The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Garnham, Nigel William (1995) Motion compensated video coding. PhD thesis, University of Nottingham.

Access from the University of Nottingham repository:
http://eprints.nottingham.ac.uk/13447/1/thesis.pdf

Copyright and reuse:

The Nottingham ePrints service makes this work by researchers of the University of Nottingham available open access under the following conditions.

This article is made available under the University of Nottingham End User licence and may be reused according to the conditions of the licence. For more details see: http://eprints.nottingham.ac.uk/end_user_agreement.pdf

A note on versions:

The version presented here may differ from the published version or from the version of record. If you wish to cite this item you are advised to consult the publisher's version. Please see the repository url above for details on accessing the published version and note that access may require a subscription.

For more information, please contact eprints@nottingham.ac.uk

MOTION COMPENSATED VIDEO CODING

Thesis submitted for the degree of

Doctor of Philosophy

by

Nigel William Garnham BEng CEng MIEE

University of Nottingham

Department of Electrical and Electronic Engineering

October 1995

To my parents

One picture is worth ten thousand words

Attributed to Frederick Barnard

Printers' Ink

March 1927

Abstract

The result of many years of international co-operation in video coding has been the development of algorithms that remove interframe redundancy, such that only changes in the image that occur over a given time are encoded for transmission to the recipient. The primary process used here is the derivation of pixel differences, encoded in a method referred to as Differential Pulse-Coded Modulation (DPCM) and this has provided the basis of contemporary research into low-bit rate hybrid codec schemes. There are, however, instances when the DPCM technique cannot successfully code a segment of the image sequence because motion is a major cause of interframe differences. Motion Compensation (MC) can be used to improve the efficiency of the predictive coding algorithm.

This thesis examines current thinking in the area of motion-compensated video compression and contrasts the application of differing algorithms to the general requirements of interframe coding. A novel technique is proposed, where the constituent features in an image are segmented, classified and their motion tracked by a local search algorithm. Although originally intended to complement the DPCM method in a predictive hybrid codec, it will be demonstrated that the evaluation of feature displacement can, in its own right, form the basis of a low bitrate video codec of low complexity.

After an extensive discussion of the issues involved, a description of laboratory simulations shows how the postulated technique is applied to standard test sequences. Measurements of image quality and the efficiency of compression are

made and compared with a contemporary standard method of low bitrate video coding.

Acknowledgements

It is a pleasure to thank all members of staff in the Department of Electrical and Electronic Engineering at the University of Nottingham for their assistance with this research project.

In particular, I am most grateful to my supervisor, Dr M K Ibrahim, for his guidance and valuable suggestions, which stimulated my thoughts and provided impetus to my practical efforts.

Thanks are extended to Professor Greg Donahue at the University of New Mexico, for introducing me to the nuances of the Khoros image processing system and to Chris Greenhalgh and Dave Snowdon in the Department of Computer Science, for helping in the production of conference videotapes. The help given by Karl Lillevold at Telenor Research in supplying H.263 simulation binaries is also greatly appreciated.

Finally, I am indebted to my family and friends for the steadfast support and encouragement they have given me over the years, without which this endeavour would not have been possible.

Statement of originality

The work contained in this thesis is that of the author, unless otherwise stated. This work has not been submitted for any other publication, examination or qualification.

Nigel William Garnham

Publications

ADAPTIVE SUB-REGION VARIABLE SHAPE MOTION COMPENSATED PREDICTION

N W Garnham and M K Ibrahim

Conference on Applications of Digital Image Processing XVII, San Diego, USA.

Proceedings of the International Society for Optical Engineering (SPIE),

volume 2298, chapter 78, pages 35-44, July 1994.

CLASSIFIED SUBREGION MOTION ESTIMATED INTERFRAME CODING

N W Garnham and M K Ibrahim

Proceedings of the IEE Colloquium on Low Bit Rate Image Coding

Digest number 1995/154, pages 3/1-3/6, June 1995.

ADAPTIVE FEATURE-BASED MOTION ANALYSIS AND ITS APPLICATION TO INTERFRAME CODING

N W Garnham and M K Ibrahim

Proceedings of the Fifth International Conference on Image Processing and its Applications, Heriot-Watt University, Edinburgh, UK.

IEE Conference Publication 410, pages 31-35, July 1995.

Contents

Image reproduction - note

This thesis was printed using Apple and Hewlett-Packard laser printers, with a halftone resolution of 600 dpi. It should be noted that whilst the quality of most images is quite clear, the effect of image reproduction can serve to further degrade image resolution.

Glossary
Acronyms and abbreviations

ATD	Absolute Temporal Difference
BBMC	Block-Based Motion Compensation
BMMC	Block-Matching Motion Compensation
BT	British Telecommunications
CCIR	International Radio Consultative Committee
CCITT	International Telegraph and Telecommunications Consultative Committee (see ITU)
CD-ROM	Compact Disc Read-Only Memory
CIF	Common Intermediate Format
Codec	Coder-decoder
CRT	Cathode Ray Tube
DAT	Digital Audio Tape
DBMC	De-composed Block Motion Compensation
DC	Direct Current
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
EOB	End of Block
FIFO	First-in first-out
HDTV	High Definition Television
IS	International Standard
ISDN	Integrated Systems Digital Network
ISO	International Standardisation Organisation

ITU	International Telecommunication Union
ITU-T	International Telecommunication Union Telecommunication Standardisation Sector
KLT	Karhunen Loève Transform
MAP	Maximum *a posteriori* Probability
MC	Motion Compensation
MCP	Motion Compensated Prediction
ME	Motion Estimation
Modem	Modulator-demodulator
MPEG	Moving Picture coding Experts Group
NICAM	Near Instantaneous Companded Audio Multiplex
NTSC	National Television System Committee
PAL	Phase Alternating Line
PSNR	Peak Signal-to Noise Ratio
PSTN	Public Switched Telephone Network
QCIF	Quarter Common Intermediate Format
SAC	Syntax-based Arithmetic Coding
Sub-QCIF	Sub-Quarter Common Intermediate Format
VLC	Variable Length Code or Variable Length Coding
VLSI	Very Large Scale Integration
VOD	Video on Demand

Symbols

$\vec{V}_n$	Motion vector
$\vec{V}_n$	Motion vector component
$\vec{V}_\varnothing$	Null displacement vector
η	Pixel luminance value
$\varnothing$	The null set
ΔQ	Quantisation step interval
d	Quantisation error
ds	Spatial pixel intervals
dt	Temporal interval
dx, dy	Spatial intervals
e	Interframe error (the result of DPCM)
$H(\mathbf{z})$	Picture entropy
I_n	Frame at instant n
I_n'	Previous frame with respect to I_n
L_n	Feature perimeter runlength label
L_p	Feature primitive label
N_B	Number of bits
$N_\mathrm{n}(\mathbf{p})$	n-neighbour pixel relationship
$\mathbf{p}_c$	Corrected pixel value
P_c	Pixel search parameter (current frame)
$\mathbf{p}_n$	Seed pixel
P_p	Pixel search parameter (previous frame)
qz	Quantisation indicator flag

R	Image area subregion set
R_n	Component subregion
R_p	Component subregion (previous frame)
s	Scale factor
u, v	Frequency co-ordinates in the transform domain
V	A value set
v	Motion vector bitstream
w_n	Input level (probability)
x, y	Spatial co-ordinates in the pixel domain

Chapter 1

Introduction

Motion compensated video coding

1.1 Overview

Of all the technological achievements in the 20th century, television has perhaps had the greatest effect on our everyday lives. For many people, a television set is an obscure box in the corner of their living room - providing news, education and entertainment. Children are now said to be addicted to it and there is no doubt that the nature of leisure time activities has radically changed over the past thirty years to accommodate television. Telecommunications systems have also invaded the home and people can now hold a telephone conversation as comfortably as they would face to face.

But the evolution in television and telecommunications systems have followed different paths. Since the introduction of colour television in the 1950's, there have been no significant changes to the mechanism of picture transmission and display. In this country, the 625 line format has been with us for a long time and for many people, their perception of improvements in the "quality" of television has been assisted by advances in associated *audio* reproduction, particularly since the advent of NICAM digital stereo. The telecommunications system, on the other hand, has been able to take advantage of new technology to provide a modern, digital network, available to everyone. Recent advances in mobile communications offer the potential for telephones to be associated with individuals, rather than their homes and offices.

Taking account of this background, it is perhaps surprising that the concept of combining pictures and sound into a single PSTN channel for video conferencing has taken so long to evolve. The essential difficulty is that bandwidth is limited in the services provided by telephone companies on the basis that to transmit speech, only 14 kHz is required for acceptable quality. Broadcast quality digital television, on the other hand, requires over 100 Mbits/s to supply pictures. Even existing terrestrial channels allocated for television cannot accommodate this amount of data. Consequently, video compression and coding appear to be the best approach to the problem, until someone provides a mass communications system in which bandwidth is not a limitation.

International co-operation has proved important in the development of video codec algorithms. Under the auspices of the CCITT, now known as the International Telecommunication Union (ITU), a recommendation was published in 1990, describing the framework of a video codec intended for use on the ISDN system on channels of 64 kbits/s. Its primary concern is the removal of redundancy, which occurs within and between picture frames. Intraframe coding can be used to compress a single frame and redundancy is said to be present where the picture comprises groups of adjacent equal value picture elements, or *pixels*. Similarly, where pixel values have not changed over time, interframe coding can remove temporal redundancy. Only changes in picture content need to be supplied to the decoder and, as a result, an efficient mechanism of picture coding is developed.

In most cases, however, video codecs are said to be *lossy*, since additional processing tends to lower the resolution and introduce errors. This said, provided certain requirements of quality are kept, most users are unable to detect coding errors and those who do will probably be able to tolerate them.

The implementation of video codecs has also been limited by the technology available. Where real-time processing is required, compression and coding must be performed at high speed - a requirement that VLSI technology has only recently appeared able to satisfy. A new generation of software video codecs is being proposed in current ITU recommendations, to work on the growing number of personal computers connected to the PSTN by a modem. As the processes are refined and the technology is improved, video conferencing codecs will become less expensive and more widely available. Whether they become more popular is, however, a different matter. It took many years for televisions and telephones to get into most homes and wariness about seeing the person the user is talking to may, for some while, make the videophone something the public feels it can do without.

1.2 The scope of this thesis

This thesis examines the current state of video technology and assesses different aspects of video compression and coding. A comprehensive introduction to the outline ITU-T H.261 recommendation is given and comparisons made with its sister MPEG algorithm, more commonly associated with picture storage and HDTV. The concepts of redundancy removal, using DPCM/DCT coding will be covered and it will be shown that, whilst useful, it is not exhaustive and other methods are needed to

work in conjunction, providing a more accurate coding of picture content over changes in time.

One such method is referred to as *motion compensation*. If a video sequence is divided into individual frames, it is possible to see if particular spatial components have been displaced as a result of overall motion. Contemporary approaches to motion compensation have been block-based and it will be shown that there are so many combinations of blocks in an image that, for real time applications, an exhaustive search is not cost-effective. Furthermore, block-based searching is rather unreliable, since the segments considered represent arbitrary areas of the picture rather than easily identifiable features.

A more advanced technique of video coding uses models of known picture features and then re-maps them against vectors supplied by a motion detection algorithm. This is perhaps more useful, although defining the original set of features has proved difficult and it has been found that images do not necessarily move consistently, making an accurate assessment of motion difficult.

The fundamental basis of this thesis is a proposal for a novel algorithm, using the best characteristics of block and model-based coding. Using the simple parameters of constituent pixel values, outline and location, a new feature classification technique compiles, for each frame, a sourcebook of subregion data. Comparison of such sourcebooks shows where any displacements have occurred and motion compensation allows a decoder to reconstruct features with the supply of motion vectors.

Effective feature classification has been made possible by the introduction of spatial processing. It is shown that the use of pixel subsampling reduces the resolution of an image and subsequent quantisation limits the range of values which can occur. Spatial processing therefore serves to deliberately introduce errors. However, it will be demonstrated that such errors are no worse than those seen in contemporary coding algorithms, by means of both visual inspection and calculating measures of spatial quality. The process of reconstruction has been found to introduce displacement errors, where successful feature replacement has not occurred. To counter this, a simple method of error detection and correction is proposed, removing many interframe errors and restoring quality and data content to acceptable levels.

All work in the development of video codec systems is undertaken with compromise in mind. There is an inherent trade-off between image quality, process efficiency and compression. An increase in compression usually leads to a reduction in quality and greater processing needs. Results from the feature classification based coding algorithm will be considered against the background of contemporary methods and comparisons made.

1.3 Outline

Chapter 2 provides the reader with an insight into contemporary techniques of video compression and coding. Using the framework of the H.261 algorithm, concepts of DPCM/DCT coding will be described, along with associated processes for efficient codeword generation and constant bitrate supply. This chapter also considers the ISO/MPEG standard and comes up to date with the latest H.263 recommendation for very low bitrate video codecs.

Although motion compensation is a very wide-ranging topic, chapter 3 concentrates on the principles of block and model-based motion compensation. It will be shown that, whilst conventional block-based motion compensation is a simple process, it is, computationally highly intensive and often impractical to implement in a real-time codec. Model-based coding, on the other hand, is more complex as it is difficult to set the parameters of model descriptions in a finite set.

Chapter 4 considers the application of spatial processing as an initial stage in a novel coding algorithm. It is shown that an image can be reduced in resolution and still maintain an acceptable level of quality. Furthermore, pixel-value quantisation can be introduced to limit the range of values associated with a given image. Frames from standard test sequences are processed and a measure of relative quality produced by both illustrations and calculations of signal-to-noise ratio.

A novel method of feature classification is described in chapter 5. Taking a low-resolution image, it is possible to identify features as subregions of equal value pixels. A comprehensive description of the classification process is made, identifying

consistent seed pixels and then clustering against predicates of pixel connectivity. With segmentation complete, a method of boundary coding is described, using runlength and primitive shape labels. A description of the implementation shows these processes at work and defines the nature of sourcebook data structures, subsequently used for the detection of motion.

Chapter 6 describes the process of motion vector generation and shows how sourcebooks are assessed in order to deduce interframe motion. Different forms of motion vector are described, covering stationary features, linear motion and perspective changes. It is also suggested that, where motion detection is unsuccessful, features can be re-classified to reduce the likelihood of error propagation.

The process of motion compensation inevitably introduces errors and chapter 7 develops a simple method of error correction, restoring some spatial quality to the images in a sequence. This chapter also illustrates and quantifies image quality and contrasts the results from the feature classification based algorithm against a simulation of the H.263 algorithm. Information theory is also used to demonstrate how error correction restores the entropy of frame data, close to its input levels.

In the conclusions of chapter 8, it will be suggested that feature classification forms the basis of a novel video codec of low complexity. Where contemporary codecs use motion compensation in conjunction with other coding techniques, feature classification works alone in determining the nature of interframe differences.

Suggestions for further work are made, particularly in the area of compression efficiency, expressing the trade-off between quality and data content.

Chapter 2

Video Compression and Coding

A review of contemporary techniques

2.1 Introduction

The principle of combining images and sound into a single communications systems is, by no means, unfamiliar. Over fifty years have passed since the introduction of broadcast television in the United Kingdom. However, it is only recently that the concept of using moving pictures for interactive video and multimedia has received interest, as the costs of transmitting a television signal over anything other than short distances has proved prohibitive. We have been limited to sending mainly still images over the public telephone network, using facsimile and low-scan techniques, made slow by the restriction in bandwidth available to most users.

It seems paradoxical that whilst the technology of digital television has advanced in remarkable leaps in recent years, we still have no efficient, widespread means of sending high quality video over the telephone network for the purposes of videotelephones. One of the fundamental costs of colour television is the bandwidth required to transmit a channel of sound and pictures. The four terrestrial channels allocated in the United Kingdom have equivalent digital bandwidths of 12-24 Mbits/s, which would be insufficient to carry sound, chrominance (colour) and luminance (brightness) signals. It is the scarcity of space in the radio-frequency spectrum that has limited the extent of broadcast television.

In its uncompressed state, conventional broadcast-quality digital television requires bit rates of typically 166 megabits per second - well over that available for even the newest Integrated Services Digital Network (ISDN) [i] channel. Given this primary constraint, contemporary research has focused on the compression of video images, allowing transmission of low resolution images over the multimedia digital network. In most cases, compression is easy to achieve, removing spatial and temporal redundancies naturally occurring in sequences of images.

Figure 2.1.1: A frame from the sequence Claire, demonstrating picture redundancies.

Consider the image of figure 2.1.1. This could be regarded as typical of a videoconferencing scene, where during the conversation, most of the picture will not change other than, say, the lips, eyes and occasional head movements. This feature can be used to good effect, such that only information about differences that have occurred will need to be sent to the recipient. This process is called *interframe*

coding and is ideal for the low level of temporal changes, associated with videoconferencing.

It is also possible to extract information about differences between spatially adjacent pixels at a given instant in time. This process of *intraframe* coding makes large areas of consistent colour and shade (the plain background in this example) easily represented. Boundaries are easy to detect, where significant changes in luminance and chrominance occur. Interframe and intraframe coding are two methods of redundancy compression that have been used to good effect in the development of videoconferencing hardware for transmission over telecommunications channels of up to 64 kbits/s.

The intrinsic effect of redundancy coding is not to reduce picture quality, indeed the efficiency of a differencing algorithm should not serve to affect spatial resolution. However, subsequent processing of the difference information can take place, where "useful" information can be described as those aspects of the image that convey meaning to the human viewer, even if that is only a small proportion of the image content. The contrast sensitivity function [ii] allows understanding of the human ability to detect spatial and temporal detail and this will be described in chapter 4. Assuming the human eye can resolve down to two minutes of an arc, it can take in the equivalent of a million pixels of information without moving. By moving the eye, but not the head, the field of view is at least an order of magnitude greater. We know the head is likely to remain stationary whilst a person is doing something specific, but the eyes are moving continuously. If we assume that to represent the colour and

luminance of a pixel, 12 bits are required, over 100 million bits of information are needed to represent the user's static scene.

Consideration of these factors gives an understanding of the essential nature of video compression algorithms. It is necessary to take a picture, which under normal circumstances would require extensive data representation and code it to the constraints of digital telecommunications network, whilst maintaining an image satisfactory to the human perception.

At an early stage, the international telecommunications community identified the need for close collaboration to ensure the adoption of a system which could be applied in all countries and make videotelephony available to a world market. Even though a European standard specification did emerge in the 1980's [iii], for a 2Mbits/s, 625 line, 25 frames per second PAL system, demand in North America required plans using the 525 line, 30 frames per second NTSC system. Subsequently, the conversion between these standards was regarded as the focal point of international co-operation and under the auspices of the organisation now known as the International Telecommunication Union[1] (the *ITU*), a videophone algorithm was recommended, meeting the needs of the new ISDN systems and working for all bit rates between 64 kbits/s and 2Mbits/s.

The resulting ITU-T Recommendation, H.261 [iv][v], forms the basis of the international development of videoconferencing systems using the new ISDN

[1] The International Telecommunication Union was formed from an amalgamation of the CCITT and the CCIR.

networks being installed throughout the world. However, many concepts used are equally applicable to other areas of video codec design, such as high-definition digital television, where an increased amount of picture data is to be transmitted within the constraints of existing terrestrial bandwidths.

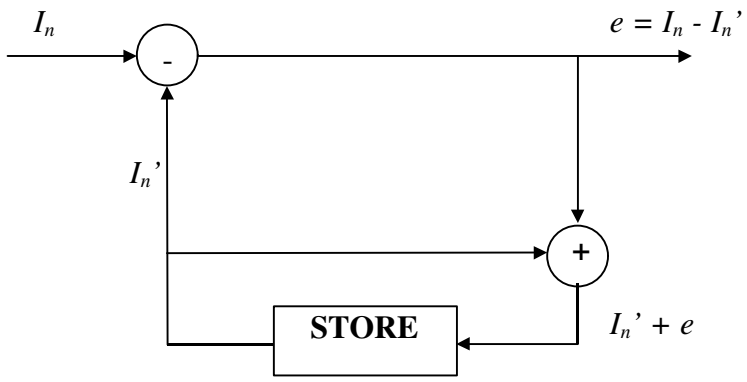
2.2 Differential Pulse Code Modulation (DPCM)

At the heart of the videoconferencing system proposed in the ITU-T H.261 document, is the use of Differential Pulse Code Modulation, or DPCM [vi]. Whereas conventional Pulse Code Modulation (PCM) uses a fixed-length set of binary codes to represent temporally-current data, DPCM extracts difference data between temporally adjacent data sets. When applied to adjacent images which constitute a video sequence, the DPCM value is that which results from differencing the pixel values, at a given spatial co-ordinate, in successive frames.

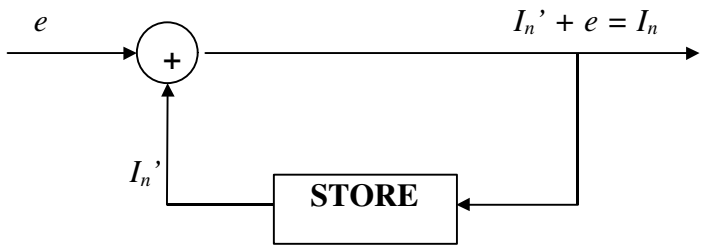
The information which results from this process can be transmitted over the communications network using variable-length codesets, where codes described by the least number of bits are used to represent the most frequently-occurring values. Practically, if little or no change occurs in the sequence, the amount of data to be coded and sent to the recipient is minimal.

In addition to considering temporally adjacent picture frames and extracting difference data (interframe coding), groups of similar pixels which are spatially adjacent in a given frame can be DPCM encoded. The use of intraframe coding is particularly helpful in the first few frames of a sequence, where all known values need to be sent to the receiver.

To explain the principle behind the DPCM process, consider the simple loop in figure 2.2.1:



(i)



(ii)

Figure 2.2.1: A simple DPCM loop (i), and receiver (ii)

Current frame data, held in the store, is updated with the generation of the difference value e which is to be sent to the decoder. Initially, the value of a given pixel magnitude held in store is subtracted from its equivalent next value and hence the current frame values are subtracted from the next-frame values. The difference signal is received at the decoder, where another store holds the current values, updated by the addition of e to produce a representation of the next frame.

Algebraically, if I_n is considered the next frame value, I_n' the current frame value and e the difference between them, then:

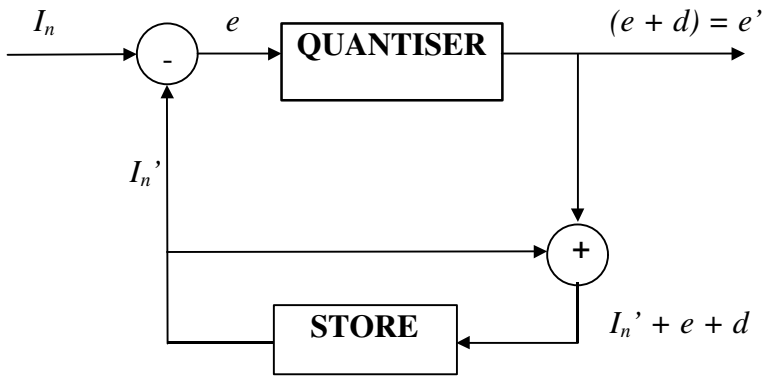
$$e = I_n - I_n' \qquad \text{[Equation 2.1]}$$

and $$I_n = I_n' + e \qquad \text{[Equation 2.2]}$$

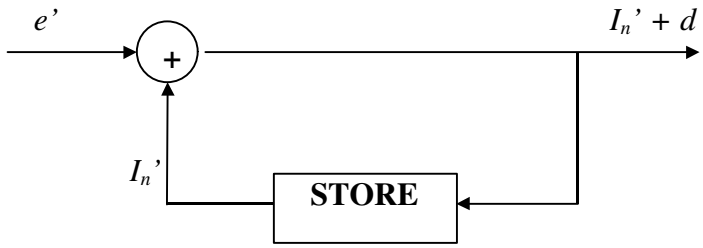
which allows the next frame to be reproduced at the decoder. An example of the effect of taking a DPCM difference image is shown in figure 2.2.3.

2.2.1 Quantisation

A key feature of the H.261 recommendation is the introduction of quantisation to reduce the large number of values that are possible for chrominance and luminance. This results in less bits being required for transmission and storage, although there is a natural addition of errors and whilst not noticeable under quiescent picture conditions, coarse quantisation can produce a poor reproduction of the original image.



(i)



(ii)

Figure 2.2.2: A simple DPCM quantised loop showing (i) encoder and (ii) decoder

The quantisation error is carried through to the output of the decoder and can be described:

$$e' = I_n - I_n' + d = e + d \qquad \text{[Equation 2.3]}$$

thus $$I_n + d = I_n' + e' \qquad \text{[Equation 2.4]}$$

where d is the quantisation error

and e' is the resulting difference signal

(i)

(ii)

(iii)

Figure 2.2.3: Two temporally-adjacent frames (i), (ii) and their DPCM difference image (iii) {reverse video is used for clarity}.

One method of controlling the extent of quantisation errors is the use of buffer feedback that allows the quantisation step interval to be related to the amount of data being produced by changes in the picture, principally due to motion..

2.3 The ITU-T H.261 Algorithm

At an early stage, the complexity of various proposals for video codec design made it clear that initially building real-time hardware to prove algorithms would be difficult. Under the supervision of the ITU, international collaboration brought together novel algorithms and flexible software-based simulation systems. Once the material had been downloaded into bit form, it could easily be processed by a mainframe computer. A video sequence of about thirty seconds real-time data can practically be stored and the results time-scaled, reflecting the increase in performance that dedicated hardware would produce.

The first stage in the process is to digitise a sequence for each pixel to assign values for both chrominance and luminance, storing the results in a random access memory. The video luminance signal is sampled at 6.75MHz and digitised to a resolution of eight bits per pixel (the equivalent of 256 grey levels). A similar process applies for chrominance, but at 3.375 MHz. The stored data for each successive frame can then be manipulated and the DPCM values extracted and coded. A block diagram of the outline H.261 hybrid codec is shown in figure 2.3.1.

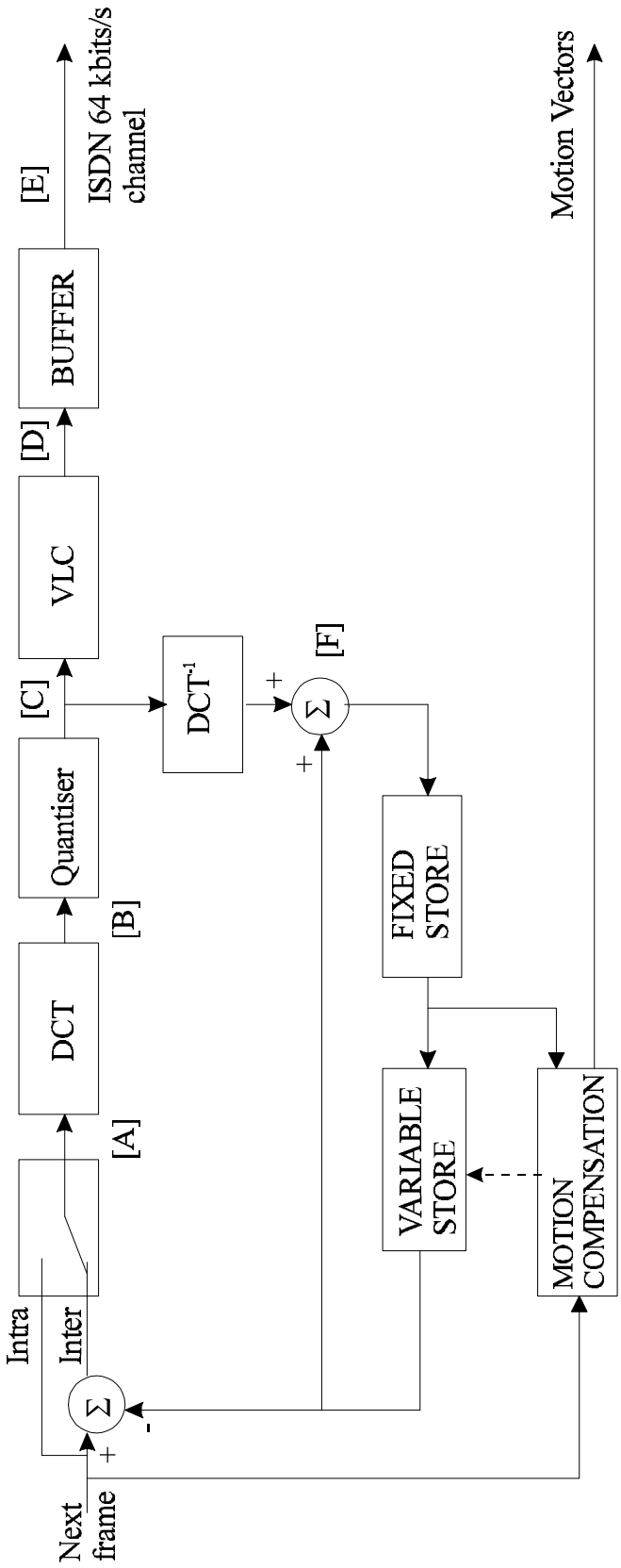


Figure 2.3.1: An outline block diagram of the H.261 hybrid DPCM/DCT encoder

2.3.1 Spatial and temporal encoding

Whilst the primary aim of a hybrid video codec is to remove interframe redundancies that occur through time, it has already been mentioned that a lot of intraframe redundancy is to be found in many image sequences. The H.261 algorithm uses a threshold mechanism, such that it normally works in temporal interframe coding mode. However, there will be occasions when a scene changes very significantly - either due to sudden movement of a large object across the scene, panning, or a wipe to a new scene. In these cases, intraframe coding is used and the codec will operate to encode difference information between spatially-adjacent pixels on the new frame. The switch to intraframe coding inevitably results in a surge of new data to produce a new frame, providing the values to which subsequent interframe DPCM coefficients may be added.

2.3.2 Transform Coding

Having obtained the DPCM values (point [A], figure 2.3.1), data compression can be applied. The picture is divided into 8×8 pixel blocks and a discrete cosine transform (DCT)[vii] applied to each block.

Transform coding does not directly cause data compression - there is still an 8×8 array of coefficients in the transform domain. Compression is achieved by the subsequent extraction of the transform coefficients representing basic spatial relationships and the choice of transform method is an important factor. Following transformation by the DCT, the transform coefficients in each 8×8 block can be

considered as the magnitudes of the various spatial frequencies present in the spatial pixel domain.

Consider a block in which all the pixel values are identical, as is the case in the plain areas of a picture. After transformation, the only non-zero transform domain value will be the D.C. coefficient, which occupies the top left-hand corner of the block. Hence the block can be said to have been compressed by a ratio of 64:1, provided the inverse transform algorithms knows that all other coefficients are zero. Unless the frame is completely plain, there will be many blocks having more non-zero transform coefficients. In almost all frames there will be a large number of transform coefficients sufficiently close to zero to be either not transmitted, or represented by data of less than eight bits.

The nature of the DCT transform and its resulting coefficients is shown in figure 2.3.2.

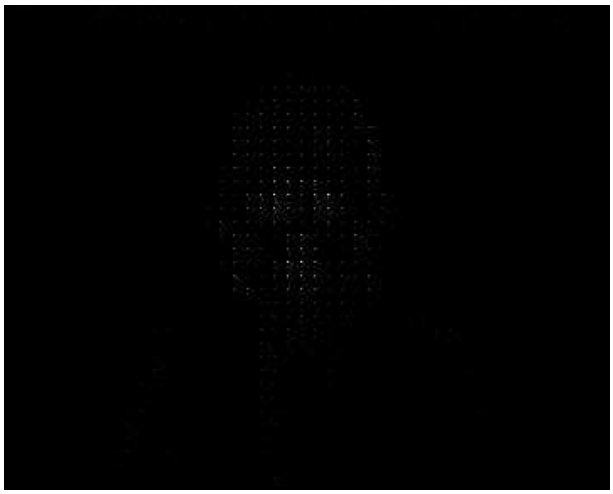


Figure 2.3.2: The application of the discrete cosine transform to a DPCM image of 8 × 8 pixel blocks (Claire). The higher intensity dots represent the location of each DC coefficient.

The transfer function of the DCT is given by:

$$F(u,v) = \frac{1}{4}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos[\pi(2x+1)\frac{u}{16}]\cos[\pi(2y+1)\frac{v}{16}]$$

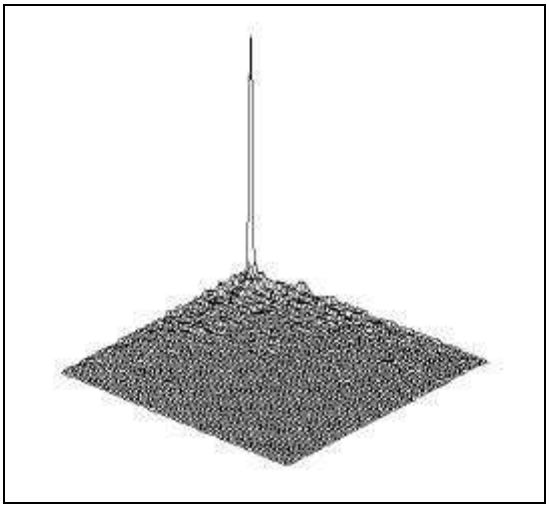
with $u,v,x,y = 0,1,2,...,7$ [Equation 2.5]

where x,y are spatial co-ordinates in the pixel domain

and u,v are co-ordinates in the transform domain.

Whilst the DCT is often considered to be the most suitable transform method available, less sophisticated methods such as the Walsh-Hadamard transform [viii] can be used with limited success in more basic applications. The Karhunen Loève transform (KLT)[ix] has been demonstrated as the most efficient technique, but has proved more complex to implement. Normally, the magnitude of the coefficients in the transform domain can be related to the frequency of the pixel values in the spatial 8×8 block.

Since it is normally assumed that humans cannot readily detect sharp, high-frequency spatial transitions [46], the lower transform coefficient values (which represent the high-frequency elements in the 8×8 block) can be discarded to allow subsequent data compression. The effect of the high-coefficient grouping is demonstrated to good effect in figure 2.3.3, which shows the effect of applying the DCT to a much larger square area of an image.



(i)

(ii)

Figure 2.3.3: Distribution of transform coefficients (i) for a selected 64×64 pixels area of interest (ii).

2.3.3 Quantisation and transform data compression

With data in each 8×8 block now reduced to a set of coefficients in the DCT domain (point [B], figure 2.3.1), the compression which follows makes use of the transform domain block characteristics already known. Each spatial 8×8 block can reliably be represented by a few transform coefficients grouped into the top left-hand corner of

the 8 × 8 coefficients in the transform domain, representing the D.C. and a few low-frequency coefficients. Quantisation is then performed on the transform coefficients, having the effect of setting all small values to zero and quantising larger values to a set of preferred magnitudes ready for coding and transmission. The quantised transform coefficients (point [C], figure 2.3.1) are then scanned in a zigzag manner, shown in figure 2.3.4.

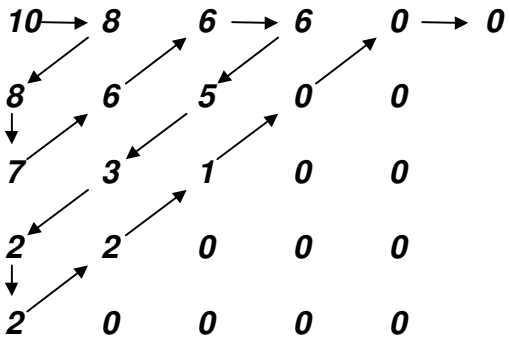


Figure 2.3.4: Codec scanning of transform domain coefficients, starting at D.C. and ending after three consecutive zero values

The transform coefficients are scanned in an order which indicates descent from the highest transform coefficient (the D.C. value) to the lowest. However, the effect of quantisation produces a large quantity of zero coefficients. Their existence makes compression possible, since it is not necessary to provide the decoder with long streams of zero values. Having encountered a run of three zero coefficients, the assumption can be made that all remaining ones are also zero and a single End-Of-Block (EOB) symbol is produced, denoting the end of all non-zero coefficients for the current 8 × 8 transform block. In the example of figure 2.3.4, sixteen codes would be

needed (including the EOB character), representing a compression of about 25% from the original set of transform coefficients.

2.4 Methods of data representation for Low Bit-Rate Coding

The video codec, by extracting the DPCM difference data and then performing a DCT upon it, will have formed a data set considerably compressed from the original broadcast-quality image values. To efficiently transmit these over the communications network, a set of codes is developed which relate to the frequency of occurrence of each quantised transform coefficient. Practically, this is achieved by the use of variable-length coding (VLC), where codes of the least bit length are used to represent the most frequently occurring coefficients. The length of the codes then increases by inverse proportion to coefficient frequency.

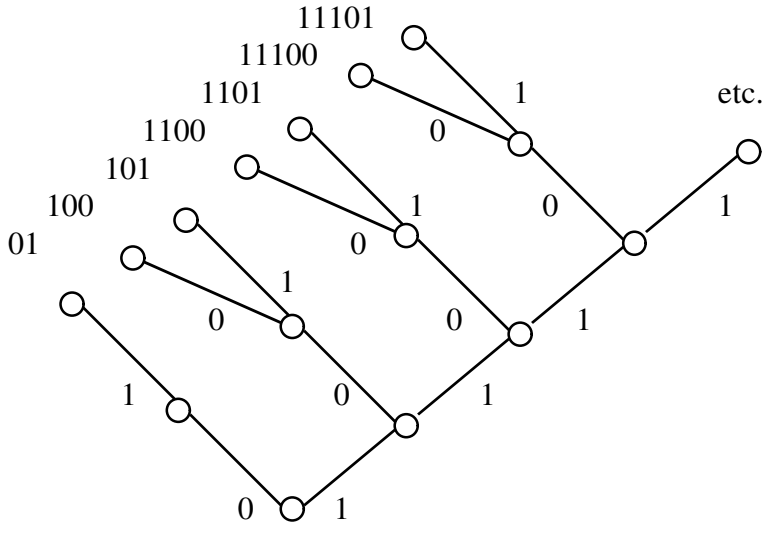


Figure 2.4.1: Binary tree representation of variable length code generation.

Use of variable length codes removes the redundancy that would occur if a fixed-length word were used. In the case of an image sequence where any one of 256 values could occur, fixed length coding would require that the minimum data word was eight bits long. Generation of VLCs is related to probability and a simple method of developing the required codeset is shown by the binary tree mechanism in figure 2.4.1). Starting at the root, a left-hand branch at each node adds a 0 to the code and a right-hand branch adds a 1. The assignment of these codes requires the probability of occurrence of each scanned quantised transform coefficient.

Consider the example shown in the table of figure 2.4.2, where the probability of occurrence of one hundred coefficients is known.

Coefficient value	Frequency	Code	Bits required
12	40	01	2
11	15	100	3
13	14	101	3
10	12	1100	4
9	8	1101	4
8	8	1111	4

Figure 2.4.2: Assignment of variable length Codewords.

By multiplication of the number of bits required to represent each value by the frequency, it can be seen that 291 bits would be required to represent this sequence of 100 coefficients. Had four bits been employed, given that all values are less than fifteen, 400 bits would have been needed and so the benefit of variable length coding is easily shown.

An algorithm for the adaptive assignment of Variable Length Codes was proposed by Huffman (1952) [x]. The compact code can be derived by first ordering the input probabilities with respect to their magnitudes, as illustrated in the table of figure 2.4.3.

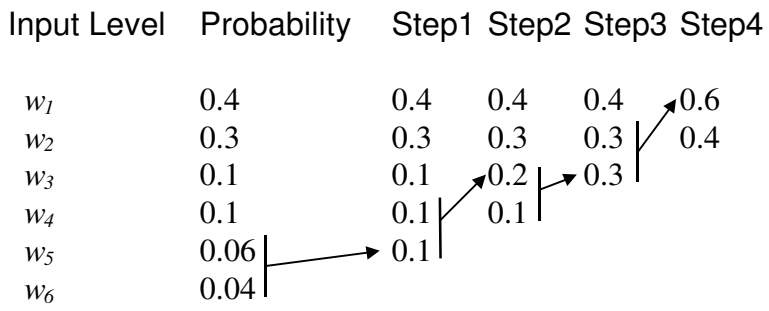


Figure 2.4.3: Construction of a Huffman Code

The two smallest probabilities are added together on a step-by-step basis to form a new set of probabilities until only two values are left. In the example, this is at the fourth step.

Codewords are generated by starting at the last step and working backwards, assigning a 0 and 1 at each step, decomposing probabilities. This is depicted in figure 2.4.4.

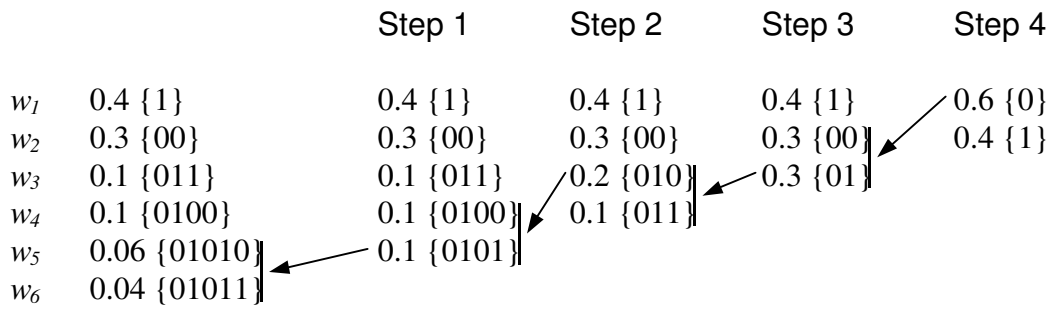


Figure 2.4.4: Generation of Huffman Codewords

Starting at (in this case) step 4 and working backwards, the 0 associated with the 0.6 remains the first bit value of its decomposed codewords and the 1 associated with 0.4 remains the first bit value in each step back. When the input probabilities are reached, a compact code has been generated to reflect the frequency of occurrence.

The codes produced are uniquely identifiable in a serial communications system. That is to say that, provided the start point is known, a bit stream can only be sub-divided into the correct codewords as smaller codes are not similar to the constituent components of larger ones (figure 2.4.5)

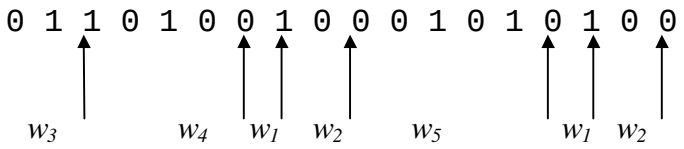


Figure 2.4.5: The resolving of unique Huffman Serial Codewords

2.5 Video Data Buffering

The final interface to the ISDN channel is a buffer (figure 2.3.1 point [E]), ensuring the constant flow of 64kbits/s data into one of the B channels [xi], even though the buffer input could be anything between 0 and 384kbits/s. It has already been shown that the resulting bit stream from the DPCM process is likely to be variable, depending on how much interframe activity has occurred. A high rate of unpredictable motion has to be transmitted at the same data rate as would apply for no interframe difference. The practical approach to this is the use of variable step size quantisation of the transform coefficients, where the magnitude of the sampling interval is determined by control parameters generated at the buffer.

Although a buffer does have the capacity to store data on a first-in first-out (FIFO) basis, the technology available for real-time data buffering still limits performance where increased capacity is normally accompanied by greater propagation delays. The use of quantisation feedback is shown in figure 2.5.1.

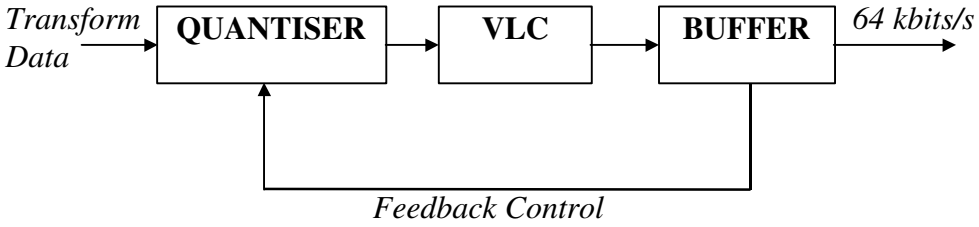


Figure 2.5.1: Quantisation step control feedback to produce constant output bitrate.

The use of this feedback control mechanism enables the coarseness of quantisation step values to be affected by buffer status. As the buffer is filled with data (particularly during significant interframe motion), the effect of the feedback is to coarsen the quantisation step interval employed and thus reduce the rate at which data is presented to the buffer. The primary drawback to this idea is that as the buffer reaches capacity and the quantisation step interval is at a maximum, picture quality can be very poor. Even though the loss of quality may only be temporary, with subsequent data correcting the quantisation distortion, it happens at a time when the viewer's temporal perception is most sensitive to changes in resolution. It would not be acceptable to use this approach for broadcast-quality television under the MPEG standard, where high transmission quality is required all the time.

In a videoconferencing situation where the video signal is frequently switching between the interactive parties, picture distortion is quite a problem because of the buffering factors mentioned. Whilst large buffers, where practical, will reduce the dependence on quantisation step control, only the availability of a broadband packet video system, such as B-ISDN, will provide a more feasible solution to this problem.

2.6 Codec Storage

In the simple DPCM loop of figure 2.2.1, the incorporation of picture feedback allows current picture data to be stored in the codec for subsequent interframe comparison. Having committed spatial video to the transform domain and applied quantisation, the coefficients must be transformed back to the pixel domain to facilitate like-for-like differencing in spatial terms.

Once again, the effect of quantisation will cause difficulties, as the distorted coefficient values will be differenced and, as a result, errors may propagate. An inverse DCT function is employed to re-form pixel difference data from the transform coefficients at point [F] on figure 2.3.1, where the differences are added to the current frame value in the fixed store to update the retained frame to that of the new frame undergoing coding.

2.7 ISO/MPEG Algorithms

Like the ITU-T Recommendation, the algorithms specified by the Moving Picture coding Experts Group (MPEG) [xii] employ a degree of both loss-less and lossy coding techniques. However, whilst the H.261 algorithm is specifically designated as the framework of video codecs working on ISDN channels of $p \times 64$kbits/s, the scope of MPEG is more wide-ranging. In the late 1980's an obvious relationship began to emerge between personal computers, digital storage on inexpensive media (such as CD-ROM) and the sale of video entertainment and educational software. The extent of MPEG has also been applied to the compression of video for the purposes of Video-on-Demand [xiii] and for HDTV.

The primary objective of MPEG was to produce a compression algorithm for storage media having a throughput of 1 - 1.5 Mbits/s, with other goals of up to 60 Mbits/s. Whilst the direct application of CD-ROM was an obvious one, the brief of MPEG was to produce a standard that would apply to other storage techniques and applications. Data-DAT, which originated in the digital audio tape standard has a data throughput rate of typically 1.5Mbits/s, has two hours per cassette and unlike CD-ROM has easy re-recording by the end user. The capacity of hard disks has also grown in recent years making them a practical option for video storage.

The MPEG-1 video coding algorithm [xiv] resulted from the requirements of CD-ROM and was greatly influenced by preliminary results in the formulation of the ITU-T H.261 algorithm. The development and evaluation of the algorithm was performed at bit rates in the region of 1 Mbits/s and video resolutions of 352 pixels $\times$ 288 lines, 25 frames per second, for PAL and 352 pixels $\times$ 240 lines, with an average of 29.97

frames per second for the NTSC system. However, whilst the H.261 algorithm effectively works on a predetermined spatial format (CIF), these rates are not fixed and can be varied according to the requirements of different applications.

The essential difference of MPEG-1, compared with H.261, is that, by the nature of the application to CD-ROM, *random access* is required. This allows the end user to arbitrarily choose any point in the video sequence from which to start viewing the moving images. To achieve this, MPEG-1 has a number of frames which are encoded on their own and without any reference to other frames in the sequence, which are referred to as *key frames* and occur typically once in every twelve. As a result, MPEG-1 deliberately forces intraframe coding on some frames, whilst the majority are formed as an interframe prediction with reference to temporally adjacent frames. The result of this is a maximum waiting time of twelve frame-delays between the user-defined start location and the commencement of playback - about 0.4 s for the PAL format.

The presence of regularly occurring intraframe coding is one of the reasons why MPEG-1 is unsuitable for real-time coding in audiovisual communications. The time taken to process and transmit an intraframe coded frame is considerably higher than for interframe difference data, causing considerable variations in the quantity of bits per frame. If the I-frames were to be taken as primary start frames for an interframe sequence, they would have to be encoded with minimal losses, rendering the availability of data for the subsequent interframe coding relatively low in a given time period. The arrangement of MPEG-1 frame data is shown in figure 2.7.1.

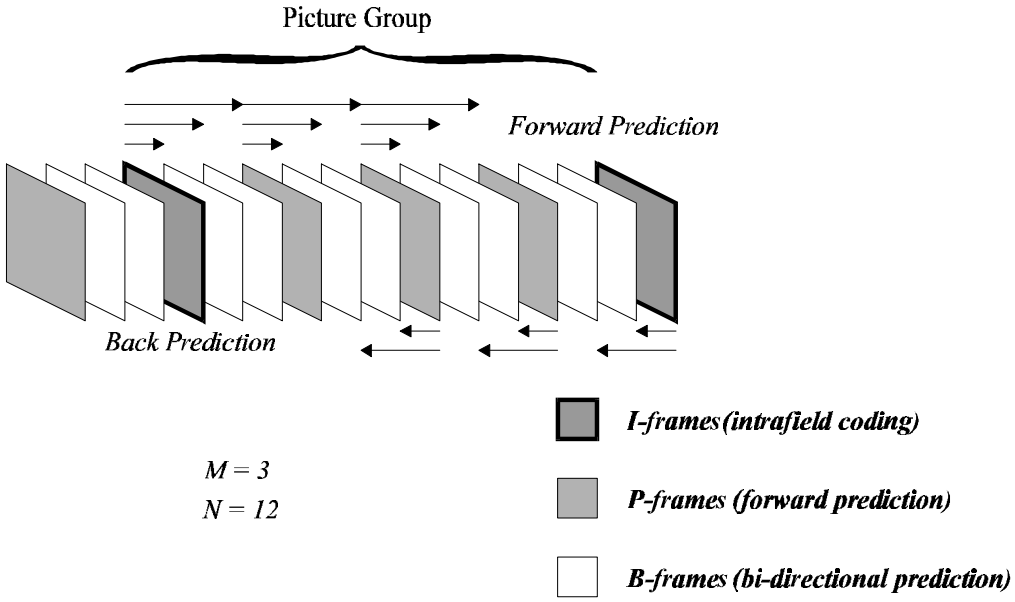


Figure 2.7.1: MPEG-1 Frame Origins

The predictive codec used in MPEG-1 solutions is once again a hybrid of the DPCM/DCT algorithm already detailed in this chapter. The DCT coefficients are scanned and quantised and assigned minimum redundancy codes. They are then passed through a buffer and out to the storage media. The buffer provides quantisation control to the forward coder, allowing smoothing of data rates and a constant bitrate to the storage device. MPEG also has the ability to produce motion vectors for the interpolative decoding of large interframe differences. Functions used in the decoding process are also employed at the encoder to ensure an equivalent prediction to the final decoded output. A block diagram of the MPEG-1 scheme is shown in figure 2.7.2.

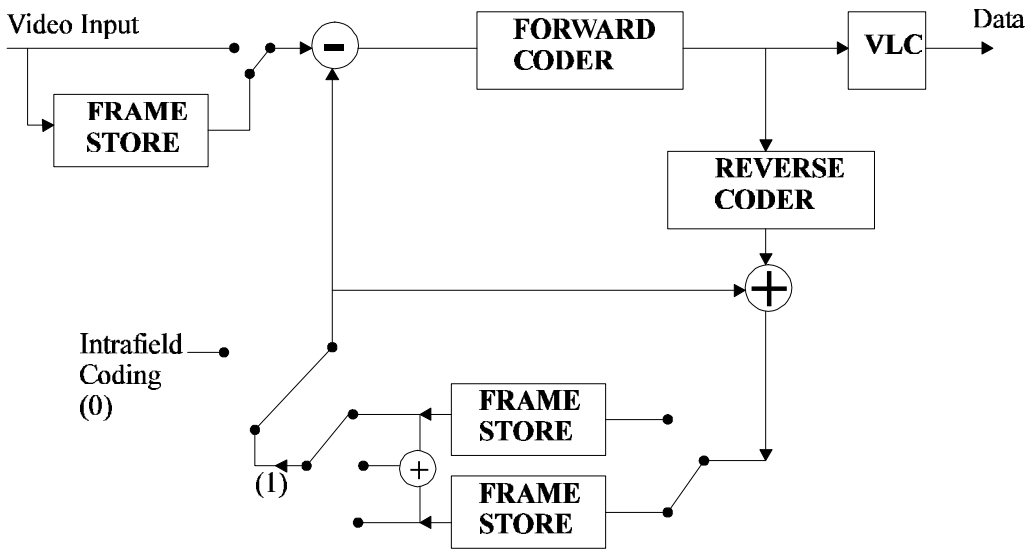


Figure 2.7.2: The MPEG-1 Coding Algorithm

One of the essential differences between MPEG-1 and the H.261 algorithm is the way in which interframe predictions are made. H.261 is primarily interframe coding using the previous frame as the main prediction source for the generation of the next frame. However, since MPEG-1 applies mainly to pre-recorded video sequence, subsequent frames can also be used to make a better prediction of the current frame. Figure 2.7.1 shows that in addition to the I-frames, occurring at every twelfth frame instant, there are groups of two B-frames, bounded by single P-frames between them. P-frames use forward prediction only from previous P- or I-frames in the sequence. However, B frames have both forward and back prediction from P- or I-frames. As B-frames are formed by the interpolation of earlier and later pictures, two motion vectors per block are needed. The effect of this is that MPEG codecs require enough storage space to hold two pictures for the coding process, as opposed to the one needed in H.261 coding. B-frames are not used as the basis of prediction for other frames in the sequence and so it has been found useful to encode them with a coarser quantisation

step interval than would be the case for I- and P-frames in the same sequence, since error propagation is not an inherent problem.

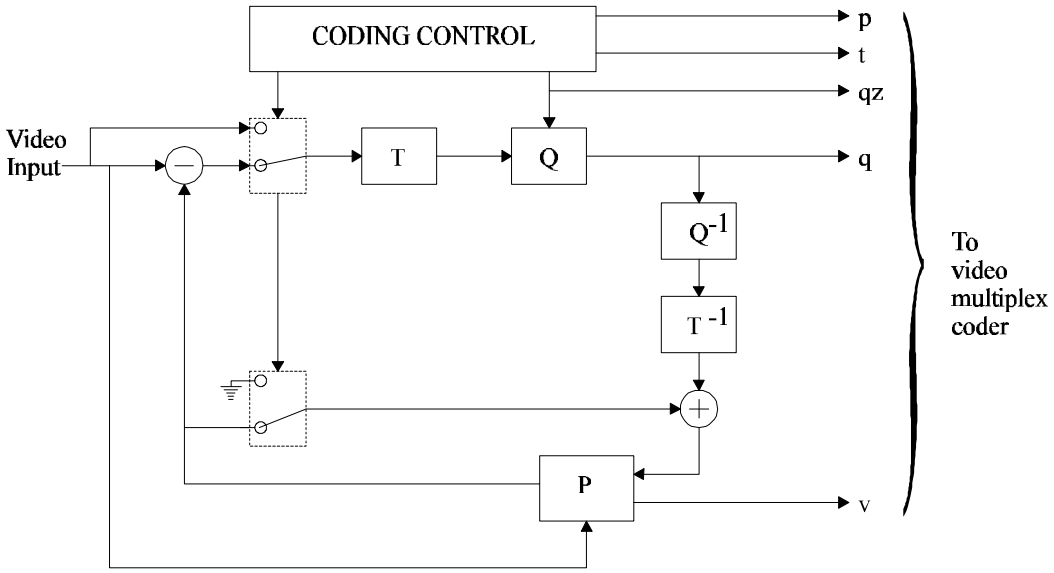
Motion vectors used by MPEG have a greater range than would be required for videoconferencing applications, since the nature of a wide range of video comprises more interframe motion than would be anticipated in a typical head-and-shoulders scene.

Subsequent work on MPEG standards has considered the application of the algorithm for data rates of up to 40Mbits/s. MPEG-2 [xv] has been adopted for direct satellite broadcasting in Europe and by the US Advanced Television Committee (FCC) for HDTV. It is effectively the same as MPEG-1, except that interlace scanning can be retained and interframe delays are less, resulting in a picture of improved quality.

2.8 The ITU-T H.263 Algorithm

As the latest development in low bitrate video compression algorithms, the H.263 algorithm [xvi][xvii] is intended to provide a framework for transmission at bitrates lower than the $p \times 64$kbits/s associated with H.261. The recommendation takes account of the growing popularity of home personal computers, connected to the PSTN by a modem, having bitrates of 14.4kbits/s or 28.8kbits/s. In consequence, recommendation H.263 is part of a set of ITU documents, relating to the line transmission of non-telephone signals. The outline of the system is contained in recommendation H.324 and peripheral information, such as the V.34 modem standard, relates directly to it.

H.263 is broadly based upon the H.261 algorithm, using block-based DPCM/DCT coding with associated motion compensation. However, there are some changes in the basic implementation and optional processes are available to improve the interframe prediction (figure 2.8.1).



T	Transform Coding
Q	Quantisation
P	Picture memory with motion compensated variable delay
p	Flag denoting INTRA/INTER
t	Flag for transmitted or not
qz	Quantiser indicator
q	Quantising index for transform coefficients
v	Motion vectors

Figure 2.8.1: H.263 draft recommendation encoder block diagram

2.8.1 H.263 Base Functions

As with H.261, H.263 supports the CIF format, however the QCIF and sub-QCIF (128×96 pixels) resolutions are more appropriate to the low bitrate environment.

The H.261 algorithm incorporates a spatial low-pass filter in the encoder feedback loop, which has been omitted from H.263. It has been shown that the pixel interpolation function involved in the half-pixel motion compensation process has the effect of low-pass filtering, without the need for a specific spatial function to remove high frequency noise caused by the quantisation of transform coefficients and also evident at the boundaries of blocks in the motion compensation process.

Motion compensation is optional, but all decoders must be capable of it. Macroblocks for colour video sequence comprise 16×16 pixels luminance, plus two corresponding 8×8 chrominance blocks. Vectors can take the form of one per macroblock, or on a block basis, where four vectors per macroblock would exist. The latter forms part of the Annex F 'Advanced Prediction Mode' of H.263.

The basis of macroblock sampling is shown in figure 2.8.2.

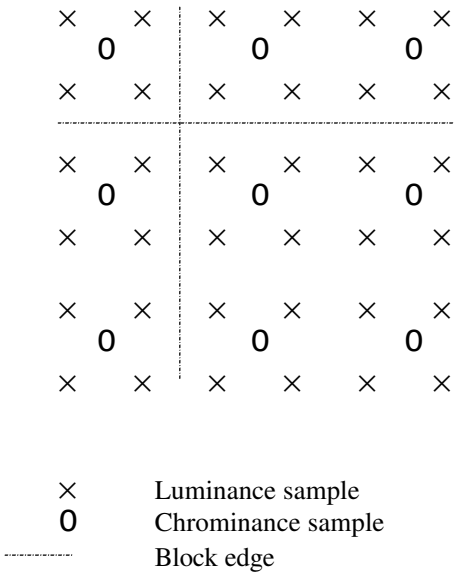


Figure 2.8.2: Positioning of block luminance and chrominance samples

Both types of motion compensation operate at half pixel resolution in the horizontal and vertical components. Motion vectors usually take the form of an absolute vector for a macroblock (or block, in the Annex F mode) and a prediction is formed as the median of the vectors for the three previously transmitted macroblocks (or blocks) to the left and above the current block. A positive value of the horizontal or vertical component of the motion vector signifies that the prediction was formed from pixels in the previous picture to the right or below the pixels being predicted. Vectors for the two 8×8 chrominance blocks in a macroblock are calculated as half the value of the single macroblock vector, or in the case of Annex F as 1/16 of the average of the four luminance block vectors, rounded to half pixel resolution.

H.263 uses a linear quantisation function with a fixed step-size for the DC component of an INTRA coded DCT block. Quantisation step size can be altered to accommodate variations in encoded bit-rate, caused by picture changes, together with the switch between intraframe and interframe coding and the time interval between coded pictures (variable frame rate). This allows the codec to adaptively trade-off the spatial and temporal resolution parameters of the system.

2.8.2 H.263 Optional Functions

A suite of additional functions is available to improve the interframe prediction.

2.8.2.1 Unrestricted motion vectors (Annex D)

In the default prediction mode of H.263, the search for motion vectors can only take place inside the normal picture. In the Unrestricted Motion Vector mode, this requirement is removed and motion vectors are allowed to point outside the picture. To do this, edge pixel values are extrapolated in *x* and *y* directions as appropriate, producing a virtual search area outside the normal boundaries (figure 2.8.3).

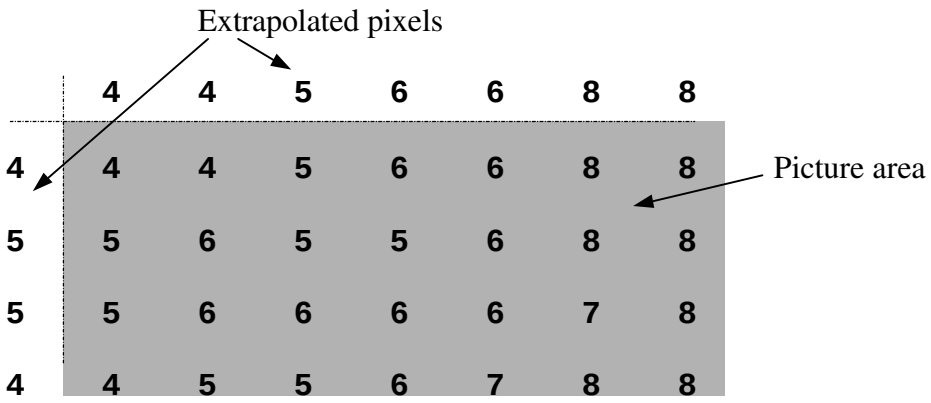


Figure 2.8.3: Extrapolation for Unrestricted Motion Vectors

Unrestricted motion vectors can improve the image prediction, particularly where there is motion involving objects entering or leaving the scene, or where the camera itself is moving in a pan. This mode is optional as it does not improve the prediction for static camera and central objects (which would be common in videoconferencing).

2.8.2.2 Syntax-based Arithmetic Coding SAC (Annex E)

SAC is a variant of Arithmetic Coding, used in place of the traditional Variable Length Code for minimum-redundancy serial transmission. The optimum length of Variable Length Codes is derived from the entropy of the data and tends to be non-integer. Arithmetic Coding, on the other hand, allows fractional bits per symbol and removes this inefficiency. It works together with a modeller to evaluate the probability of a particular symbol in the bit stream and the process varies depending on the type of information being coded, be it a Coded Block Pattern or Motion Vectors.

The implementation of SAC is, however, rather complex and it is impossible to recognise individual symbols in an encoded bit stream. Recovery from errors is difficult, since SAC does not re-synchronise after a few false symbols, as Variable Length Codes do.

2.8.2.3 Advanced Prediction Mode (Annex F)

This uses four motion vectors instead of one per macroblock and uses overlapped block motion compensation [xviii], which tends to provide a smoother prediction image and a better spatial quality at the decoder. It is required that this mode operates in conjunction with the Unrestricted Motion Vector Mode (Annex D), to

make a consistent prediction from the availability of extrapolated luminance and chrominance pixels.

The four 8×8 pixel luminance blocks in a macroblock allow a better representation of motion to be made, albeit at the price of a greater data overhead. It is therefore the responsibility of the implementing organisation to decide the value of this additional motion data.

2.8.2.4 PB-Frames (Annex G)

The algorithm also allows for the use of forward and bi-directionally predicted frames, similar to those described in section 2.7. Motion vectors can be used from the P-frames to generate predictions for the B-frames. Additional vectors may also be transmitted as an optional mode, which effectively doubles the temporal resolution of the image with only a small increase in the coded video data rate. However, this tends to produce a less satisfactory prediction in sequences having very fast or complex motion, or low initial frame rates.

2.8.3 Summary

The H.263 algorithm has been demonstrated as a versatile low bitrate video coding procedure, with particular application for PSTN communications, where 'software codecs', using the processing power of a contemporary personal computer can do away with the need for an expensive custom receiver.

Comparisons with the H.263 algorithm will be made later in this thesis, contrasting the parameters of bitrate and signal-to-noise ratio in both the base-level functionality and with the incorporation of annexes D-G.

Chapter 3

Motion Compensation

Algorithms describing feature displacement

3.1 Introduction

Chapter 2 described methods of picture coding using the difference between spatially or temporally adjacent pixels. It is certainly true that these techniques provide an effective means of picture coding. However, the need to maintain a constant data rate during moments of more significant motion can result in an image reproduction of low quality.

The performance of interframe coding can be significantly improved by the use of motion compensation. If movement has occurred, it is possible to map the displacement of groups of pixels, representing either image features or regular blocks, in the next frame. For example, the pixels representing part of a moving lip in the head and shoulders examples can be grouped and a single vector used to denote their origin and destination, provided no significant change in topography has occurred.

Motion compensation and estimation techniques can broadly be divided into four main categories, the method of differentials, Fourier techniques, block matching and model coding. For completeness this chapter gives coverage to each method, even though the novel algorithm described in this thesis is based on feature description and classification.

3.2 The nature of interframe motion

One fundamental concept to emphasise is that each frame in an image sequence is merely a snapshot of some real-life event at a given moment in time. A video sequence is no different to cinema or an animated cartoon, in that it is a collection of still images displayed in temporal sequence, at high speed, to give the viewer an *impression* of motion. This may sound rather basic, but the effectiveness of human psychovisual perception (see chapter 4), depends to a great extent on the nature of the image subject, as well as how it is displayed.

The tendency we have to see linear motion in a sequence of still images is referred to as *temporal aliasing*. Normally this is a natural process that causes us no difficulty, although sometimes the mechanics of temporal aliasing can easily be seen. Occasionally real time motion is filmed at a frame rate incompatible with the motion taking place. An ideal example of this is in old Western films, where wagon wheels seem to rotate slowly backwards, even though the wagon itself is travelling forwards at speed. The reason for this is that the temporal sampling of the wheel is such that it rotates slightly less than one revolution between each frame and the difference it should have made up is seen as being a small rotational movement in the opposite direction, as shown in figure 3.2.1.

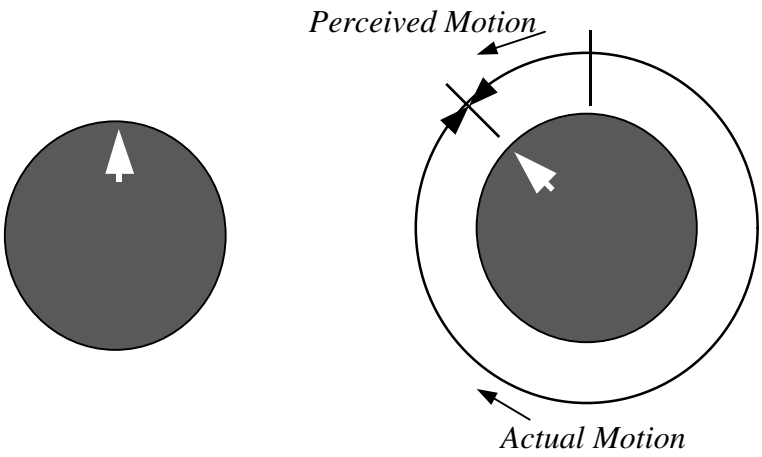


Figure 3.2.1: The effect of temporal aliasing

Moving objects can be considered as residing in a spatio-temporal spectrum of their own, that allows an understanding of the processes at work. The motion of a particular feature in an image scene can distinguish it from other stationary objects, as well as the background. Consequently, the temporal activity of an object provides the viewer with more interest in its features.

3.2.1 The spectrum of a moving object

The spatio-temporal spectrum of an object undergoing displacement is described by Thomas [xix][xx]. If we consider a point moving horizontally at fixed speed, its trajectory in one dimensional space can be plotted graphically (figure 3.2.2). The line will be vertical if the velocity is zero, with angle of skew increasing for higher velocities. Spatial and temporal axes are labelled m and f respectively. For a stationary point, its spectrum in the domain lies along the $f = 0$ axis, with the spectrum skewed due to motion, as shown in figure 3.2.2(ii). To illustrate the cause of skew, consider the temporal frequency of a fixed point in the image - the contribution to the

temporal frequency from a given spatial frequency is the product of the spatial frequency and the velocity of displacement. It follows, therefore, that higher spatial frequencies give rise to higher temporal frequencies.

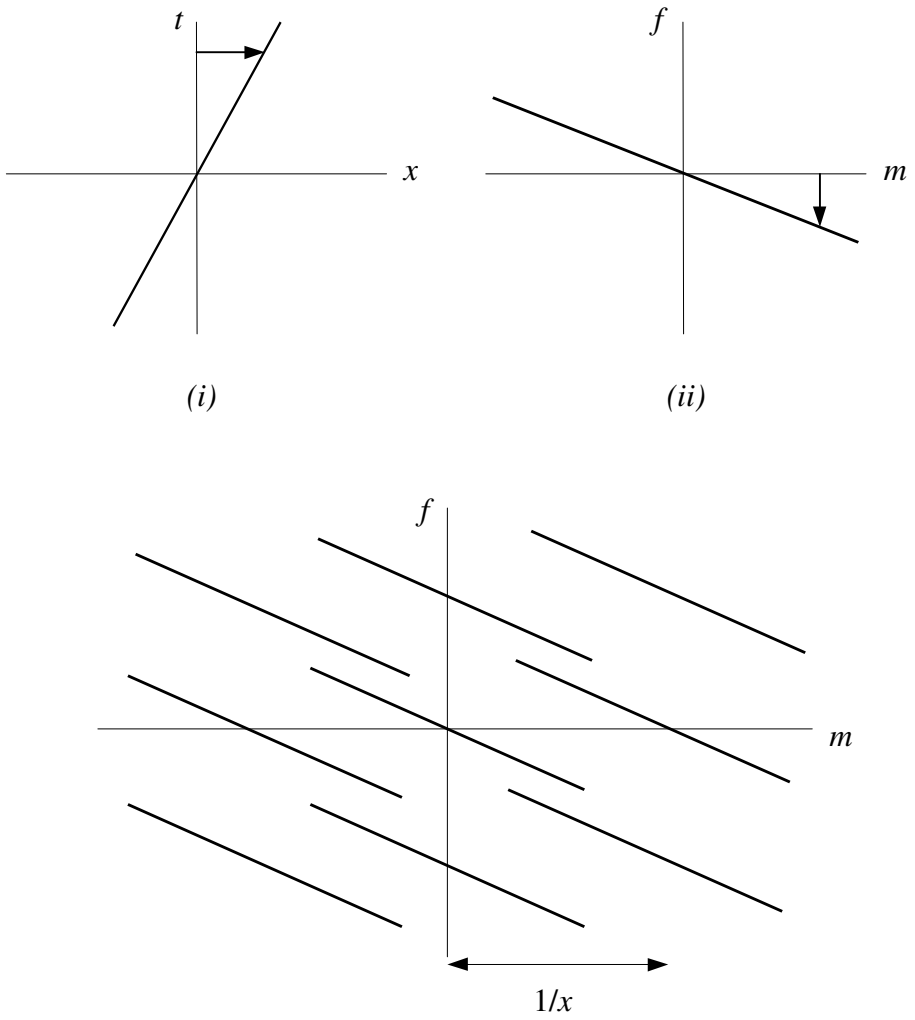


Figure 3.2.2: The spectrum of a moving object. (i) the space-time description of a moving point, (ii) spatio-temporal frequency spectrum before sampling, (iii) spatio-temporal frequency spectrum after sampling.

When the image is sampled, either in space or time, the spectrum is repeated at intervals of the reciprocal sampling frequencies. Figure 3.2.2(iii) shows a replicated spectrum in the case of orthogonal sampling. Notice that there is insufficient pre-filtering, prior to sampling, to prevent the repeat spectra from overlapping both spatially and temporally at particular motion speeds.

3.2.2 Post- and pre-filters

The most important post-filter process, occurring after the image has been displayed on a screen, takes place in the human psychovisual perception system. Different effects have been observed for the separate cases of an eye staring at a fixed spatial point on a screen and an eye which is tracking a moving point. Greater detail on the human spatio-temporal response is found in Budrikis (1973)[xxi].

Figure 3.2.3.1 shows the spatio-temporal frequency response of a staring human eye, superimposed on the spectra produced from supplying the sampled signal described in the previous section into a CRT display. Sampling frequencies used are set so as to make the first temporal repeat spectrum just visible - an example of which is the flicker that occurs on a 50Hz television picture. High spatial frequencies cannot easily be detected on the retina due to blurring.

Figure 3.2.3.2 shows an exaggerated example of fast motion, still with a fixed gazing point. At this speed, components from the first temporal repeat spectrum lie close to the axis of zero temporal frequency, appearing as almost stationary objects (an explanation of the slowly-reversing wagon wheels mentioned earlier). There is a very marked loss of spatial resolution at this speed. However, if the eye is to track an

object, the eye's post-filtering mechanism skews in the same way as the object being tracked, such that spatial resolution is enhanced and only flicker causes temporal low-quality.

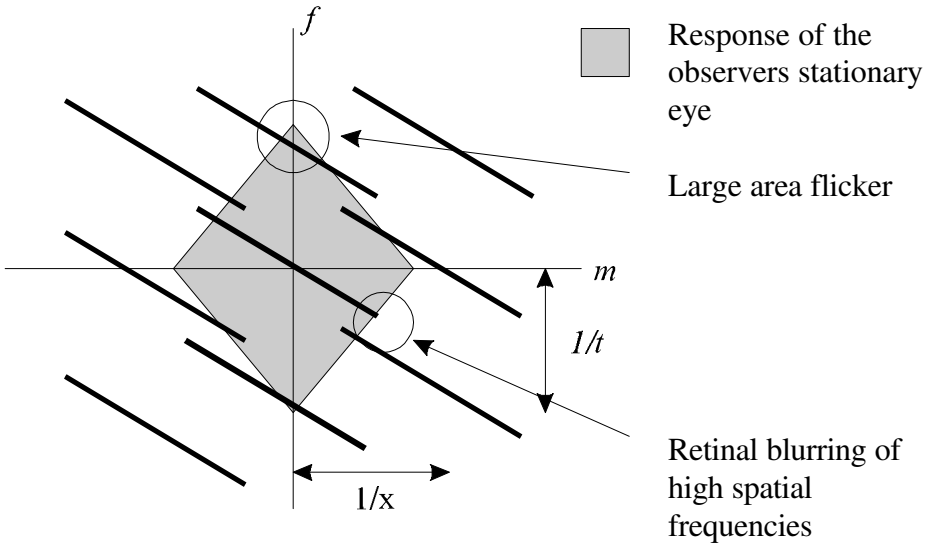


Figure 3.2.3.1: The spatio-temporal response of the stationary eye to slow motion

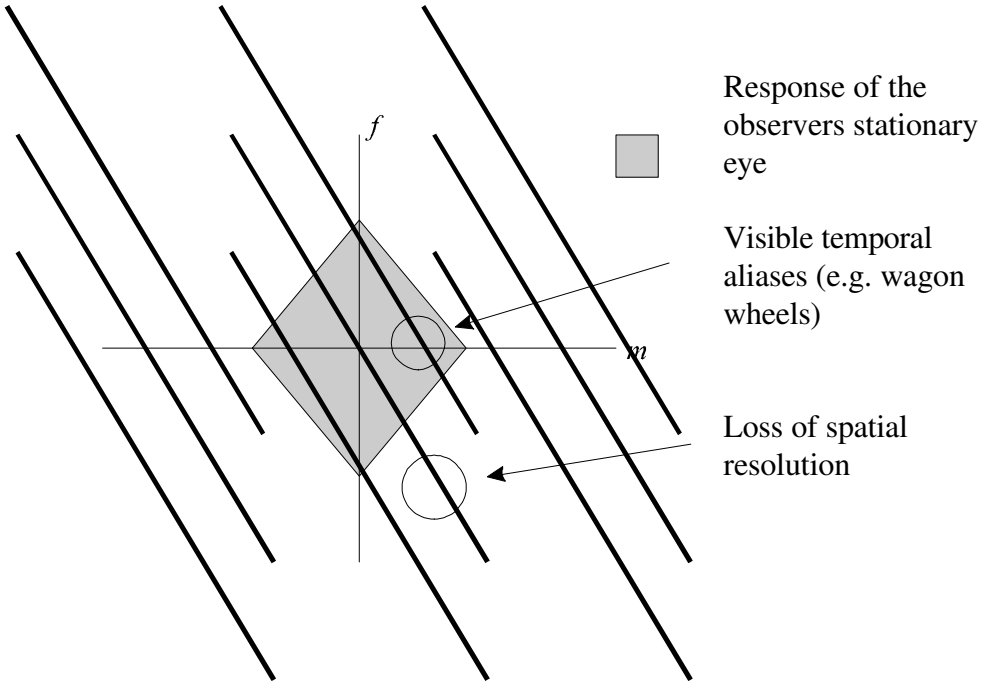


Figure 3.2.3.2: The spatio-temporal response of the stationary eye to fast motion

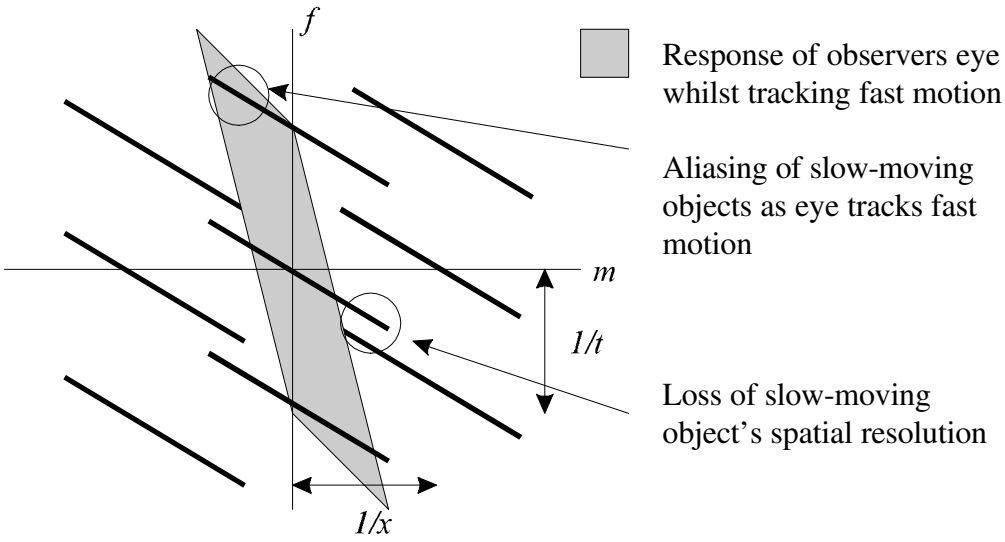


Figure 3.2.3.3: The spatio-temporal response of the eye tracking fast motion, with slower motion also present

This is a satisfactory explanation of the spatio-temporal response of the case where a single component is moving in an image scene. However, it is more usually the case that there are several objects moving at the same time in different directions. In the case of television coverage of an ice skater, if the observer was to track the moving background whilst the camera panned to follow the skater, the observer would see multiple images of the skater on the periphery of their vision (figure 3.2.3.3).

Such problems are normally overcome by increasing the frame rate in the display so as to provide more than enough information for the human perception mechanism to effectively perform temporal aliasing. Unfortunately, this is not often acceptable from an engineering stance, since the provision of more frames requires more bandwidth in the radio-frequency spectrum and is incompatible with the desire to actually reduce the amount of data required to reproduce an image sequence of acceptable quality.

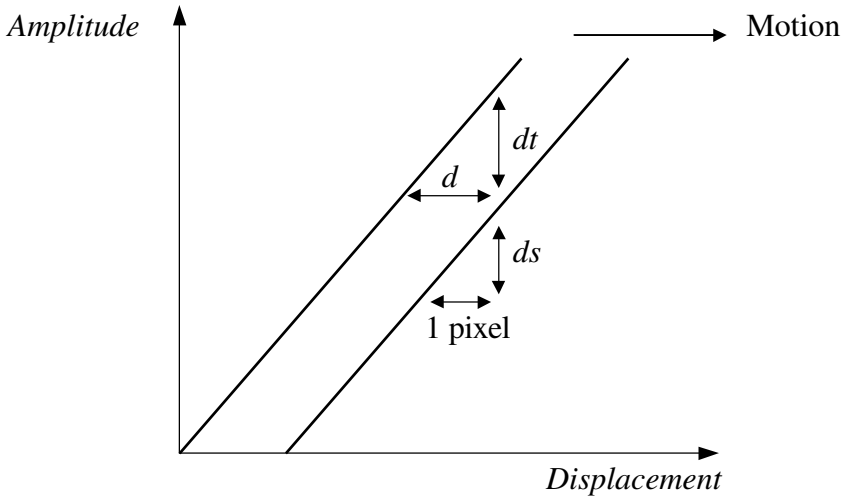
3.3 Methods of motion detection and estimation

Many methods of motion estimation and representation have been investigated and whilst an exhaustive review would be a major undertaking, an overview of the four main areas of interest has been completed.

3.3.1 Method of differentials

This is a simple technique that relies fundamentally on two basic assumptions. Firstly, that luminance is a linear function of position over the distance an object could move in a given frame period - i.e. the displacement is small in comparison with the highest image frequency present. Secondly, the luminance of objects remains constant as they move. The second of these conditions is common to most basic methods of motion estimation, because features are not necessarily regarded as objects they represent, but rather arbitrary groups of similar-value pixels. As will be described later, this is a limitation on the effectiveness of any algorithm.

The process of differentials is shown in figure 3.3.1, illustrating the process in a single dimension. The luminance difference between corresponding pixels in temporally adjacent frames, *dt*, and spatially adjacent pixels in a single frame, *ds*, is calculated. A ratio *dt:ds* shows the interframe displacement of pixels, which can easily be observed from the diagram. Limb and Murphy [xxii] proposed the use of this technique to measure the speed of motion in video sequences, with two modes - optimised for both slow and fast motion.



——— Luminance level in adjacent frames

dt temporal amplitude difference

ds spatial amplitude difference

d interframe displacement (for calculation)

Figure 3.3.1: Estimating interframe displacement by differentials

The assumptions already mentioned can be described, by modelling the image using the equation:

$I(x, y, t + \Delta t) = I(x - v_x\Delta t, y - v_y\Delta t, t)$ [Equation 3.1]

where $I(x, y, t)$ is the intensity of a pixel at (x, y) at time t

and v_x, v_y are average components of motion between time t and $t + \Delta t$

for all regions of an image, except those through which an object has already passed during time Δt. A Taylor expansion of equation 3.1 gives:

$$v_x \frac{\partial I}{\partial x} + \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} + E(x, y, t) = 0 \qquad \text{[Equation 3.2]}$$

where $E(x,y,t)$ represents the high-order terms of I.

Theoretically, the two directional velocity components, v_x and v_y, can be obtained if the spatial and temporal derivatives of image brightness are available at the positions of anticipated motion. It is assumed that the higher-order components of E can be ignored.

Whilst the method of differentials has been used to some effect in low-resolution schemes, it is unable to cope with large displacements. The use of a *pel recursive* method, such as that described by Netravali and Robbins [xxiii], does help and some form of spatial pre-filtering can be applied to the picture to remove high-frequency components, allowing an approximate measure of motion to be made. The technique is, however, unsatisfactory in the generation of motion vectors for temporal interpolation, since it matches similar areas rather than producing vectors which correspond to actual motion between frames.

3.3.2 Fourier Methods

Use of the Fourier transform provides a more satisfactory estimate of true motion. However, the algorithms that result are often complex and impractical for use in real-time coding schemes. The phase information, produced by taking the 2-D transform of each frame, is used to determine relative interframe displacements.

If we consider two successive frames, I_1 and I_2 of a scene undergoing some form of translatory motion with a motion vector of (v_x, v_y) pixels per field (frame) period, then:

$$I_2 (x, y) = I_1 (x - v_x, y - v_y) \qquad \text{[Equation 3.3]}$$

ignoring edge and interlacing effects.

Taking the Fourier transform of each side and employing the shifting theorem gives:

$$F_2 (m, n) = F_1 (m, n)\, e^{-\pi j(m \vec{x} + n \vec{y})} \qquad \text{[Equation 3.4]}$$

where F_1 is the Fourier transform of I_1 and m, n represent spatial frequencies. $\vec{x}$ and $\vec{y}$ are equivalent to the vector components v_x, v_y.

The Fourier transform of the (circular) cross-correlation of the images is:

$$\Im (C) = F_1 . F_2^* \qquad \text{[Equation 3.5]}$$

$$= F_1 . F_1^* . e^{2\pi j(m \vec{x} + n \vec{y})}$$

where $\Im$ () represents the Fourier transform.

Dividing this expression by $F_1 . F_1^*$ before taking the inverse transform, yields:

$$C(x, y) = \delta(x - v_x, y - v_y) \qquad \text{[Equation 3.6]}$$

showing that the correlation function has become a delta function located at the required displacement. Generally speaking, if I_2 was not a pure translation of I_1, but differed in overall luminance, we would calculate:

$$C(x, y) = \Im^{-1} \left\{ \frac{F_1 F_2^*}{|F_1 F_2^*|} \right\} \qquad \text{[Equation 3.7]}$$

where $\Im^{-1}$ represents the inverse Fourier transform.

The effect of this process has been to normalise the spectrum of the two adjacent images I_1 and I_2 before performing a cross-correlation. The normalising process has extracted the phase information from the transforms. This technique is specifically known as phase correlation, which differs primarily from normal cross-correlation in that the sharpness of the peaks allows us to distinguish several types of motion. Other features of this technique are that scene brightness changes do not affect measurement, provided they have a low spectral bandwidth (if, for example, the overall luminance drops due to a large shadow being cast on the scene). Fourier transforms are more efficient than using 'long-hand' cross-correlation, particularly for large displacements, with less multiplications required. However, the computational complexity for a real-time sequence is still a problem and hence Fourier techniques tend not to be used for real-time video processing.

3.4 Block matching algorithms

Having already shown that pixel-recursive techniques have inherent computational intensity, experience has demonstrated the effectiveness of block-matching methods. These are a fairly large category and are well-documented [xxiv], ranging on one hand from arbitrary full-search, regular block size, algorithms to more adaptive variable-

size block searching, where the blocks have a greater correlation with actual shapes in the scene.

3.4.1 Full-search block matching

This technique, refined by Jain and Jain [xxv] involves the division of an image into small blocks of a given size. Using a larger search window, all possible blocks of the same size in the previous frame are evaluated. The position at which least errors occur is assumed as the origin of a current block and a displacement vector produced, as shown in figure 3.4.1. If we denote the size of the search subject block as being a square of dimensions n × n and the search window a square of $N \times N$ pixels, then a simple summation can be employed to test each of the possible $n \times n$ blocks in the previous frame. This can be expressed by the following function:

$$\sum e^2 = \sum_{x=0}^{n} \sum_{y=0}^{n} \left| P_c(x,y) - P_p(x,y) \right|^2 \qquad \text{[Equation 3.8]}$$

where P_c is a pixel value on the current frame in the subject block R_c

and P_p is a pixel value on the previous frame in test block R_p.

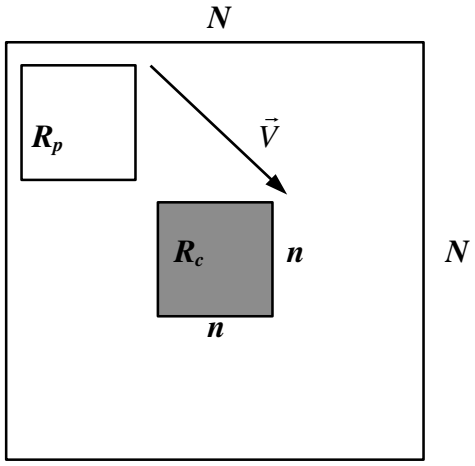


Figure 3.4.1: Generating a displacement vector by full-search block matching.

With the generation of a motion vector $\vec{V} = (dx, dy)$, the previous frame subregion, R_p, can be re-mapped in the current frame to the location R_c. Although this would appear to be a generally satisfactory method of interframe motion estimation, it is important to note that the blocks used for searching contain completely arbitrary picture information. Calculating the errors between blocks of interest is not an exhaustive technique. It is possible that in a given search window, $N \times N$, there will be several blocks from the previous frame that meet the required minimum error criteria, simply because they possess the same net pixel intensities. However, there is no mechanism to ensure that the location of given values within the block under consideration *uniquely* correlates with those in the current subject block, R_c. For images having large areas of uniform texture and intensity, the method of minimum error calculation fails, as no reliable estimate of interframe motion can be produced.

However, the full-search block matching technique has proven feasible to implement as a relatively simple algorithm, for the purposes of interpolative motion compensation in predictive hybrid DPCM/DCT codec schemes. Indeed, it was adopted by CCITT Study Group XV [xxvi] in their base models for a videoconferencing codec system operating on channels of $p \times 64$ kbits/s. Hardware solutions of varying efficiency have been constructed [xxvii][xxviii], however whilst this technique is simple, the process of block-matching motion compensation (BMMC) requires a significant computational overhead. To demonstrate this, we can use the example of BMMC in a standard Common Intermediate Format (CIF) sequence, using only luminance values as the comparison for temporally adjacent frames. If we set the block size ($n \times n$) to be 8×8 pixels and the search window ($N \times N$) to be 24×24 pixels, then:

- for each search window, there are 17×17 possible locations of 8×8 blocks = 289 positions per search window;
- the calculation of least errors requires 64 multiplications and 64 subtractions, per block. Hence, for each search window we have 128 arithmetic operations in 289 positions = 36992 operations;
- Each frame has 400 windows of 24×24 pixels.
  So 400×36992 operations = **14 million operations per frame**
- and with 30 frames per second, the real-time computational overhead is in the order of **480 million operations per second**.

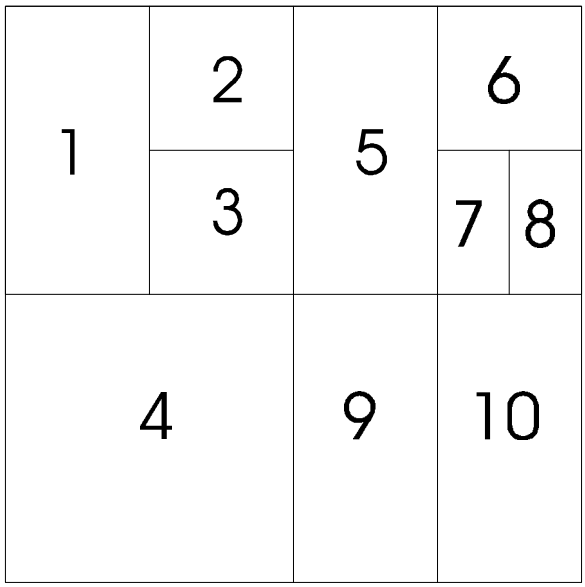
It is clear that for the purposes of real-time video codec design, this alone raises significant implementation issues. The inclusions of fast integrated circuits to perform this level of processing is impractical in low-cost video codec hardware.

A further drawback to the simple full-search method is that as motion vectors are estimated on a block by block basis, it is assumed that motion within the block itself is uniform. To make this assumption more practicable, small blocks (8×8 or 16×16) are used, but there is an essential trade-off between the number blocks present in the image, the amount of processing needed and the reconstructed quality. Were it not for the real-time processing limitations of the system, simple block matching with small blocks would be ideal. Where motion vectors are employed as "side" information in predictive codecs, this overhead cannot be accommodated.

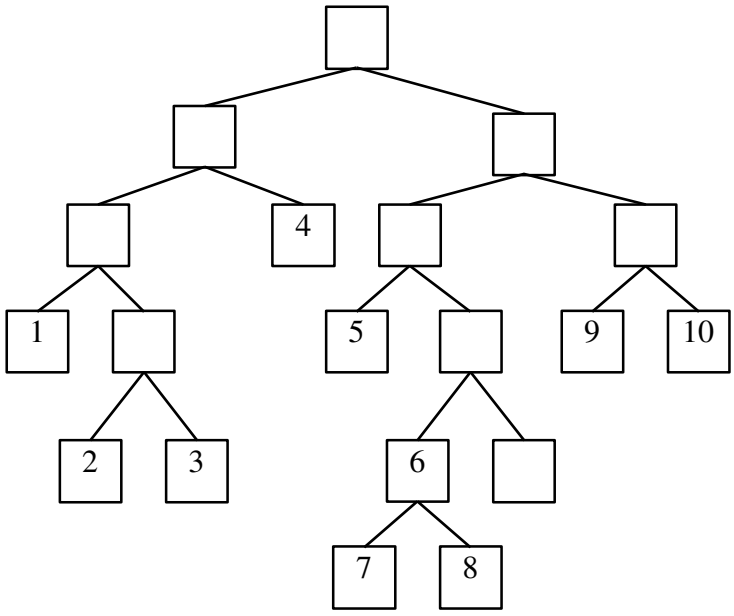
3.4.2 Variable-size block matching

3.4.2.1 Image decomposition

A refinement to the conventional theme of block matching is described by Chan, Yu and Constantinides [xxix]. Using different block sizes, such that smaller blocks describe areas of most detail, a more efficient algorithm can be developed, whilst retaining the basic search method for blocks of similar sizes. The technique is similar to the algorithm proposed by Horowitz and Pavlidis [xxx], in which a binary- or quad-tree traversal algorithm is used to hierarchically segment each frame into regions of uniform intensity. Figure 3.4.2 shows the application of binary decomposition to a square image segment.



(i)



(ii)

Figure 3.4.2: Variable size block matching motion compensation showing (i) an example of decomposition and (ii) the resultant binary tree. [After Chan, Yu and Constantinides.]

Each frame is subdivided using the binary tree process shown. For each block, an attempt is made to match it to blocks in a search window on the previous frame, assuming that only a certain number of blocks of the same shape may exist in that region. Sub-division into smaller blocks is performed using the calculation of a sum-square, employing an algorithm similar to that shown in equation 3.8. The splitting continues until the error approximation for the block falls below a certain threshold, or until the smallest tolerable block size has been reached. In this way, a bottom-up process is developed, starting at the largest block available which is sub-divided into many, smaller, constituent blocks. The blocks are recognised as representing regions of uniform intensity which is a good platform from which to commence a more adaptive method of block-based coding. However, the blocks themselves are still of regular dimensions and only follow statistical trends in image content, rather than describing the nature of spatial features.

A further adaptation of this approach is made by Seferidis and Ghanbari [xxxi], extending the binary-tree decomposition to a full quad-tree method where each node, unless it is a leaf, generates four children (figure 3.4.3).

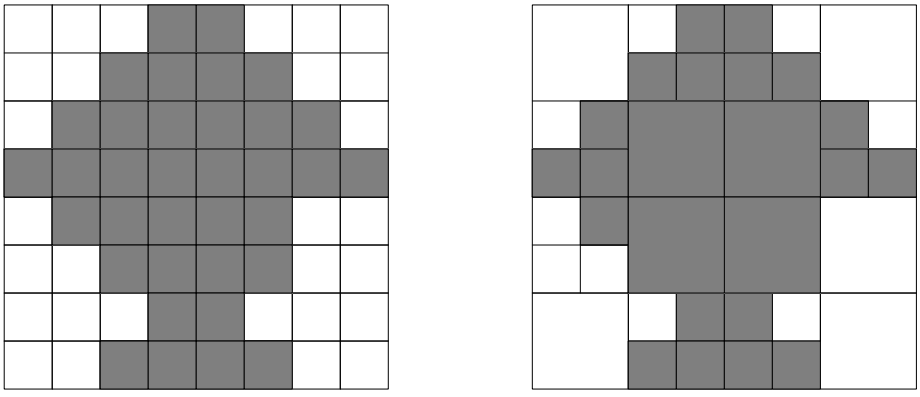


Figure 3.4.3: (i) Original fixed-block image and (ii) its quad-tree decomposition

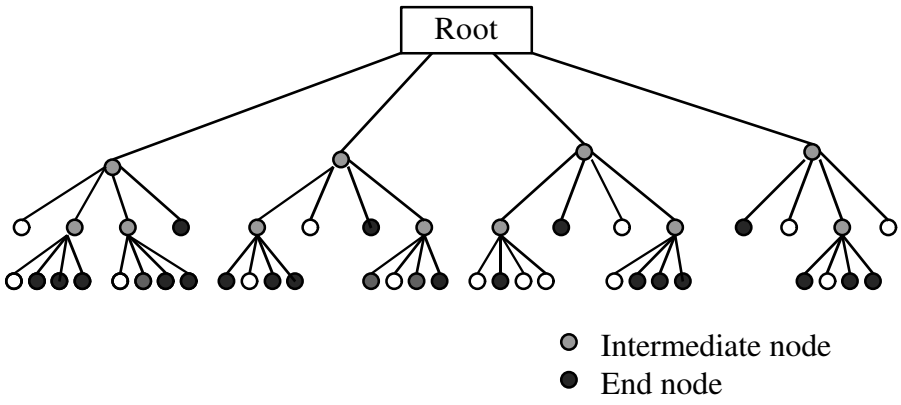


Figure 3.4.4: The process of quad-tree segmentation [after Seferidis and Ghanbari]

The basis of both binary and quad-tree decomposition is that, to limit the extent of processing required, maximum and minimum sizes are set for the root and smallest block respectively. Whilst it may be desirable to specify the entire image as the root, results for still and moving images [xxxii] have shown that it is practical to start with smaller root-blocks of 32×32 pixels working down to small blocks of 8×8 pixels. Hence, the tree diagram shown in figure 3.4.4 is the extent of the processing required for the variable block size decomposition of the image segment of figure 3.4.3.

The absolute temporal difference (ATD) is defined as the measure by which blocks are subdivided in a hypothesis test. Simply this is the sum of the absolute differences for a spatially-equivalent block in the previous frame and those of the corresponding block in the current frame. If this value is greater than some pre-determined threshold, further division of the image segment is necessary. It is clear that this process is not wholly satisfactory, since the information represented by the ATD says very little about the nature of interframe motion. However, since the application of motion vectors is primarily to reduce the interframe error signal, rather than provide

an accurate description of motion, it may not necessarily be considered a disadvantage. To ensure that subdivision only takes place where there is still a significant change in luminance, the threshold value employed is proportional to the number of pixels in the block. For blocks having an ATD above the minimum threshold T_{min}, a second adaptive threshold is used to divide the blocks into four children (for the case of quadtree decomposition). An algorithm for the assignment of blocks is then followed:

(1) choose four adjacent blocks of 32 × 32 pixels, each representing a child;

(2) calculate the absolute temporal difference for each as a, b, c, d;

(3) set the adaptive threshold T_a = (a+b+c+d)/8;

(4) if $T_a < T_{min}$, then set $T_{min} = T_a$;

(5) if the absolute temporal difference of any child is above this threshold, subdivide it;

(6) stop if the minimum block size (8 × 8) has been reached, else return to step (2).

Results have shown that the effect of this algorithm on a CIF image (352 × 288 pixels), yields 99 blocks of the maximum block size, 32 × 32 pixels. The coding overhead can be significantly reduced by the use of Huffman variable-length codes according to the probability of occurrence of each block. Assuming codes of 0, 01 and 11 for blocks of 8 × 8, 16 × 16 and 32 × 32, respectively, the binary tree can be encoded with a very small overhead. For the image *Claire* (figure 2.1.1), the application of quadtree segmentation yields a data overhead of 304 bits for tree encoding. Seferidis and Ghanbari claim this to be comparable or less (depending on

picture activity) than the basic macro-block structured bit pattern used in the standard H.261 codec for addressing coded/non-coded blocks.

3.4.2.2 Displacement vectors

Having a range of different size blocks provides a useful basis on which to develop an adaptive method of interframe motion detection. It has been shown that smaller blocks relate to areas of detail in an image, and it can be assumed that the moving edges of the foreground subject, compared with the background, will be represented by the smallest permissible block size. This assumption allows us to suggest that much larger blocks will exist to represent a lack of interframe activity, particularly in the background. Block matching can be performed using the full-search method described in section 3.4.1. Using a local search template of ±16 pixels in the previous frame, with respect to the reference block, the search overhead is the same for each decomposed block, regardless of size. However, if it is assumed that motion is much less, the search window can also be a variable size. 32×32 pixel blocks tend to exhibit a low interframe activity and can normally be compensated by only a translational motion vector. Smaller blocks, on the other hand, may demonstrate complex rotational and perspective changes for which a further test may be required.

Generally speaking, the spatial translation for block displacement between two adjacent frames can be depicted by two mapping functions, f_1 and f_2, where in the current frame:

$x_i^c = f_1(x_i^p, y_i^p)$ and $y_i^c = f_2(x_i^p, y_i^p)$ [Equation 3.9]

where x_i^c and y_i^c are the current block values

and $f_n(x_i^p, y_i^p)$ is the translation of pixels in the displacement of a block from the previous frame.

The effect of the variable shape block matching method is shown in figure 3.4.5.

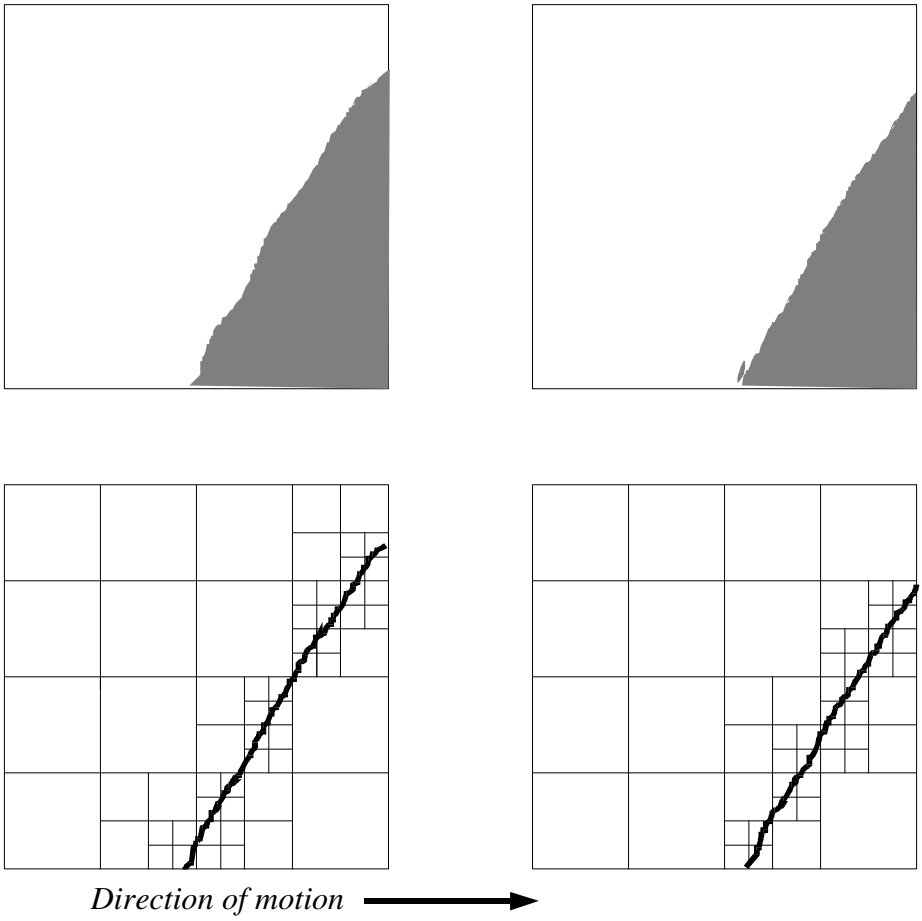


Figure 3.4.5: Edge motion affecting only smaller variable shape blocks

3.4.2.3 Edge block classification

Whilst these smaller blocks provide a more accurate representation of feature contrast than might be the case with regular block sizes, there is still an arbitrary nature to the detail they represent. Recent work by Lee and Crebbin [xxxiii] has extended the scope of segment classification to provide a set of primitives to describe high-detail 4 × 4 pixel regions. Using eight edge classes and one mixed class in the normalised DCT domain, pattern matching each of the blocks can be linked to six DCT coefficients. The DCT of a 4 × 4 image vector is given by:

$$\boldsymbol{F} = \begin{bmatrix} F_{0,0} & F_{0,1} & F_{0,2} & F_{0,3} \\ F_{1,0} & F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,0} & F_{2,1} & F_{2,2} & F_{2,3} \\ F_{3,0} & F_{3,1} & F_{3,2} & F_{3,3} \end{bmatrix} \qquad \text{[Equation 3.10]}$$

where the relationship of $F_{u,v}$ to the spatial values $F_{x,y}$ is analogous to that stated in equation 2.5.

Components in the horizontal and vertical domains can be represented by two edge feature sets, horizontal feature F_{hor} and vertical feature F_{ver}:

$$\left.\begin{aligned} F_{hor} &= \{F_{u,0} \;\; u = 1,2,3\} \\ F_{ver} &= \{F_{0,v} \;\; v = 1,2,3\} \end{aligned}\right\} \qquad \text{[Equation 3.11]}$$

The set can be expanded such that diagonal components, such as 45° and 135°, can be represented by a combination of vertical and horizontal features.

This technique has been implemented such that it only extracts edge features on the basis of their location and orientation. For reference, the DCT edge features of equation 3.11 are grouped into one DCT set $E = \{E_k;\ k = 1, ..., 6\}$ and the largest value between the two dominant DCT coefficients E_1 and E_4, which correspond to $F_{1,0}$ and $F_{0,1}$ in equation 3.11, is defined as the pivot point for the DCT set. Each member of the set is the normalised so that the pivot point is unity, so that it is then found that $\boldsymbol{E}' = \{E_k/E_1\}$ (or $\boldsymbol{E}' = \{E_k/E_4\}$, for $k = 1, ..., 6\}$. This is then used to represent a given edge feature, without reference to the value of pixels in the spatial domain. Lee and Crebbin have specified 24 edge models for 4×4 segmented blocks, examples of which are shown in figure 3.4.6.

F01=	0.0000	0.0000	1.0000	-1.0000
F02=	0.0000	0.0000	0.3170	0.0000
F03=	0.0000	0.0000	-0.0711	-0.0711
F10=	1.0000	-1.0000	1.0000	-1.0000
F20=	0.7654	-0.7654	0.3170	0.0000
F30=	0.4142	-0.4142	-0.0711	-0.0711

Figure 3.4.6: Use of DCT coefficients to describe sample 4×4 pixel edge models [after Lee and Crebbin]

To generate the basis of a fixed-length codebook, groups of 4×4 pixels are taken from the input of the codec and their normalised DCT coefficients compared with each of the 24 combinations in the edge model set. The groups are then represented by the model which is closest.

3.4.2.4 Codebook design

Lee and Crebbin's work demonstrates that a codebook cannot be exhaustive in terms of the features it represents. Some limitation has to be imposed, where the quality of reconstructed images has to be traded-off against the overall data requirements of the coding system. Contemporary techniques of codebook design, to represent in the most efficient way each type of block encountered, are based on the technique know as the LBG algorithm, after its founders Linde, Buzo and Gray (1980) [xxxiv]. A primary codebook comprises twelve different sub-codebooks - three for variable size blocks in low-detail regions and nine for edge and mixed blocks in high-detail regions. All the sub-codebooks are based on the descriptor residing in the DCT domain.

As will be discussed in chapter 4, the human perception mechanism is critical of local areas of concentrated low-quality, compared with a smaller, overall reductions in image quality. Hence the size sub-codebooks must be selected such that the optimal size produces the least average distortion in the reconstructed image. Ramamurthi and Gersho [xxxv] suggest a weight set, $\{r_i\}$, to satisfy this requirement, which has a relationship to the asymptotically optimal sub-codebook sizes, $\{ N_i^* \}$, given by the equation:

$$\frac{r_i P_i D_i^*}{N_i^*} = \text{constant} \qquad \text{[Equation 3.12]}$$

where P_i is the probability of a block residing in the *i*th class. However, it is hard to find an analytical method to derive the set $\{ N_i^* \}$ that produces the optimal picture distortion $\{ D_i^* \}$, since the probability density of the input image is unlikely to be known. It is normally assumed that the distortion criteria is constant throughout

an image sequence, which is acceptable provided there is no significant change in interframe activity. It has been shown [xxxvi] that the perceived quality of low-detail regions where sub-codebook sizes are the same, is not subject to a noticeable variation. However, a variation in the sizes of sub-codebooks representing high-detail areas, seemed to produce an approximately equal variation in spatial quality.

Hence it is desirable to have a greater range of block classification for high-detail areas of an image than would apply for regions of low-detail. The probability of a block residing in any class within a codebook is reduced as the size of the codebook is increased, simply because there are more descriptors available to describe a block. The statistical relationship of block descriptions and the probability of block occurrence has been examined at length by Bergeron and Dubois [xxxvii]. They describe a technique of maximum *a posteriori* probability (MAP) criteria, used to set a descriptive function and parameters, that can represent local block motion vector fields.

3.4.2.5 Variable block size motion estimation - summary

The techniques described have only covered one area of block-based motion compensation and many other techniques have been pursued by the video coding community. However, in this thesis, it is intended to build on the methods described. It has been shown that an exhaustive search of all areas in the image is not necessary and, by introducing blocks of different sizes to represent the range of detail present in an image, the overhead required for both searching and block description is significantly reduced. Using statistical measures, the size of the codebooks needed to describe interframe blocks can be adaptively suited to the image and the level of

temporal activity present. It has been shown that meeting a requirement for low bit-rate coding, is as much a goal of these techniques as a constraint.

Whilst the process of block matching can use the statistical or transform domain characteristics of a pixel block to uniquely identify it from others in the area, blocks still cannot, inherently, describe the nature of features appearing in an image. An edge is merely the transition in contrast between neighbouring groups of pixels and even though a group of blocks, all transversed by an edge, may show statistical trends, there is still no direct relationship between the topography of features and the motion they undergo.

3.5 Model-based coding

Model-based coding is a technique which models the detail of image objects so they may be represented in coded form. Applications are diverse ranging from the interpretation of handwriting and automated facial recognition, to the identification of diagnostic trends in medical image processing. As an example, we could require that the receiver displays a cube undergoing some kind of rotation or translation. Using a model for the description of the cube, the information needed to generate a moving sequence at the receiver would be the size, colour, position and other static properties of the cube, together with a description of its motion. Whilst it is easy to produce computer animated sequences of simple objects undergoing motion, it is not so straightforward for the head and shoulders of one of the parties in a videoconference.

Once again, a measure of image quality must be employed to maintain a minimum acceptable level of detail. An image having many blurred or misplaced artefacts

would demonstrate a direct relationship between the relative observed spatial quality and the calculated signal to noise ratio. It is also possible to continue the approach where a trade off is accepted between the spatial quality of a particular object and its temporal motion.

The application of model-based techniques to low bit-rate image coding is discussed by Welsh [xxxviii], who describes two approaches. The first is purely synthetic, where a sequence of frames is coded and transmitted in such a way as to allow the receiver to effectively produce an object animation. Differences encountered will be tolerated, provided they are small and non-propagating. The second approach [xxxix] employs model-based methods to form a good prediction of successive frames for a moving sequence, with errors between the predicted and actual frames coded using the conventional techniques outlined in chapter 2.

Model-based predictors produce an estimate of true motion superior to block-based approaches, allowing the data rate to be reduced without any significant reduction in picture quality. Since features are more easily classified, the predictive algorithm can concentrate effort on areas where the viewer requires greater spatial resolution, such as the eyes and mouth in a videophone scene.

3.5.1 Model-based image synthesis

A common method of representing two- and three-dimensional objects in computer graphics is by a network of connecting polygons. The model can then be stored in the computer as a list depicting the relevant shape relationships. These arrays take two forms. Firstly, the *x*, *y* and *z* co-ordinates of each polygon vertex in object space *X*[*V*], *Y*[*V*], *Z*[*V*], where *V* is the vertex address. Secondly, a pair of two dimensional arrays is used *LINV*[*L*][*E*] gives the addresses of the vertices at the end of line *L*, where *E* is 0 or 1, depending on which end of the line is involved. *LINL*[*P*][*S*] gives the line address for each side *S* of a polygon *P*. A wire frame image can then be plotted to depict the object by examining all the values of *L* in *LINV*[*L*][*E*] giving the vertex addresses, which in turn supply the *X*[*V*] *Y*[*V*] and *Z*[*V*] data. A small example of this technique is shown in figure 3.5.1.

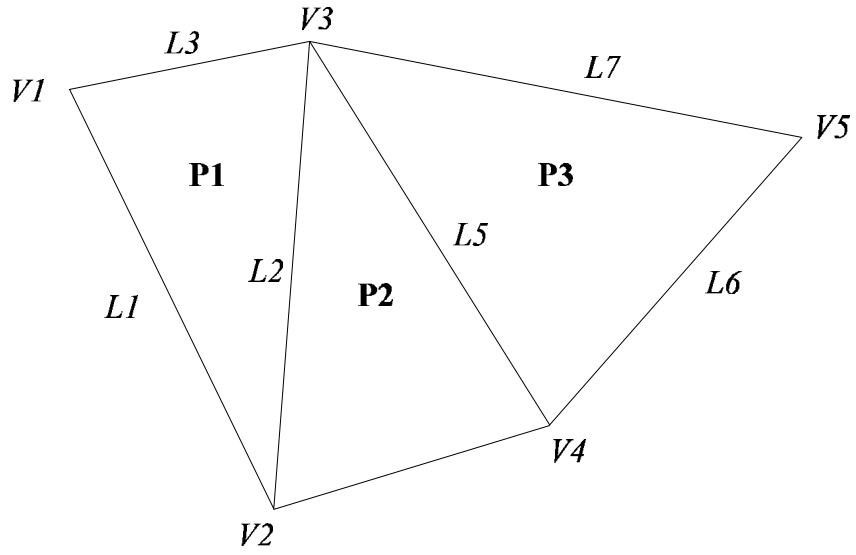


Figure 3.5.1: Section of a simple wire-frame model

In the example shown, the co-ordinates of vertex *V*5 are *X*[5], *Y*[5] and *Z*[5], each indexed by vertex addresses. The addresses of each end of line *L*6 are given by

LINV[6][0] and *LINV*[6][1]. The addresses of each side of polygon *P*3 are given by *LINP*[3][0], *LINP*[3][1] and *LINP*[3][2].

To give the model an impression of depth, shading can be employed to depict contours with respect to light reflected from a given source, as shown in figure 3.5.2.

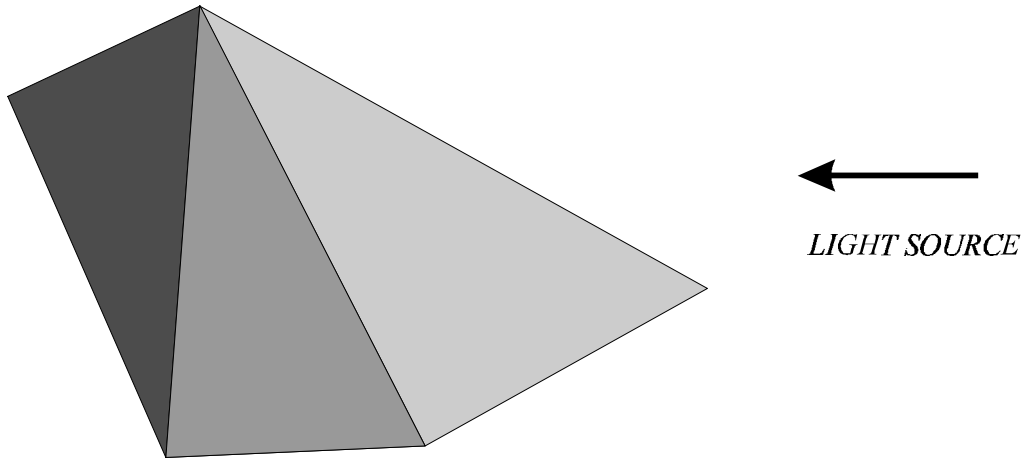


Figure 3.5.2: Shading a polygon net to enhance perceived depth

Further processing can employ shading techniques [xl] which interpolate a smooth transition in luminance between polygons, giving the effect smooth curved surface.

Wire-frame models of faces were constructed by Frederic Parke [xli] to produce facial animations. The face can be made to move in a global way by applying common rotations and translation to the vertices as a whole, with local changes made to emphasise changes in facial expression - a technique that corresponds to the evaluation of facial actions (the facial action coding system, or FACS) [xlii]. Parke took 50 categorised actions and translated them into a set of vectors that could be applied to the vertices.

Parke's work has been taken further by Akimoto, Suenaga and Wallace [xliii], who have developed a generic head model based on the wire-frame principle. They worked on the basis that it is difficult and computationally impractical to generate a full set of vertices, for a given person, on a real-time basis. So, it was proposed that a generic model could be adjusted to reflect the unique facial features of an individual. Using a front and side view, the feature extraction system extracts the base vertices of a new subject. A template matching technique is used to identify primary profile features, such as chin tip, mouth, nose tip and nose bridge. Hence the areas of searching are limited to those of most subjective interest to the coding application. The generic model is then adjusted to take account of these changes, altering the vertices co-ordinates and polygon topography as required. The structure of this base model is shown in figure 3.5.3.

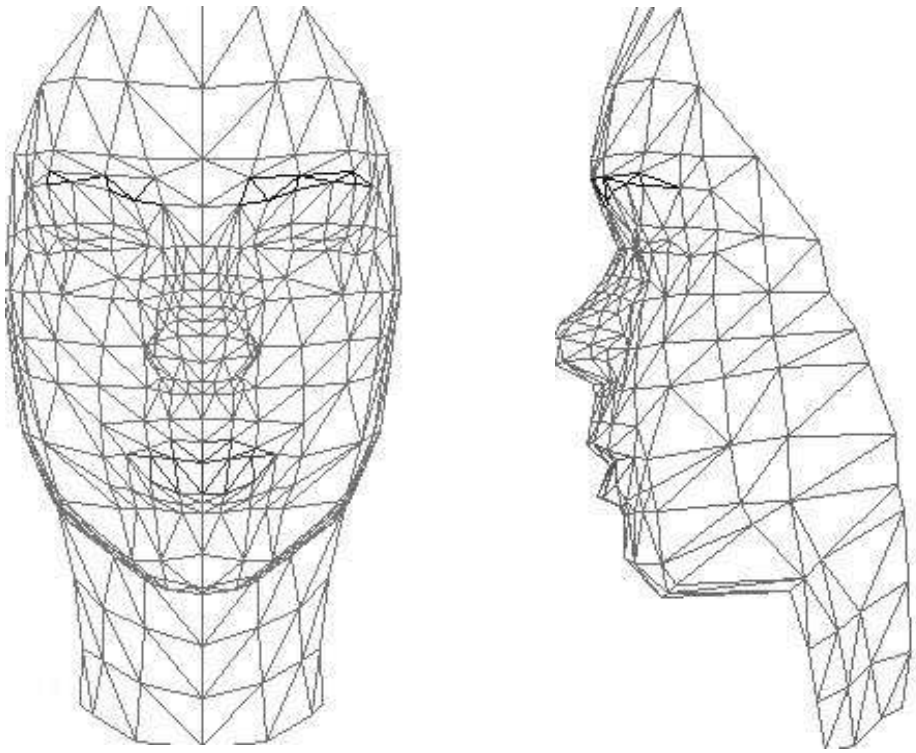


Figure 3.5.3: The structure of a human head base model

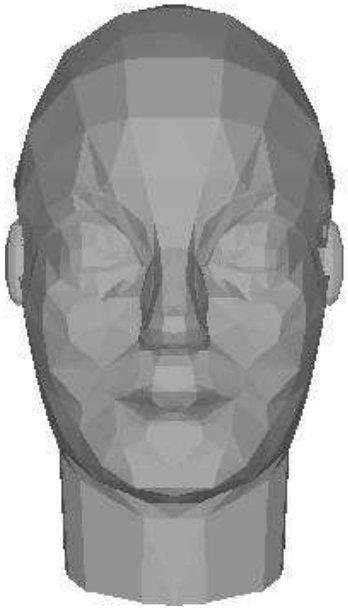


Figure 3.5.4: The application of depth shading

The technique developed by Welsh *et al* uses the synthesis of certain areas of the face to produce a montage of the expressions to be applied to an individual's base model at the decoder. A codebook is developed for each face, with up to 10 different subimages of both the eyes and the mouth, which allows up to 100 facial expressions to be modelled. To improve the spatial effect, the edges of the subimage can be contoured against the vertices of the head model to give the effect of smoothness, even though changes and perspective and orientation may, in some part, have occurred.

The primary difficulty of model-based synthesis lies in the evaluation of the face when generating the codebooks. Locating the eyes and mouth, especially from an undulating head and shoulders image of low contrast, is not a simple proposition. Furthermore, the selection of appropriate templates to generate the expressions is a difficult task. It is perhaps analogous to quantisation - where we try to describe a feature having a large range of possibilities with a small set of values. Interpolation between actual and montage vertices produces a spatio-temporal impression of smoothness, although this is practically the introduction of errors. For cases of more significant change, the tracking of individual vertices may be difficult to achieve, where several vertices may make up a feature such as the mouth.

The wireframe structure shown in figure 3.5.3 is unlikely to be adopted for a real-time coding system, simply because it would be difficult to model the face against all the vertices provided. The model can be represented with less points in a three-dimensional space, effectively producing an image of lower spatial resolution. The

emphasis of coding can then be applied to the eyes and mouth as areas having most spatial significance to the viewer.

Practical implementations of the synthesis technique employ binary extraction to locate the important facial features. Using a simple edge enhancement algorithm, such as a Sobel filter, the transition zones between areas of contrast can form the basis of a simple search process. Nagao [xliv] used a sample of 700 faces and claimed a feature extraction success of 90%. The Welsh model is 50% efficient, however this work has been carried out with an application to real-time coding. The sourcing of eye and mouth features requires a significant computational search overhead and the transmission of codebooks to the receiver has a high initial data transfer requirement.

Whilst this method of model based coding examines more closely the spatial detail of an image, it is still concerned only with the *boundaries* between areas of detail in making a prediction of motion. A more intuitive algorithm would not only look at the edges, but also consider the content of subregions.

3.5.2 Model-based coding - summary

This section has described work which provides coding parameters more directly associated with the physical nature of features in an image. However, there is some degree of compromise required in the extent model notation. Model-based coding has a foundation in computer graphics and static image analysis, where real-time constraints are not an issue. However, the application to interactive video coding indicates that models must be of lower resolution, and in the case of model-based

synthesis, codebooks must be a fixed set to reflect a range of likely changes in feature orientation, whilst limited by the processing capability of the system.

There is a strong argument that feature based coding techniques along these lines are the way forward in video coding and whilst model-based coding, in general, is not suited to a realistic implementation as a video codec algorithm, it does provide us with some important pointers for future development.

Chapter 4

Spatial Processing

Methods of low-resolution image representation

4.1 Introduction

It has already been shown that a mechanism which encodes an image or video sequence in terms of its constituent features is closely linked to the human psychovisual perception mechanism. The fundamental issue explored in this chapter is the extent to which image quality can be compromised to yield a more efficient data representation for subsequent coding.

The approach to this is made in several ways. Initially, the expectations of human perception will be considered - more simply, what can we "get away with" ? Essentially, it will be shown that there is a useful trade-off between the amount of information needed to represent a picture and the quality the perception mechanism can tolerate. Methods of data reduction can be very effective, providing a lower resolution image. Inevitably an image of inferior quality will result, but it may be the case that the image, when viewed as part of a video sequence, will be satisfactory. It will then be suggested that pre-processing can be employed to reduce the quantity of subregions required to represent an image, such that low bit-rate coding can be used to transmit a sequence of satisfactory quality to a decoder.

This chapter will also introduce some of the benchmarks used to quantify image quality - contrast sensitivity and the signal-to-noise ratio of single images and video sequences. Ideally, it would be desirable not to have to reduce image quality at all.

However, the scope of this thesis is the demonstration of a novel mechanism to perform classified subregion motion estimation and as such, it must be expected that compromises will be made between quality and codec efficiency.

4.2 Spatio-temporal perception

Chapter 3 dealt extensively with the concept of object displacement and its relationship to the spatio-temporal frequency spectrum. In a more qualitative way, we can assess the application of this relationship on contemporary standards of video production. Throughout Europe and Australasia, video is supplied with a spatial resolution of 625 lines of pixels per frame, operating at 25 frames per second, whereas in the Americas and the Far East, the 525 lines, 30 frames per second, system is used. In each system, there is a balance between spatial and temporal quality, with the 625/25 system offering greater spatial resolution, at a slightly slower framerate.

Relating the constraints of these standards back to the evaluation of spatio-temporal perception, we know that the video has to be displayed at a frame rate fast enough to prevent visible flicker and temporal aliasing, and at a spatial resolution so as to provide sufficient information to allow the recognition of features.

The spatial resolutions specified by the ITU-T for videoconferencing codecs under recommendations H.261 and H.263 are shown in appendix 1. Three distinct image formats are shown, based on the primary Common Intermediate Format (CIF), with a spatial resolution of 352×288 pixels. Nominally, temporal resolution is sampled at 25 frames per second, however most practical H.261 codecs employ a degree of variable frame rates, as a further trade-off in the provision of better spatial quality at

times of significant interframe motion. The effect of these spatial resolutions is that if the QCIF image were enlarged to the same actual size as a CIF image, it would have one quarter of the resolution. However, that is not to say that the psychovisual perception mechanism will perceive a similar qualitative reduction in detail. Extensive work evaluating the human perception of spatial quality has been carried out by Watson [xlv], who not only assesses the effects of a change in spatial resolution, but also considers the characteristics relating to different types of spatial quantisation.

Work by some psychologists, particularly Triesmann [xlvi], has suggested that in general, the psychovisual perception system tends to take more interest in the spatial qualities of objects undergoing motion. This infers that the amount of detail in the surroundings and background are not an issue.

4.3 Spatial pre-processing

If we are to relate these observations to the type of information held in a videoconferencing scene, they would be most useful as the basis of a pre-processing mechanism. For example, would it be practicable to introduce a resolution-reduction technique and spatial quantisation to deliberate attenuate the quality of an image? There would naturally be an effect on the quality observed by the viewer, but there would also be a significant reduction in the amount of data required to represent the image as part of a video sequence. To further explore this idea, consider the three CCIR standard images shown below.

Figure 4.3.1: Frame 001 from CCIR base sequence Susie

Figure 4.3.2: Frame 001 from CCIR base sequence Miss America

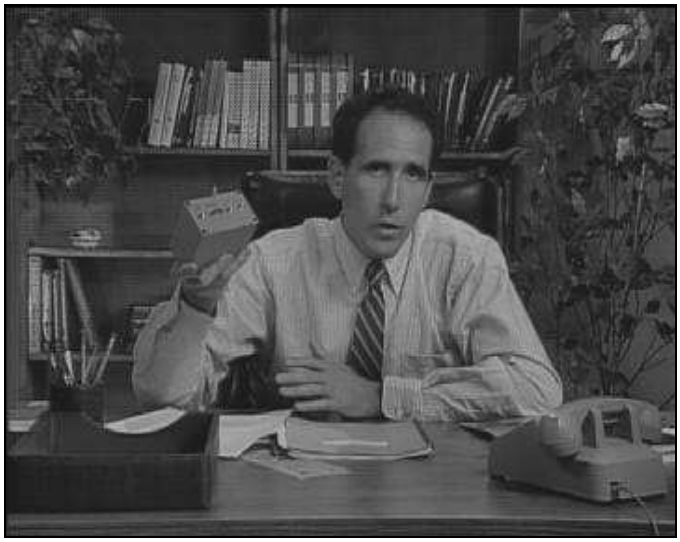


Figure 4.3.3: Frame 001 from CCIR base sequence Salesman

Looking at each of these images, we see they represent a range of spatial detail likely to occur in a *fixed-background* videoconferencing situation. This distinction is important, since other image sequences contain elements of pan, where the background changes as the camera tracks a foreground subject. For the purposes of this work, we shall assume that the camera is static in common with most videoconferencing codecs. The frames from *Susie* and *Miss America* show a foreground subject of principally facial detail, set against a fairly plain background. However, the *Salesman* image is more complex, having a background of much greater detail. If we consider the postulates of spatial perception already mentioned, we should still be able to reduce the resolution of the images and gain a data compression advantage, with only a small cost in image quality.

4.3.1 Reduction in spatial resolution

To test this idea, each of the images was reduced from a full CIF original to a QCIF equivalent, using two different techniques.

4.3.2 Image sub-sampling

Image subsampling is a simple method of reducing image resolution without the need for any processing of the pixel intensity value. An image is reduced in size by a scale factor *s*, simply by skipping to every *s*'th pixel. The relationship between the area of the resulting lower resolution image and the original is proportionate the square of *s*, hence:

$$[M(R') \times N(R')] = \frac{1}{s^2} [\, M(R) \times N(R) \,] \qquad \text{[Equation 4.1]}$$

where $M \times N$ is the image area (width $\times$ height)

and R and R' are the original and resultant images, respectively.

In the case of a CIF original image, the area will be $352 \times 288 = 101376$ pixels. The application of a scale factor of 2 will be a QCIF image of $176 \times 144 = 25344$ pixels, one quarter the original resolution. Although used in many primitive systems, arbitrary sub-sampling produces an unsatisfactory representation of the original image, simply because so much data (in this case, 75%) is discarded. This effect is shown in figure 4.3.4.

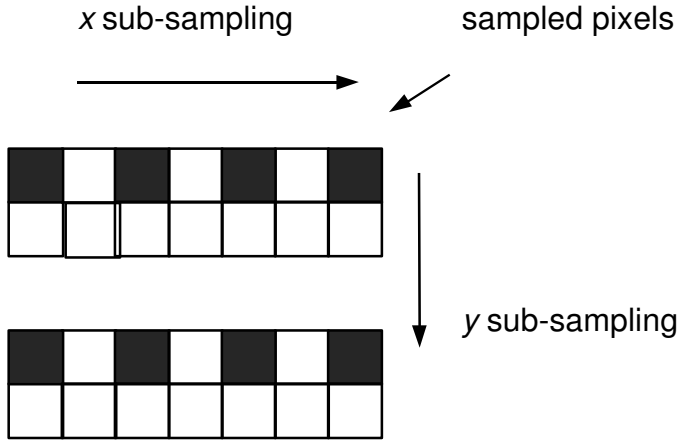


Figure 4.3.4: Image sub-sampling

Simple sub-sampling was applied to each of the test image frames, using a scale factor, *s*, of two to produce QCIF resolution. The resultant images were scaled to CIF equivalent size, such that each group of four pixels represents one single sub-sampled pixel from the resulting QCIF image. This allows for the visual comparison of the images.

Figure 4.3.5: Susie 001 - QCIF result of sub-sampling

Figure 4.3.6: Miss America 001 - QCIF result of sub-sampling

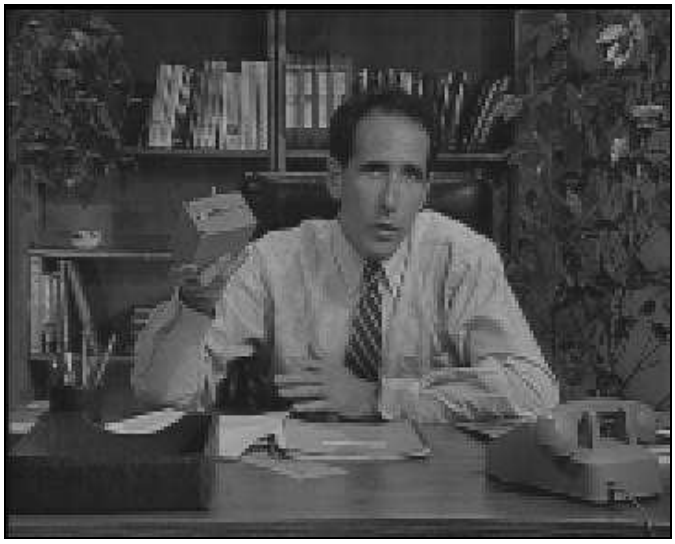


Figure 4.3.7: Salesman 001 - QCIF result of sub-sampling

The effect of this process has a greater influence on picture quality in areas of detail and high-frequency transitions in luminance. For example, the edge of the telephone handset in *Susie* has a stepped profile, as does the shirt sleeve in *Salesman*. This noticeable degradation must be addressed, as it will inevitably cause error propagation in subsequent processing.

For a more quantitative appreciation of image quality, we can compare the resultant images to their originals using the *mean square signal-to-noise ratio* [54], which for images is given by:

$$snr_{\text{ms}}\,(\text{dB}) = 10\log_{10}\frac{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} f(x,y)^2}{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}\left[\hat{f}(x,y) - f(x,y)\right]^2} \qquad \text{[Equation 4.2]}$$

where m and n are the image width and height

and $f(x,y), \hat{f}(x,y)$ are the original and resulting image pixel values.

Image	snr_{ms} (dB)
Susie 001	40.2101
Miss America 001	43.6039
Salesman 001	42.2735

Figure 4.3.8: Sub-sampling signal-to-noise ratio

4.3.3 Mode-value sampling

In spite of its obvious limitations, the sub-sampling technique, illustrated in figure 4.3.4, is quite satisfactory for images having large areas of consistent pixel intensity, but in areas of detail, extracting one pixel in every four may reduce the overall representation of trends in the image. To help deal with this, another technique was developed to sample the original CIF image and extract one value for each four that represented the four pixels as a group.

Statistical sampling is a technique used predominantly in spatial filtering, where individual pixel errors can be compensated by considering the surrounding pixel values. However, for the purposes of this exercise we can consider the possibility of extracting the mean, the median or the mode value of each $s \times s$ pixel block, where s is the scale factor. The mean value would represent most accurately the trend of pixel intensity in a block, however it would inevitably result in the introduction of many new values not present in the original image. As images are normally represented by integers only, the rounding up of a floating-point value would introduce errors to the

low-resolution image. Given that the median is simply the middle value and does not represent the spread of intensities, it was decided to select the mode as a sample parameter for the reconstructed picture. So for an $s \times s$ sample block, the mode is the most frequently-occurring value.

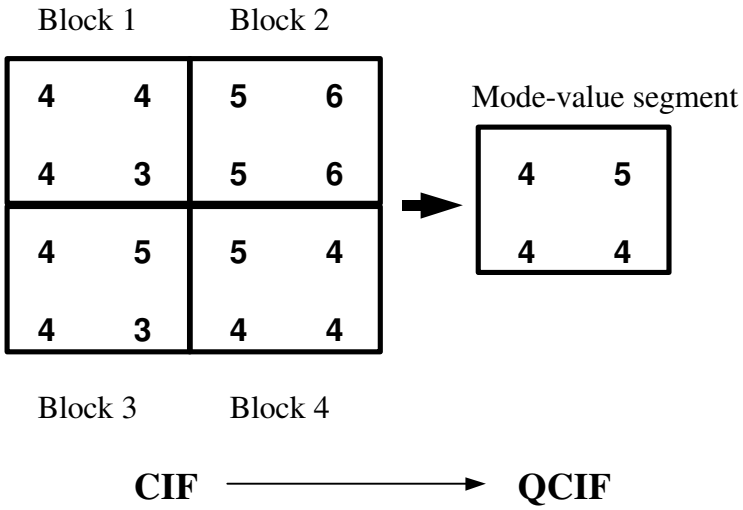


Figure 4.3.9: Extraction of mode value samples, s = 2

For the usual case where the scale factor, *s*, is an even integer, there will be occasions where the mode value could be either of *s* values, as is the case in figure 4.3.9, Block 2, where the scale factor is 2. The algorithm deals with this situation by selecting the lower of the two values. In areas of consistent pixel intensity, the mode value is normally representative of all the pixels, whereas in areas of transition, the mode value represents the dominant pixel value for that block and the QCIF image represents local trends more efficiently than may be the case for simple sub-sampling.

The mode sampling algorithm was applied to each of the three test images and the results are shown below. Once again, the images have been re-scaled for comparison with the original CIF images. Hence, each block visible is simply four picture elements representing the one sample made for the resultant QCIF image.

Figure 4.3.10: Susie 001 - QCIF result of mode sampling

Figure 4.3.11: Miss America 001 - QCIF result of mode sampling

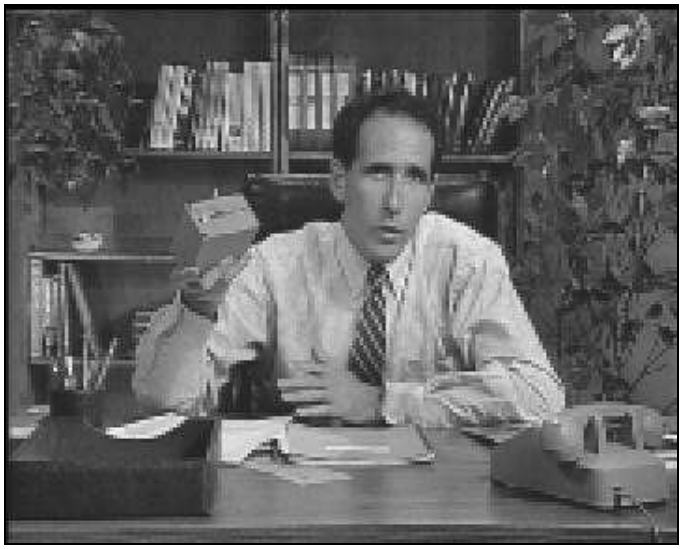


Figure 4.3.12: Salesman 001 - QCIF result of mode sampling

The evaluation of signal-to-noise ratio for each of these images, compared with the original CIF image was made and the results are shown in figure 4.3.13.

Image	**snr_{ms} (dB)**
Susie 001	56.7098
Miss America 001	61.4962
Salesman 001	59.6213

Figure 4.3.13: Mode sampling signal-to-noise ratio

Once again, a visual inspection of each of the resultant images shows a coarse rendition of changes in contrast. However, there is a significant improvement in the signal-to-noise ratio, based upon the original images. This suggests that the selection of mode value as a sample parameter is more effective than the previous method of image sub-sampling. The interpolation of the QCIF result back to a CIF format for this comparison has produced an image more closely resembling the original. If a

scheme is to be selected for image sampling, it is clear that mode value extraction is preferable.

4.4 Spatial quantisation

The amount of data required to represent an image can be further reduced by the introduction of quantisation, which is a pixel-level approach to achieving a lower resolution image. Quantisation is used throughout signal and image processing to reduce a large set of values to one much smaller for the purposes of coding. It has already been shown in chapter 2 that the introduction of a quantiser to the DPCM/DCT loop in a predictive video codec reduces the range of DCT coefficients and makes coding more efficient.

Having achieved a reduction in image resolution by the use of mode sampling, the feasibility of further pixel quantisation was investigated. Figure 4.4.1 shows a histogram of the pixel luminance values in the base image *Susie001*. The data shown is for the value of pixel luminance - this is of most significance in video processing as the psychovisual perception system is more sensitive to changes in brightness than to changes in colour.

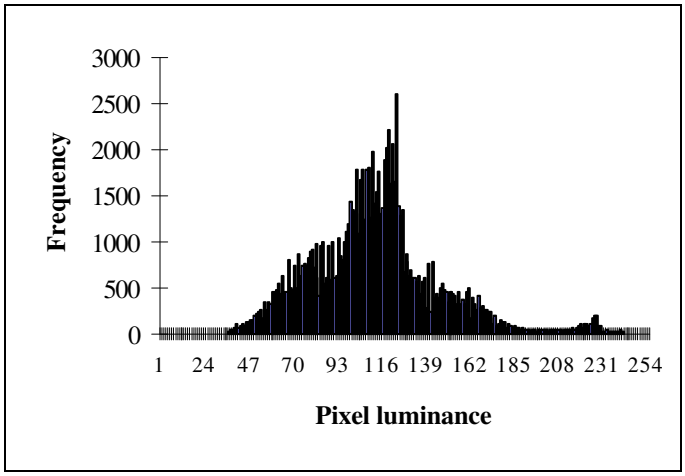


Figure 4.4.1: Luminance histogram for the base image Susie001

The representation of the luminance for any CIF image, without quantisation will be a fixed value of 352×288 pixels $\times$ 8 bits, if there are 256 grey levels. This overhead of 811008 bits per frame can be reduced by quantisation. By halving the total number of steps of quantisation employed, the datagram is reduced by one bit.

An algorithm for linear quantisation was developed, allowing the user to specify any even quantisation step interval, ΔQ. If an input value lies within the range η_n to $\eta_n + \frac{\Delta Q}{2}$, or $\eta_n - \frac{\Delta Q}{2}$ then it is assigned to the quantised luminance value η_n (figure 4.4.2).

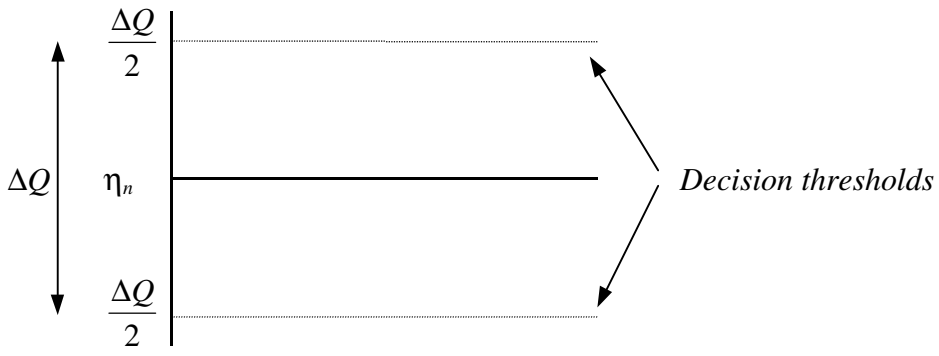


Figure 4.4.2: Assignment of quantisation values

Quantisation was performed on each of the three base images, using steps of ΔQ=8 and ΔQ=16 luminance levels. As figures 4.4.3. and 4.4.4 show, there is a considerable reduction in the range of luminance values for *Susie001*, without a significant change in spatial quality. Figures 4.4.5 and 4.4.6 show the application of the same quantiser to the *Salesman* sequence. Notice that the overall contrast of adjoining objects has improved, as quantisation has had the effect of increasing some spatial frequencies.

Figure 4.4.3: Susie001 - CIF image with quantisation ΔQ=8

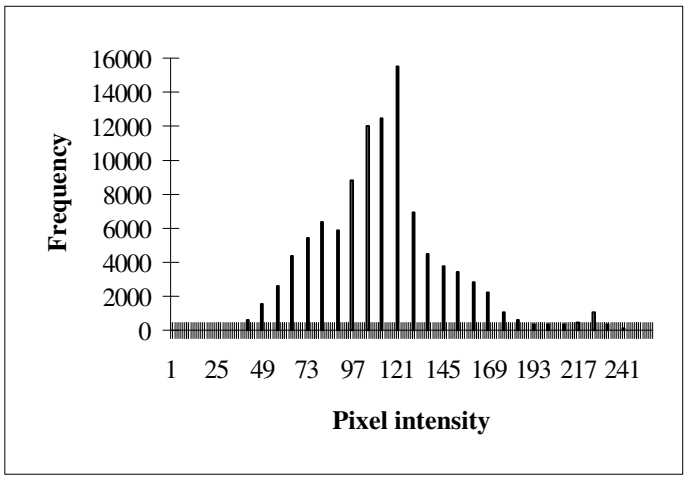


Figure 4.4.4: Luminance histogram for base image Susie001, ΔQ=8

Once again, there are now only a small range of values required to represent the image. Overall, *Salesman* is notably darker than *Susie* and so the most frequently occurring values are at the lower intensities (figure 4.4.6).

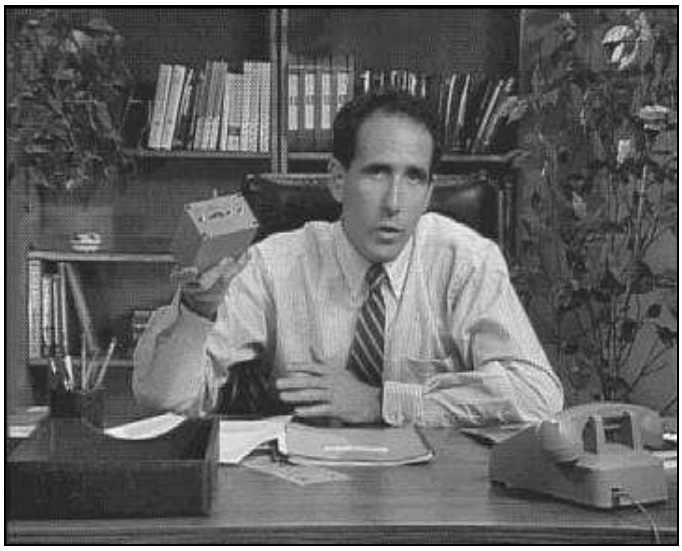


Figure 4.4.5: Salesman001 - CIF image with quantisation ΔQ=8

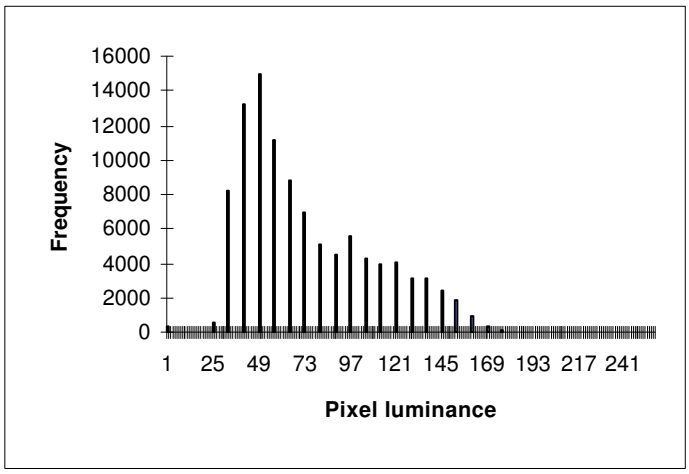


Figure 4.4.6: Luminance histogram for base image Salesman001, ΔQ=8

In both cases, the effect of a quantisation step interval of 8 luminance levels is to reduce the range of values from 0-255, single step, to 0-255, eight steps. To represent 255 levels, eight bits are required, whereas for the quantised image the maximum number of values is 32, which would be coded using five bits. In general, the number of bits required, $N_{B,}$ following quantisation is:

$$N_B = \log_2 \frac{\eta_{max}}{\Delta Q} - 1 \qquad \text{[Equation 4.3]}$$

where η_{max} is the maximum number of grey levels

and ΔQ is the quantisation step interval.

For a full CIF image, the number of bits required to represent an image is:

$$352 \times 288 \times 8 = 811008.$$

However, for a full CIF image processed with a quantisation step interval, ΔQ=8, the number of bits required is:

$$352 \times 288 \times 5 = 506880$$

which is 62.5% of the original.

Having considered the bitrate, the signal-to-noise ratio of the quantised image provides a quantitative measure of image quality. The linear quantiser was applied to each of the test images for ΔQ=8 and ΔQ=16 (figure 4.4.7).

Image	**snr_{ms}(dB) ΔQ=8**	**snr_{ms} (dB) ΔQ=16**
Susie 001	41.6001	39.3813
Miss America 001	39.7326	37.6130
Salesman 001	40.0457	38.2237

Figure 4.4.7: The effect of quantisation on signal-to-noise ratios

In the case of ΔQ=16, there will be a reduction in bits required per pixel to four, reducing the data overhead to half that required for the original representation. However, images resulting from this level of quantisation were found to be of poor spatial quality and it was decided not to pursue any level of less than 5 bits (ΔQ=8).

4.5 Hybrid pre-processing

Having evaluated the separate effects of image spatial sampling and pixel quantisation, the processes can be combined to produce a low-resolution image which can be represented by a much smaller data set. In other words, we can both reduce the number of pixels and the number of values they can take. The nature of the algorithm is shown in figure 4.5.1.

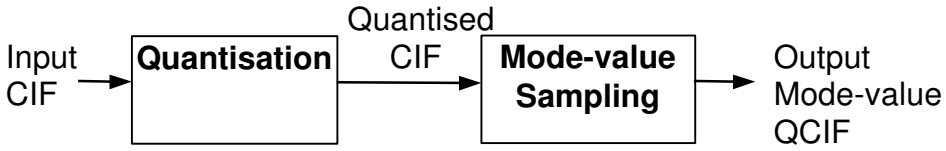


Figure 4.5.1: A hybrid spatial pre-processing algorithm

It was noted that the most efficient method of combining the two processes was to have quantisation before sampling. Whilst this adds slightly to the process load (the quantisation is performed on all the pixels in a CIF image), the effect on signal-to-noise ratio of reversing the process is that more errors result as the mode-values selected may represent a much larger variance in luminance than was originally the case. Each of the test sequences was processed and the resulting bitrates and signal-to-noise ratios evaluated (figures 4.5.2 to 4.5.5).

Figure 4.5.2: Susie frame 001 after hybrid pre-processing, $s=2$, $\Delta Q=8$

Figure 4.5.3: Miss America frame 001 after hybrid pre-processing, s=2, ΔQ=8

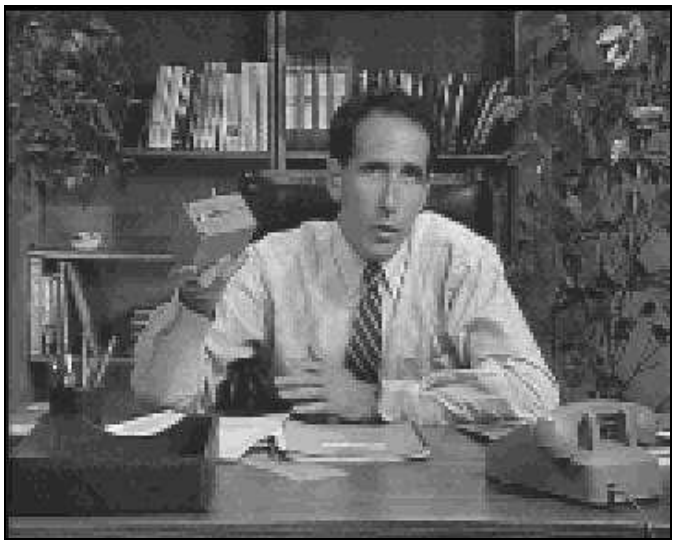


Figure 4.5.4: Salesman frame 001 after hybrid pre-processing, s=2, ΔQ=8

Image	snr_{ms}(dB) s=2, ΔQ=8
Susie 001	40.8508
Miss America 001	39.1163
Salesman 001	39.5421

Figure 4.5.5: Hybrid pre-processing single frame signal-to-noise ratios

The effect of hybrid pre-processing on the range of pixel values produces histograms of the same overall shape as before, however with lower frequencies, due to the reduced resolution in QCIF format (figures 4.5.6 and 4.5.7).

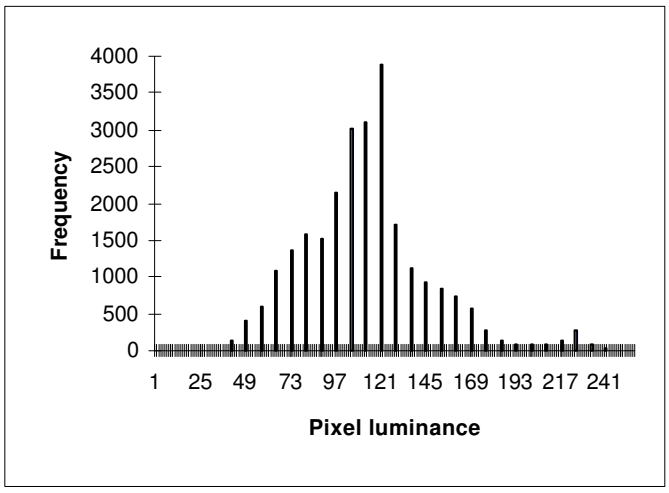


Figure 4.5.6: Luminance histogram for base image Susie001 after hybrid pre-processing, s=2, ΔQ=8

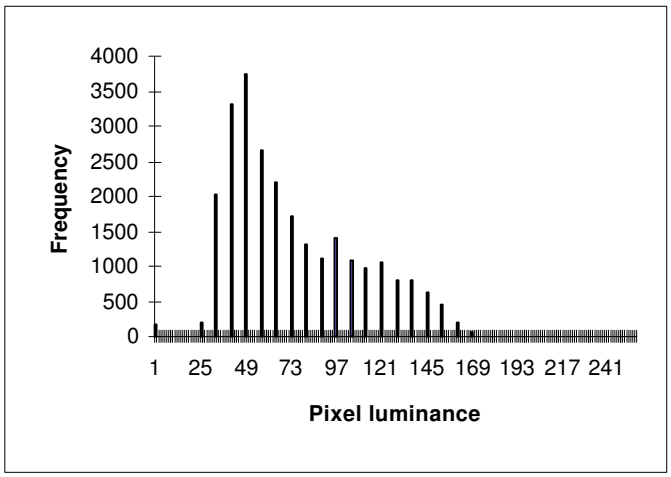


Figure 4.5.6: Luminance histogram for base image Salesman001 after hybrid pre-processing, s=2, ΔQ=8

Having retained the overall pixel distribution, albeit at a fraction of the original image magnitudes, it is clear that a significant reduction in bitrate will be possible, whilst the image quality has been lowered to an acceptable level. The use of a quantisation interval of 8 levels (5-bits), was adopted for all remaining work, as it was considered the best trade-off between bitrate reduction and the maintenance of acceptable quality.

To consider the associated bitrate implications, we can evaluate the effects of applying both quantisation and mode-value sampling to the single frame images:

Quantisation

If ΔQ is set to eight grey levels, the effect of quantisation will be:

$$N_B = \log_2 \frac{256}{8} - 1 = 5 \text{ bits per value}$$

Resolution

If CIF is mode-value sampled to QCIF, there will be a 25% reduction in data content, hence:

$$176 \times 144 \;=\; 25344 \text{ pixels per frame}$$

Bitrate reduction

To describe the original, CIF image, with 256 luminance levels, the required bitrate for each frame is:

$$352 \times 288 \times 8 \;=\; 811008 \text{ bits per frame}$$

compared with

$$176 \times 144 \times 5 \;=\; 126720 \text{ bits per frame}$$

a reduction to 15.6% of the original data overhead.

4.6 Spatial Processing - Summary

This chapter has described an effective method of picture pre-processing, which realises the efficient reduction of data content. It has been shown that an image of lower resolution can quite adequately represent the range of spatial information required for low bit-rate transmission, using a novel application of mode-value sampling. This ensures that the resultant image more closely reflects trends in pixels values that may not be accurately represented by simple two-dimensional sub-sampling.

It has also been shown that quantisation can be employed to reduce the range of values a pixel can take. The histograms demonstrate that this method of sampling maintains the characteristic pixel frequency curve shape, albeit with many less levels of luminance.

The combined effect of these techniques is an algorithm which can easily be applied to images, having no intraframe recursion and working at real-time requirements. It will subsequently be demonstrated that the pre-processing technique yields further benefits in terms of designating the contrast levels between adjoining groups of similar value pixels.

More fundamentally, the application of spatial processing is a method of conditioning the resolution of images, such that control exists as to the extent of reduction in image quality outside the machinery of interframe coding. In many video codecs, image quality is determined by the constraints imposed by the compression algorithm and the bitrate requirements. It is therefore a useful departure from conventional thinking to deliberately manage the resolution of images before compression and coding takes place.

Chapter 5

Feature Classification

The detection and identification of image subregions

5.1 Introduction

This chapter introduces the infrastructure of a novel method of image classification, representing groups of pixels by the shape they take. Chapter 3 examined closely the range of techniques currently employed for motion compensated video coding, ranging from regular-shape blocks, image decomposition and segmentation and model-based coding. Each of these techniques, whilst adequate for the purposes of motion compensation, is rather cumbersome, tends not to directly represent spatial features and may not uniquely represent any particular group of pixels, making a good prediction of motion difficult to achieve.

The iterative binary and quadtree segmentation approaches are, however, a step in the right direction. The fundamental idea explored here is whether these processes can be reversed, such that a subregion is formed from the pixel level *up*, rather than from the whole image *down*. Using a method of unique value clustering, it will be shown that an algorithm of low computational complexity can be developed. It will also be demonstrated that the methods of spatial processing described in chapter 4 are of particular assistance in regulating the quantity of features to be detected and classified.

Having extracted groups of pixel values from a given image, a simple method of topological classification is demonstrated. In subsequent chapters, it will be seen that

the efficient generation of motion vectors will depend almost totally on the success of feature classification.

For the purposes of nomenclature, a whole image frame *R* is composed of many subregions, or features, which for the purposes of this work are simply groups of similarly valued picture elements, conforming to some basic predicates.

5.2 Pixel Clustering

5.2.1 Seed Pixels

Pixel clustering is a simple method of region-growing and is familiar to users of computer graphics applications. One use of this technique is the extraction of non-uniform areas of interest by shading an area of pixels, which conforms to given parameters, with a high intensity colour. Often, the user randomly selects a seed pixel as the start location and the process 'grows' the sub-region around the seed.

To extract sub-regions for feature classification, a refinement to this approach is required. Firstly, the selection of a seed pixel must be made in accordance with simple criteria, common to all sub-regions and so the method selected uses the raster-scan principle to pass all possible seed pixel locations. Consider figure 5.2.1.

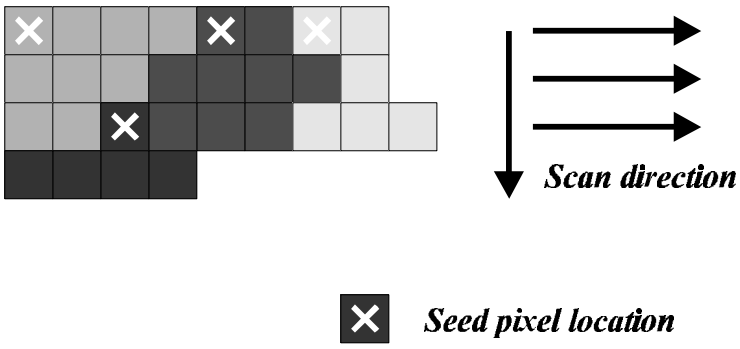


Figure 5.2.1: Scanning for the location of seed pixels

The criteria used is simple and non-recursive. The seed pixel is the first pixel to be encountered in the scan, which is different from its immediately scanned predecessor and which does not belong to a sub-region associated with a previously marked seed pixel. In the example shown, for many sub-regions, the seed pixel tends to be located at the top left-hand corner, however in the lower sub-region, the seed pixel is a single point above the main body of it's related group.

The process of clustering is known as pixel aggregation. Starting with a series of seed locations, neighbouring pixels are appended to a set, provided they meet certain spatial predicates and have a similar value to the seed pixel. The practical implementation of this is a linked-list, which is a set of values and *(x, y)* co-ordinates identifying related pixels and their luminance values.

For a given image, R, we may consider segmentation as a process which partitions R into a set of n adjoining sub-regions, hence:

$$\bigcup_{i=1}^{n} R_i = R \qquad \text{[Equation 5.1]}$$

where R_i is a connected sub-region, $i = 1, 2, ..., n$

and $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$

where $\varnothing$ is the null set, demonstrating that adjacent subregions are disjoint.

For all the constituent pixels of R, we can define a subregion using a predicate $P(R_i)$ = TRUE for $i = 1, 2, ..., n$ which shows that the pixel values (in this case) will be the same and predicate $P(R_i \cup R_j)$ = FALSE for $i \neq j$, indicating that pixels in adjacent regions are different.

5.2.2 Pixel Relationships

Having specified the nature of sub-region uniqueness, we can consider the aggregation process for the formation of image sub-regions R_1 to R_n. This can take two forms. A pixel **p** at co-ordinates (x, y) has four horizontal and vertical neighbours, whose co-ordinates are given by:

$(x + 1, y)$ $(x - 1, y)$ $(x, y + 1)$ $(x, y - 1)$.

This set, called the four-neighbours of **p**, is denoted by $N_4(\mathbf{p})$. Each pixel is a unit distance from (x, y) and some of the neighbours of **p** will lie outside the image area if (x, y) is on the border.

The four diagonal neighbours of **p** reside at:

$$(x + 1, y + 1)\ (x + 1, y - 1)\ (x - 1, y + 1)\ (x - 1, y - 1)$$

and are denoted by $\boldsymbol{N}_D(\mathbf{p})$.

These points, together with the four-neighbours are called the eight-neighbours of **p**, $\boldsymbol{N}_8(\mathbf{p})$. As before, some of the points of $\boldsymbol{N}_D(\mathbf{p})$ and $\boldsymbol{N}_8(\mathbf{p})$ will lie outside the image boundary.

5.2.3 Pixel Connectivity

An important concept in establishing boundaries of sub-regions is whether adjoining pixels are neighbours in terms of satisfying the set criteria, as well as having some kind of luminance similarity. It is not necessarily the case that pixels are wholly related, simply because they belong to the four-neighbour set.

If a further parameter is set, a sub-region can be formed against both spatial and value measures. For example, we could select a sub-region as comprising pixels having luminance values in the range η=32 to η=64, forming a value set:

$$V = \{32, 33, ..., 64\}.$$

Four-connectivity would apply if two pixels are in the set $\boldsymbol{N}_4(\mathbf{p})$ and have values from V. Similarly, eight connectivity would be satisfied if two pixels are in the set $\boldsymbol{N}_8(\mathbf{p})$ and have values from V (figure 5.2.2).

$\mathbf{q} \in \{N_4(\mathbf{p})\}, \mathbf{q} \in \{V\}$ $\mathbf{v} \in \{N_8(\mathbf{p})\}, \mathbf{v} \in \{V\}$

Four-connectivity Eight-connectivity

Figure 5.2.2: Pixel connectivity relationships

A refinement of these relationships is referred to as *m-connectivity*. Generally speaking, two pixels **p** and **q** with values from V are *m*-connected if :

$\mathbf{q} \in N_4(\mathbf{p})$

or $\mathbf{q} \in N_D(\mathbf{p})$ and $N_4(\mathbf{p}) \cap N_4(\mathbf{q}) = \varnothing$

This is a useful extension of the set relationship as it avoids multiple path connections used when eight-connectivity is employed. Consider figure 5.2.3.

Eight-neighbours *m*-neighbours

Figure 5.2.3: Modification of pixel connectivity

Notice that the multiple connection to the top right value, which was double connected under the set $N_8(\mathbf{p})$ is now connected only horizontally and vertically. It

has a relationship to the centre pixel, but only via the value in the middle of the top row.

The application of these predicates to a sub-region clustering algorithm is quite straightforward. We select a set V, comprising the range of values permissible and then specify the type of connectivity required. It was considered most appropriate to have only orthogonal connectivity, as this allowed a simple perimeter description, described later. Using the idea of *m*-connectivity, diagonal relationships are not allowed, unless the path can be made via a mutually adjacent pixel (figure 5.2.4).

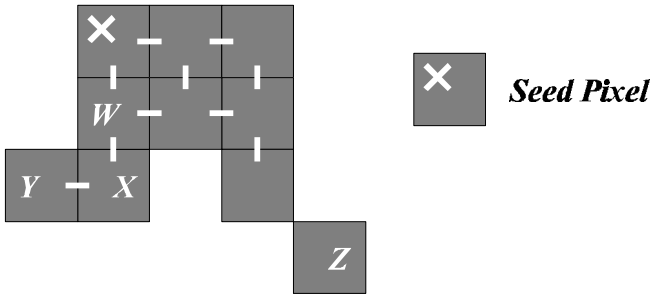


Figure 5.2.4: Sub-region connectivity

In the group, pixel *Y* is not connected to pixel *W* in the set $\boldsymbol{N}_4(W)$, however, it is connected via the set $\boldsymbol{N}_4(X)$. Pixel *Z*, on the other hand, is not four-neighbour connected to any pixels in the group and so would not be considered as part of this subregion.

5.3 Image Segmentation

Chapter four showed how the introduction of spatial pre-processing significantly reduced the overhead in data required to represent an image. One useful side effect of this process is that it makes image segmentation much easier. Practically, we find that adjoining pixels tend to vary by only small luminance values, unless there is an obvious boundary between the objects in an image. If quantisation is introduced, these small fluctuations are removed and the result is a larger region comprising pixels of the same value. The contrast between such regions is improved to a minimum equal to the quantisation step interval, ΔQ. If an image, R, is composed of up to n subregions, the value of n will be reduced as ΔQ is increased.

In order to test this effect, quantisation was applied to the image sequence *Miss America*, with ΔQ set to a minimum of eight grey levels and then increased to 16 and 32 grey levels. A sample frame of eight-level quantisation is shown in figure 5.3.1.

Figure 5.3.1: *Miss America* QCIF resolution with ΔQ=8

Sub-regions were segmented on the basis of the *m*-connectivity approach, but instead of using a range of values in a set V, a single quantised value was used for that locality. The total quantity of segmented sub-regions was counted and the results are shown in figure 5.3.2. It is clear that quantisation directly affects the quantity of sub-regions in a given image.

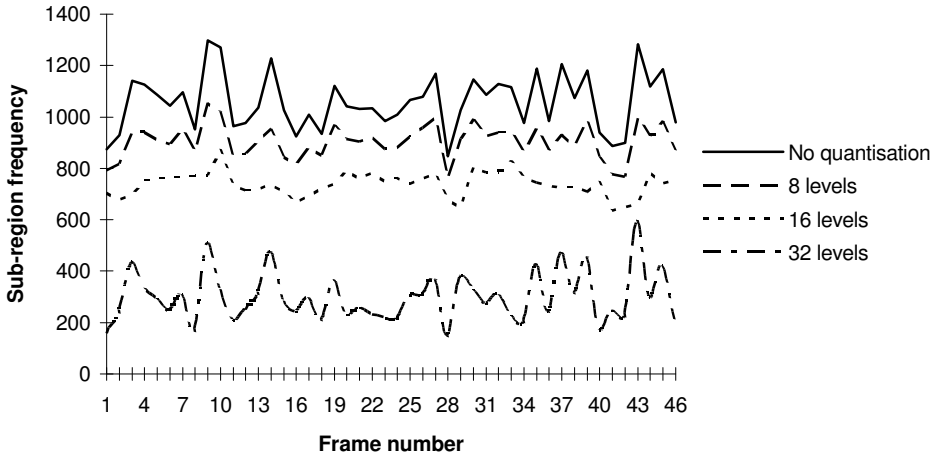


Figure 5.3.2: The effect of quantisation on sub-region frequency (Miss America)

5.4 Sub-region boundary tracking

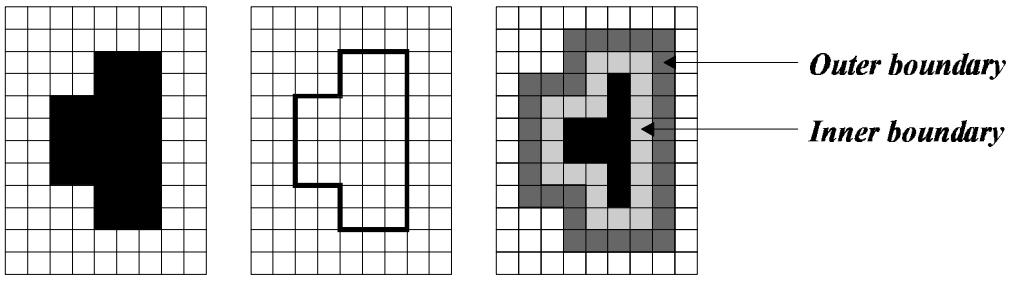
The process of orthogonal pixel connectivity allows for several approaches to the unique classification of a particular type of feature, together with its location in the image space, *R*.

There are many approaches to the description of sub-region topography for the purposes of digital image processing. We can consider the perimeter of a shape as being a description related to the pixels bounded to it, the pixels outside it, or the

"cracks" between such pixels. Rosenfeld [xlvii] introduced the concept of adjacency graphs, where a digital image can be represented as a graph whose vertices are the pixels. An edge of the graph corresponds to each pair of adjacent pixels and connected subsets containing similar pixels formed consistent subgraphs. A neighbourhood was defined as the boundaries found in an image and was introduced as a set of adjacent graph vertices.

In defining what constitutes a sub-region boundary, Rosenfeld [xlviii] went on to define a crack as a finite element separating two pixels which, for the purposes of this analysis, are considered as squares. To generate the boundary of a sub-region, we require that for any two adjacent sets, R_i and R_j in an image region R, $R_i \cap R_j = \varnothing$.

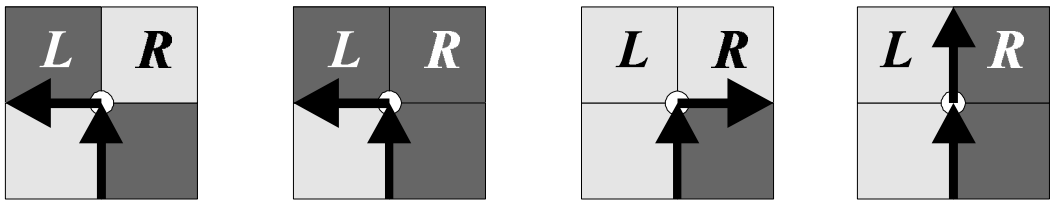
The boundary, or frontier, of a sub-region can be extracted as a series of cracks, comprising no pixels. It may be a closed polygon, or several polygons if the sub-region has holes in it. For the purposes of this application of boundary tracking, it is assumed that a sub-region wholly contained within another will be classified in its own right and so the latter of these characteristics will not apply. Consider figure 5.4.1. If we take the "inner" and "outer" boundaries of the sub-region, then we could say that they were the same, as they achieve the topological classification of a common perimeter. However, Pavlidis [xlix] makes a distinction between them as there is a difference in the numbers of pixels involved.



(i) (ii) (iii)

Figure 5.4.1: (i) An image sub-region, (ii) its boundary comprised of a series of "cracks", (iii) its "inner" and "outer" boundaries under eight-adjacency.

As we already have the sub-region pixel information as a set of connected values, we simply need to describe the shape they take as a unique classification. This is done by a method similar to that described by Kovalevsky [l], where boundaries are tracked and a complete shape description taken when the outline is closed. (figure 5.4.2).



(a) (b) (c) (d)

Figure 5.4.2: Turn rules for boundary tracking

We select the perimeter as being the series of cracks bounding the entire sub-region and the boundary is tracked, starting and finishing at the seed pixel location. When a node (the intersection of four pixels) is reached, a decision is made on the next tracking direction by evaluating the four adjoining pixel values. If *L* and *R* are in the object, turn left, but if *L* and *R* are in the background turn right. If *L* is in the background and *R* is in the object retain the old direction. Kovalevsky allows for a further condition to track left if *L* is equal to the object value and *R* is not (rule (a)), however this only satisfies the connectivity of $N_8(\mathbf{p})$, which is not being used in this feature extraction process. When complete, a code can be constructed to describe the path followed by the boundary tracking algorithm and, given the seed pixel location and the sub-region pixel value, the feature can be reconstructed.

5.5 Boundary Coding

5.5.1 Runlength Coding

Figure 5.5.1 shows the resulting boundary for a given sub-region, R_n, in image space *R*.

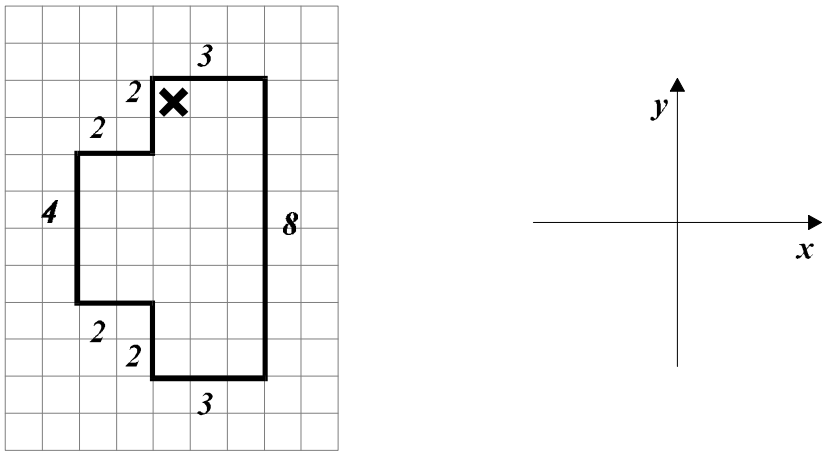


Figure 5.5.1: A connected sub-region boundary

The perimeter is assigned a *boundary label* $\boldsymbol{L}$, which uses a runlength code to describe, in relative Cartesian co-ordinates, the magnitude of each side and the series of turns made to complete the boundary. In the case of the example shown, the perimeter can be coded:

$$\boldsymbol{L}_n = +3\ -j8\ -3\ +j2\ -2\ +j4\ +2\ +j2$$

To test that the boundary is closed at the seed pixel location, we simply require that the sum total of $\boldsymbol{L}_n$ is zero. An essential feature of the boundary label is that it is simply shape-descriptive. That is to say that a much larger version of the same shape requires the same amount of data to describe its outline. The result of this is that runlengths are relatively inefficient for smaller shapes found in an image region.

5.5.2 Feature Primitives

A practical approach to improving the efficiency of small sub-region coding is by the use of feature primitives. In a given image, particularly one which has been spatially pre-processed, it has been found in this work that many subregions tend to be regularly-occurring shapes, albeit of different size. By producing a set of primitives, we can remove the need for a full boundary label and instead apply a single code relating to a known primitive in a look-up table. The definition of a primitive is that it is unit-length, such that at least one side has a length of one pixel/block. Some typical primitives are shown in figure 5.5.2.

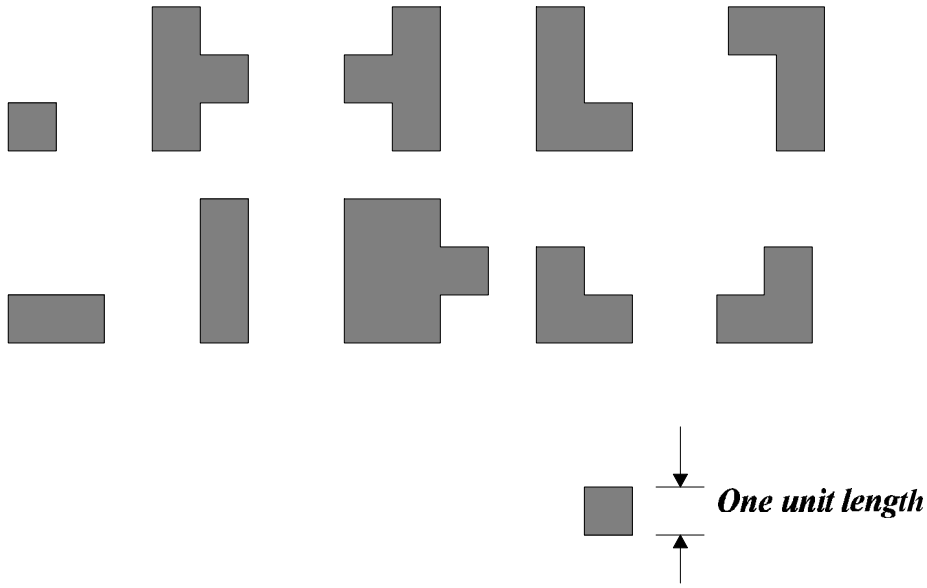
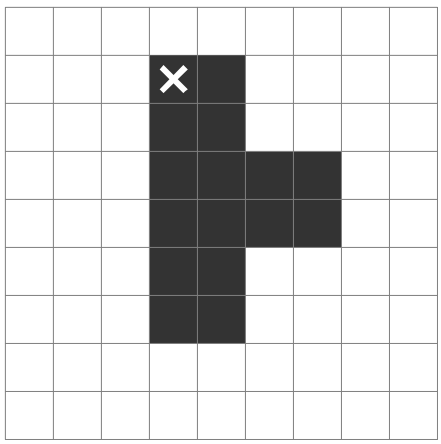


Figure 5.5.2: Some examples of unit-length sub-region primitives

If a single byte were to be used for the sub-region primitive description and another for a positive scale factor, *s*, it would be possible to considerably reduce the transmitted data overhead. The format of the data for each sub-region is [*s*:(*code*)] (figure 5.5.3).



$\boldsymbol{L}_n$ = +2 -j2 +2 -j2 -2 -j2 -2 +j6

Primitive description = 2:(21)

Figure 5.5.3: Use of primitive descriptions reduces the data overhead

5.6 Implementation of a classification algorithm

5.6.1 Clustering

Having developed the fundamental processes of feature classification, the implementation of a practical algorithm is fairly straightforward. The clustering technique, where we start with a seed pixel and list all adjoining pixels meeting the *m*-neighbour criteria, indicates the use of a linked list. By simply tagging each pixel value and location, we can show its relationship to adjoining pixels, as well as indicate that it no longer needs to be considered as a possible component of another subregion.

Information about file handling is shown in appendix 2. Using a two-dimensional array, having the same maxima as the width and height of the image frame, we can classify groups of values according to their value and the relative position (*x, y*) in the array. The framework of this technique is shown in the pseudo-code below:

```
for y = 0 to (frame_height - 1) {
      for x = 0 to (frame_width - 1) {
            if pixel_value[x,y] is not already marked
                  mark pixel_value(seed);
                  mark pixels at [(x+1),y][x,(y+1)] if = seed;
                  for the pixels locations [(x+1),y][x,(y+1)] {
                  loop to iteratively mark pixels at [(x+1),y]
                        [x,(y+1)][(x-1),y][x,(y-1)] if = seed };
      increment x by 1 };     (Pixel scan left-right)
increment y by 1 };           (Line scan top-bottom)
```

This approach is shown diagrammatically in figure 5.6.1.

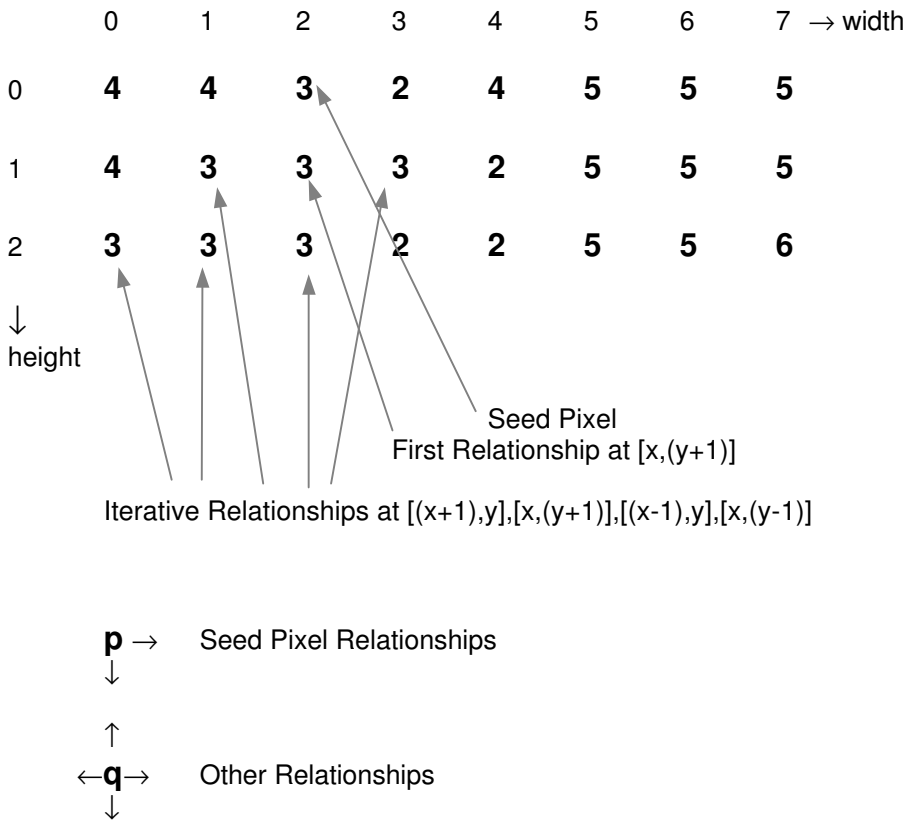


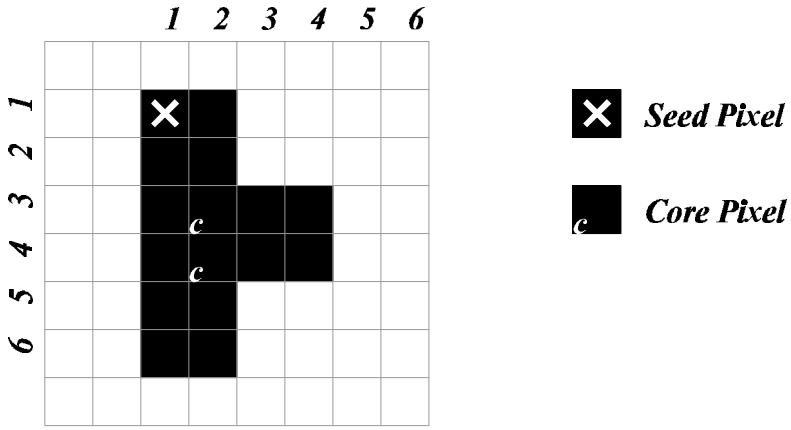
Figure 5.6.1: Array implementation of a clustering algorithm

Clustering is quite efficient, since the effect in areas having large subregions is that once pixels are linked, the scan jumps over them until it finds a pixel location which has not been clustered. This is taken as the seed for another subregion and so on.

5.6.2 Conversion to runlength labels

Having scanned a full frame, a set of linked lists exists to describe the pixels which go to make up the constituent subregions. Consider the shape shown in figure 5.6.2. The corresponding linked list contains all the information needed to generate the runlength label, since the boundaries of the subregion occur at the maximum and minimum values in the directions $+x$, $-x$, $+y$ and $-y$. If we extract these maxima and

minima, the feature runlength label is easy to construct. For large regions, this is a considerable compression in the amount of data to be handled, since the core pixels in the subregion play no further part in the classification if their value is the same as the seed pixel.



Linked list maxima and minima (*x,y*):

(1,1),(2,1),(2,2),(3,3),(4,3),(4,4),(3,4),(2,5),(2,6),(1,6),(1,5),(1,4),(1,3),(1,2),(1,1)

Figure 5.6.2: Linked list maxima and minima

.

In this case, we have dropped only two core pixels at (2,3) and (2,4). We can extract the runlength directions and magnitudes by simply ordering the linked list maxima and minima, as shown. Starting with the seed pixel (1,1), the next value is (2,1) and the next is (2,2). A runlength component is taken from two adjacent values, where one co-ordinate is the same. So (2,1) has a similarity to (1,1), but (2,2) does not. Hence the *y*-component is constant, but the *x*-component comprises two pixels, including the seed pixel, increasing in order. Therefore, the runlength component for this segment will be +2. The value (2,1) also forms a set with (2,2). Here the *x*-component is

constant, whereas the *y*-component comprises two-pixels, so the runlength component will be -j2.

For the longer runlength on the left-hand side of this shape, we have a set comprising the linked values (1,6),(1,5),(1,4),(1,3),(1,2)(1,1), returning to the seed pixel location. Here *x* is constant, but *y* decreases over six pixels, so the runlength component will be +j6.

Hence, the runlength label, for the subregion located at seed pixel location (1,1) will be:

$$L_{(1,1)} = +2\text{-}j2+2\text{-}j2\text{-}2\text{-}j2\text{-}2+j6$$

To check on the coding, these values are added together and their sum is found to be zero.

5.6.3 Conversion to feature primitives

The subregion primitives shown in figure 5.5.2 are examples from a pre-defined set of 80 shapes, found to occur frequently in a variety of video sequences. All primitives have at least one side which is unit-length - i.e. they cannot be scaled down any further. The shape shown in figure 5.6.2 corresponds to primitive 21 from the reserved set, scaled by a factor of two. To express this, we take the runlength and recursively divide it by two until one of the values is unit length.

In the case of $L_{(1,1)}$, only one iteration is required:

(+2-j2+2-j2-2-j2-2+j6) ÷ 2 = (+1-j1+1-j1-1-j1-1+j3)

so $L_{(1,1)}$ = 2:(21)

The generation of primitive codes is done by simple comparison with a look-up table, which would be available to both the encoder and decoder. A section of the look-up table is shown in appendix 3.

Runlength labels which cannot be matched to subregion primitives are retained in their existing state. Whilst this is an addition to the data overhead for the description of a given subregion, it is found that most features correspond to the set of primitives, as shown in figure 5.6.3.

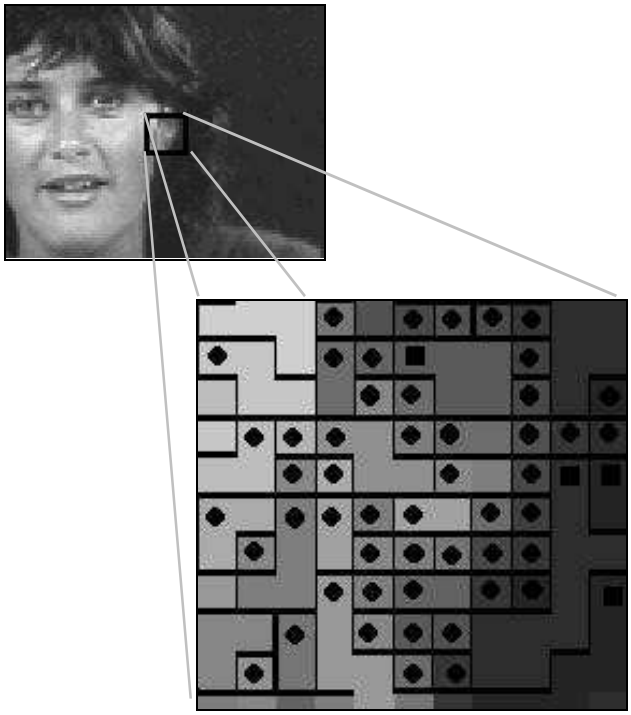


Figure 5.6.3: Distribution of classified subregions in Miss America frame 001, mode-value subsampled, $\Delta Q = 8$.

In this example, subregions corresponding to the set of primitives have their seed pixel locations marked by the filled circle, ?, whereas those requiring full runlength descriptions are denoted by a filled square, ¦. This segment of the image is transition area of fairly high frequencies, with the ear contrasting against the darker background of the hair. In such areas, there tend to be high concentrations of single-pixel subregions, *s*:(01), which are needed to form a luminance gradient from one object to another. However, for much of this image, features are larger and easily defined.

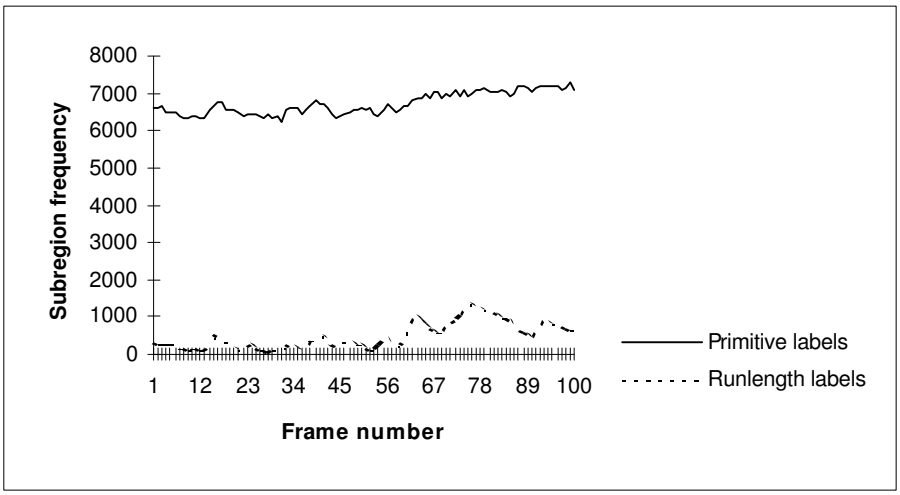


Figure 5.6.4: Subregion runlength and primitive code frequencies Miss America CIF, ΔQ=8

Figure 5.6.4 shows the distribution of runlength labels and primitive codes for the sequence *Miss America*. For clarity, this was processed in the full-CIF format. It can clearly be seen that full runlength codes occupy only a small part of the total subregion description, typically less than 10% at the start of the sequence. However, as the amount of motion increases, there is a greater quantity of runlength labels, due mainly to a corresponding increase in single-pixel primitive codes, seen here between

frames 56 and 100. There is an increase in data overhead for this period, however it reduces once the motion has passed (in this case, the head moved from side to side).

Primitive codes are thus not only efficient in terms of data content, but also in their frequency of occurrence in the video sequence. As will be seen in chapter 6, this greatly assists the prediction of interframe motion.

5.6.4 Data Representation

Where a subregion can be represented by either a runlength label or a primitive code, it is necessary to distinguish which is to be used. The format used for sourcebook encoding is shown in figure 5.6.5, using as an example the first two lines of pixels in the enlarged area of figure 5.6.3.

Seed Location	η value	L_n or L_p
(4,1)	72	1:01
(6,1)	56	1:01
(7,1)	48	1:01
(8,1)	64	1:01
(9,1)	40	1:01
(1,2)	128	1:72
(4,2)	64	1:06
(5,2)	72	1:01
(6,2)	64	+3-j2-2+j1-1+j1
(9,2)	48	1:01

Figure 5.6.5: Sourcebook representation of classified subregions

For the purposes of laboratory simulation, the sourcebook needs no further encoding. However, in a serial communications system, these descriptors would have to be encrypted as datagrams, uniquely identifiable in a bitstream. The method proposed by Huffman (section 2.4) for minimum-redundancy coding could be applied equally to this application, as it can be seen that there is a high frequency of single-pixel features in the example area. Proposals for further data compression will be outlined in the conclusions of chapter 8.

5.7 Summary

This chapter has presented a novel method for the evaluation and classification of image subregions as discrete features for the purposes of video coding. Using a clustering algorithm and a few simple predicates specifying the relationships of neighbouring groups of pixels, it is possible to segment a locality into clearly-defined features.

The practical implementation of the classification technique uses a linked list to show the relationship between pixels in their relative x and y axes. Then, by examining only the maxima and minima for the co-ordinates in each of the feature lists, a runlength label, L_n ,of directions and magnitudes in Cartesian notation can be used to depict the topography of a subregion.

The runlength labels can further be coded with equivalent, pre-defined feature primitive codes L_p, which relate directly to the runlengths of subregions found to occur frequently.

The data is then held in a sourcebook, comprising the components of seed pixel location (x,y), the subregion pixel value, η and a code describing the shape using either runlength label or primitive code notation. It will subsequently be shown that this information can be used as the basis of a simple method of interframe coding.

This process of subregion segmentation, description and classification is clearly a method of intraframe coding in its own right. The sourcebooks of subregion descriptions provide all the information needed for subsequent interframe coding and it is their size governed, in this case, by the effects of spatial processing, that will affect coding efficiency and reconstructed image quality.

Chapter 6

Interframe Coding

Evaluating the displacement of subregions

6.1 Introduction

In the development of this novel algorithm for the detection of interframe motion, it is the feature classification aspect which holds the key to its success. In chapter 3, it was seen how the designation of blocks or segmented subregions allowed a simple search for motion to take place. Feature classification is no different, although now we have a sourcebook of data relating specifically to the attributes of features in the image. It would be reasonable to say this provides a very good starting point for the estimation of interframe motion, compared with the rather arbitrary approach of block-based motion compensation (BBMC).

A further consideration is that, generally speaking, BBMC tends to be used in conjunction with conventional DPCM/DCT coding, as a hybrid approach to interframe coding. As a technique having a fairly high computational intensity, it forms only a small part of the process of predictive interframe coding. It will be suggested in the conclusion to this chapter that feature classification provides an effective means of displacement vector generation and, in its own right, offers the basis of a novel video codec.

It has already been shown that a sourcebook can be generated for each frame in the sequence. This chapter will show how an update process delivers information about

interframe differences to the decoder. Having transferred a set of initial parameters, the data overhead drops to levels consistent with the requirements of a low bit-rate environment. The origins of base vectors will be used to provide information on the extent of interframe motion and spatial plots will show how these vary over a test sequence. The quantity of errors will also be plotted, showing the effect of interframe vector coding on the quality of the reconstructed images.

6.2 Codec Structure

The video coding algorithm described in this thesis is shown in the schematic of figure 6.2.1. It differs from conventional approaches to interframe coding in that a lot of processing takes place on a frame *before* it is compared with temporally adjacent neighbours, for the production of motion vectors. The process of spatial quantisation and sub-sampling from mode values, together with the feature classification technique, provides a sourcebook of codes, each describing the content and topography of image subregions.

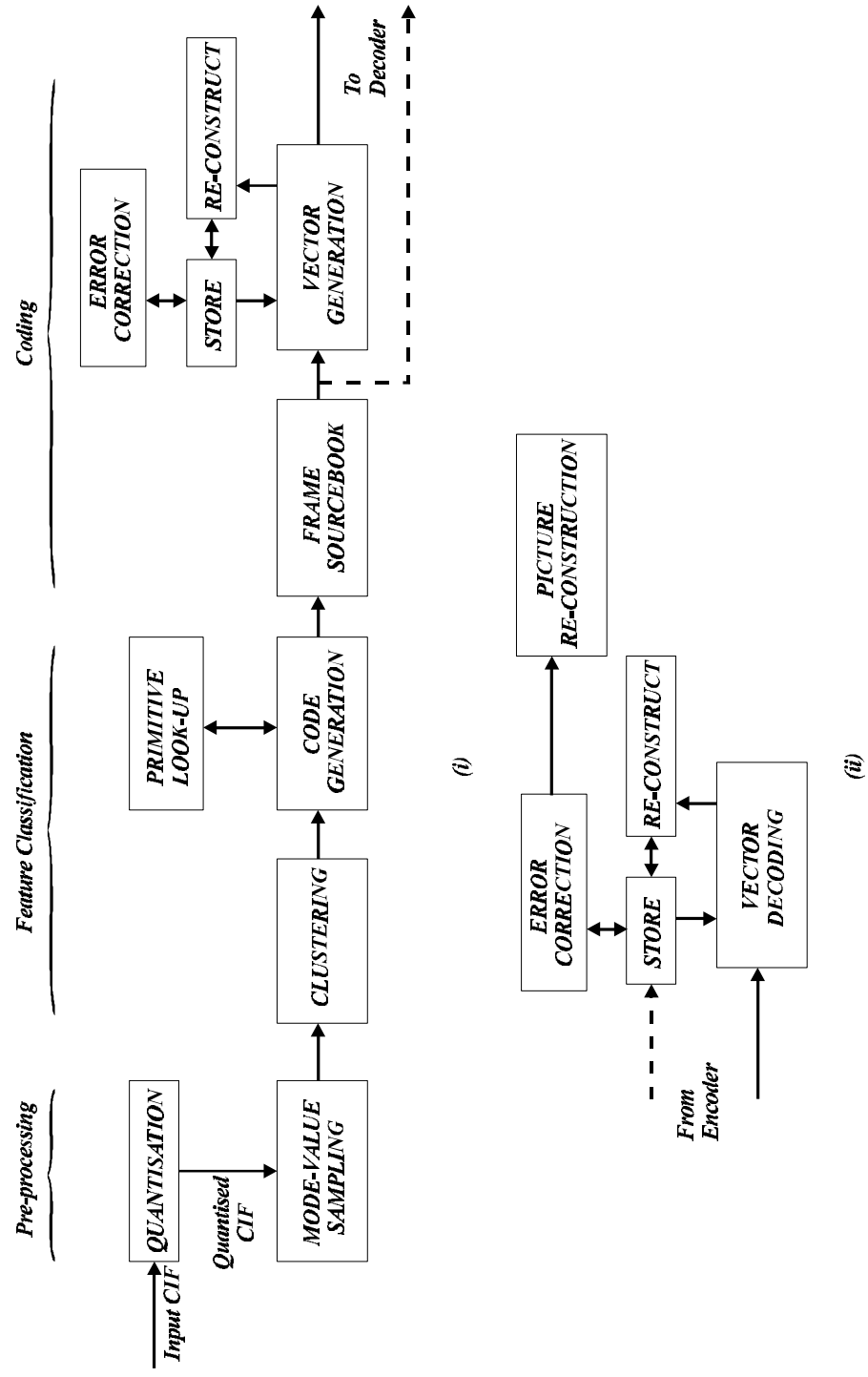


Figure 6.2.1: Classified feature interframe coding schematic showing (i) encoder and (ii) decoder structures.

In the same way that managed coefficient quantisation retains an acceptable level of quality for the DPCM/DCT process in the H.261 algorithm, a vector generation technique using feature sourcebooks must make an efficient trade-off between picture quality and the data compression requirements of the system.

As with any algorithm where errors are created as part of the coding method, it is important that error propagation is minimised. To accomplish this, components of the vector encoding algorithm are common to those needed to decode and reconstruct the sourcebook at the decoder. Where the H.261 algorithm uses an updated picture store in the encoding and decoding loops, the feature classification method uses an updated sourcebook, comprising component subregion descriptions.

6.3 Motion Vectors

The process of motion vector generation is not significantly different to the BBMC or DBMC techniques, outlined in chapter 3. There are still two sets of data, relating to adjacent frames in a video sequence. For each known artefact at a given location, we search the equivalent spatial vicinity in its neighbouring frame to see if any motion has occurred. For most of the time, much of an image may be stationary, in which case the location of a feature in adjacent frames will be constant.

For a pair of temporally adjacent sourcebooks, where motion is seen to occur it may take one of three forms. Firstly, a feature can be displaced, retaining its outline, with all constituent pixels shifted by the same magnitude and direction. This effect is known as *linear interframe motion* and an example is shown in figure 6.3.1. The second possible outcome of a displacement might occur where the fundamental

outline of a shape is unchanged, but it has become larger or smaller. For such *perspective motion*, the location of the seed pixel may not actually have changed, and a motion vector may have to be designed such as to allow the alteration in perspective, without displacing the whole object. Finally, a feature may have been displaced, retained its outline and been rotated at the same time. It would be difficult to describe this *complex motion* with a single vector, however the practical effect of rotation would be that a feature could only have retained its exact topography if the shift were a multiple of 90°, as shown in figure 6.3.5.

6.3.1 Linear motion vectors

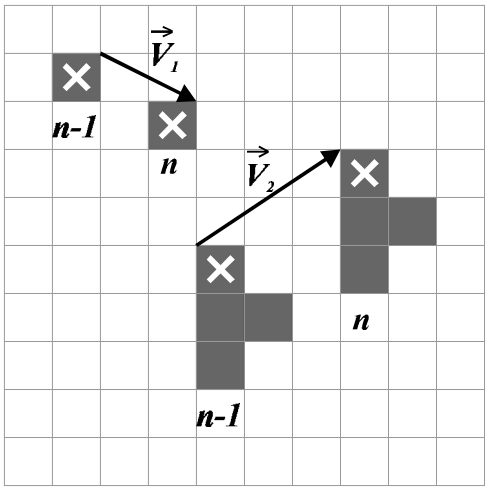


Figure 6.3.1: Linear interframe displacement of feature primitives

Providing the features have retained their topography, the majority of motion vectors produced for the sequences *Miss America* and *Salesman* are found to be linear. Linear motion vectors are the most simple to code, since the decoder will know that all pixels forming a feature will experience the homogeneous displacement specified by the motion vector.

Figure 6.3.1 shows the development of linear motion vectors for given primitive features. We choose as a common reference point the seed pixel, $\mathbf{p}_{n-1}$, in the previous frame, I_{n-1}. A search of feature runlength labels and values for the equivalent spatial vicinity in the current frame, I_n, shows any displacement of the feature. The value of a linear motion vector is taken as the equivalent spatial separation of the seed pixels $\mathbf{p}_n$ and $\mathbf{p}_{n-1}$. The vectors are then recorded in Cartesian notation,

$$\vec{V}_n = (\pm x \pm jy) \qquad \text{[Equation 6.1]}$$

$\vec{V}_1$ = +2-j1

$\vec{V}_2$ = +3+j2

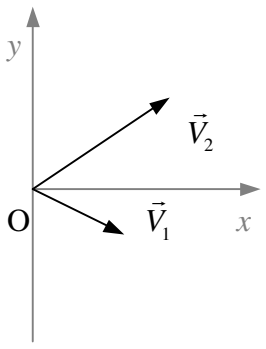


Figure 6.3.2: Linear displacement vectors (for example of figure 6.3.1)

Displacements are taken as relative to the origin (O) of the seed pixel, $\mathbf{p}_{n-1}$, rather than their absolute location in the whole frame. This allows local reconstruction to take place.

In the example, the features shown are both primitives and it may be the case that several features located in the search area match both the topography and value of the reference subregion, R_{n-1}. Any displacement is therefore to be assumed as relating to the location nearest the seed pixel $\mathbf{p}_{n-1}$.

6.3.2 Perspective motion vectors

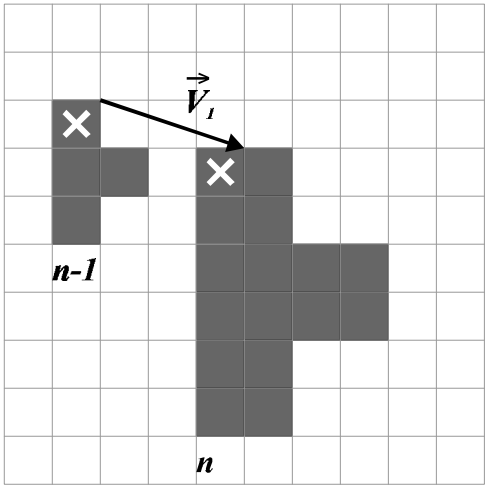


Figure 6.3.3: Perspective interframe motion of a feature primitive

For the case of a displacement which occurs in conjunction with a perspective transform (figure 6.3.3), the process of feature searching will be similar to that employed for the derivation of linear motion vectors. However, the search algorithm will need to recognise a feature as being scaled from the reference feature.

Practically, we could expect the process to take a form similar to the feature primitive classification technique, where a simple filter scales the runlength labels to unit-distance values before making a comparison with pre-determined criteria.

$\vec{V}_1$ = 2:(+3-j1)

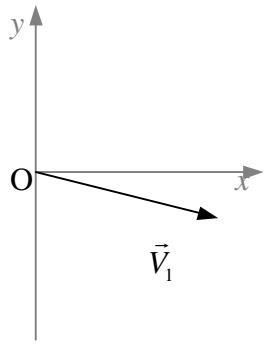


Figure 6.3.4: A perspective displacement vector

Figure 6.3.4 shows the introduction of a scale value for a perspective transformation and displacement of the example feature. The form of the vector notation is similar to that employed for feature primitive labels, where a universal scale factor, *s*, precedes the displacement vector components.

6.3.3 Stationary motion vectors

It seems perhaps rather an abstract idea, but if the reconstruction is to be complete, codes must also be produced to show where no motion has occurred. Stationary, or *null* vectors are used to tell the decoder to keep features where they were in the previous frame.

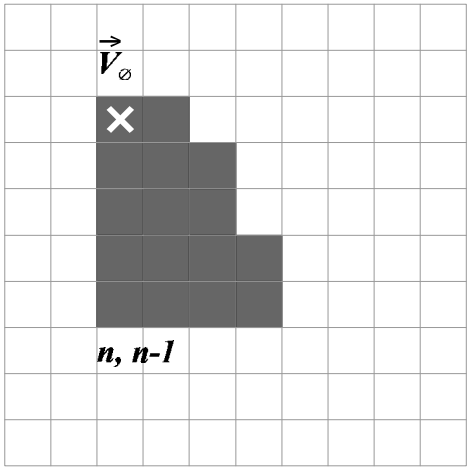


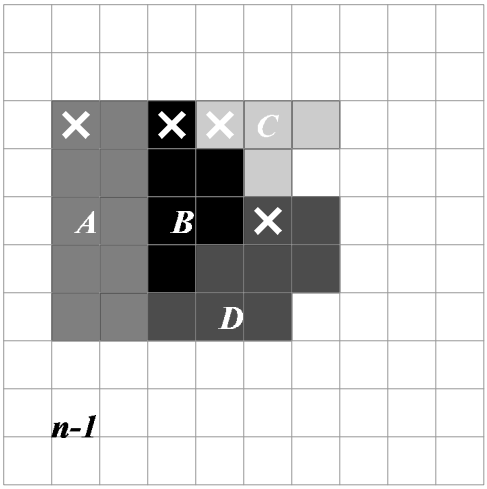
Figure 6.3.5: A null (stationary) motion vector

This approach is shown in figure 6.3.5. A null vector, $\vec{V}_{\varnothing}$, is produced to show the location of a stationary object. The use of null vectors has proved important to the success of a displacement coding algorithm, since they form the majority of all interframe codes produced. As will be seen later in this chapter, vectors describing motion account for only a small part of the total interframe data produced in the test sequences.

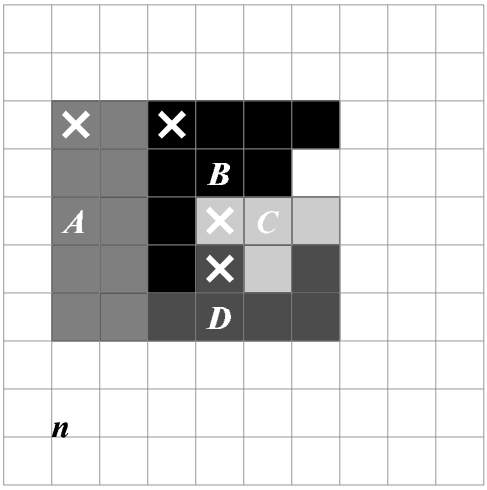
6.4 Changes in feature topography

Motion vectors are very useful in portraying the displacement of simple subregions, however there are occasions when a search fails to detect any shape in the selected neighbourhood, likely to be the destination of a feature in the previous frame. This could be for two reasons. Either the motion has been so great that the feature has moved outside the designated search area, or the nature of interframe activity, caused principally by neighbouring features, will have distorted the topography of the feature.

In these cases, motion vectors cannot adequately describe changes in the feature parameters and a full re-classification will be needed for the unaccounted pixels. Consider figure 6.4.1. Here we see an area comprising several subregions, all classified, with the seed pixel locations known in the frame I_{n-1}. However, the effect of displacement is that some features have moved and can still be identified by their descriptors, whilst their motion has caused a change in topography to another feature which may not have been subject to displacement in its own right.



(i)



(ii)

Figure 6.4.1: The re-classification of distorted features

In the example, subregions *A* and *C* have retained their topography - *A* is stationary and *C* has undergone linear displacement. Subregions *B* and *D*, on the other hand, have been directly affected by the displacement of subregion *C*. A good example of this would occur where a foreground object was moving over a background, with the effect of revealing more of the background feature. Whilst *A* and *C* can be represented by null and linear motion vectors, subregions *B* and *D* could not be located in a search of this area. Consequently new seed pixel locations would be chosen and the features described.

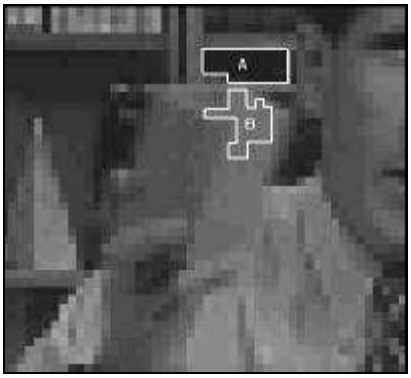
This is an addition to the processing overhead and is not the most optimal method of feature re-classification. However, it is important to keep the decoder sourcebook updated with as much contemporary data as possible and the supply of re-classified runlength and primitive labels seems to be a good solution.

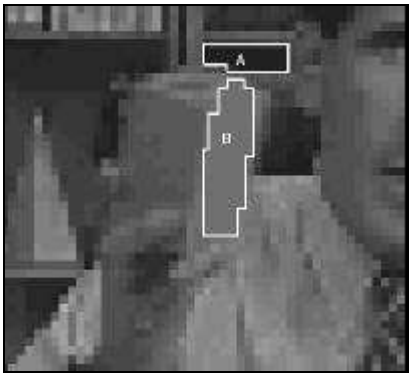
Practically, it is found that re-classification accounts for typically 8% of the features in the *Miss America* sequence and typically 14% for the *Salesman* sequence ($\Delta Q=8$). An illustration of the need for re-classification is shown in figure 6.4.2.

The data overhead of motion vectors increases with spatial complexity. The background of *Salesman* is more detailed than was seen for *Miss America*, however most motion vectors are null. Only when the background is exposed or covered by the motion of a foreground object will interframe differences occur. This example shows a stationary background feature (*A*), near to a displaced foreground feature (*B*). In this case, re-classification of the foreground feature is required, since at this

point the box is being rotated and changes in overall luminance will occur for that part of the image.

(i)





(ii) (iii)

Figure 6.4.2: (i) Frame 58 from the Salesman sequence with (ii) an enlarged area showing selected subregions and (iii) their displacement in frame 59. ΔQ=8, QCIF resolution.

6.5 Implementation of a displacement vector algorithm

6.5.1 Searching

Figure 6.5.1 shows the base schematic for a feature classification motion vector algorithm. Primarily, we use two sourcebooks for the searching - the previous frame I_{n-1}, from which features have originated and the current frame I_n, where displacements may occur. Once again, the scanning process starts in the top-left and

works down to the bottom-right corner of the array In the process, all seed-pixel values in the x direction are checked before incrementing y. For each seed pixel $\mathbf{p}_{n-1}$ in the frame I_{n-1}, we search I_n to determine the displacement.

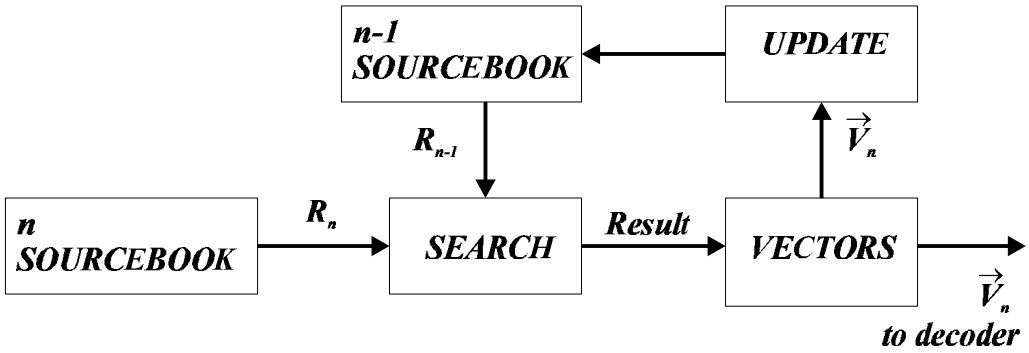


Figure 6.5.1: Vector coding search schematic

Consider again the example features illustrated in section 5.6.4 (figure 6.5.2).

Seed Location	η value	L_n or L_p
(4,1)	72	1:01
(6,1)	56	1:01
(7,1)	48	1:01
(8,1)	64	1:01
(9,1)	40	1:01
(1,2)	128	1:72
(4,2)	64	1:06
(5,2)	72	1:01
(6,2)	64	+3-j2-2+j1-1+j1
(9,2)	48	1:01

Figure 6.5.2: Feature sourcebook structure

Let this set act as the sourcebook structure for the frame $I_{n\text{-}1}$. The first feature is at (4,1). The initial step in the search algorithm would be to see whether a similar feature exists in the sourcebook at the same spatial location in I_n, matching pixel value 72 and primitive label 1:01. If this is the case, a null vector $\vec{V}_{\varnothing(4,1)}$ would be produced and supplied to the decoder. Any failure to match the equivalent seed pixel in I_n would invoke the full search algorithm.

Full searching works along the same lines as many other schemes for motion estimation. We take the seed pixel location $\mathbf{p}_{n\text{-}1}$ in the previous frame and around it, search all seed pixel locations residing within a search window in I_n. For each seed pixel location, the value and description of the associated feature is compared with that relating to the reference feature. A linear motion vector can then be plotted from the reference feature seed pixel to the seed pixel location in frame I_n that matches $R_{n\text{-}1}$ and is spatially closest to its seed pixel. This process is shown in figure 6.5.3.

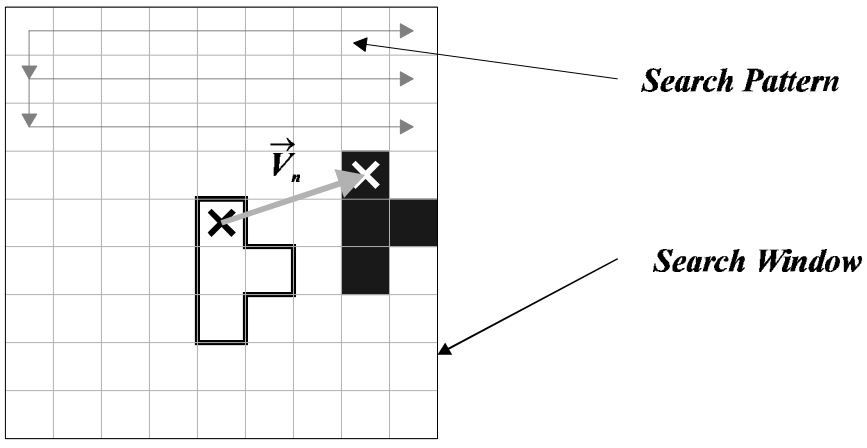


Figure 6.5.3: Local feature searching

It may be the case that, although a seed pixel occurs within the search window, part of the body of its associated subregion extends beyond this area. This is not a problem, since it is the seed pixel location used for a reference marker in local searching. In conventional BBMC, the size of search window is usually important as it has a direct effect on the computational overhead. However, in this case the effect is of less concern, as the search parameters are provided in the sourcebooks and no additional calculations have to be made. The comparisons between the subregions associated with each seed pixel are made on a topological basis. The algorithm is also less intensive as only the seed pixels are used in a search - conventional BBMC tries every pixel location for a match with its reference block. The structure of the searching algorithm is shown in figure 6.5.5.

6.5.2 Data Structure

Where vectors have been produced successfully, they can describe feature displacement with one of three data structures, shown in figure 6.5.4.

Description	Vector Notation	Data Structure
Null Vector	$\vec{V}_{\varnothing(x,y)}$	`VNULL (X,Y)`
Linear	$\vec{V}_{n(x,y)}$	`V(X,Y) (±X±JY)`
Perspective	$\vec{V}_{n(x,y)}$	`V(X,Y) S(±X±JY)`

Figure 6.5.4: Vector data structures

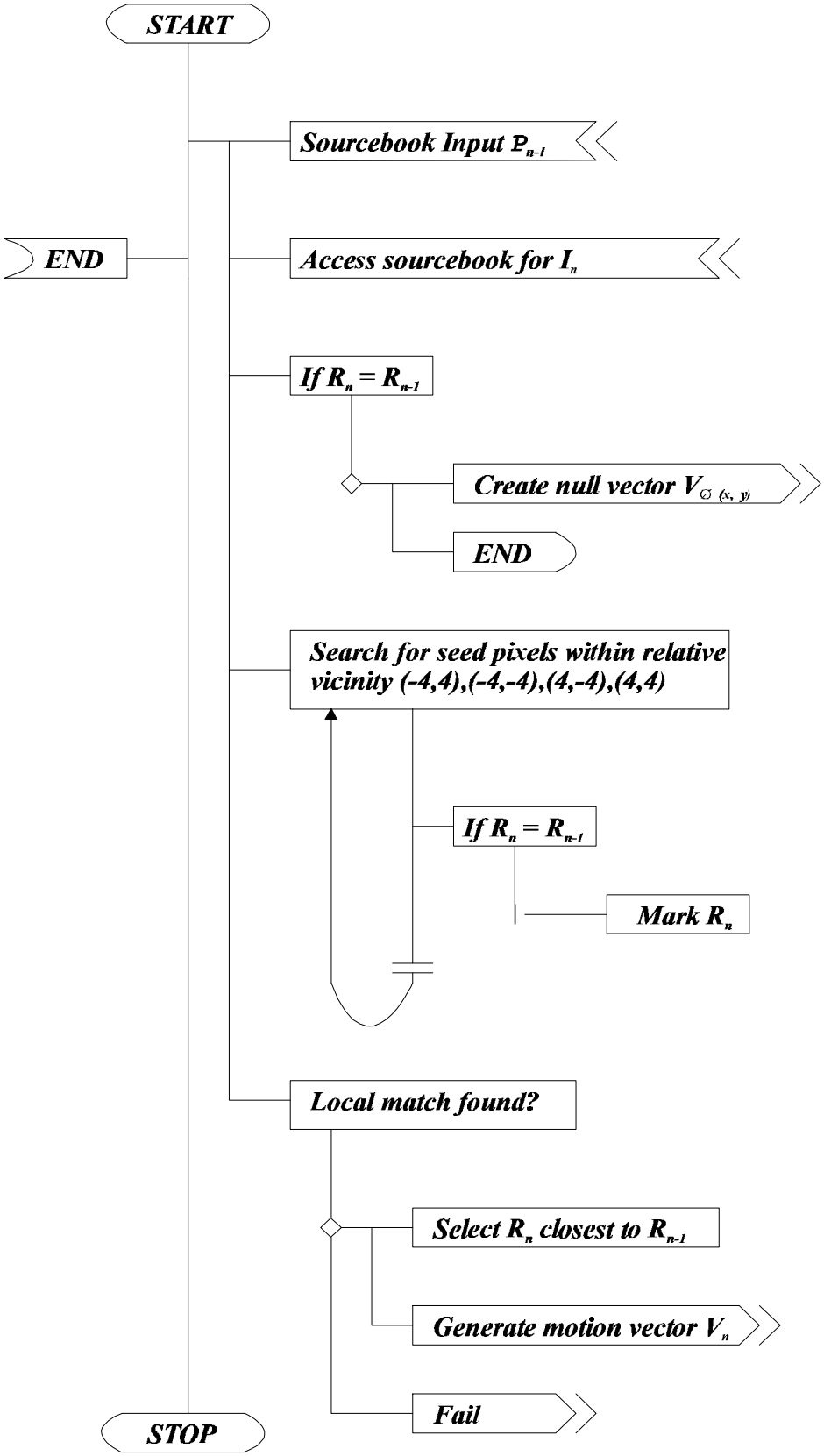


Figure 6.5.5: Design Structure Diagram for the vector search algorithm: BS6224 [li][lii]

6.5.3 Classification of new features

Subregions residing in the sourcebook are marked once a vector is produced, to describe their interframe activity. However, this leaves a number of pixel groups not matched to any (*n*-1) feature. Their classification is simple and easy to perform, along the lines described in chapter 5. Consider again the group illustrated in figure 6.4.1. Subregions *A* and *C* were successfully coded with null or linear motion vectors. Subregions *B* and *D* were distorted and could not be matched. If subregions *A* and *C* are removed, we can classify the remaining features (figure 6.5.5).

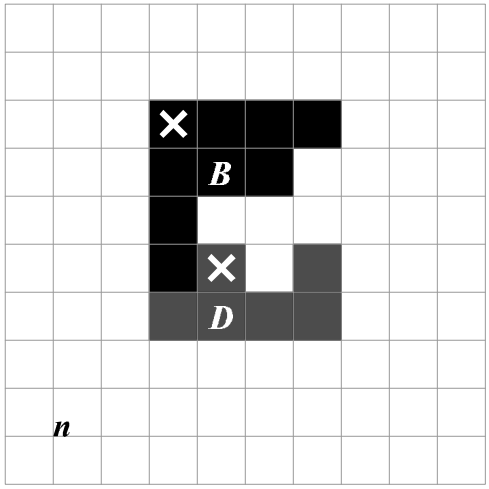


Figure 6.5.5: Feature re-classification

For each of the new subregions, seed pixels are allocated using the criteria specified in chapter 5. Clustering and boundary detection then follow and feature runlength or primitive labels are assigned. For the new frame, new feature data, together with the existing feature displacement vectors are sent to the decoder. For the purposes of simulation, they were held in a file.

6.6 Application to standard test sequences

6.6.1 Motion vector generation

The vector coding algorithm was applied to frame sourcebooks produced for the sequences *Miss America* and *Salesman*. *Miss America* was found to be quite simple to code - the plain background generates very few feature descriptions and the head is mainly stationary. The location of seed pixels relating to subregions undergoing motion is shown in figure 6.6.1.

Figure 6.6.1: Location of motion vector origins, Miss America frame 001, ΔQ=8, QCIF resolution

It can be seen that the distribution of motion vectors around the face is significant - indeed when viewing the sequence there is a concentration of facial activity caused by a change in expression over the first few frames. By frame 32, interframe differences have been reduced as the subject is virtually stationary (figure 6.6.2).

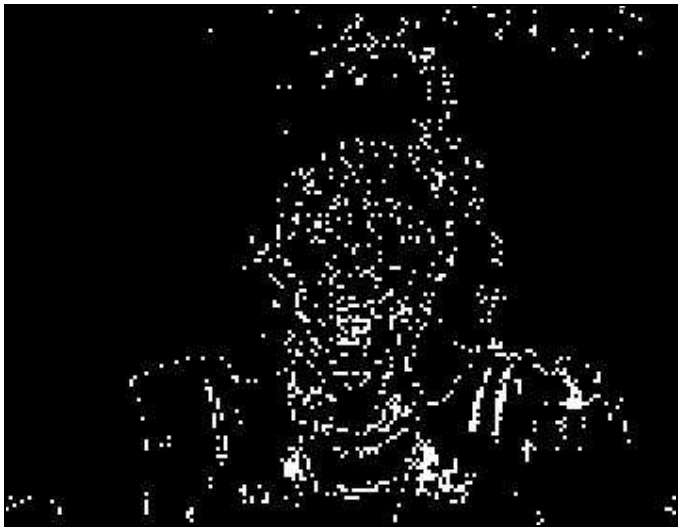


Figure 6.6.2: Location of motion vector origins, Miss America frame 032, ΔQ=8, QCIF resolution

Between frames 60 and 90, there is a lot of interframe activity as *Miss America* moves her head from side to side. This is manifested by a considerable increase in the total of runlength labels and primitives, shown in figure 5.6.4. The maximum extent of head displacement, occurs in frame 79 (figure 6.6.3).

Figure 6.6.3: Location of motion vector origins, Miss America frame 032, ΔQ=8, QCIF resolution

A similar range of effects applies in the *Salesman* sequence. For much of the time, motion is concentrated around the foreground box object, however frame 68 shows interframe activity involving body motion, illustrated in figure 6.6.4.

Figure 6.6.4: Location of motion vector origins, Salesman frame 068, ΔQ=8, QCIF resolution

Salesman is more demanding on the feature classification algorithm, primarily because of the extent of background detail. It is no more difficult to encode, but requires a significantly greater computational overhead to do so. One feature experienced with *Salesman* was that spatial errors were introduced by an aliasing effect during the process of mode-value sub-sampling. This characteristic is discussed further in chapter 7.

6.6.2 Reconstruction

The process of re-constituting new images requires the decoder to displace subregions, in accordance with the motion vector parameters. Using the vector sets for *Miss America* and *Salesman*, spatial reconstruction took place. Equivalent spatial illustrations for the *Miss America* set, corresponding to the base vectors shown in figure 6.6.1 - 6.6.3, are seen in figures 6.6.5 - 6.6.7.

Figure 6.6.5: Reconstructed frame Miss America 002

Figure 6.6.6: Reconstructed frame Miss America 033

Figure 6.6.7: Reconstructed frame Miss America 080

It can be seen that overall shifts caused by the subject have been effectively reconstructed. However, there is a noticeable reduction in spatial quality and this is caused by a process of overlapping, where the reconstruction process places parts of displaced subregions over others and pixel values are added. Chapter 7 presents a discussion of this effect and proposes a solution of non-recursive error detection and correction.

Figure 6.6.8 shows the overall distribution of errors for the *Miss America* sequence. These values are absolute, although it is of more analytical use to consider the plots of signal-to-noise ratio, seen in chapter 7.

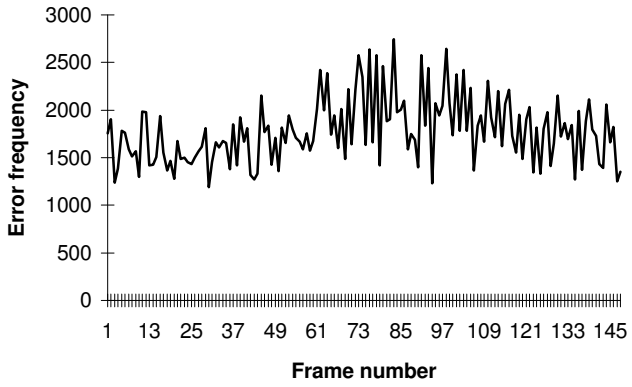


Figure 6.6.8: Distribution of reconstruction errors for Miss America

6.7 Summary

This chapter has demonstrated the implementation of a novel algorithm for interframe displacement vector coding, based on the feature classification technique. Unlike BBMC and other full-search algorithms, the evaluation of seed pixel locations in a local area and subsequent comparison of their related subregions, requires a process of low computational complexity.

Vectors may take several forms, with null vectors showing stationary features and, for the test sequences, accounting for most vectors produced. Linear displacements are calculated by a local full-search algorithm, where feature descriptors are compared by their parameters, without recourse to statistical calculations. Where displacements cannot be detected, the method of feature classification is invoked to classify the parameters of new pixel groups.

The effect of these techniques is to supply to the decoder a set of values from which a reconstruction of interframe activity can be made. The process is not error-free and,

in particular, the overlapping of displaced subregions has been identified as a problem. However, a plot of such error quantities show that they form only a small part of the reconstruction. To enhance the mapping of features, a technique is required to improve the spatial quality of image reconstruction.

Having already described the process of feature description and classification as a useful novel method of *intraframe* coding, the generation of motion vectors is simply an extension of this for *interframe* coding. All the information needed to generate motion vectors, where applicable, is to be found in temporally adjacent intraframe sourcebooks. Where vectors cannot be produced, the update is simply provided by the description of new subregions where the presence of previously classified features has not been detected.

Chapter 7

Image Quality

The detection and correction of prediction errors

7.1 Introduction

This chapter explores the nature of interframe errors caused by various effects of the displacement vector algorithm. In the previous section, it was seen that a reasonably comprehensive set of data is supplied for decoding, comprising motion vectors and new feature labels. Whilst the algorithm is fairly efficient in describing overall trends in interframe motion, small errors do occur where features overlap. The result is that each frame contains areas of unwanted motion artefacts. Using the concepts of spatial filtering introduced in chapter 4, a method of error detection and correction is presented, with the overall objective of improving spatial image quality. It will be suggested that, whilst the spatial quality is still far from that observed in the original images, it is less noticeable as a temporal effect when the images are animated as part of a video sequence.

The signal-to-noise ratio of an image has already been used as a measure of reconstructed image quality, with respect to the base sequence. However, where this was once used for single images only, it is now calculated for all the component frames in a video sequence, revealing changes in signal-to-noise ratio caused by different types of motion. Sequential signal-to-noise ratio plots are also used to show

the effect of error correction, with comparisons made with both the H.263 algorithm and conventional block-based motion compensation.

In addition to quality measures, this chapter considers the anticipated efficiency of the algorithm. The concluding remarks of chapter 8 will suggest that as an area for further development, the frame rate of the feature classification technique could be varied adaptively in order to supply the needs of a constant bitrate communications system.

7.2 Error detection and correction

7.2.1 Displacement errors

Chapter 6 described how interframe errors accumulate where an inaccurate prediction of motion has been made, with the decoder placing shapes over others so as to cause overlaps. It is usually seen to be the case that overlaps affect only one or two pixels for a displaced subregion (figure 7.2.1.), since most simple primitive shapes have at least one side equal to a single pixel width.

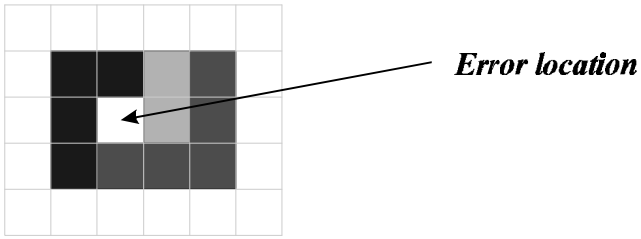
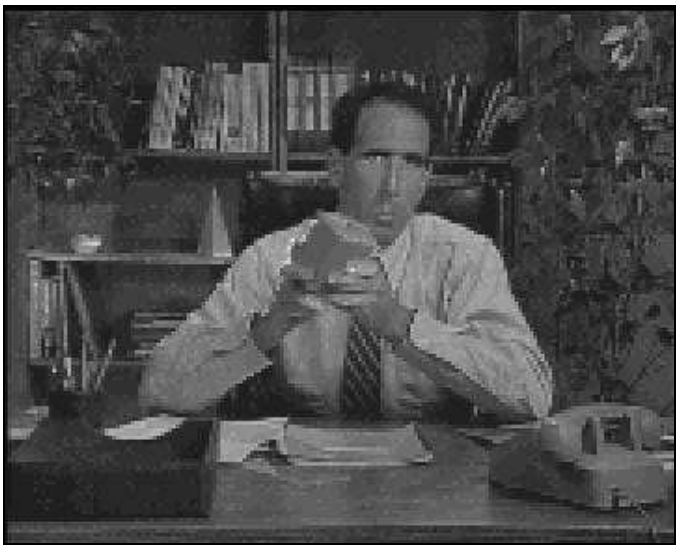


Figure 7.2.1: The nature of interframe errors

The spatial effects of such displacement errors are manifested by a 'frosting' effect of white pixel around the margins of large displaced objects. Figure 7.2.2 shows an example from the *Salesman* sequence. Notice that around the head, shoulders and

sleeves, there are small displacement errors and in particular on the margins of the box (enlarged). *Salesman* is particularly prone to such errors where foreground objects have been subject to motion over a rather detailed, stationary background. Errors are also in evidence for the reconstructed sequences of *Miss America*, but are less frequent, primarily because foreground motion is made with respect to a plain background.



(i)

(ii)

Figure 7.2.2: (i) *Salesman reconstructed frame 052 showing* (ii) *the location of displacement errors.* $\Delta Q=8$, *QCIF resolution.*

7.2.2 Error Correction

Spatial filtering is a technique widely used in the correction of image errors, where noise has been introduced and it is also suitable for this application. Using a filter template, a pixel error can be corrected, not necessarily to its original value, but to a good approximation, based on neighbouring trends. Using either the mean, median or mode value of a particular set, the template centre pixel can be corrected. From a quality viewpoint, the optimum method would be to extract the mean. However, this has the drawback of possibly adding to the image a value unused elsewhere. Also, since pixels can only take integer values, the correction may be a truncation of the mean and thus an approximation. In a low-resolution system, where pixels are being quantised by (in this case) an interval of eight grey levels, a mean value would possibly fall somewhere within the range of a quantisation step interval.

Considering again the work outlined in chapter 5, it is clear that use of the mode value is the best solution as a method of error correction. The mode is taken as the most frequently occurring value in a group covered by a fixed size filter template, with dimensions 3×3 pixels. If there are two values occurring at equal highest frequency, the median is selected instead. The basis of this algorithm is illustrated in figure 7.2.3.

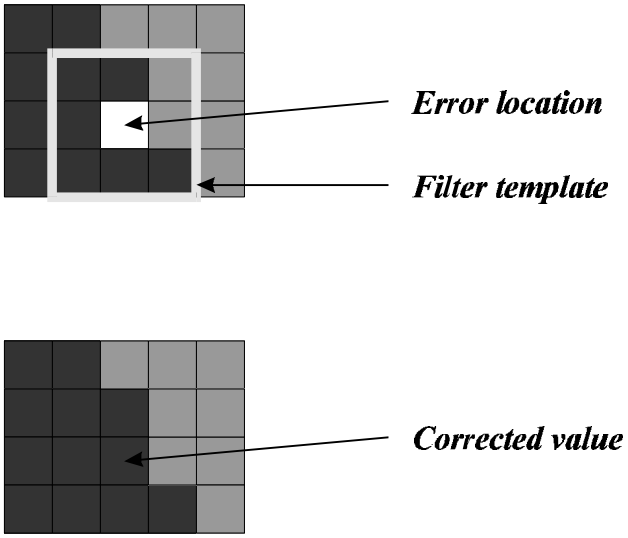
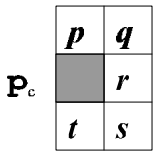
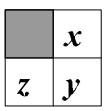


Figure 7.2.3: The detection and correction of displacement errors

The process of error detection is fairly simple. Given that most errors are seen to be single pixels, all single pixel features found in a reconstructed image are extracted. Then, ignoring known seed pixels having the primitive label 1:(01), a search list is produced showing the location of errors (*x, y*). Returning to the reconstructed image, the filter template is centred on the error locations and the corrected pixel value, $\mathbf{p}_c$, is set.



$\mathbf{p}_c$ if (0, *y*), (*xmax*-1, *y*), (*x*, 0), (*x*, *ymax*-1)
$\mathbf{p}_c$ = mode (*p, q, r, s, t*)



$\mathbf{p}_c$ if (0, 0), (0, *ymax*-1), (*xmax*-1, 0), (*xmax*-1, *ymax*-1)
$\mathbf{p}_c$ = mode (*x, y, z*)

Figure 7.2.4: Templates for edge and corner pixel correction

Fixed size filter templates have also been developed for the special cases where errors are found at the edges or corners, seen in figure 7.2.4.

This method of adaptive error correction is useful, since it is of low complexity and does not affect most other features, whose displacement has successfully been reconstructed. Since the encoder will supply both motion vectors and re-classified feature data for the next frame, the results of error correction are not stored, as they would cause error propagation.

7.3 Measurements of spatial quality

Although the method of error correction described in section 7.2 has only a cosmetic effect, its application has been very successful, given that corrected values may still be errors if they have not reverted back to their base sequence value. Figure 7.3.1 shows reconstructed frame 075 from the *Miss America* sequence, before and after the use of error correction filtering.

Figure 7.3.1 (i)

(ii)

Figure 7.3.1: (i) The presence of reconstruction errors - Miss America 075 and (ii) the spatial effects of their correction. ΔQ=8, QCIF resolution

It is clear the spatial quality has been improved in terms of the reduction of displacement errors, however since many such errors occurred at boundary locations, the filtering effect has also reduced high frequency spatial components. Whilst a viewing single frame does not give a good impression of this effect, the temporal improvement in quality is much better. To quantify this, figure 7.3.2 shows a plot of the signal-to-noise ratio for the first 100 frames of the sequence. It can be seen that an improvement of typically +5dB has been achieved.

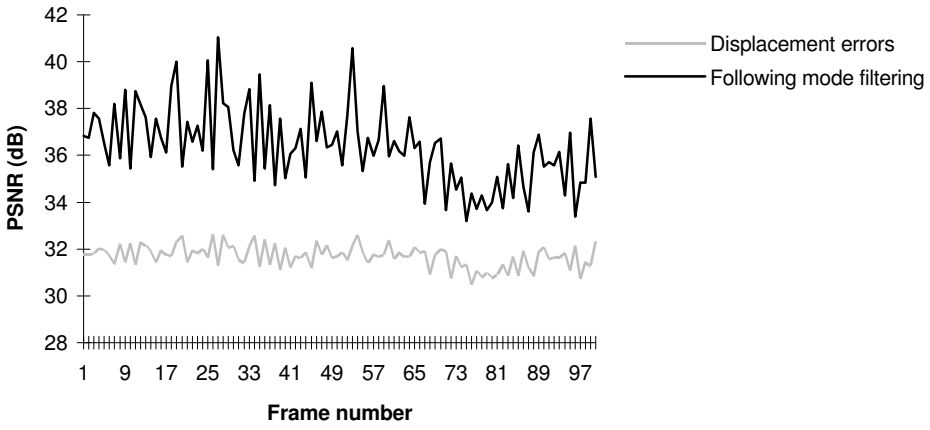
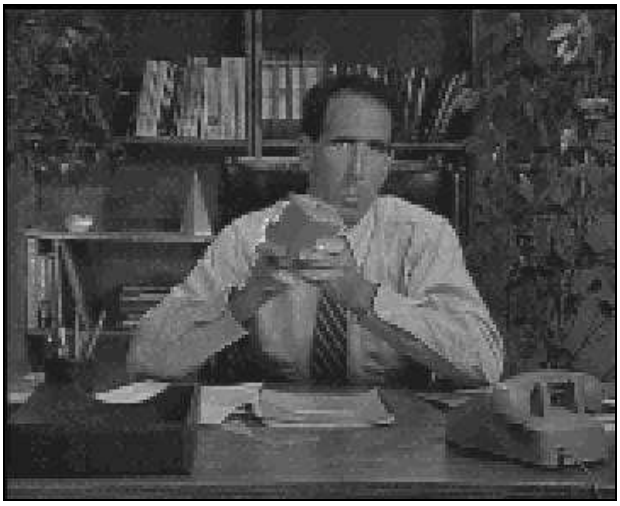


Figure 7.3.2: Signal-to-noise ratio for the reconstructed sequence Miss America, ΔQ=8, QCIF resolution

Signal-to-noise ratio values are calculated with respect to the input frame, with ΔQ set to eight grey levels, using equation 4.2.

A similar improvement is observed for the *Salesman* sequence. The 'frosting' effect seen in the reconstruction has gone and been replaced by corrected values, giving an impression of improved spatial quality (figure 7.3.3).



(i)

(ii)

Figure 7.3.3: (i) The presence of reconstruction errors - Salesman 052 and (ii) the spatial effects of their correction. $\Delta Q=8$, QCIF resolution

Figure 7.3.4 shows a plot of signal-to-noise ratio for the *Salesman* sequence over 100 frames. Once again, the effect of error correction filtering is an improvement in signal-to-noise ratio, of typically +3dB.

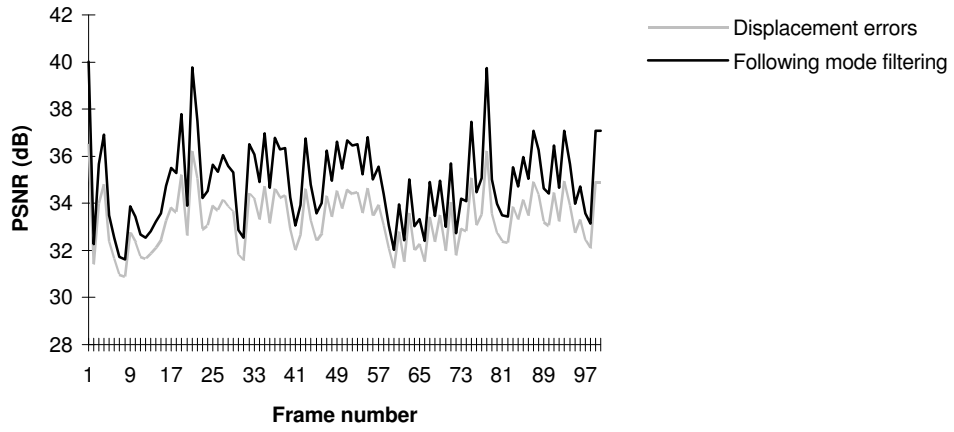


Figure 7.3.4: Signal-to-noise ratio for the reconstructed sequence Salesman, ΔQ=8, QCIF resolution

The high peaks apparent of the signal-to-noise ratio for the corrected sequence are a sign of greater improvements where there have been less errors, resulting from only a small amount of interframe motion.

7.4 Comparisons with the H.263 algorithm

Section 2.8 introduced the infrastructure of the ITU-T H.263 algorithm. This is the latest development in low bit rate video compression algorithms and, working on similar principles to its parent H.261 hybrid DPCM/DCT algorithm, is intended for bitrates of typically 14.4kbits/s to 28.8kbits/s. The half-pixel motion compensation process used in the H.263 codec renders a good interframe prediction and ensures a low level of noise caused by the quantisation of transform coefficients. Block effects, usually observed in H.261 codecs, are less evident.

Using a simulation algorithm, developed primarily by Telenor Research [liii] and BT Laboratories, the *Miss America* and *Salesman* sequences were encoded using the

H.263 parameters. For each sequence, two sets of resulting bitstreams were produced - the base H.263 coding and base H.263 with optional annexes D-G applied (section 2.8.2). Whilst the quality of the reconstructed frames is improved by the incorporation of PB frames (annex G), the unrestricted motion vector searching, outlined in annex D, have little effect since the background is stationary.

Figure 7.4.1 shows a reconstruction of frame 75 from the *Miss America* sequence, with H.263-base and H.263-annex coding applied respectively.

Figure 7.4.1 (i)

(ii)

Figure 7.4.1: Reconstruction of (i) H.263-base and (ii) H.263-annex encoded sequences, Miss America 075. Frame rate = 28.8 kbits/s

Plots of signal-to-noise ratio show the relative improvement to image quality provided by the use of annex functions. It can be seen that the signal-to-noise ratio improvement generally increases further into the sequence.

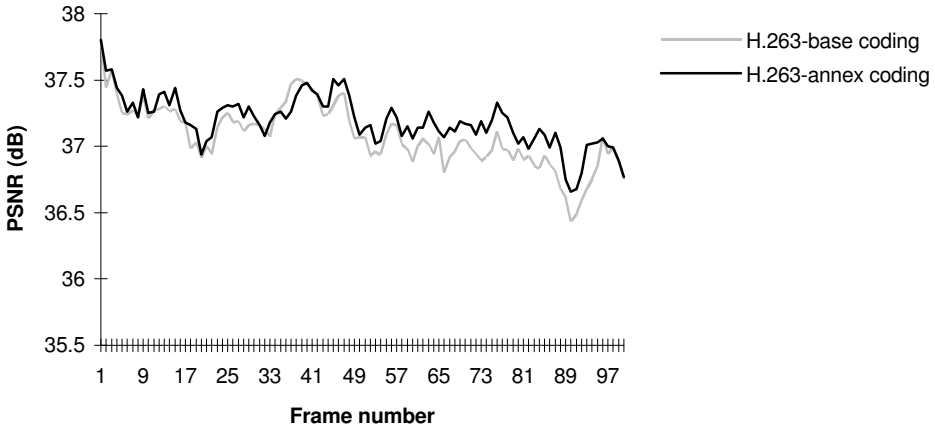


Figure 7.4.2: Reconstructed sequence signal-to-noise ratio for H.263 coding. Sequence Miss America. Frame rate = 28.8 kbits/s

It is interesting to see that not only does the overall signal-to-noise ratio seem to drop during the sequence, but the difference between H.263-base and H.263-annex coding increases. The implementation algorithm used intraframe coding for the first frame of the sequence. As the frames are further from the intraframe frame, the prediction is less reliable, since it is based on the cumulative effects of interframe coding in previous frames.

Similar effects can be seen in the *Salesman* sequence, which is more demanding on the H.263 algorithm. Compare the pictures in figure 7.4.3 with those in 7.3.3.

Figure 7.4.3 (i)

(ii)

Figure 7.4.3: Reconstruction of (i) H.263-base and (ii) H.263-annex encoded sequences, Salesman 052. Frame rate = 28.8 kbits/s

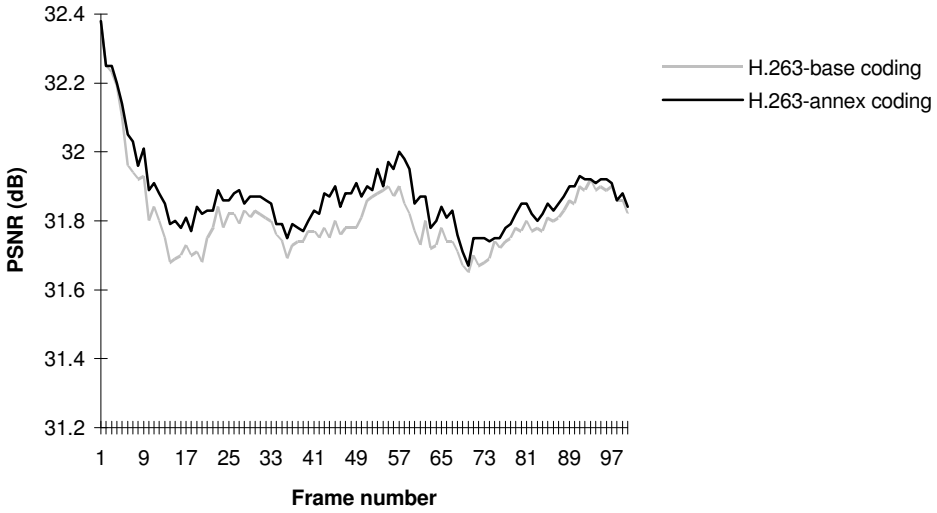


Figure 7.4.4: Reconstructed sequence signal-to-noise ratio for H.263 coding. Salesman sequence. Frame rate = 28.8 kbits/s

Once again, signal-to-noise ratio is improved by the annex options, but this is less when further away from the intraframe coded first frame (figure 7.4.4).

7.5 Video compression

Following on from image quality, it is useful to consider the effectiveness of video coding algorithms in terms of the compression they achieve. It has already been shown that the primary benchmark for image compression is the bitrate, either in terms of bits per second to transmit a sequence, or the relative quantity of bits required to rebuild an individual frame. The latter case will apply in a system which is not constrained by constant bitrate requirements and is suitable for the feature classification technique.

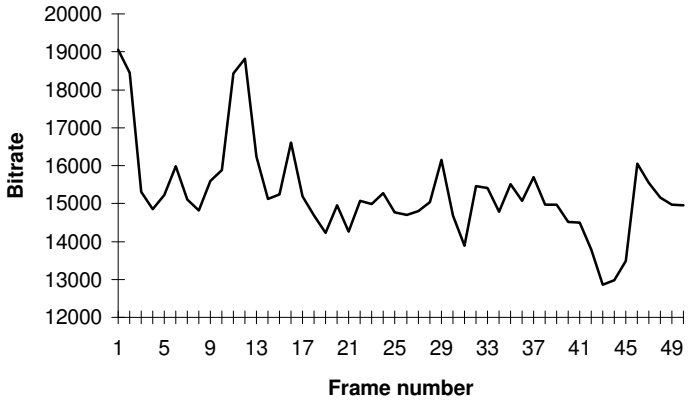


Figure 7.5.1: Bitrate calculations for the Miss America sequence

Taking as an example the *Miss America* sequence, the bitrate is calculated from the assignment of one byte = 8 bits for the encoded interframe data. So for the simplest primitive, *s*:(01), we require two bytes, or 16 bits to produce a sourcebook code. For motion vectors, two bytes are required for the seed pixel location identifier and two bytes for the displacement notation In the case of null vectors, only two bytes are required. For all other runlength codes, a single byte is assumed for each component. It is clear there is quite a lot of waste here, since one byte could describe a runlength

magnitude of up to 255 units, unlikely for most classified features. In the conclusions of chapter 8, it will be suggested that the runlength and vector coding methods could be made more efficient by the use of variable length codes.

Considering the bitrates shown in figure 7.5.1, it shows an initially high overhead caused by the transfer of the first sourcebook. It could be suggested that this is analogous to the increased data rate resulting from intraframe coding in the first frame of an H.263 sequence.

Bitrates for the equivalent H.263 coded sequence are shown in the table of figure 7.5.2.

Video sequence	Mean kbits/s	Mean frames/s	Mean kbits/frame
Miss America Feature-based	382.04	25.00	15.281
Miss America H.263-base	21.18	9.85	2.150
Miss America H.263-annex	20.18	10.96	1.841

Figure 7.5.2: Comparative video bitrates for the Miss America sequence.

The implementation of the H.263 algorithm aimed for a constant bitrate of approximately 28.8 kbits/s. This yielded variable frame rates at 9.85 and 10.96

frames/second, improved by the H.263-annex options. The feature classification technique, on the other hand used reconstruction at a fixed framerate, equal to the source images, of 25 frames/second. The result was a much greater mean bitrate, although spread over all frames, it was just over seven times the H.263 base result.

If these values were to be translated into video compression ratios, it would be seen that for all techniques, the mean compression was still of a useful level. The calculation of image compression is based on a mode-value sub-sampled QCIF image, having 176 × 144 pixels, at eight bits per pixel = 202752 bits. This is then compared with the mean frame bitrates shown in figure 7.5.2.

Video sequence	Mean compression ratio
Miss America Feature-based	13.3:1
Miss America H.263-base	94.3:1
Miss America H.263-annex	110.1:1

Figure 7.5.3: Mean compression ratios for the Miss America sequence

7.6 Information theory

One final technique used to evaluate the effects of image compression and coding is the use of information theory. By considering the range of values occurring in a given

image, the average information per source input, or *entropy*, can be calculated to show the spread of information. A high entropy value shows a large range of data, which would be less efficient to encode than for an image exhibiting a low entropy value. Generally speaking, for a range of pixel values, the entropy is given by:

$$H(\mathbf{z}) = - \sum_{k=1}^{m} P_k \log_2 (P_k) \qquad \text{[Equation 7.1]}$$

where $H(\mathbf{z})$ is the entropy of input source $\mathbf{z}$

P_k is the probability of occurrence of k in the set k=1 to m.

Entropy calculations have been made in two groups. Firstly, we can consider the effect of spatial pre-processing on the image sequences for *Miss America* (figures 7.6.1 and 7.6.2). It can be seen that whilst the entropy profile has remained almost the same, the values have been reduced to approximately 40% of the original, a combined effect of quantisation and mode-value subsampling.

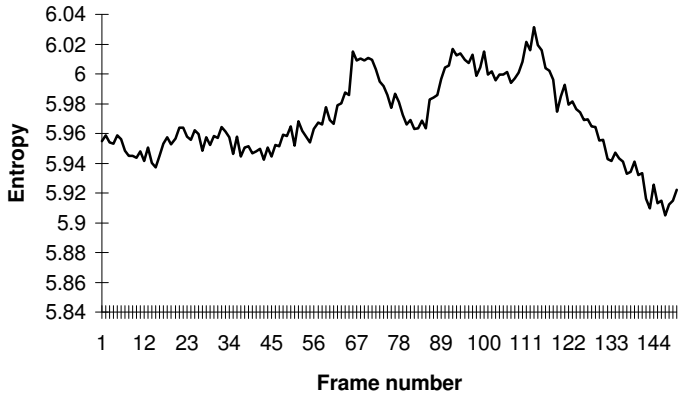


Figure 7.6.1: Frame entropy - Miss America CIF base sequence

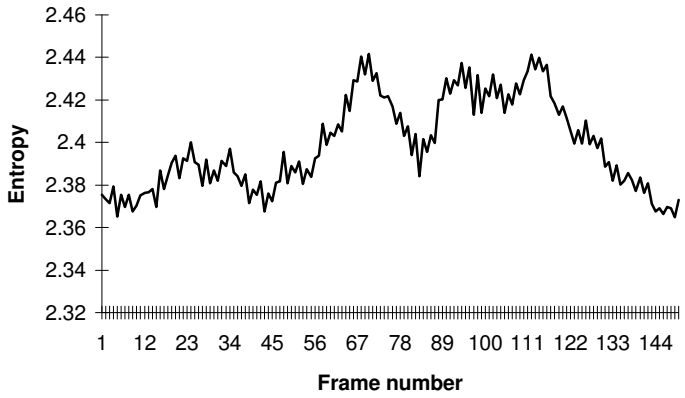


Figure 7.6.2: Frame entropy - Miss America ΔQ=8 QCIF resolution

The second comparison is made between the reconstructed sequence and the effect of error correction (figure 7.6.3).

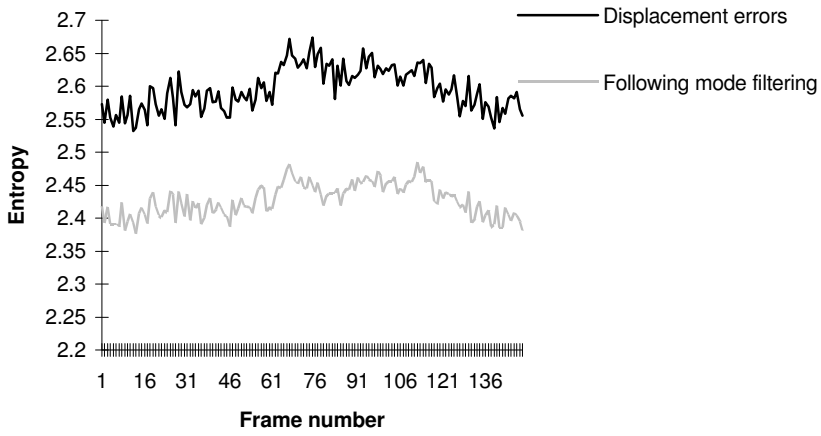


Figure 7.6.3: Frame entropy for the reconstructed Miss America sequence showing the effects of error correction

Whilst entropy values have not re-gained their QCIF input levels, the effect of error correction is useful in removing unwanted artefacts and restoring a low overall entropy. A similar effect can be seen in the reconstructed *Salesman* sequence (figure 7.6.4).

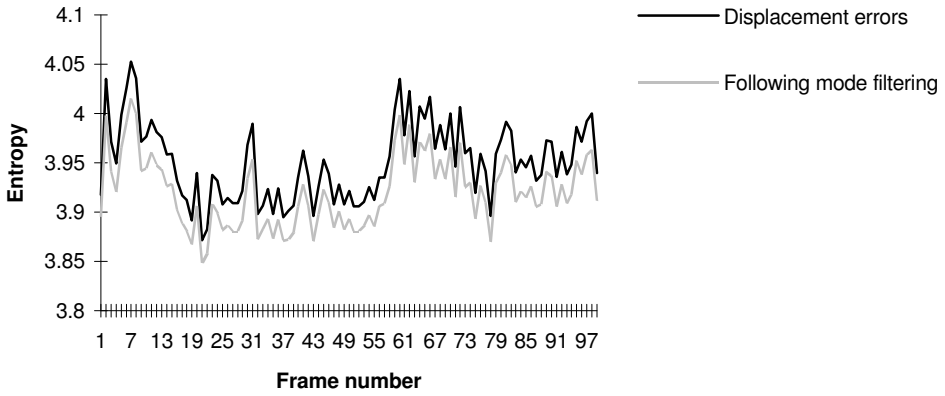


Figure 7.6.4: Frame entropy for the reconstructed Salesman sequence showing the effects of error correction

In this case, the result of error correction is less significant because *Salesman* has a greater degree of interframe detail than *Miss America*. However, both plots show that it is useful to consider entropy in the evaluation of process efficiency.

7.7 Summary

The results presented in this chapter display a range of interesting characteristics observed during the testing of the feature classification algorithm, together with a comparison to a simulated H.263 implementation. In the first instance, it was known that errors were likely to occur in the reconstruction of individual frames in both test sequences. Empirical evidence had suggested that the majority of such errors occurred only on a single pixel basis. The effect of the error correction algorithm, therefore, was simply to extrapolate known error locations from the reconstructed image and then correct their values using the mode or median, calculated from the values of other pixels in the vicinity.

If we consider both the illustrations of reconstructed images and their associated signal-to-noise ratios, it can be seen that there is a marked difference in perceivable quality. For *Miss America*, figure 7.2.1 (ii) shows errors corrected and whilst there is still good contrast between the subject and the plain background, facial details appear blurred. This is an effect of the error correction process, but the success of the algorithm in showing overall motion, at a point in the sequence where it is quite high, indicates that such errors are not to be unexpected in a low-resolution codec. The H.263 implementation of the same sequence is also prone to spatial blurring, caused by low-pass filtering associated with the interpolation function in its half-pixel motion estimation. However, the temporal quality of the sequence is satisfactory and playing a sequence tends to diminish the importance of such spatial defects.

The video compression ratios for the reconstructed *Miss America* sequence are quite variable and whilst a mean ratio of 13.3:1 appears good in its own right, it seems to compare rather unfavourably with the H.263 mean compression values, with a figure of 110.1:1 for the H.263-annex coding. An obvious explanation for these differences is the absence of an efficient coding structure in the feature classification technique. Chapter 2 described the use of Huffman codes in the context of video coding. Given the high quantities of both primitive labels and null or linear motion vectors, further work would be well directed towards the use of variable length coding.

It is also interesting to consider the effects of feature classification and coding on the frame entropy values. Although there is an increase in entropy for the first iteration of reconstruction, use of error correction appears to have the effect of restoring entropy levels close to those observed after spatial pre-processing. This justifies the use of

mode-value filtering in the error correction process, where the new value remains within the set of values normally occurring within the picture.

Chapter 8

Conclusions

Discussion and recommendations for further work

8.1 Introduction

This chapter presents a general view of the results described in this thesis and discusses some of the issues raised by the implementation of the feature classification algorithm.

In the introduction, it was suggested that the low bitrate objectives of video coding and compression would inevitably lead to an element of compromise. It has been shown that an inherent trade-off exists between image quality and compression efficiency. At first glance, this might appear to be unsatisfactory. However, contemporary algorithms for very low bitrate coding seek to sustain a reduced, yet constant, level of quality as a design parameter. Under these circumstances, the results exhibited by the feature classification algorithm are certainly within the bounds of acceptability. It is considered, therefore, that the novel feature classification algorithm, described in this work, contributes a useful development in video coding and displays, in its own right, the basis of an efficient, low resolution technique.

8.2 Overview

The review section, comprising chapters 2 and 3, examined in detail the current state of video coding and compression technology. Particular emphasis was given to the ITU-T recommendations for the general structure of low bitrate video codecs. It was seen that DPCM/DCT coding is an important part of a predictive algorithm which

seeks to extract pixel difference as a method of removing redundancy. The survey then continued to examine different techniques for motion compensation and, in particular, block and model-based coding methods.

It is demonstrated in section 3.4.1 that the full-search block matching motion compensation algorithm, whilst simple, requires a level of computational intensity unsuitable for implementation with personal-computer based software codecs. Since the BMMC technique tends to be used in conjunction with DPCM/DCT coding, as part of hybrid video codec systems, it may be difficult to reconcile the processing overhead involved, unless motion vectors take a much greater role in the coding representation. There have been developments, such as decomposed block segmentation, where different block sizes are allocated to areas of detail. These assist by reducing the overheads, but the fundamental nature of block matching is that blocks represent arbitrary spatial regions and bear no direct relationship to the nature of features occurring in an image.

Model-based coding goes further in resolving this constraint by attempting to associate motion with a known set of feature primitives. It was seen, however, that codebooks are limited in range and consequently, best fits are made to the restricted set of parameters. In view of this, model-based motion compensation requires a complex algorithm which may introduce errors or unsatisfactory quality where a good prediction cannot be made.

Feature classification and coding can be considered as comprising elements of both block and model-based coding. Chapter 5 described an approach to the detection and

identification of image subregions, using some simple predicates of pixel connectivity and set notation. Whilst the classifications that result from boundary tracking the pixel groups do not represent objects we would recognise, they do produce features that have a greater certainty of unique identification in the area where they reside.

The extraction of feature descriptions is assisted by the deliberate introduction of low-resolution sampling and pixel value quantisation. The effect of this is that errors tend to be introduced before processing, so that they do not tend to propagate in the decoder. With fewer subregions to process, the resulting QCIF image provides a good starting point for the feature classification algorithm.

Successful feature classification makes motion vector generation straightforward. It is known that, provided the background is stationary, many features will not be displaced. Direct comparison of interframe seed pixel locations in temporally adjacent sourcebooks generates null vectors, allowing dynamic searching to concentrate on areas subject to motion.

8.3 Results

The processes of spatial subsampling and pixel value quantisation produce some interesting effects. Whilst a bitrate reduction of almost 85% is possible, the resulting QCIF images continue to give an appearance of satisfactory quality, quantified by the measurement of signal-to-noise ratio (figure 4.5.5).

Spatial quality of the reconstructed sequences is also quite acceptable. Figure 7.3.1 shows the nature of displacement errors on *Miss America* and their subsequent correction. Whilst the error filtering process serves to remove some high frequency spatial components at object boundaries, there is a considerable improvement in signal-to-noise ratio, from typically 32dB to 37dB. Considering the spatially pre-processed *Miss America* sequence exhibits a signal-to-noise ratio of about 40dB, the net improvement of approximately 3dB associated with the new algorithm is notable. The *Salesman* sequence, on the other hand, comprises significantly more spatial detail and, whilst pre-processing results in a reduction in quality to 42dB, the effect of displacement error correction is to restore the sequence to about 35dB, a net improvement of 7dB.

However, when comparing these results with those obtained for the H.263 simulations, it appears that the feature classification method provides a comparable reconstruction. For the *Salesman* sequence, a measurement of signal-to-noise ratio shows values of typically 31.8dB for the H.263-annex processing, a reduction of over 3dB with respect to the feature classification process. The comparative spatial quality is clearly visible in figure 7.3.3 and 7.4.3 for the feature classification and H.263 simulations respectively.

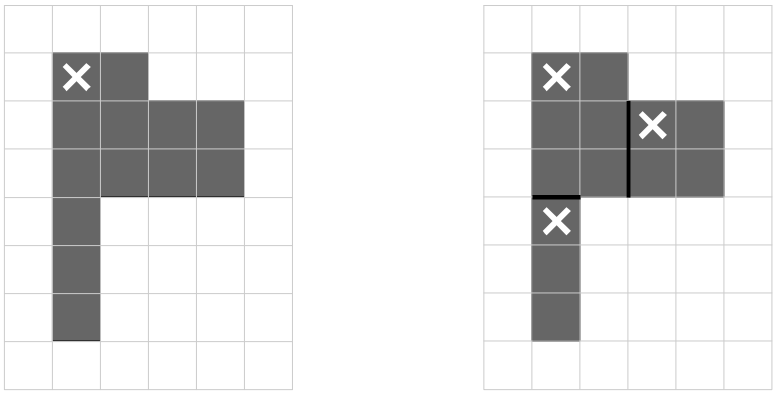
Measurements of bitrate show that the compression ratios for the feature classification process, whilst good, are not as efficient as those calculated for H.263. However, there is clearly potential for further bitrate reduction by a more efficient coding method, discussed in section 8.4. Generally speaking, it is difficult to make some direct comparisons between feature classification and H.263 coding, because the different approaches and criteria have been adopted.

8.4 Recommendations for further work

8.4.1 Adaptive feature classification

This thesis has demonstrated two connected approaches to feature classification. The runlength labels produced by the pixel clustering and boundary detection process can, in most cases, be reduced to a simple primitive label. However, some runlength labels need to be retained, where they describe shapes outside the scope of the feature lookup table.

This requirement could be removed by the introduction of an adaptive technique to subdivide features into known primitives (figure 8.4.1). The result would be a considerable reduction in the parameters needed to describe all subregions and an associated reduction in bitrate.



+2-j1+2-j2-3-j3-1+j6

(i)

1:(11) 2:(:01) 1:(07)

(ii)

Figure 8.4.2: (i) runlength encoded feature description and (ii) its resolution to a set of primitive labels

8.4.2 Variable length coding

Variable length codes are used in predictive codecs as a way of efficiently coding data, according to its frequency of occurrence. Chapter 2 showed how the H.261 algorithm employs Huffman coding to produce shorter datagrams for coefficients which occur most often. This could easily be applied to the feature classification technique for both sourcebook encoding and motion vector generation. If all features could be expressed as primitives, then codes could simply be allocated against their frequency without needing to provide a mechanism for coding runlength labels.

The application of further compression techniques will depend upon the application of the feature classification algorithm. If low bitrates, equivalent to the application of the H.261 and H.263 recommendations, are sought, further work may also have to consider an adjustment in the pre-processing parameters to achieve a further reduction in image quality.

Generally speaking, the use of pre-processing is an important development in conditioning the input image sequence to levels of spatial resolution and pixel magnitude, compatible with the compression activity of the codec. At the input stage, errors are introduced, but there is control over their extent. The novel algorithm described in this thesis uses sampling of both spatial and magnitude parameters, whereas the H.263 technique discards some frames during moments of greater interframe activity. A hybrid system could sample *each* of the three parameters of space, magnitude and time, making a trade-off between the spatial and temporal quality of the reconstructed image sequence.

Further work could also consider more closely the use of reconstruction error correction, using passive techniques to improve visual quality. The method of mode-value filtering outlined in this thesis is only one of many techniques which could be applied, *independent* of the codec, to produce a cosmetic improvement.

8.4.3 Vector generation

With hindsight, it can be seen that many of the errors associated with feature displacement and reconstruction are caused because the algorithm encodes on the basis of where features originated in the *previous* frame. If the encoder were to be genuinely predictive, looking to the *next* frame for its interframe information, there would have been a more significant improvement in quality, without any real increase in processing overhead.

Further work would be well directed to re-considering this approach and implementing the displacement algorithm in such a way as to minimise the interframe errors for which subsequent filtering has proved necessary.

References

1 Griffiths J. M., *ISDN Explained - Worldwide Network and Applications Technology*, John Wiley, 1990.

2 Sekuler R. and Blake R., *Perception*, 3rd Edition, McGraw-Hill 1994, pp 158-178.

3 Thompson J., "European collaboration on Picture Coding Research for 2Mbits/s transmission", *IEEE Trans*, **COM-29**, no. 12, December 1981, pp 2003-4.

4 ITU-T Recommendation† H.261 *Video codec for audiovisual services at p × 64 kbits/s*, July 1990. [† Formerly "CCITT Recommendation"]

5 Carr M.D., "Video codec hardware to realise a new world standard", *British Telecom Technology Journal*, vol. 8, no. 3, July 1990, pp 28-35.

6 Aggoun A., *DPCM Video Signal/Image Processing*, University of Nottingham PhD Thesis, May 1992.

7 Clarke R.J., *Transform Coding of Images*, Academic Press, 1985, pp 111-115.

8 *Ibid*. pp 100-105.

9 *Ibid*. pp 91 - 97.

10 Huffman D.A., "A method for the construction of minimum-redundancy codes", *Proc IRE*, September 1952, pp 1098-1101.

11 Ramteke T., *Networks*, Prentice-Hall 1994, pp 294-299.

12 Morrison D.G., "Standardization by ISO/MPEG of Digital Video Coding for Storage Applications" , in *Audiovisual Telecommunications*, ed. Kenyon N.D. and Nightingale C., Chapman and Hall, 1992, pp 177-193.

13 Hodge W., Mabon S. and Powers J.T. (Jr.), "Video on Demand: Architecture, Systems and Applications", *Journal of the Society of Motion Picture and Television Engineers*, vol. 102, no. 9, 1993, pp 791-803.

14 ISO/IEC "Coding of moving pictures and audio for digital storage media at up to about 1.5 Mbits/s", IS 11172-2.

15 ISO/IEC "Generic coding of moving pictures and associated audio", IS13818-2.

16 ITU-T Draft Recommendation H.263, Study Group XV Document, Q2/15, Expert's Group on Very Low Bitrate Video Telephony, Leidschendam, April 1995.

17 Whybray M.W. and Ellis W., "H.263 - Video Coding Recommendation for PSTN Videophone and Multimedia", *Proc IEE Colloquium on low bit-rate image coding*, Jun 1995, pp 6/1-6/9.

18 Orchard M.T. and Sullivan G.J, "Overlapped Block Motion Compensation: An Estimation-Theoretic Approach", *IEEE Trans on Image Processing*, vol. 3, no. 5, Sep 1994, pp 693-699.

19 Thomas G.A., "Motion and motion estimation", *Image Processing*, ed. Pearson D.E., McGraw-Hill, 1991

20 Thomas G.A., "Television motion measurement for DATV and other applications*", BBC Research Department Report*, no. 1987/11, 1987

21 Budrikis Z.L., "Model approximations to visual spatio-temporal sine-wave threshold data*", Bell System Technical Journal*, vol. 52, n. 9, 1973, pp 1643-67

22 Limb J.O. and Murphy H.A., "Measuring the speed of moving objects from television signals", *IEEE Trans* **COM-23**, no. 4, 1975, pp 474-8

23 Netravali A.N. and Robbins J.D., "Motion compensated television coding: part 1", *Bell System Technical Journal*, vol. 58, no.3, 1979, pp 631-70

24 Mussman H.G., Pirsch P. and Grallert H.J.,"Advances in picture coding", *Proc IEEE*, 1985, vol 73, pp 523-548

25 Jain J.R. and Jain A.K., "Displacement measurement and its application to interframe coding", *IEEE Trans*, **COM-29** (12), pp1799-1808

26 CCITT "Description of reference model 6 (RM6)", SGXV, Specialists Group on Coding for Visual Telephony, Doc 396, 1988.

27 Parke I. and Morrison D.G., "A hardware motion compensator for a videoconferencing codec*", Proc IEE Colloquium on Motion Compensated Image Processing*, 1987, pp 1/1-1/5.

28 De Vos L. *et al*, "VLSI architectures for the full-search block matching algorithm", *Proc ICASSP '89*, **M5.22**, May 1989, pp 1687-1690.

29 Chan M. H., Yu Y.B. and Constantinides A. G., "*Variable size block matching motion compensation with applications to video coding*", Proc IEE-I, vol 137, pt 1, no 4, August 1990, pp 205-212

30 Horowitz S. L. and Pavlidis T., "Picture segmentation by a tree traversal algorithm", *J. Assoc. Comput. Mach*., 23, 1976, pp 368-388

31 Seferidis V. and Ghanbari M., "Generalised block-matching motion estimation using quad-tree structured decomposition", *IEE Proceedings-I*, vol. 141, no. 6, December 1994, pp 446-452

32 Seferidis V. and Ghanbari M., "General approach to block-matching motion estimation", *Optical Engineering*, vol. 32, no. 7, July 1993, pp 1464-1474

33 Lee M. H. and Crebbin G., "Image sequence coding using quadtree-based block-matching motion compensation and classified vector quantisation", *Proc IEE-I*, vol. 141, no. 6, December 1994, pp 453-460

34 Linde Y., Buzo A. and Gray R. M., "An algorithm for vector quantiser design", *IEE Trans. Commun.*, **COM-28**, January 1980, pp 84-95

35 Ramamurthi B. and Gersho A., "Classified vector quantisation of images", *IEEE Trans, Commun.*, **COM-34**, November 1986, pp 1105-1115

36 Lee M. H. and Crebbin G., "Classified vector quantisation with variable block-size DCT models", *Proc IEE-I*, vol. 141, no. 1, pp 39-48

37 Bergeron C. and Dubois E., "Gradient-based algorithms for block-oriented MAP estimation of motion and application to motion-compensated temporal interpolation*", IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 1, March 1991, pp 72-85

38 Welsh W. J., "Model-based coding of videophone images", *IEE Electronics and Communication Engineering Journal*, February 1991, pp 29 - 36

39 Musmann H. G., Hotter M. and Ostermann J., "Object oriented analysis-synthesis coding of moving images", *Image Commun.*, vol. 1, no. 2, October 1989, pp 117-138

40 Newman W. M. and Sproull R., *Principles of interactive computer graphics*, McGraw-Hill, 1981

41 Parke F. I., "Parameterised models for facial animation*", IEEE Computer Graphics and Applications*, November 1982, pp 61-68

42 Ekman P. and Friesen W. V., "Manual for the facial action coding system", Consulting Psychologists Press, Palo Alto, California, 1977

43 Akimoto T., Suenaga Y. and Wallace R. S., "Automatic creation of 3D facial models" *IEEE Computer Graphics and Applications*, September 1993, pp 16-22

44 Nagao M., "Picture recognition and data structure", *Graphic Languages*, ed. Nake and Rosenfeld, North Holland Press, 1972, pp 48-69

45 Watson A.B., "Efficiency of a model human image code", *Journal of the Optical Society of America* (A), vol. 4, no. 12, December 1987, pp 2401-2417

46 Treismann A., "Features and objects in visual processing", *Scientific American*, no. 255, 1986, pp 114-125

47 Rosenfeld A., "Connectivity in Digital Pictures", *Journal of the ACM*, vol. 17, 1970, pp 146-160

48 Rosenfeld A. and Kak A. C., *Digital Picture Processing*, 2nd ed., Academic Press, New York, 1982

49 Pavlidis T., *Structural Pattern Recognition*, Springer-Verlag, Berlin, 1977

50 Kovalevsky V. A., "Topological Foundations of Shape Analysis", *Shape in Picture - Mathematical Description of Shape in Grey-Level Images*, ed. Ying-Lie O., Toet A., et al, NATO ASI Series, 1992, pp 21-36

51 *Guide to Structure Diagrams for Use in Program Design and other Logic Applications*, BS6224, British Standards Institution, 1992

52 Macro A. and Buxton J., *The Craft of Software Engineering*, Addison-Wesley, 1987, pp 203-209

53 WWW Hypertext source `http://www.nta.no/brukere/DVC/` "Digital Video Coding at Telenor R&D", Telenor Research Limited, Norway.

54 Gonzalez R. C. and Woods R.E., *Digital Image Processing*, Addison-Wesley, 1993, p 319

Appendix 1

Test sequence spatial resolutions

1 Introduction

This appendix details comparative sizes and resolutions of the Common Intermediate Format (CIF) and its variants. CIF was adopted by the ITU to overcome standards conversion between PAL and NTSC, has a spatial aspect ration of 4:3 and is encoded from an input frame rate of typically 25 frames per second.

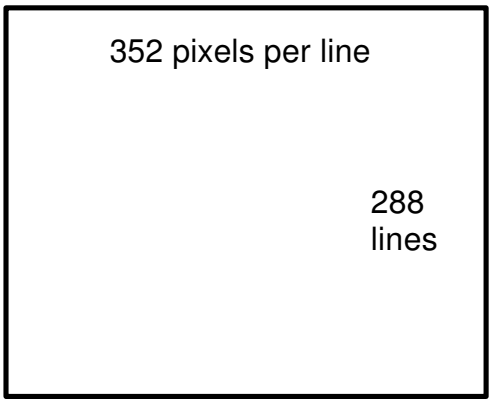
Luminance is sampled at 6.75 MHz - twice the frequency of chrominance (3.375 MHz). The result is that luminance images have twice the spatial resolution of chrominance, shown in figure A1.1.

	Luminance		*Chrominance*	
Format	Pixels/line	Lines/picture	Pixels/line	Lines/picture
CIF	352	288	176	144
QCIF	176	144	88	72
Sub-QCIF	128	96	64	48

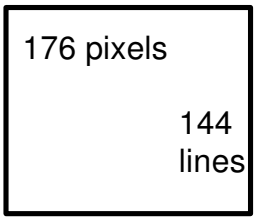
A1.1: Comparative format resolutions

An illustration of the sampling technique is shown in figure 2.8.2. To demonstrate the relative sizes and resolutions involved, figures A1.2-A1.4 show comparisons between the base CIF, QCIF and Sub-QCIF image sizes. Figures A1.5-A1.10 go on to illustrate the different effects of image size and resolution for both luminance (greyscale) and colour images. Sampling for luminance is greater because the human psychovisual perception mechanism is more sensitive to changes in brightness than colour.

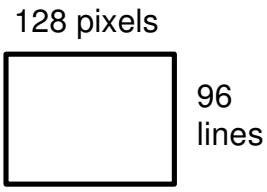
A1.2: Common Intermediate Format (CIF)



A1.3: Quarter Common Intermediate Format (QCIF)



A1.4: Sub-Quarter Intermediate Format (Sub-QCIF)



2 Comparative luminance resolutions

A1.5: Susie 001 CIF

A1.6: Susie 001 QCIF

A1.7: Susie 001 Sub-QCIF

3 Comparative colour resolutions

A1.8: Susie 001 CIF

A1.9: Susie 001 QCIF

A1.10: Susie 001 Sub-QCIF

Appendix 2

Image data format and file handling

1 Introduction

This appendix describes the format of image files used for the simulation of the feature classification algorithm. The related software was mainly compiled from C++ code generated by the author. Additional functions were sourced from the Khoros™ image processing software. Different file formats were required for each implementation and shell scripts were composed to manage the file sequencing, conversion and output to sourcebook and data files.

2 Image file formats

2.1 Portable Bit Maps (PBMs)

The portable bitmap/pixmap (PBM) format was introduced to the public domain by Jeff Poskanzer in 1989 and since then it has become popular as an X-image format. PBM files represent the lowest common denominator monochrome file format. They were originally designed to facilitate bitmap mailing between different types of machines, using mailers that can only handle ASCII characters. The format now serves as the common language of a large family of bitmap conversion filters.

The format of PBM files allows very simple implementation. It has a three line ASCII header comprising: the PBM type; the width and height of the image; the maximum pixel value. There then follows the pixel data, separated by single spaces. For monochrome images, this is simply the luminance value and for colour images, each

adjoining triplet represents the red, green and blue component of the pixel value. This is illustrated in figure A2.1.

Comment lines — PBM type — Image dimensions — Maximum value

```
# Created by viff2pbm
# missa000.pbm
P2
352 288
255
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
 90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90  90
```

Figure A2.1: PBM File format

The example shows part of one of the CIF frames from the Miss America sequence, converted into PBM monochrome (P2). The presence of the header information in a simple ASCII format is very useful as it allows C++ code to read-in the file data

directly to an array, without any further conversion in types. This is demonstrated in the following code extract for `main()`.

```
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>

char * f;
char * o;

// string for the first ID of pgm files
char pgmtype[4];

int xmax, ymax, gmax, count, x, y, pixval;

// arrays to hold line data
int image_array [400][300];

// array to hold block info

int block [10];

main (int argc, char * argv[] ) {

// setup default files
f = argv[1];
o = argv[2];

      ifstream fin (f);
      if (! fin.is_open() ) {
            cerr << "Error with input file \n";
            exit (1);
            }  // if (! fin):

      ofstream fout (o);
      if (! fout.is_open() ) {
            cerr << "\nError with output file \n";
            exit (1);
            } // if (! fout)

// Read in file from input source


      // get the header information

      fin >> pgmtype >> xmax >> ymax >> gmax;

      // read pixel values into array

      for (y=0; y < ymax; y++) {

            for (x=0; x < xmax; x++) {

                  fin >> pixval;
                  image_array[x][y] = pixval;
                  } // read in lines x

            } // read in columns y
```

{ Program body }

```
      // output header

      fout << pgmtype << "\n";
      fout << xmax << " " << ymax << "\n";
      fout << gmax << "\n";


      //output pixel values

      for (y=0; y < ymax; y=y+2) {

            for (x=0; x < xmax; x=x+2){

                  // force <CR> to tidy PBM file
                  if (x % 20 == 0 && x != 0) fout << "\n";
                  fout << image_array[x][y] << " ";


                  } // for output of pixels
            } // for output of lines

} // main
```

In the simulations, PBM files were passed without comment lines. The example shows how a string is allocated for the header PBM identifier and integers set for the header values `xmax`, `ymax` and `gmax` (the maximum greyscale value). The `xmax` and `ymax` values are passed to a loop to read the remaining PBM pixel values into an array, directly matching the *x, y* positions of the image.

For writing, the header, `xmax`, `ymax` and `gmax` are written to a new file, with appropriate carriage returns to keep the header format. The loop then goes through the image array and writes the processed values to the output file, separated by a single space. This example comes from the `main()` body of a program, where file parameters are passed from shell command-line arguments in `main (int argc, char * argv[] )`. The first argument `f` is set to the `ifstream fin` parameter used to designate the source file and the second argument `o` is used to designate the

`ofstream` destination file. A check can be made to ensure the file arguments are correctly presented and comprised the `.ppm` file extension.

This format was most useful and the direct relationship between image and array value locations allows simple processing of subregions. Given a specific location *(x, y)*, local calculations for adjacent horizontal and vertical pixels are made using *(x*-1*, y)(x*+1*, y)* and *(x, y*-1*)(x, y*+1*)*, respectively.

2.2 Visualisation/Image File Format (VIFF)

The VIFF format is an advanced file structure designed to facilitate the interchange of data, primarily within the functions of the Khoros system, but also between researchers who want to exchange data and information. VIFF was designed to be comprehensive, covering the many applications of image processing that require different information about an image. The VIFF header comprises information necessary for a component application to interpret the data and perform basic error checking. The VIFF data structure has been applied to one and two-dimensional data processing applications, as well as three-dimensional visualisation. There is currently spare capacity in the 1024 byte header to extend the data field contained in a VIFF file.

The 1024 byte header is followed by map(s), location data and then image data. The header also information identifying the header format used. The file format (on disk) or data structure (while in memory) have evolved to be directly compatible, in a way analogous to the PBM and array data format previously mentioned. The data may be

an image in the normal sense, or sets of vectors. The difference is indicated by carefully examining the values of the various fields.

The fields can be divided into five categories:

- administration - file management information;
- data storage - how data is stored, but not how it is interpreted;
- location data - describes the spatial location of the data (optional);
- data mapping - describes how the stored data should be mapped or interpreted;
- colour space - in the case of images, indicates what co-ordinate space and model is being used.

A full explanation of the VIFF format can be found in the Khoros Manual [liv], volume II, chapter 1. In general the it is a format which contains a lot more than the simple image data. For the design of a large image processing system, it is useful to have additional information and file, image and spatial parameters, all held in the same location.

This explanation of VIFF is included, since some Khoros functions were used during the simulation of the feature classification.

3 File handling

Image sequences were stored as sets of individual frame files. This made the process of interframe manipulation quite straightforward, as a relationship existed between the temporal location of a frame and its position in the file set.

Image sequence files were held in a specific directory in the UNIX™ filestore. Each had a common identification in its filename, with a numeric separator determining the temporal sequence. In the case of Miss America, files were stored under `missa(frame_number).vif`, using the compressed VIFF format, as shown in the listing below:

```
   4 drwxrwxrwx  2 nwg          4096 Aug 14 16:18 .
   1 drwxrwxrwx 23 nwg           512 Jun 17 20:09 ..
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1000.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1001.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1002.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1003.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1004.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1005.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1006.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1007.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1008.vif
 112 -rwx------  1 nwg        104704 Sep  8  1993 missa1009.vif
```

Following processing, the initial identifier is changed, but the frame number is retained. UNIX shell script was developed to control file handling in the simulation process. Consider the example listed below, for the spatial pre-processing algorithm:

```
#!/bin/sh

for I in `ls missa10*.vif`
do
      FILENO=`echo $I | awk -Fmissa '{print $2}'| awk -F. '{print
$1}'`
        echo "Processing file $FILENO"

      viff2pbm -i missa$FILENO.vif -o missa$FILENO.pbm
      quant missa$FILENO.pbm missaq$FILENO.pbm 8
      cifmode3 missaq$FILENO.pbm missaqcif$FILENO.pbm
      pbm2viff -i missaqcif$FILENO.pbm -o missaqcif$FILENO.vif
      vconvert -i missaqcif$FILENO.vif -o missaqcif$FILENO.vif -t
      "byte"
      rm *.pbm

done
```

This example shows how all files matching the wildcard `missa10*.vif` are listed. The frame number is extracted from the listing by use of the `awk` command, used for non-interactive text processing. This is set as an environment variable `$FILENO`. At

each stage of processing, files are created having different prefixes, but retaining their position specified by the current value of `$FILENO`. The script loops until all files in the set `missa*.vif` have been processed. In this case, the source files are in VIFF format, they are converted to PBM, quantised to a step interval of 8 grey levels, mode value subsampled and then returned to VIFF format. The resulting files will therefore be:

```
112 -rwx------  1 nwg        missaqcif1000.vif
112 -rwx------  1 nwg        missaqcif1001.vif
112 -rwx------  1 nwg        missaqcif1002.vif
112 -rwx------  1 nwg        missaqcif1003.vif
112 -rwx------  1 nwg        missaqcif1004.vif
112 -rwx------  1 nwg        missaqcif1005.vif
112 -rwx------  1 nwg        missaqcif1006.vif
112 -rwx------  1 nwg        missaqcif1007.vif
112 -rwx------  1 nwg        missaqcif1008.vif
112 -rwx------  1 nwg        missaqcif1009.vif
```

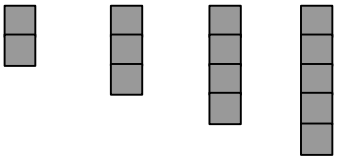
The use of shell script processing made file handling very simple and allowed a fully modular approach to programming in the context of image sequence file sets.

Appendix 3

Primitive feature look-up table

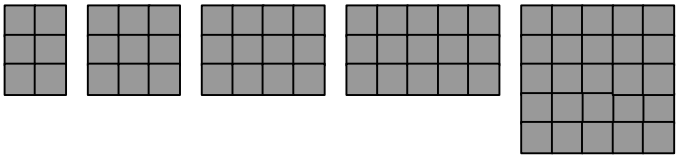
Strips

01 02 03 04 05



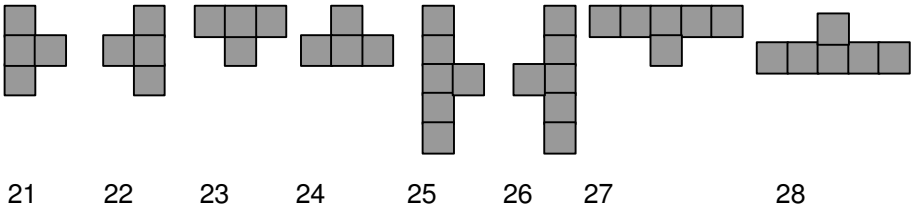
06 07 08 09

Rectangles

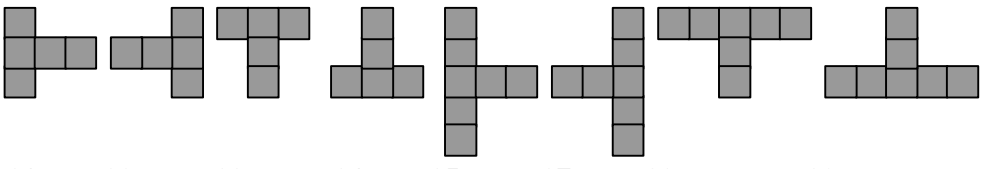


11 12 13 14 15

T-shapes

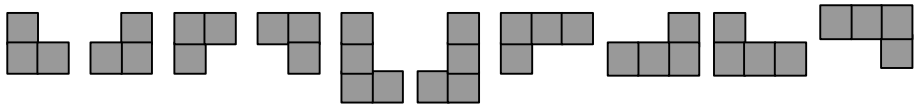


21 22 23 24 25 26 27 28

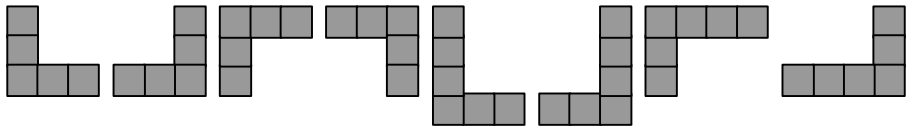


31 32 33 34 35 37 38 39

L-shapes

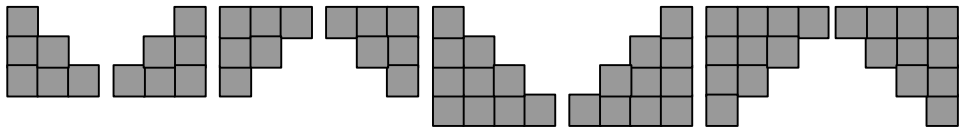


40 41 42 43 44 45 46 47 48 49



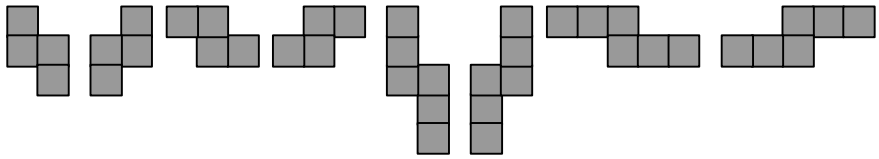
50 51 52 53 54 55 56 57

Steps (diagonal edges)



60 61 62 63 64 65 66 67

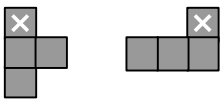
Single steps



70 71 72 73 74 75 76 77

Notation

In all cases, the seed pixel is the first to be encountered in a scan from top-left to bottom-right, e.g.:



i Griffiths J. M., *ISDN Explained - Worldwide Network and Applications Technology*, John Wiley, 1990.

ii Sekuler R. and Blake R., *Perception*, 3rd Edition, McGraw-Hill 1994, pp 158-178.

iii Thompson J., "European collaboration on Picture Coding Research for 2Mbits/s transmission", *IEEE Trans*, **COM-29**, no. 12, December 1981, pp 2003-4.

iv ITU-T Recommendation[†] H.261 *Video codec for audiovisual services at p* × *64 kbits/s*, July 1990. [† Formerly "CCITT Recommendation"]

v Carr M.D., "Video codec hardware to realise a new world standard", *British Telecom Technology Journal*, vol. 8, no. 3, July 1990, pp 28-35.

vi Aggoun A., *DPCM Video Signal/Image Processing*, University of Nottingham PhD Thesis, May 1992.

vii Clarke R.J., *Transform Coding of Images*, Academic Press, 1985, pp 111-115.

viii *Ibid*. pp 100-105.

ix *Ibid*. pp 91 - 97.

x Huffman D.A., "A method for the construction of minimum-redundancy codes", *Proc IRE*, September 1952, pp 1098-1101.

xi Ramteke T., *Networks*, Prentice-Hall 1994, pp 294-299.

xii Morrison D.G., "Standardization by ISO/MPEG of Digital Video Coding for Storage Applications" , in *Audiovisual Telecommunications*, ed. Kenyon N.D. and Nightingale C., Chapman and Hall, 1992, pp 177-193.

xiii Hodge W., Mabon S. and Powers J.T. (Jr.), "Video on Demand: Architecture, Systems and Applications", *Journal of the Society of Motion Picture and Television Engineers*, vol. 102, no. 9, 1993, pp 791-803.

xiv ISO/IEC "Coding of moving pictures and audio for digital storage media at up to about 1.5 Mbits/s", IS 11172-2.

xv ISO/IEC "Generic coding of moving pictures and associated audio", IS13818-2.

xvi ITU-T Draft Recommendation H.263, Study Group XV Document, Q2/15, Expert's Group on Very Low Bitrate Video Telephony, Leidschendam, April 1995.

xvii Whybray M.W. and Ellis W., "H.263 - Video Coding Recommendation for PSTN Videophone and Multimedia", *Proc IEE Colloquium on low bit-rate image coding*, Jun 1995, pp 6/1-6/9.

xviii Orchard M.T. and Sullivan G.J, "Overlapped Block Motion Compensation: An Estimation-Theoretic Approach", *IEEE Trans on Image Processing*, vol. 3, no. 5, Sep 1994, pp 693-699.

xix Thomas G.A., "Motion and motion estimation", *Image Processing*, ed. Pearson D.E., McGraw-Hill, 1991

xx Thomas G.A., "Television motion measurement for DATV and other applications", *BBC Research Department Report*, no. 1987/11, 1987

xxi Budrikis Z.L., "Model approximations to visual spatio-temporal sine-wave threshold data", *Bell System Technical Journal*, vol. 52, n. 9, 1973, pp 1643-67

xxii Limb J.O. and Murphy H.A., "Measuring the speed of moving objects from television signals", *IEEE Trans* **COM-23**, no. 4, 1975, pp 474-8

xxiii Netravali A.N. and Robbins J.D., "Motion compensated television coding: part 1", Bell System Technical Journal, vol. 58, no.3, 1979, pp 631-70

xxiv Mussman H.G., Pirsch P. and Grallert H.J.,"Advances in picture coding", *Proc IEEE*, 1985, vol 73, pp 523-548

xxv Jain J.R. and Jain A.K., "Displacement measurement and its application to interframe coding", *IEEE Trans*, **COM-29** (12), pp1799-1808

xxvi CCITT "Description of reference model 6 (RM6)", SGXV, Specialists Group on Coding for Visual Telephony, Doc 396, 1988.

xxvii Parke I. and Morrison D.G., "A hardware motion compensator for a videoconferencing codec*", Proc IEE Colloquium on Motion Compensated Image Processing*, 1987, pp 1/1-1/5.

xxviii De Vos L. et al, "VLSI architectures for the full-search block matching algorithm", *Proc ICASSP '89*, **M5.22**, May 1989, pp 1687-1690.

xxix Chan M. H., Yu Y.B. and Constantinides A. G., "*Variable size block matching motion compensation with applications to video coding*", Proc IEE-I, vol 137, pt 1, no 4, August 1990, pp 205-212

xxx Horowitz S. L. and Pavlidis T., "Picture segmentation by a tree traversal algorithm", *J. Assoc. Comput. Mach*., 23, 1976, pp 368-388

xxxi Seferidis V. and Ghanbari M., "Generalised block-matching motion estimation using quad-tree structured decomposition", *IEE Proceedings-I*, vol. 141, no. 6, December 1994, pp 446-452

xxxii Seferidis V. and Ghanbari M., "General approach to block-matching motion estimation", *Optical Engineering*, vol. 32, no. 7, July 1993, pp 1464-1474

xxxiii Lee M. H. and Crebbin G., "Image sequence coding using quadtree-based block-matching motion compensation and classified vector quantisation", *Proc IEE-I*, vol. 141, no. 6, December 1994, pp 453-460

xxxiv Linde Y., Buzo A. and Gray R. M., "An algorithm for vector quantiser design", *IEE Trans. Commun.*, **COM-28**, January 1980, pp 84-95

xxxv Ramamurthi B. and Gersho A., "Classified vector quantisation of images", *IEEE Trans, Commun.*, **COM-34**, November 1986, pp 1105-1115

xxxvi Lee M. H. and Crebbin G., "Classified vector quantisation with variable block-size DCT models", *Proc IEE-I*, vol. 141, no. 1, pp 39-48

xxxvii Bergeron C. and Dubois E., "Gradient-based algorithms for block-oriented MAP estimation of motion and application to motion-compensated temporal interpolation*", IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 1, March 1991, pp 72-85

xxxviii Welsh W. J., "Model-based coding of videophone images", *IEE Electronics and Communication Engineering Journal*, February 1991, pp 29 - 36

xxxix Musmann H. G., Hotter M. and Ostermann J., "Object oriented analysis-synthesis coding of moving images", *Image Commun.*, vol. 1, no. 2, October 1989, pp 117-138

xl Newman W. M. and Sproull R., "Principles of interactive computer graphics", McGraw-Hill, 1981

xli Parke F. I., "Parameterised models for facial animation", *IEEE Computer Graphics and Applications*, November 1982, pp 61-68

xlii Ekman P. and Friesen W. V., "Manual for the facial action coding system", Consulting Psychologists Press, Palo Alto, California, 1977

xliii Akimoto T., Suenaga Y. and Wallace R. S., "Automatic creation of 3D facial models" *IEEE Computer Graphics and Applications*, September 1993, pp 16-22

xliv Nagao M., "Picture recognition and data structure", *Graphic Languages*, ed. Nake and Rosenfeld, North Holland Press, 1972, pp 48-69

xlv Watson A.B., "Efficiency of a model human image code", *Journal of the Optical Society of America* (A), vol. 4, no. 12, December 1987, pp 2401-2417

xlvi Treismann A., "Features and objects in visual processing", *Scientific American*, no. 255, 1986, pp 114-125

xlvii Rosenfeld A., "Connectivity in Digital Pictures", *Journal of the ACM*, vol. 17, 1970, pp 146-160

xlviii Rosenfeld A. and Kak A. C., *Digital Picture Processing*, 2nd ed., Academic Press, New York, 1982

xlix Pavlidis T., *Structural Pattern Recognition*, Springer-Verlag, Berlin, 1977

l Kovalevsky V. A., "Topological Foundations of Shape Analysis", *Shape in Picture - Mathematical Description of Shape in Grey-Level Images*, ed. Ying-Lie O., Toet A., et al, NATO ASI Series, 1992, pp 21-36

li *Guide to Structure Diagrams for Use in Program Design and other Logic Applications*, BS6224, British Standards Institution, 1992

lii Macro A. and Buxton J., *The Craft of Software Engineering*, Addison-Wesley, 1987, pp 203-209

liii WWW Hypertext Source `http://www.nta.no/brukere/DVC/` Telenor Research Limited, Norway.

Reference

liv *Khoros User's Manual*, Release 1.0, The Khoros Group, University of New Mexico, USA