.*

a large area of an image can be described as stationary background and as such a proportional decrease in computational overhead is expected. Experiments with the Miss America sequence show that stationary background prediction results in a decrease in a execution time of 40-60%.

5.6 Comparison with related work

There are many inherent difficulties in comparing different image coding schemes. One of the biggest problems is due to the fact that there is little or no agreement on what represents a reasonable measure of coding distortion. In general, distortion results are quoted in terms of mean square error or SNR and are then qualified by a subjective statement concerning the perceived image quality. Although it is possible to directly compare such measures as mean square error or SNR it is almost impossible to compare the perceived image quality without laboriously recreating a codec so that direct comparison between reconstructed images is possible. Therefore, in the following only general remarks are made about the comparison with this and other related work.

The variable block size conditional replenishment encoder is similar to Strobach's tree-structured scene adaptive coder. Both algorithms are based upon a quadtree decomposition of the prediction error. However, in comparison to Strobach's encoder the variable block size conditional replenishment encoder uses an enhanced motion estimation algorithm to allow blocks to be directly copied between frames. This results in less distortion being introduced but only allows one quadtree decomposition to be calculated per block, which as discussed in section 5.4.5 may result in a suboptimum decomposition being employed.

A minimum rate of 0.3 bit/pixel was reported, which corresponds to a bandwidth of approximately 76 kbits/s for a frame rate of 10 Hz and an image size of 176 × 144 pixels. This is greater than the 61 kbits/s reported by Strobach but as yet entropy encoding of the vectors has not been included in the variable block size conditional replenishment scheme. Also if the suggestions for future work which are described in chapter 6 are investigated it is anticipated the bandwidth of less than 61 kbits/s could be achieved.

5.7 Summary and conclusions

A new algorithm was introduced which encodes sequences of images based upon conditional replenishment. The algorithm employs a variable size block matching algorithm to segment an image into regions which can be encoded using a displacement vector and regions which must be replenished.

The threshold used to control the VSBMA allowed greater distortion to be introduced in brighter regions of the image where the HVS is less sensitive to intensity perturbations. The threshold was found to ignore single isolated pixels but was still sensitive to small isolated regions of pixels and hence produced accurate segmentations.

The regular structure of a decomposition allowed simple rules to be developed which determine whether compression will or will not occur. These rules were employed for calculating the height of tree so that compression was always ensured.

The copying operation does not introduce unsightly artifacts since for sufficiently accurate motion estimates the prediction error can be approximated to be white noise.

Initial simulation results were presented based upon PCM encoding of the failed leaves. It was noted that these results represented the upper bound to the rate and the lower bound to the distortion. Compression was found to occur for a minimum block size of 2×2 pixels and gave rates in the range 0.5-0.9 bits/pixel for good image quality.

In general, PCM encoding of the failed blocks is unsuitable for most practical implementations since the rate required to encode the failed regions is too high. Different algorithms for encoding the failed blocks were discussed. It was noted that the coding gain of many spatial encoders is area dependent and that the operation of the VBCR encoder would be fundamentally modified. BTC was employed to demonstrate the effect of including such a spatial coding algorithm. Although the results from BTC were found to be satisfactory it was noted that other more sophisticated algorithms, such as transform coding, could be employed to produce better results.

Attempts at controlling rate whilst maintaining the image fidelity via the motion estimation threshold demonstrated that only a limited range of values could be employed. This is because as the threshold is increased the complexity of a decomposition decreases producing progressively coarser motion estimates. A solution to this problem was suggested, based upon relabelling fail blocks as pass blocks depending upon their contribution to the total distortion.

Finally, a stationary background prediction algorithm was employed to reduce the computational overhead of the VSBMA. A simple algorithm was employed which performed a regular decomposition on the frame difference image using the same threshold as the VBCR encoder. To obviate the need for the removal of small isolated regions in the stationary background prediction the height of the tree was set so that the smallest block size was 8×8 pixels. The stationary background prediction algorithm was found to reduce the overall computation by 40-60%.

Chapter 6

Conclusions

The research reported in this thesis has investigated methods of efficiently encoding image sequences. The fundamental aim was to generate an algorithm for encoding images which maintained the fidelity of the reconstructed images at low bit-rates. To obtain this objective, motion estimation and motion-compensated prediction were investigated in detail.

An assumption often made by many researchers is that infinite computational power is available. Algorithms are created which are far too complex to implement efficiently in hardware with the assumption that technology will eventually provide a solution. To avoid this problem, parallel implementations of image coding algorithms were also studied.

6.1 Summary

Chapter 3 investigated the problems of porting image coding algorithms to reconfigurable, distributed-memory MIMD parallel computer architectures. The aim of this research was twofold: firstly, develop heuristics and paradigms which could be reliably employed to map image coding algorithms to a MIMD architecture; and secondly, develop a high performance simulation tool for developing image compression algorithms. To illustrate the potential gains from employing a parallel architecture the H.261 standard was mapped onto an array of T800 transputers.

Since image coding algorithms are frequently applied to independent blocks of data, mapping the data space rather than the problem space was found to give a rich source of potential parallelism. Functional and data decompositions of H.261 were compared and it was demonstrated that data decompositions are more efficient.

The use of topology independent routing harnesses was found to significantly simplify the task of routing communications, thus allowing the programmer to easily experiment with different processor topologies. Three compact topologies were employed: the random graph, the greedy graph and the chordal ring. An important discovery was that for less than 20 transputers the execution time of a program is almost independent of the processor topology.

The simulation results demonstrated that although close-to-real-time performance could be achieved using 20 transputers the execution time was communication bound. This problem was caused by the T800 architecture which has a relatively small communication bandwidth. This problem could be easily solved by employing a more sophisticated processing element which has a greater communication bandwidth such as the Texas Instruments C40 range of products.

Motion estimation for image coding was discussed in chapter 4. Motion-compensated prediction, which is employed in many state-of-the-art encoders, was discussed in detail. Simulation results were presented to show how the entropy and energy of motion-compensated prediction errors vary when different block-matching algorithms are employed.

A model of the prediction error was developed based upon the assumption that the prediction error is caused by the failure to correctly estimate the displacement of the objects within the image. It was shown from this model that a typical prediction error will contain high intensity lines in regions where the original image contains high spatial gradients. The model was also used to approximate the error surface generated by a block matching algorithm. The accuracy of the motion estimate was shown to be dependent upon the contents of a block:

- Motion estimates for blocks which contain pixels with an almost uniform distribution of pixel intensities can easily be corrupted by noise.
- Motion estimates for blocks in which there are large changes in intensity are robust to noise.

Thus displacement estimates for object edges may be very accurate although conversely the magnitude of the prediction error for these displacement estimates may be large.

The approach adopted previously was further simplified to allow a model of the spectral properties of the prediction error to be developed. Motion-compensated prediction was shown, for sufficiently accurate displacement estimates, to be equivalent to a non-ideal high-pass spatial filter. This model was used to approximate the energy gain of motion-compensated prediction using an isotropic correlation function. The theoretical and experimental results were demonstrated to agree to within the limitations imposed by the model.

The signal-to-noise coding gain of the optimum hybrid motion-compensated codec was shown to be asymptotically bounded by the inverse spectral flatness measure of the prediction error. Since motion-compensated prediction can be approximated to be a non-ideal high-pass spatial filter and real-world images are approximated to contain predominantly low frequency information, the prediction error was shown have a greatly increased SFM. Spectral flatness measures of approximately 0.9 were predicted using the theoretical model, indicating that only small coding gains could be expected. Measurement of the coding gain for actual prediction errors demonstrated that a significant increase in SFM does occur. However, the measured values are smaller than those predicted by theory which was attributed to the simplicity of the adopted model.

The importance of employing a spatial encoder which is optimised for compressing the prediction error was highlighted at the end of this chapter.

A new algorithm was introduced in chapter 5 which attempts to exploit the remaining redundancy in the prediction error. A variable size block matching algorithm was employed to segment an image so that it could be encoded efficiently using conditional replenishment. The regular structure of the decomposition allowed simple mathematical rules to be formulated which ensured that compression would always occur. The reconstruction error in the copied regions was simply the prediction error, which for sufficiently accurate displacement estimates can be approximated to be white noise. It was observed that the algorithm preserved the frequency content of an image and as such maintained the fidelity of the edges.

Initially, simulation results were presented for PCM encoding of the failed leaves of the regular decomposition. It was noted that these results represent the upper bound to the rate and the lower bound to the distortion. High quality images were given for rates between 0.5-0.9 bit/pixel but the image quality degraded rapidly as the rate was further reduced.

PCM encoding of the failed leaves is, in general, unsatisfactory since if large areas of the image are labelled failed, the rate required to encode an image will increase dramatically. Different spatial coding algorithms were discussed and it was noted that the coding gain of many of these algorithms is area dependent. A BTC encoder was employed to illustrate the effect of including a spatial coding algorithm. Despite the obvious simplicity of the BTC encoder reasonable quality images were obtained in the range 0.3-0.5 bit/pixel. It was observed that more sophisticated algorithms such as transform coders or vector quantizers would give better results.

Attempts at controlling rate and distortion using the motion estimation threshold demonstrated that only a limited range of values could be employed. This problem was solved by using a small motion estimation threshold to generate a complex decomposition, leaf nodes were then adaptively relabelled and merged to achieve the required rate or distortion. It was noted that if all the leaves in a decomposition were encoded with vectors this represented the lower bound to the rate and the upper bound to the distortion. This result can be usefully employed with the merging operation.

Entropy coding the displacement vectors was also shown to reduce the rate required to encode an image. Reductions in rate of approximately 0.2 bit/pixel were observed for a motion estimation threshold of 0.001.

Finally, a simple stationary background prediction algorithm was employed to reduce the computational requirements of the VSBMA. For head and shoulder type sequences reductions in the computational overhead of 40-60% were observed.

The variable block size conditional replenishment encoder was demonstrated to maintain the fidelity of the reconstructed images, even at very low rates.

6.2 Future work

The variable block size conditional replenishment encoder was demonstrated to be a powerful algorithm for encoding images sequences. The regular nature of the decomposition allowed simple rules to be formulated which ensure compression will always occur. Since each tree and its corresponding displacement estimates can easily be stored in processor memory, the trees can be manipulated to achieve the desired rate and distortion. In section 5.4.7 an algorithm was described which employed a threshold to reduce the rate required to encode a tree. However, it is difficult to determine the magnitude of the threshold required to achieve a particular rate, a poor estimate of the required threshold can result in a reduction in the image fidelity.

The desired rate could be achieved by sorting the nodes of the trees according to their contribution to the total distortion. The nodes which least contribute to the total distortion could then be relabelled and merged. Although this approach is guaranteed to find the minimum value of distortion for a given value of rate it is very computationally expensive. Further work is required to investigate whether this operation could be achieved in real-time. However, even if real-time processing is not possible the above algorithm could be employed for encoding recorded video, where real-time encoding is not important.

Further investigation into the sensitivity of the object position within the image when using the variable block size conditional replenishment algorithm is also required. At present, this sensitivity can result in a very complex decomposition being generated when a simple decomposition would suffice. Strobach overcomes this problem in his tree-structured scene adaptive coder [9] by repeated calculation of the initial decomposition for a number of different block positions to find the best solution. However, this solution cannot be employed with the variable block size motion estimation algorithm since the computational overhead would be excessive.

A simple and elegant solution to this problem could be to perform merging across block boundaries to produce irregular regions. These irregular regions could then be described using the contour which bounds them [105]. This approach should have several potential benefits:

1. The sensitivity to object position is removed.
2. Motion estimates can be made for single pixels.

For an irregular decomposition the rate required to encode an image is dependent upon the length of the perimeter of a region. Thus, regions which have complex boundaries require a greater number of bits. For many types of images the boundaries between objects are not complex and as such the above should not be a problem. However, motion estimates for single pixels are easily corrupted by noise and ragged boundaries may occur. Therefore, before two regions could be merged a decision must be made to ensure that a sufficient gain would be achieved. Wagner has described such a scheme which could be used which is based upon comparing the improvement to the reconstructed image versus the bit cost [106].

Using an irregular decomposition could also have important repercusions for the stationary background prediction. The isolated regions caused by incorrect segmentation of the frame difference could no longer be tolerated. Algorithms described in chapter 5 could be used to remove these regions.

References

[1] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Prentice-Hall, 1984.

[2] R. J. Clarke, *Transform Coding of Images.* Academic Press, 1985.

[3] H. G. Musmann, P. Pirsch, and H. Grallert, "Advances in picture coding," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523–548, 1985.

[4] M. Kunt, A. Ikonomopolous, and M. Kocher, "Second-generation image coding techniques," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 549–574, 1985.

[5] A. K. Jain, "Image compression: A review," *Proceedings of the IEEE*, vol. 69, no. 3, pp. 349–389, 1981.

[6] R. Forchheimer and T. Kronander, "Image coding — from waveforms to animation," *IEEE Transactions on ASSP*, vol. 37, no. 12, pp. 2008–2023, 1989.

[7] "Draft revision of recommendation H.261: Video codec for audiovisual services at $p\times$ 64kbits/s," *Signal Processing: Image Communication*, vol. 2, pp. 221–239, October 1990.

[8] D. J. Gall, "The MPEG video compression algorithm," *Signal Processing : Image Communication*, vol. 4, pp. 129–140, 1992.

[9] P. Strobach, "Tree-structured scene adaptive coder," *IEEE Transactions on Communications*, vol. COM-38, pp. 477–486, April 1990.

[10] B. Girod, "The efficiency of motion-compensating predication for hybrid coding of video sequences," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 1140–1154, August 1987.

[11] A. N. Netravali and J. O. Limb, "Picture coding: A review," *Proceedings of the IEEE*, vol. 68, pp. 366–406, March 1980.

[12] B. Girod, "Psychovisual aspects of image communication," *Signal processing*, vol. 28, pp. 239–251, 1992.

[13] R. B. Ash, *Information theory.* Interscience publishers, 1965.

[14] D. Huffman, "A method of constructing minimum redundancy codes," *Proceedings of the IRE*, pp. 1098–1101, September 1952.

[15] L. D. Davisson, "Rate-distortion theory and application," *Proceedings of the IEEE*, vol. IT-60, pp. 800–808, July 1972.

[16] A. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, January 1980.

[17] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Transactions on Communications*, vol. COM-36, pp. 957–971, August 1988.

[18] M. Barnsley, "A better way to code images," *BYTE*, pp. 215–223, 1988.

[19] Y. Fisher, "Fractal image compression," in *SIGGRAPH '92*, pp. 1–21, 1992.

[20] F. I. Parke, "Parameterized models for facial animation," *IEEE Computer Graphics and Applications*, vol. 2, pp. 61–68, November 1982.

[21] W. J. Welsh, S. Searby, and J. B. Waite, "Model-based image coding," *British Telecom Technology Journal*, vol. 8, pp. 94–106, July 1990.

[22] W. J. Welsh, "Model-based coding of videophone images," *Electronics and Communications Engineering journal*, vol. 3, pp. 29–36, February 1991.

[23] G. K. Wallace, "The JPEG still compression standard," *IEEE Transactions on Consumer Electronics*, vol. CE-38, no. 1, pp. xvii–xxxiv, 1992.

[24] M. W. Whybray, "Video codec design using DSP devices," *British Telecom Technology Journal*, vol. 10, no. 1, 1992.

[25] E. Gelenbe, *Multiprocessor Performance*. John Wiley and Sons, 1989.

[26] I. LTD, *Transputer reference manual*. Prentice Hall, UK, 1988.

[27] D. J. Wallace, "Supercomputing with transputers," *Computer systems in Engineering*, vol. 1, no. 1, pp. 131–144, 1990.

[28] J. A. Elliott, J. A. Beaumont, and P. M. Grant, "Techniques for motion video processing on transputer-based reconfigurable multicomputers," in *IEE colloquium on Parallel Architectures for Image Processing Applications*, no. 086, pp. 7/1–7/5, 1991.

[29] D. May and P. Thomson, "Transputer and routers: components for concurrent machines," in *Transputer/occam* (T. L. Kunii and D. May, eds.), pp. 3–20, IOS press, Netherlands, 1990.

[30] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978.

[31] J. Wexler, *Concurrent Programming in occam 2*, vol. 3 of *6*. Edinburgh: Ellis Horwood, 9 ed., Jan 1989.

[32] J. Yantchev and C. R. Jesshope, "Adaptive, low latency, deadlock-free packet routing for networks of processors," *IEE Proceedings Part E*, vol. 136, no. 3, pp. 176–186, 1989.

[33] L. J. Clarke and G. Wilson, "Tiny: an efficient routing harness for the inmos transputer," *Concurrency: Practice and Experience.*, vol. 3, no. 3, pp. 221–245, 1990.

[34] A. Evans and N. Coulson, *CS Tools for MeikOS*, vol. 83-009 A10-01.00. Meiko Limited, 1990.

[35] S. Bokhari, "On the mapping problem," *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 433–442, 1981.

[36] S. Y. Lee and J. K. Aggarwal, "A mapping strategy for parallel processing," *IEEE Transactions*, vol. C-37, no. 4, pp. 207–214, 1988.

[37] H. Shen, "Self-adjusting maps: a heuristic mapping algorithm for mapping parallel programs onto transputer networks.," in *Proceedings of the 11th occam user group technical meetings*, pp. 89–98.

[38] N. Udiavar and G. S. Stiles, "A simple but flexible model for determining optimal task allocation and configuration on network of transputers," in *Proceedings of Transputer research and applications*, pp. 25–32, 1990.

[39] M. Ichikawa and K. Shomura, "Design and implementation of software based real-time video codec using multi-transputer architecture," in *Proceedings of Transputer/occam Japan*, vol. 3, pp. 91–100, 1990.

[40] G. Sexton and K. Hawick, "A transputer implementation of the CCITT reference model 6 video coding algorithm," *Mailshot, SERC/DTI transputer initiative*, pp. 67–70, 1989.

[41] S. H. Bokhari, "Partitioning problems in parallel, pipelined, and distributed computing," *IEEE Transactions*, vol. C-37, no. 1, pp. 48–57, 1988.

[42] B. Rudberg, M. N. Chong, S. Marshall, and J. J. Soraghan, "Transputer topologies for image sequence transmission," in *IEE colloquium for 'Parallel architectures for image processing applications'*, vol. 1991/086, pp. 9/1–9/5, 1991.

[43] A. J. G. Hey, "Scientific applications on transputer arrays – some experiments in mimd parallelism," *Proc. Transputer/occam Japan*, vol. 3, pp. 103–121, 1989.

[44] J. A. Elliott, J. M. Bueamont, P. M. Grant, and J. T. E. McDonnell, "Real time videophone image coding algorithm simulation on a concurrent supercomputer," in *Proceedings of the IEE Sixth International Conference on Digital Processing in Communications*, pp. 89–94, 1991.

[45] A. Ghafoor, S. Sheikh, and P. Sole, "Bipartite distance regular interconnection topology for fault-tolerant multi-processor systems," *IEE Proceedings part E*, vol. 137, pp. 173–184, 1990.

[46] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 3 ed., 1991.

[47] D. M. N. Prior, N. J. Radcliffe, and N. G. Norman, "What price regularity?," *Concurrency: Practice and Experience*, pp. 55–78, 1990.

[48] B. W. Arden and H. Lee, "Analysis of chordal ring networks," *IEE Transactions on Computers*, vol. C-30, no. 4, pp. 291–295, 1981.

[49] R. F. Browne and R. M. Hodgson, "Symmetric degree-four chordal ring networks," *IEE Proceedings E*, vol. 137, pp. 310–316, 1990.

[50] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. MIT press, MA, 1990.

[51] B. W. Kernighan and D. M. Ritchie, *The C programming language*. Prentice-Hall, NJ, 1978.

[52] T. S. Durrani, W. A. Sandham, and J. J. Soraghan, eds., *Applications of transputers 3*, IOS Press, Netherlands, 1991.

[53] A. N. Netravali and J. D. Robbins, "Motion-compensated television coding: Part 1," *The Bell System Technical Journal*, vol. 58, pp. 631–670, March 1979.

[54] D. Walker and K. Rao, "Improved pel-recursive motion estimation," *IEEE Transactions on Communications*, vol. COM-32, pp. 1128–1134, October 1984.

[55] R. J. Moorhead, S. A. Rajala, and L. W. Cook, "Image sequence compression using pel-recursive motion compensation technique," *IEEE Journal on Selected Areas in Communication*, vol. SAC-5, pp. 1100–1114, August 1987.

[56] J. Biemond, L. Looijenga, D. Boekee, and R. H. J. M. Plompen, "A pel-recursive Wiener based displacement estimation algorithm," *Signal Processing*, vol. 13, pp. 399–412, December 1987.

[57] R. P. Kleihorst, R. L. Lagendijk, and J. Biemond, "Motion estimation from noisy image sequences," in *Proceedings of PCS '93*, pp. 4.6–4.7, Picture coding symposium, 1993.

[58] J. R. Jain and A. K. Jain, "Displacement measure and its application in interframe image coding," *IEEE Transactions on Communications*, vol. COM-29, pp. 1799 - 1808, December 1981.

[59] M. F. Zanuy and F. T. Ruiz, "A modification of the conjugate direction for motion estimation," in *Signal Processing VI: Theories and Applications*, pp. 1311–1314, EUSIPCO, 1991.

[60] M. Bierling, "Displacement estimation by hierarchical block matching," *SPIE Visual Communication and Image Processing'88*, vol. 1001, pp. 942–951, 1988.

[61] M. H. Chan, Y. B. Yu, and A. G. Constantinides, "Variable size block matching motion compensation with applications to video coding," *IEE Proceedings Pt. I*, vol. 137, pp. 205–212, August 1990.

[62] Q. Wang and R. J. Clarke, "Motion compensated sequence coding using image pyramids," *Electronics letters*, vol. 26, no. 9, pp. 575–576, 1990.

[63] Q. Wang and R. J. Clarke, "Motion estimation and compensation for image sequence coding," *Signal Processing : Image Communication*, vol. 4, no. 2, pp. 161–174, 1992.

[64] M. H. A. Fadzil and T. J. Dennis, "A hierarchical motion estimator for interframe coding," in *IEE Colloquium on Applications of Motion Compensation*, vol. 1990/128, pp. 3/1–3/6, 1990.

[65] M. H. A. Fadzil and T. J. Dennis, "Video subband VQ coding at 64kbits/s using short-kernel filter banks with an improved motion estimation technique," *Signal processing: Image communication*, vol. 3, no. 1, pp. 3–21, 1991.

[66] R. J. Clarke, "Basic pricinples of motion estimation and compensation," in *IEE Colloquium on 'Applications of motion compensation'*, vol. 1990/128, pp. 1/1–1/7, 1990.

[67] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Transactions on Communications*, vol. COM-41, no. 4, pp. 604–612, 1993.

[68] H. Li and R. Forchheimer, "A new motion-compensated technique for video compression," in *ICASSP*, pp. v441–v444, 1993.

[69] V. Seferidis and M. Ghanbari, "General-approach to block matching motion estimation," *Optical engineering*, vol. 32, no. 7, pp. 1464–1474, 1993.

[70] C. A. Papadopoulos and T. G. Clarkson, "Data compression of moving images using geometric transformations," in *IEE 4th International conference on image processing and its applications*, (April), pp. 101–105, 1992.

[71] C. A. Papadopoulos and T. G. Clarkson, "Motion compensation using 2nd order geometric transformations," *Electronic letters*, vol. 28, pp. 2314–2315, December 1992.

[72] C. A. Papadopoulos and T. G. Clarkson, "The use of 2nd order geometric transformations in interframe image data compression," in *IEEE Global Telecommunications conference,*, pp. 1309–1304, November 1993.

[73] S. Wu and J. Kittler, "A differential method for simultaneous estimation of rotation, change of scale and translation," *Signal Processing: Image Communication*, vol. 2, pp. 69–80, 1990.

[74] A. Amitay and D. Malah, "Global motion estimation in image-sequences of 3-D scenes for coding applications," in *Proceedings of the Picture Coding Symposium '93*, pp. 10.1–10.2, 1993.

[75] K. Illgner, C. Stiller, and F. Müller, "A robust zoom and pan estimation technique," in *Proceedings of Picture Coding Symposium '93*, pp. 10.5–10.6, 1993.

[76] C. Cafforio, F. Rocca, and S. Tubaro, "Motion compensated image interpolation," *IEEE Transactions of Communications*, vol. COM-38, pp. 215–222, February 1990.

[77] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimator," *Signal Processing*, vol. 11, pp. 387–404, December 1986.

[78] I. Parke, "A hardware motion compensator for videoconferencing codec," in *IEE Colloquium on 'Applications of motion compensation'*, vol. 1990/128, pp. 6/1–6/3, 1990.

[79] D. J. Connor and J. O. Limb, "Properties of frame-difference signals generated by moving images," *IEEE Transactions on Communications*, vol. COM-23, pp. 1564–1575, October 1974.

[80] C. D. McGillem and G. R. Cooper, *Continuous and discrete signal and system analysis.* CBS Publishing Japan Ltd., 2 ed., 1986.

[81] J. B. O'Neal and T. R. Natarajan, "Coding isotropic images," *IEEE Transactions on Information Theory*, vol. IT-23, pp. 697–707, November 1977.

[82] R. J. Clarke, "On transform coding of motion-compensated difference images," *IEE Proceedings part I*, vol. 139, no. 3, pp. 372–376, 1992.

[83] K. S. Fu and J. K. Mui, "A survey on image segmentation," *Pattern Recognition*, vol. 13, pp. 3–16, 1981.

[84] R. C. Gonzalez and R. E. Woods, *Digital image processing.* Addison-Wesley Publishing Company, 1992.

[85] A. Klinger and C. R. Dyer, "Experiments on picture representation using regular decomposition," *Computer Graphics and Image Processing*, vol. 5, pp. 68–105, 1976.

[86] M. A. Oliver and N. E. Wiseman, "Operations on quadtree encoded images," *The Computer Journal*, vol. 26, no. 1, pp. 83-91, 1983.

[87] F. W. Mounts, "A video encoding system with conditional picture-element replenishment," *The Bell System Technical Journal*, pp. 2545–2554, 1969.

[88] D. N. Hein and N. Ahmed, "Video compression using conditional replenishment and motion prediction," *IEEE Transactions on Electromagnetic Compatibility*, vol. EMC-26, pp. 134–142, August 1984.

[89] M. Ghanbari, "Motion vector replenishment for low bit-rate video coding," *Signal Processing: Image Communication*, vol. 2, pp. 397–407, 1990.

[90] Q. Wang and R. J. Clarke, "A new motion-compensated image coding scheme at 64 kbit/s," *IEE Proceedings Pt I*, vol. 139, pp. 219–223, April 1992.

[91] E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," *IEEE Transactions on Communications*, vol. COM-27, no. 9, pp. 1335–1342, 1979.

[92] D. J. Healy and O. R. Mitchell, "Digital video bandwidth compression using block truncation coding," *IEEE Transactions in Communications*, vol. COM-29, no. 12, pp. 1809–1817, 1981.

[93] G. Z. Arce and J. N. C. Gallagher, "BTC image coding using median filter roots," *IEEE Transactions on Communications*, vol. COM-31, no. 6, pp. 784–793, 1983.

[94] D. R. Halverston, N. C. Griswold, and G. L. Wise, "A generalised block truncation coding algorithm for image compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 3, pp. 664–668, 1984.

[95] V. R. Updikar and J. P. Raina, "BTC image coding using vector quantization," *IEEE Transactions on Communications*, vol. COM-35, no. 3, pp. 352–356, 1987.

[96] N. C. Griswold, D. R. Halverston, and G. L. Wise, "A note on adaptive block truncation coding for image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 8, pp. 1201–1203, 1987.

[97] E. J. Delp and O. R. Mitchell, "Moment preserving quantization," *IEEE Transactions of Communications*, vol. COM-39, no. 11, pp. 1549–1558, 1991.

[98] E. J. Delp and O. R. Mitchell, "The use of block truncation coding in DPCM coding of images," *IEEE Transactions on Signal Processing*, vol. SP-39, no. 4, pp. 967–971, 1991.

[99] J. U. Roy and N. M. Nasrabadi, "Hierarchical block truncation coding," *Optical Engineering*, vol. 30, no. 5, pp. 551–556, 1991.

[100] N. Mukawa and H. Kuroda, "Uncovered background prediction in interframe coding," *IEEE Transactions on Communications*, vol. COM-33, pp. 1227–1231, November 1985.

[101] D. Hepper, "Efficiency analysis and application of uncovered background prediction in a low bit rate image coder," *IEEE Transactions on Communications*, vol. 38, pp. 1578–1584, September 1990.

[102] M. Hotter and R. Thoma, "Image segmentation based on object-oriented mapping estimation," *Signal Processing*, vol. 15, pp. 315–334, 1988.

[103] R. Thoma and M. Bierling, "Motion compensating interpolation considering covered and uncovered background," *Signal Processing: Image Communication*, vol. 1, pp. 191–212, 1989.

[104] N. Diehl, "Object-oriented motion estimation and segmentation in image sequences," *Signal Processing: Image Communication*, vol. 3, no. 1, pp. 23–56, 1991.

[105] M. Soryani and R. J. Clarke, "Image segmentation and motion-adaptive frame and interpolation for coding image sequences," in *ICASSP 89*, pp. 1882–1885, 1989.

[106] P. Wagner and B. Girod, "Region-based motion field estimation," in *Picture Coding Symposium*, pp. 4.5–4.6, 1993.

Appendix A

Original publications

The work described in this thesis has been reported in the following publications:

- J. A. Elliott, C. Cubiss, P. M. Grant and J. T. E. McDonnell, 'Real-time simulation of videophone image coding algorithms on reconfigurable multicomputers', Proceedings IEE, vol. 139, Part E, Computers and Digital Techniques, No. 3, pp. 269-279, May 1992.
- † C. Cubiss, P. M. Grant and J. T. E. McDonnell. 'Variable block size motion compensated conditional replenishment coder, in the Proceedings of the Picture Coding Symposium, Lausanne, Switzerland, March 1993.

Papers submitted:

- C. Cubiss, P. M. Grant and J. T. E. McDonnell, 'Region-based motion-compensated replenishment', submitted to Euspico 1994, Edinburgh, Scotland.

In preparation:

- C. Cubiss, P. M. Grant and J. T. E. McDonnell, 'Tree-structured motion-compensated replenishment', in preparation for Proceedings IEE, Vision, Image and Signal Processing.

† Reprinted in this appendix.

Appendix B

Rate-distortion theory for transform coding

The KLT is an optimum transform [2](Chapter 3), [1](Chapter 12) for image coding. In practice it is rarely used because it is computational expensive, but it is useful for deriving an upper bound to the transform coding gain [15].

The basis vectors of the KLT are given by the eigenvectors, ψ_i, of the co-variance matrix, $\mathbf{R}_{xx}$, *i.e.*

$$\mathbf{R}_{xx}\psi_i - \lambda_i\psi_i = 0 \tag{B.1}$$

where λ_i are the eigenvalues associated with the eigenvectors.

The effect of the KLT is to make the transform coefficients statistically independent so that the co-variance matrix of the transform coefficients, $\mathbf{R}_{\theta\theta}$, is diagonal:

$$\mathbf{R}_{\theta\theta} = \begin{pmatrix} \lambda_{11} & 0 & 0 & \cdot\ \cdot & 0 \\ 0 & \lambda_{22} & 0 & \cdot\ \cdot & \cdot \\ 0 & 0 & \lambda_{33} & & \\ \cdot & & & & \\ \cdot & & & & \\ 0 & \cdot & \cdot & \cdot\ \cdot & \lambda_{N^2N^2} \end{pmatrix} \tag{B.2}$$

The diagonal elements of the matrix are the variances of the transform coefficients which are the eigenvalues of equation (B.1). If the transform coefficients are assumed to be stationary Gaussian variables then a rate-distortion function can be derived.

The parametric rate-distortion function of a stationary Gaussian source [15] is given by,

$$D(\phi) = \min\{\phi, \sigma^2\} \tag{B.3}$$

$$R(\phi) = \max\left\{0, \frac{1}{2}\log_2 \frac{\sigma^2}{\phi}\right\} \tag{B.4}$$

where σ^2 is the variance of the Gaussian source. For N^2 independent Gaussian variables (KLT transform coefficients) with variances, σ_{kl}^2, the rate-distortion function is given by the average of N^2 Gaussian rate-distortion [15] functions,

$$D_N(\phi) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} \min\{\phi, \sigma_{kl}^2\} \tag{B.5}$$

$$R_N(\phi) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} \max\left\{0, \log_2 \frac{\sigma_{kl}^2}{\phi}\right\} \tag{B.6}$$

For small distortions, $\phi < \sigma_k^2$, then the rate-distortion function is given by,

$$R_N(D) = \frac{1}{2}\log_2 \frac{\left(\prod_{k=0}^{N-1}\prod_{l=0}^{N-1}\sigma_{kl}^2\right)^{1/N^2}}{D} \tag{B.7}$$

Recall that the variances of the transform coefficients are the eigenvectors of the auto-correlation matrix. Hence, multiplying the eigenvectors is equivalent to calculating the determinant of the auto-correlation matrix $\mathbf{R}_{xx}$ and hence equation (B.7) becomes,

$$R_N(D) = \frac{1}{2}\log_2 \frac{|\mathbf{R}_{xx}|^{1/N^2}}{D} \tag{B.8}$$

The transform coding gain is defined to be the signal-to-noise gain over PCM encoding of the image for the same average rate. From equations (B.8), (B.3) and

(B.4) the coding gain for an $N \times N$ block, ${}^{N\times N}G$, is

$$
\begin{aligned}
{}^{N\times N}G &= \frac{|\mathbf{R}_{xx}|^{1/N^2}}{\sigma_x^2} && \text{(B.9)} \\
&= \frac{(\prod_{i=1}^{N^2} \lambda_{ii})^{1/N^2}}{\frac{1}{N^2}\sum_{i=1}^{N^2} \lambda_{ii}} && \text{(B.10)}
\end{aligned}
$$

The minimum rate at which the image can be encoded for a given average distortion is given as $N \to \infty$. The asymtotic versions of equations (B.5) and (B.6) are given by,

$$
\begin{aligned}
D(\phi) &= \frac{1}{(2\pi^2)} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \min\{\phi, S_{xx}(\omega_x, \omega_y)\}\, d\omega_x\, d\omega_y && \text{(B.11)} \\
R(\phi) &= \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \min\left\{0, \frac{1}{2}\log_2 \frac{S_{xx}(\omega_x, \omega_y)}{\phi}\right\} d\omega_x\, d\omega_y && \text{(B.12)}
\end{aligned}
$$

where $S_{xx}(\omega_x, \omega_y)$ is the power spectrum of the image.

For small distortions, $\phi \le S_{xx}(\omega_x, \omega_y)$, the rate-distortion function is given by,

$$R(D) = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \frac{1}{2}\log_2 \frac{S_{xx}(\omega_x, \omega_y)}{D}\, d\omega_x\, d\omega_y \qquad \text{(B.13)}$$

The spectral flatness measure of signal with a power spectrum, $S_{xx}(\omega_x, \omega_y)$, is defined to be:

$$\gamma_x^2 = \frac{\exp\left[\frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} \log(S_{xx}(\omega_x, \omega_y))\, d\omega_x\, d\omega_y\right]}{\frac{1}{(2\pi)^2}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} S_{xx}(\omega_x, \omega_y)\, d\omega_x\, d\omega_y} \qquad \text{(B.14)}$$

For zero mean signals the variance of the signal, σ_x^2, is given by,

$$\sigma_x^2 = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} S_{xx}(\omega_x, \omega_y)\, d\omega_x\, d\omega_y \qquad \text{(B.15)}$$

Substituting equations (B.14) and (B.15) into equation (B.13) gives,

$$R(D) = \frac{1}{2}\log_2 \frac{\gamma_x^2 \sigma_x^2}{D} \qquad \text{(B.16)}$$

The asymptotic coding gain [1, Chapter 12] is given by,

$$G = \lim_{N \to \infty} {}^{N \times N}G = (\gamma_x^2)^{-1} \tag{B.17}$$

Hence for low distortions the asymtotic coding gain for a Gaussian source is given by the inverse of spectral flatness measure of the original image. Physically this result makes sense since an image with a white spectrum has no correlation between elements and hence transform coding can be expected to give no gain.

Appendix C

Contents of disk

The disk attached to the back of this thesis contains the software necessary to generate the variable block size conditional replenishment encoder. The software is written in ANSI 'C' for UNIX based machines.

The software can be unpacked with the following command,

```
uncompress mc.tarZ; tar xf mc.tar
```

The software can be compiled using the *Makefile* provided with the software. The encoder is called *mc* and can be invoked with the following command line arguments,

-f <filename>: Filename or name of basename of file sequence.

-g <threshold>: Fail threshold, default = 0.

-p: Use stationary background prediction.

-s: Show statistics for individual frames. The default mode shows average statistics.

-t <threshold>: Motion estimation threshold, default = 0.001.

-v: Run in verbose mode.

-z: Use BTC encoder.