

**School of Electrical and Computer Engineering**

**Signal Compression for Digital Television**

by

**Mr. Huy So Truong**


**This thesis is presented as part of the requirements for**

**the award of the Degree of Master of Engineering**

of

**Curtin University of Technology**

**April, 99**

PROJECT DOCUMENTATION SHEET				
TITLE				
<p style="text-align: center;">Signal Compression for Digital Television</p>				
AUTHOR		<p style="text-align: center;">Huy So Truong</p>		
DATE	April 18, 1999	SUPERVISOR	Dr. Stephen C. Y. Ho	
DEGREE	Master of Engineering	OPTION	Electrical	
<p><b>ABSTRACT</b></p> <p>Still image and image sequence compression plays an important role in digital television. For still image compression, Block Adaptive Classified Vector Quantisation (BACVQ) algorithm has been developed which combines variable block size coding and CVQ. Segmentation and classification decisions are made based on spatial and temporal domain criteria. For image sequence compression, an adaptive coding technique has been developed which divides an image sequence into groups of pictures using adaptive scene segmentation before BACVQ and variable block size motion compensated predictive coding techniques are applied. Both compression techniques have demonstrated good performance with high compression gain and reconstruction quality.</p> <p>A small scale parallel DSP system has been constructed with a 486DX33 IBM/PC serving as a master processor and two DSP (PC-32) cards as parallel processors. Dualport memory has been used to provide high speed interconnection and its access is arbitrated by hardware semaphore and mailbox messaging. The host PC shall dynamically distribute processing tasks to idle PC-32 cards for parallel processing. Optimal parallel operation has been achieved during 4x4 CVQ processing, variable block size motion compensation and residual frame processing.</p>				
<p><b>INDEXING TERMS</b></p> <p>Block Adaptive Classified VQ, intraframe coding, interframe coding, motion compensated predictive coding, variable block size motion estimation, progressive motion estimation, adaptive scene segmentation, quadtree segmentation, parallel still image compression, parallel image sequence compression, statistical redundancy, spatial and temporal redundancy.</p>				
		GOOD	AVERAGE	POOR
TECHNICAL WORK				
REPORT PRESENTATION				
EXAMINER	CO-EXAMINER			

Mr. Huy So Truong

---

18 April, 1999

Prof. T. Smith  
Dean of Engineering  
Curtin University of Technology  
Kent Street,  
Bentley, W.A. 6120

Dear Sir,

Please accept this thesis, entitled "Signal Compression for Digital Television", as part of the requirements for the Degree of Master of Engineering (Electrical Engineering).

Yours faithfully,

---

Mr. Huy So Truong

## Synopsis

Still image and image sequence compression plays an important role in the development of digital television. Although various still image and image sequence compression algorithms have already been developed, it is very difficult for them to achieve both compression performance and coding efficiency simultaneously due to the complexity of the compression process itself. As a results, improvements in the forms of hybrid coding, coding procedure refinement, new algorithms and even new coding concepts have been constantly tried, some offering very encouraging results.

In this thesis, Block Adaptive Classified Vector Quantisation (BACVQ) has been developed as an alternative algorithm for still image compression. It is found that BACVQ achieves good compression performance and coding efficiency by combining variable block-size coding and classified VQ. Its performance is further enhanced by adopting both spatial and transform domain criteria for the image block segmentation and classification process. Alternative algorithms have also been developed to accelerate normal codebook searching operation and to determine the optimal sizes of classified VQ sub-codebooks.

For image sequence compression, an adaptive spatial/temporal compression algorithm has been developed which divides an image sequence into smaller groups of pictures (GOP) using adaptive scene segmentation before BACVQ and variable block-size motion compensated predictive coding are applied to the intraframe and interframe coding processes. It is found the application of the proposed adaptive scene segmentation algorithm, an alternative motion estimation strategy and a new progressive motion estimation algorithm enables the performance and efficiency of the compression process to be improved even further.

Apart from improving still image and image sequence compression algorithms, the application of parallel processing to image sequence compression is also investigated. Parallel image compression offers a more effective approach than the sequential counterparts to accelerate the compression process and bring it closer to

real-time operation. In this study, a small scale parallel digital signal processing platform has been constructed for supporting parallel image sequence compression operation. It consists of a 486DX33 IBM/PC serving as a master processor and two DSP (PC-32) cards as parallel processors. Because of the independent processing and spatial arrangement natures of most image processing operations, an effective parallel image sequence compression algorithm has been developed on the proposed parallel processing platform to significantly reduce the processing time of the proposed parallel image compression algorithms.

## Acknowledgments

I would like to acknowledge the support and encouragement of my supervisor, Dr. Stephen C. Y. Ho. His expertise in the area of digital signal processing and image processing has provided me a significant insight into many issues encountered throughout the course of my study. I am also grateful for his endorsement and arrangement for me to obtain financial assistance from Australian Postgraduate Award scholarship and funding for acquiring parallel DSP hardware from Curtin University; without which my Master study would not be possible.

I would like to acknowledge financial supports from my School for me to attend several International Conferences. Staff members at my School have also offered valuable assistance throughout the course of my study. In particular, I would like to thank Mrs Ingrid Hastie and Mrs. Susan Summers for their administrative work.

Special thanks to Dr. Kheong Chee with whom I have had so many exiting and enjoyable discussions on various topics of image compression due to his enthusiasm and expertise in the field. I am also grateful for his valuable advice on how to organising research materials and his assistance in utilising HP workstations in the Image Technology lab for conducting my research activities.

Finally, I would like to thank Mr. Sarfraz Khokhar and Mr Yukihiro Nakazato for their friendship and interest in my work. I would also like to thank all of my family members. Without their continuing supports and encouragement, my Master study would have been very difficult.

## Table of Contents

Synopsis.....	i
Acknowledgments.....	iii
Table of Contents .....	iv
List of Figures.....	viii
List of Tables.....	x
List of Abbreviations.....	xi
Chapter 1: Introduction .....	1
1.1 Introduction.....	1
1.2 The growing importance of digital technology.....	2
1.3 The importance of digital image compression .....	4
1.4 Thesis contributions .....	5
1.5 Thesis organisation.....	6
Chapter 2: Still Image Compression.....	9
2.1 Introduction.....	9
2.2 Transform coding.....	12
2.2.1 Discrete Fourier transform (DFT):.....	17
2.2.2 Discrete Hartley transform (DHT): .....	17
2.2.3 Karhunen-Loeve transform (KLT): .....	17
2.2.4 Discrete Cosine transform (DCT):.....	18
2.3 Subband and wavelet coding.....	19
2.4 Vector quantisation .....	24
2.4.1 Overview.....	24
2.4.2 Vector quantisation .....	27
2.4.3 Shannon's rate-distortion theorem.....	29
2.4.4 Optimality conditions for vector quantisation .....	31
2.4.5 Nearest Neighbour Quantisation.....	32
2.4.6 Structurally constrained vector quantisers .....	33
2.4.7 Fast nearest neighbour encoding techniques.....	42
2.4.8 Vector quantisation codebook generation.....	46
2.4.9 Generalised Lloyd algorithm.....	49

2.5 Proposed block adaptive classified vector quantisation.....	52
2.5.1 Variable block-size coding with adaptive block segmentation.....	52
2.5.2 Classified VQ.....	55
2.5.3 Image sub-block classification.....	56
2.5.4 BACVQ Codebook design.....	60
2.5.5 Bit allocation.....	64
2.5.6 Simulation results.....	65
2.6 Huffman coding and arithmetic coding.....	71
2.7 Summary.....	72
Chapter 3: Image Sequence Compression.....	76
3.1 Introduction.....	76
3.2 Human Visual System.....	82
3.3 Scene Segmentation.....	84
3.4 Motion compensated coding.....	87
3.4.1 Motion compensated predictive coding.....	87
3.4.2 Motion compensated interpolative coding.....	92
3.4.3 Motion estimation strategy.....	94
3.5 Adaptive coding techniques.....	100
3.6 Basic coding structures of MPEG.....	103
3.7 Proposed adaptive image sequence compression algorithm.....	104
3.7.1 Adaptive scene segmentation.....	104
3.7.2 Variable block-size motion compensation.....	108
3.7.3 Progressive motion vector estimation.....	110
3.7.4 Block adaptive classified vector quantisation (BACVQ).....	113
3.7.5 Simulation results.....	116
3.8 Summary.....	120
Chapter 4: Parallel Processing.....	132
4.1 Introduction.....	132
4.2 Parallel processing architectures.....	136
4.2.1 Single instruction stream-single data stream (SISD).....	136
4.2.2 Single instruction stream-multiple data stream (SIMD).....	137
4.2.3 Multiple-instruction stream-Single data stream (MISD).....	138
4.2.4 Multiple-instruction stream-Multiple data stream (MIMD).....	139



4.3	Connectivity in parallel processing systems .....	141
4.4	Processing synchronisation .....	142
4.4.1	<i>Synchronisation in shared-memory processing</i> .....	142
4.4.2	<i>Synchronisation in message passing computation</i> .....	145
4.5	Partitioning and scheduling .....	146
4.6	Processing performance .....	147
4.7	Parallel processing system design issues .....	150
4.7.1	<i>Message-passing MIMD</i> .....	150
4.7.2	<i>Shared-memory MIMD</i> .....	152
4.8	Summary .....	153
 Chapter 5: Parallel Image Sequence Compression .....		156
5.1	Introduction .....	156
5.2	Parallel digital signal processing platforms .....	158
5.2.1	<i>Overview</i> .....	158
5.2.2	<i>Texas Instruments TMS320C80 MVP DSP system</i> .....	161
5.2.3	<i>Sonitech SPIRIT-40 system</i> .....	163
5.2.4	<i>Atlanta Signal Processors ELP DSP platform</i> .....	163
5.2.5	<i>Innovative Integration PC32 DSP system</i> .....	164
5.2.6	<i>Other parallel DSP systems</i> .....	165
5.3	Hardware configuration of the prototype parallel DSP platform .....	165
5.4	Basic operations of the prototype parallel DSP platform .....	169
5.5	Dualport memory operation .....	171
5.6	Semaphore operation .....	175
5.7	Parallel processing algorithm and experimental results .....	177
5.7.1	<i>Parallel HOD computation</i> .....	177
5.7.2	<i>Parallel still image compression</i> .....	182
5.7.3	<i>Parallel image sequence compression</i> .....	187
5.8	Summary .....	201
 Chapter 6: Conclusions and Recommendations .....		206
6.1	BACVQ algorithm .....	206
6.1.1	<i>Algorithm overview</i> .....	206
6.1.2	<i>Coder Evaluation</i> .....	207
6.1.3	<i>Recommendations</i> .....	208

6.2 Adaptive spatial/temporal image sequence compression.....	209
6.2.1 Algorithm overview .....	209
6.2.2 Coder Evaluation .....	211
6.2.3 Recommendations.....	212
6.3 Parallel image sequence compression.....	213
6.3.1 Algorithm overview .....	213
6.3.2 Coder evaluation .....	216
6.3.3 Recommendations.....	218
References .....	220
Appendix A: Conference Paper #1 .....	226
Appendix B: Conference Paper #2.....	238
Appendix C: Conference Paper #3.....	244
Appendix D: Still Image Compression Flowchart.....	250
Appendix E: Image Sequence Compression Flowchart .....	267
Appendix F: Parallel Processing Arrangements .....	272
Appendix G: Boot loading operations .....	275
Appendix H: Influence of high level languages on processing speed..	277
Appendix I: Samples of Parallel Processing Results.....	279

## List of Figures

Figure 2.1: Block diagram of transform coding.....	14
Figure 2.2: A typical zigzag scanning pattern for transformed coefficients quantisation.....	15
Figure 2.3: A typical subband decomposition of still images.....	20
Figure 2.4: Block diagram of a subband coding system.....	22
Figure 2.5: Block diagram of vector quantisation coding systems.....	25
Figure 2.6: Typical codebook structure of TSVQs.....	34
Figure 2.7: Block diagram of classified VQ.....	36
Figure 2.8: Block diagram of transform VQ.....	38
Figure 2.9: Block diagram of multi-stage VQ.....	39
Figure 2.10: Triangle inequality codebook search method.....	45
Figure 2.11: Adaptive block segmentation structure.....	53
Figure 2.12: Dynamic range variation of 8x8 smooth blocks for image 'Lenna'.....	53
Figure 2.13: Illustration diagram of BACVQ.....	54
Figure 2.14: Classified VQ operation.....	56
Figure 2.15: An original and transformed high texture block.....	58
Figure 2.16: Various classes for blocks containing edges.....	58
Figure 2.17: Codebook structures for BACVQ.....	60
Figure 2.18: Partial distortion among various edge-like codebooks.....	61
Figure 2.19: Partial codebook search range.....	64
Figure 2.20: Bit allocation for BACVQ code words.....	65
Figure 2.21: Various coding results for 'Lenna' image.....	68
Figure 2.22: Sample results of BACVQ coding process (set #1).....	70
Figure 2.23: Sample results of BACVQ coding process (set #2).....	71
Figure 3.1: Typical temporal DPCM coding.....	79
Figure 3.2: General coding structure of a motion compensated DPCM coder.....	89
Figure 3.3: Operation principle of temporal interpolation.....	93
Figure 3.4: Predicting missing image samples using bidirectional temporal interpolation.....	93
Figure 3.5: Window search range for motion estimation.....	96
Figure 3.6: Example of 2D logarithmic search strategy.....	97
Figure 3.7: Example of modified 2D search strategy.....	97
Figure 3.8: Example of 3-step search strategy.....	98
Figure 3.9: Example of modified one-at-a-time search strategy.....	98

Figure 3.10: Basic GOP frame structure for MPEG coding without B-frame inclusion.....	103
Figure 3.11: Basic GOP frame structure for MPEG coding with B-frame inclusion.....	104
Figure 3.12: Histogram of difference for the test sequence.....	105
Figure 3.13: Magnitude of HOD variation of the test sequence.....	106
Figure 3.14: Dynamic GOP structure .....	107
Figure 3.15: Flowchart for the proposed image sequence compression algorithm .....	109
Figure 3.16: Typical search locations for the proposed motion estimation strategy.....	111
Figure 3.17: Conventional and progressive motion estimation strategies.....	113
Figure 3.18: Typical pixel value distribution of original and residual frames .....	116
Figure 3.19: Block diagram of the proposed image sequence compress algorithm .....	117
Figure 3.20: Bit allocation for coding motion vector fields .....	117
Figure 3.21: Bit rate performance on the test sequence.....	118
Figure 3.22: PSNR performance on the test sequence.....	118
Figure 3.23: Sample pictures for the "Salesman" test image sequence.....	124
Figure 3.24: Sample pictures for the "Claire" test image sequence.....	125
Figure 3.25: Sample pictures for the "Miss American" test image sequence.....	126
Figure 3.26: Sample pictures for the "Susie" test image sequence.....	127
Figure 3.27: Sample pictures for the "Caltrain" test image sequence .....	128
Figure 3.28: Sample pictures for the "Table Tennis" test image sequence .....	129
Figure 3.29: Sample pictures for the "Football" test image sequence .....	130
Figure 3.30: Sample pictures for the "Flower Garden" test image sequence .....	131
Figure 4.1: Typical connectivity for a three processing element system .....	141
Figure 4.2: Physical connections of a 3-D hypercube machine.....	150
Figure 4.3: Pseudo code for message based synchronisation .....	152
Figure 5.1: Typical processing arrangement of C80-based development system. ....	162
Figure 5.2: Block diagram of the prototype parallel DSP platform.....	166
Figure 5.3: Dualport memory mapping between the host PC and the PC32 .....	172
Figure 5.4: Mailbox data structure and its relative position in dualport memory. ....	173
Figure 5.5: Image data distribution for the moderate parallel processing condition.....	179
Figure 5.6: Parallel HOD computation algorithm.....	180
Figure 5.7: The effect of image data transfer overheads on parallel HOD computation.....	182
Figure 5.8: Execution timing for 4x4 image sub-block intraframe processing .....	185
Figure 5.9: Residual block structure and returned codeword index arrangement. ....	194

## List of Tables

Table 2.1: Adaptive allocation of codebook sizes to various edge codebooks.....	67
Table 2.2: Results of coding non-training images with BACVQ .....	67
Table 2.3: Results of coding training images with BACVQ .....	67
Table 2.4: Results obtained from various BACVQ for non-training image "Lenna".....	67
Table 3.1: Maximum search steps and search points for a $\pm 8$ -pixel search window. ....	99
Table 3.2: Composition of test image sequence .....	105
Table 3.3: BACVQ codebook optimisation for intraframe coding.....	115
Table 3.4: BACVQ codebook optimisation for residual frame processing .....	116
Table 5.1: Various HOD computation time for image frame #001 and #061. ....	182
Table 5.2: Execution time for processing frame CIF061 as a still image.....	185
Table 5.3: Relative parallel performance gain of various processing stages.....	188
Table 5.4: Performance gain of parallel image sequence compression.....	195

## List of Abbreviations

ADSP: Advanced digital signal processor.

BACVQ: Block adaptive classified vector quantisation.

BH: Block histogram difference.

BMA: Block matching algorithm.

BOPS: Billion operations per second.

bpp: Bits per pixel.

BV: Block variance difference.

CTC: Combined transform coding.

CVQ: Classified vector quantisation.

DCT: Discrete cosine transform.

DFT: Discrete Fourier transform.

DHT: Discrete Hartley transform.

DOH: Difference of histogram.

DPCM: Differential pulse-code modulation.

DSP: Digital signal processing or digital signal processor.

DST: Discrete sine transform.

DVD: Digital versatile disc or digital video disc.

GLA: Generalised Lloyd algorithm for VQ codebook optimisation.

GOP: Group of pictures.

HDTV: High definition television.

HOD: Histogram of difference.

HVS: Human visual system.

JPEG: Joint Photographic Experts Group.

KLT: Karhunen-Loeve transform.

LBG: Linde, Buzo and Gray algorithm for VQ codebook optimisation.

LP, BP and HP: Low-pass, band-pass and high pass.

MAE: Mean absolute error.

MCE: Motion compensation error.

ME: Motion estimation.

MEE: Motion estimation error.

MFLOPS: Million floating-point operations per second.

MHOD: Magnitude of HOD variations.

MIMD: Multiple instruction stream multiple data stream.

MIPS: Million instructions per second.

MISD: Multiple instruction stream single data stream.

MPEG: Motion Picture Experts Group

MSE: Mean squared error.

NNQ: Nearest neighbour quantiser.

PE: processing element.

PSNR: Peak signal-to-noise ratio.

QMF: Quadrature mirror filter.

RISC: Reduced instruction set computer

SIMD: Single instruction stream multiple data stream.

SISD: Single instruction stream single data stream.

SSE: Sum of squared errors.

TSVQ: Tree-structured vector quantiser.

VLSI: Very large-scale Integration.

VQ: Vector quantisation or vector quantiser.

# **Chapter 1: Introduction**

## **1.1 Introduction**

In human history, establishing communications over a distance has always been an important part of human activities. In the early days, various forms of communications like smoke signals, semaphore, mirror-flashing, drums, light or fire beacons, well-trained pigeons, or even human messengers were used to deliver messages over a distance. All these primitive forms of communications, however, suffer from many disadvantages such as distance limitation, low message-carrying capacity, low speed, unreliability and high cost.

The invention of telegraph and radio transmission in the 19<sup>th</sup> century has revolutionised the way human beings communicate (Webster's Concise Encyclopedia, 1994). In 1837, telegraph was used mainly as a signalling device over an electrical circuit in British railway systems. Subsequently, the theory of electromagnetic waves was put into practical use in 1895 by Guglielmo Marconi. The significance of Marconi's contribution is that for the first time in human history, human beings have been able to communicate over a distance using radio transmission. After the successful demonstration of radio transmission, Marconi developed wireless telegraph communications to exchange text messages between England and France in 1899 and across the Atlantic in 1901. Radio communication of human speech was then demonstrated in 1906. The development of radio communications has allowed not only news to be broadcast from radio stations to reach larger groups of audience instantly but also education and entertainment programs to be delivered to them more effectively. Most significantly, the invention of black-and-white television in the 1930's and colour television in the 1950's has had even greater impacts on people's lifestyle. Naturally, visual communications have much greater appeal than the voice counterparts, especially when high quality sound and picture transmissions are deployed. Visual communications inherently provide a friendlier atmosphere and more comprehensive coverage for news reporting and offer a more exciting entertainment environment to enhance human experience.



## **1.2 The growing importance of digital technology**

In addition to the deployment of various forms of communications, more recent development of digital technology has opened up a whole new world of opportunities. The advent of VLSI and microprocessor technology together with the flexibility and cost effectiveness of digital processing techniques have fuelled the proliferation of digital applications in almost every area of human activities. So much so that it has resulted in a swift transition from analog to digital systems in many fields.

Even in analog environment digital applications have also been widespread. In the music industry, the introduction of digital audio compact discs (CDs) in 1982 has virtually marked the end of an era in which analog phonograph records were used as a recording medium for high quality music reproduction (Ely, 1996). Being recorded in digital form and smaller in physical size, CDs are less susceptible to the encoding of unwanted sounds and to physical damage. Furthermore, CDs improve over conventional records and tape recordings with a more uniform and accurate frequency response, a complete absence of background noise, and a wider dynamic range. Similarly, in the telecommunication industry, digital communications systems have quickly replaced their analog counterparts. This is because digital transmission is more efficient and inherently more robust against circuit deficiencies and transmission impairments. In addition, signal processing in the digital domain is in general far more flexible and cost effective (Zou, 1993). Digital technology has also been applied to photographic imaging whereby a picture is broken up into small dots, called pixels, with average value of light intensity being assigned to each pixel. More recently, digital video and digital television have been actively explored.

Up until the early 1970s, television signals were handled almost entirely in analog forms, that is, video signal of an optical scene was represented as a continuously variable voltage, whose instantaneous value reflected the luminance and chrominance of picture elements being transmitted. However, as digital techniques develop, much of analog video-processing has been replaced by digital signal processing and high-speed sampling that converts continuously varying video signals into digital samples, each representing a discrete signal level. The advantages of handling video signals in

digital forms are that they can be easily manipulated with special effects or used for image overlays or multiple-imaging mixing without any degradation in image quality. In addition, several digital signals can be multiplexed, expediting the transmission and processing of auxiliary information. Also, digital coding provides many benefits for message compression and serves as the basis for interactive telecommunications systems. For cable and direct broadcast satellite television services, digital compression techniques allows their transmission capacity to be instantly increased by more than 10 folds. Digital coding can also be mixed for various media forms, including computer communications on the same carrier, enabling integrated transmission of voice, image, graphics, and data in the same digital data stream.

The emergence of digital versatile disc (DVD) in 1996 marked a new era of consumer video entertainment, in which a digital video format was brought to the mass consumer market for the first time (Ely, 1996). This new format increases the density of the standard 650MB CD to up to 17GB DVD storage capacity, enabling the recording time in MPEG-1 digital video format to be extended from 74 minutes to several hours. In addition, the establishment of various digital compression standards such as H.261 and MPEG-1 and MPEG -2 has contributed significantly to the development of multimedia computing, video-on-demand, digital terrestrial broadcast television, video archiving system, videophone, and video conferencing (Petajan, 1992) (Wright, 1996). Not only has such development of digital technology revolutionised the existing information technology, but it has also brought about the convergence of entertainment, computing, and communications, thereby further enriching human experience.

Being an advanced television standard, high-definition television (HDTV) offers a significantly greater number of scanning lines, and therefore a clearer picture than the conventional 525/625 line television systems (Webster's Concise Encyclopedia, 1994). In 1989, the Japanese broadcasting station NHK and a consortium of manufacturers launched the Hi-Vision HDTV system, an analog HDTV system with 1,125 lines and a wide-screen format. In 1993, however, the United States decided to adopt digital transmission systems for its future HDTV systems. The major

advantage of the proposed U.S. standard for HDTV over the Japanese one is that it is based on digital format. Digital HDTV transmission is preferred to its analog counterpart for several reasons (Aoki, 1994). Firstly, it can be implemented at lower power levels, thus causing significantly less interference with existing television signals, and its quality is also constant throughout a given service area. Secondly, digital TV receivers are more likely to be compatible with a wide variety of computer and telecommunications services. In Australia, the television industry and the government are planning to introduce digital HDTV transmission by Year 2000, coincident with the Olympic Games in Sydney.

Alongside the successful transition from analog to digital, rapid advances have also been made to existing telecommunications infrastructure. Integrated-Services Digital Network (ISDN) has been established as a versatile, high capacity communication networks to provide various digital services such as voice, videophone, computer data or high-quality fax. Satellites have been deployed to provide digital direct broadcasting television services to subscribers. Digital transmission has enabled these satellites to increase the number of television channels substantially. In addition, recent advances in fibre optic technology have led to the deployment of fibre-optic cable instead of the usual copper wire for communications links. While fibre-optic technology has been widely applied in many areas, its greatest impact is perhaps in the field of telecommunications, where optical fibre offers the ability to transmit audio, video, and a variety of digital information. Its advantages over metallic cable include vastly increased information carrying capacity, lower transmission losses, lower cost, much smaller cable size, and almost complete immunity from stray electrical fields or interference.

### **1.3 The importance of digital image compression**

Although advances in transmission systems have substantially increased the transmission capability of modern digital telecommunication networks, conserving transmission system resources remains a critical issue. This is mainly because the rapid development of digital imaging and video technology has placed an ever increasing demand on those transmission systems to cater for newer and more demanding applications. Therefore, apart from improving digital system capability,

applying digital compression in general and still image/image sequence compression in particular has been considered as a key approach for dealing with the rapid growth of visual information being exchanged over communication channels or stored in mass storage medium (Gray, 1984).

## **1.4 Thesis contributions**

To date, numerous techniques have already been developed for still image and image sequence compression. However, although considerable effort has been made in developing these techniques, it is generally very difficult for a compression technique to actually achieve optimality in compression performance and computational simplicity simultaneously due to the complexity of the still image/image sequence compression process itself. For this reason, the prime objectives of this research are to explore various possibilities to improve the compression performance of both still image and image sequence compression and at the same time to reduce its computation complexity by adopting more efficient processing operation. In particular, as an alternative approach for still image compression, the proposed Block Adaptive Classified Vector Quantisation (BACVQ) achieves good compression performance and coding efficiency by combining classified VQ and variable block-size coding technique. More consistent classification and adaptive image-block segmentation algorithms based on both spatial and transform domains have also been developed to further improve the operation of CVQ and variable block-size coding. In addition, highly efficient algorithms have been proposed to accelerate codebook searching operation and to determine the optimal sizes of classified VQ sub-codebooks.

For image sequence compression, an adaptive spatial/temporal compression algorithm has been developed with improved compression performance and coding efficiency. To improve the consistency of the scene segmentation process, an adaptive scene segmentation scheme based on the magnitude of the variation in histogram of difference (MHOD) has been developed without adding significant complexity. In addition, a variable block-size motion compensated predictive coding algorithm has been developed to improve the efficiency of the interframe coding process. These improvements include the use of motion estimation errors as a

segmentation criterion to provide more consistent image segmentation operation, the addition of extra search points at the centre of image search windows to eliminate possible motion estimation errors in low motion regions, and the development of an alternative progressive motion estimation strategy to improve the efficiency of motion estimation and to ease implementation difficulties.

Apart from improving still image and image sequence compression algorithms, the application of parallel processing to image sequence compression is also investigated in this study. This is because parallel image compression offers a more effective approach than the sequential counterparts to accelerate the compression process and bring it closer to real-time operation. In particular, because of the independent processing and spatial arrangement natures of most image processing operations, an effective parallel image sequence compression algorithm has been developed to significantly reduce the processing time of the proposed parallel image compression algorithms.

## **1.5 Thesis organisation**

The topic of still image compression is investigated in Chapter 2. Following an overview of still image compression in Section 2.1, a discussion of various still image compression techniques is given in the subsequent sections. Typical still image compression techniques include transform coding, subband and wavelet coding, and vector quantisation (VQ). Although VQ has been considered as an optimum compression technique, its computation complexity has severely limited its applications in practice. As such, this has motivated us to develop the proposed block adaptive classified VQ algorithm (BACVQ) for still image compression. A detailed description of the BACVQ algorithm and a discussion of its simulation results are given in Section 2.5, followed by a discussion of lossless compression techniques in Section 2.6.

Various problems relating to lossy image sequence compression are discussed in Chapter 3. Following an overview of various technical issues involved in image sequence compression in Section 3.1, a survey of human visual system is conducted to identify some of its deficiencies which are beneficial to image sequence

compression operations. The importance of scene segmentation and various segmentation criteria are discussed in Section 3.3, followed by an investigation of the operation principle of various motion compensated coding techniques. How adaptive coding techniques such as block size and coding mode adaptation can be applied to improve the performance of image sequence compression is then examined. The basic coding structure of MPEG is briefly discussed in Section 3.6 and some of its terminology has been used to describe the proposed adaptive image sequence compression algorithm in Section 3.7. In this algorithm, a new adaptive scene segmentation technique is investigated, which offers accurate scene transition detection, high tolerance to spurious image activities and computational simplicity. In addition, an alternative approach to variable block-size motion compensation and progressive motion vector estimation is introduced.

Chapter 4 discusses the basic concept of parallel processing. Following an overview of parallel processing in Section 4.1, various parallel processing architectures are discussed. The issues of parallel connectivity are then investigated in Section 4.3. In order to take advantages of parallel processing hardware, considerations must be given to the design of parallel algorithms. Being the most critical operations during parallel processing, process synchronisation, task partitioning and process scheduling are examined in detail in Sections 4.4 and 4.5. To evaluate the performance of digital processing systems quantitatively, various performance measures are described in Section 4.6 with greater emphasis on their accuracy and simplicity. This is followed by a discussion of various issues relating to the design of message-passing and shared-memory MIMD systems.

The topic of parallel image sequence compression is investigated in Chapter 5. Following an overview of parallel image sequence compression and parallel DSP systems in Section 5.1, the characteristics of various commercially available parallel DSP platforms are discussed. The hardware configuration of the proposed prototype parallel DSP platform for the study of parallel still image/image sequence compression and its basic processing operations are described in Sections 5.3 and 5.4. Because dualport memory and semaphore operations contribute significantly to the proper operations of the prototype parallel DSP platform, they are examined in

detail in Sections 5.5 and 5.6. In Section 5.7, the proposed parallel image sequence compression algorithm is described and the performance of the prototype parallel DSP platform is judged by means of experimental results.

Finally, Chapter 6 concludes this thesis with an overall perspective on the algorithms developed and the results obtained in this study. It also provides some recommendations applicable to future research work. Various appendices are included at the end of this thesis to facilitate direct references. While Appendix A to C contain a copy of various conference papers submitted previously, Appendix D to F present the flowchart of the simulation software developed for the studying of still image compression, image sequence compression, and various parallel processing arrangements. Appendix G provides some notes on boot loading operations of the PC-32 cards. While Appendix H presents an example showing some inefficiency in using high level programming, Appendix I provides a sample output of simulation results obtained during the study of parallel image sequence algorithm.

## **Chapter 2: Still Image Compression**

In this chapter, the topic of still image compression is presented because it is an important process not only by itself as a key technique for compressing still images but also for its role in image sequence compression. Although many mainstream compression algorithms have been already developed for still image compression applications, most of them tend to suffer performance or implementation problems to some degree. For this reason, hybrid compression algorithms are more favourable and have been developed aiming at reducing image reconstruction distortion, increasing coding gain, and/or lowering implementation complexity. They enjoy these benefits mainly by combining the advantages of various coding techniques.

Being a hybrid coding algorithm, Block Adaptive Classified Vector Quantisation is presented in this chapter. Its enhanced coding performance is achieved as a result of combining variable block-size coding with classified VQ. Improved image-block classification and codebook generation algorithms have also been developed to provide better classification consistency and to ease implementation issues associated with VQ coding.

### **2.1 Introduction**

Although the rapid technological advance in recent years has enabled many digital systems to expand their system capability substantially in terms of transmission bandwidth and mass storage, conserving system resources remains an important issue. This is mainly because the rapid development of digital imaging technology has imposed an ever increasing demand on those systems to cater for more and more applications involving digital images. Therefore, apart from improving digital system capability, digital image compression also represents a key technology for dealing with the rapid growth of visual information which is exchanged over communication networks or stored on mass storage media (Gray, 1984).

The operation principle of all high performance digital image compression techniques relies heavily on the removal of both statistical and perceptual redundancy within normal images in order to achieve high compression gains. While image



compression techniques based merely on statistical redundancy removal allow original visual information to be fully recovered from compressed data, those based on perceptual redundancy removal or a combination of both do not possess such a property (Fedele et al, 1988). This fundamental difference between these two classes of image compression techniques has led to the development of the so-called lossless and lossy image compression algorithms. In general, while lossless image compression algorithms have very limited compression capability, lossy image compression techniques are capable of offering much higher compression gains but at the expense of some fidelity degradation. Therefore, instead of insisting on perfect reconstruction, lossy image compression techniques are more concerned with preserving the best image fidelity possible for a given rate (Gray et al, 1992). Among the most common lossy image compression techniques are transform coding, subband and wavelet coding, and vector quantisation (VQ).

For many block-based image compression techniques such as transform coding and VQ, coding efficiency can be improved by applying variable block size coding strategy in conjunction with adaptive block segmentation. For image compression applications, it is desirable that perceptually important image areas are segmented into small blocks while low activity regions being segmented into larger ones. This is because not only does this segmentation approach offer higher overall coding efficiency, but it also allows the coding process to achieve uniform spatial distribution of perceptual distortion throughout the reconstructed images, thereby significantly enhancing their perceptual quality. Due to the flexibility and efficiency of quadtree segmentation, most variable block size coding techniques have adopted quadtree segmentation principle during their adaptive block segmentation process (Shusterman et al, 1994) (Vaisey et al, 1992) (Lee & Crebbin, 1994a).

In an effort to improve the consistency of the segmentation process, various assessment criteria for evaluating local image activities within image regions have been reported. In the spatial domain, Vaisey et al (1992) have found that block variance can be used effectively to evaluate local image activities. When the variance of an image block exceeds an empirically determined threshold value, the block shall be considered as being associated with a high detail class, and therefore shall be

segmented into four smaller sub-blocks of equal size. This segmentation process is applied recursively until a low activity sub-block is detected or a minimum sub-block size is reached. Alternatively, image activity measures in the transform domain is also possible. According to Chen et al (1984), the total energy of the AC DCT-coefficients of transformed image blocks can be used as an effective image activity measure.

Being a very important coding algorithm, VQ has attracted considerable interest from the image compression community due to its compression potential. As far as quantisation is concerned, it has been widely acknowledged that optimum quantisation can only be achieved with VQ. However, high quality VQ coding operation usually insists on the use of a very large codebook and high vector dimension, leading to exceptionally high computational cost. Therefore, in an effort to make VQ more practically feasible, recent studies have shown that the coding structure of VQ can be modified with little sub-optimality, thus resulting in the development of various constraint VQs (Vetterli et al, 1992). In particular, classified VQ has been regarded as a typical variant of constraint VQs and its advantages have been well perceived. Not only does it offer a significant reduction in computation complexity as opposed to full-search VQ, but it also plays a key role in enhancing the perceptual quality of reconstructed images. In addition, when combined with variable block size coding, classified VQ would offer even greater performance benefits. As a result, this coding approach has been adopted in the proposed block adaptive classified VQ algorithm (BACVQ).

It should be noted that the performance of classified VQs depends largely on the effectiveness of vector classification process. This is because the extremely high complexity of real images makes it very difficult to maintain consistent classification across all local image textures into their corresponding classes, which in turn affects the efficiency of classified VQs. Therefore, in an effort to improve the classification consistency, various classification schemes operating in either transform or spatial domains have been investigated (Lee et al, 1994a) (Ngan et al, 1992) (Ramamurthi et al, 1986) (Wen et al, 1993). In particular, as a result of this study, an alternative vector classification algorithm operating in both transform and spatial domains has

been proposed. Its operation is based on the use of DCT coefficients, dynamic range and contrast sensitivity. Additionally, dynamic thresholding has been adopted during the classification process in order to exploit the HVS deficiency. As such, this vector classification algorithm was observed to produce highly consistent classification results.

In implementing the proposed BACVQ coding algorithm, a highly efficient partial codebook search approach has been developed for 8x8 codebook search operation. In fact, experimental results have shown that this partial codebook search can offer up to 80% reduction in search time with a small PSNR degradation of 0.03dB. In addition, a new algorithm has been developed for determining the optimal codebook size for classified VQ sub-codebooks. Compared to other codebook size determination methods, the proposed algorithm is capable of determining the optimal codebook size for individual sub-codebooks with significantly reduced computation complexity, especially when large codebook training sets are involved.

And lastly, irrespective of the type of lossy image compression techniques being used in image compression systems, it is a common practice that a statistical encoding technique such as Huffman coding or arithmetic coding is applied as the final stage of the compression process. The aim of this coding stage is to eliminate any remaining statistical redundancy in the coded data stream, and therefore achieving additional compression gain without adding any distortion. This favourable outcome is realised because statistical encoding is basically a lossless compression technique which allows full recovery of the original data (Fedele et al, 1988) (AT&T, 1993).

## **2.2 Transform coding**

One of the most popular coding techniques for still image compression is transform coding. Its operation principle is primarily based on orthogonal linear transformation of original signals to the transform domain prior to the quantisation process (Malvar, 1992). As such, its relatively simple coding algorithm coupled with its effectiveness in de-correlating image elements has accounted for its popularity for image

compression applications. In fact, it has been adopted in the specifications of JPEG, a well-known still image compression standard (AT&T, 1993).

Another advantage of applying transform coding to image compression applications is that since natural images do not contain all frequencies, many transformed coefficients in transformed image blocks have zero values, thus representing the redundancy as well as the available compression gain. In an extreme case, for example, a pure grey image block of 16x16 block size effectively has only one DC terms and 255 nil coefficients. This means that a compression gain of 256:1 can be theoretically obtained. Even when these coefficients are not zero, many of them are so small that they fall below visual thresholds and contribute very little to reconstructed image quality. As such, they can be adequately truncated to zero without noticeable visual impairment. This implies higher compression gain is achievable with transform coding when more coefficients of transformed image blocks are set to zero (Ahlgren, 1988).

In general, orthogonal linear transformation can be implemented in various forms. Among the most common forms of orthogonal linear transformation are Karhunen-Loeve transform (KLT), discrete Fourier transform (DFT), discrete Hartley transform, discrete sine transform (DST), and discrete cosine transform (DCT). Although DFT and DCT are very similar in their mathematical formulations, DCT is by far more popular for image processing applications due to its energy compaction efficiency (Malvar, 1992) (Rao et al, 1990).

However, applying conventional DCT coding techniques to image compression applications usually suffers from “blocking” effects because of the discontinuation at block boundaries. In addition, ringing at visual object boundaries is another weakness of DCT coding which severely affects the perceptual quality of reconstructed images. Basically, all these objectionable features occur due to the lossy quantisation of transformed coefficients following the DCT transformation process of image blocks. Therefore, research effort has been recently directed to eliminating such deficiencies of DCT. An example of this is the development of the combined transform coding technique (CTC). Compared to the conventional DCT

approach, CTC has been claimed to offer the desire property of reducing both blocking and ringing effects in addition to improved compression efficiency (Zhang et al, 1993).

From the statistical point of view, KLT is considered as the optimal linear transformation for image compression applications. This is because KLT can ideally produce a set of totally uncorrelated coefficients from the original signals. Furthermore, KLT is capable of maximising the energy compaction in transformed blocks. All of these are regarded as highly desirable features in image compression to lower overall distortion. However, from the practical point of view, KLT has some severe drawbacks when applying to signal coding applications. Firstly, KLT is signal dependent since its transformation requires good modeling of input signal statistics. As a result, efficient implementation of KLT is usually difficult to achieve, especially when signal statistics may change over time and real-time computation is required. Secondly, since the KLT matrix has no particular structure, efficient computation of its transformation is harder to accomplish (Malvar, 1992).

In transform coding, it is a common practice that separate quantisers are used for the quantisation of individual transformed coefficients. This is because transformed coefficients are expected to have different probability distribution functions due to the de-correlation effect of the linear transformation process. A typical transform coding scheme is illustrated in Figure 2.1.

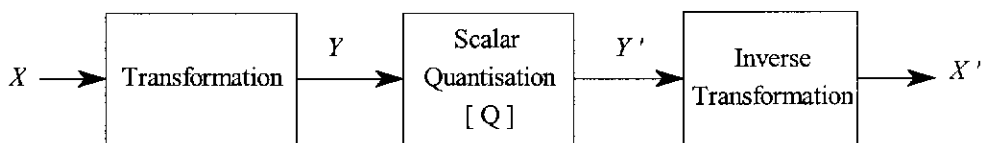


Figure 2.1: Block diagram of transform coding

To assess the performance of a coding scheme, a quantitative distortion measure based on the overall sum of squared errors is frequently used and is given by:

$$D_{TC} = \sum_{i=1}^k E[(x_i - x'_i)^2] = E[\|X - X'\|^2]$$

where,  $X$  and  $X'$  represent the original and reconstructed vectors, respectively; and  $k$  denotes vector dimension.

An orthogonal transformation of the form  $Y=TX$  is normally employed in the above transformation. It is considered as a linear transformation with respect to  $T$  satisfying the orthogonality condition, that is,

$$T^t = T^{-1}$$

As distances are preserved in orthogonal transformation, it follows that:

$$D_{TC} = E\left[\|X - X'\|^2\right] = E\left[\|Y - Y'\|^2\right]$$

This implies that orthogonal transformation does not magnify quantisation noise so that the sum of squared quantisation errors for the  $k$  quantisers is equal to the overall distortion. For this reason, optimal quantisation of transformed coefficients with optimal bit allocation plays a crucial role in determining the overall performance of transform coding.

Zig-zag scanning pattern according to the assigned number

Lowest ordered DCT coefficients	1	2	6	7	23	24	28	29
	3	5	8	13	25	27	30	43
	4	9	12	14	26	31	42	44
	10	11	15	16	32	41	45	54
	17	18	22	33	40	46	53	55
	19	21	34	39	47	52	56	61
	20	35	38	48	51	57	60	62
	36	37	49	50	58	59	63	64

Figure 2:2: A typical zigzag scanning pattern for transformed coefficients quantisation

Apart from separate quantisation, transformed coefficients are commonly quantised in the order of a zigzag pattern (Dufour et al, 1992). The major motivation for this is to achieve better energy compaction, to improve the efficiency of subsequent run-length coding, and to facilitate the implementation of hierarchical coding schemes. For instance, in HDTV compatible coding, the encoded data stream may consist of two components, namely, a compatible data stream and an additional data stream.

The compatible data stream shall contain all the information necessary for the reconstruction of conventional TV sequences. Such a data stream typically contains 16 lowest ordered transformed coefficients of 8x8 DCT blocks following the zigzag scanning process. Subsequently, when inverse DCT is applied to these coefficients, image blocks of 4x4 block size can be reconstructed, which correspond to a lower resolution representation of original 8x8 blocks. In fact, the derivation of conventional TV pictures in this way is equivalent to applying 4:1 decimation to HDTV images. Contrary to the compatible data stream, the additional data stream shall contain extra data necessary for the reconstruction of superior HDTV sequences. Another advantage of the zigzag scanning technique is that it offers a more flexible coding scheme. For example, if a decoder does not receive the total number of transformed coefficients in full, the decoder can simply set the missing coefficients, especially those with higher orders, to zero without affecting its operation. This suggests that the zigzag scanning technique also contributes to the realisation of a more robust coding system. A typical zigzag scanning pattern is illustrated in Figure 2.2.

For image processing applications, linear transformation of image data blocks would theoretically require the application of two dimensional (2D) transformation operators because images are represented as 2D data. However, since image data can also be modelled as separable random processes, separable transformation can be applied without any loss in performance. This means that 2D transform operators can be implemented more efficiently using 1D transform operators and a two-stage computation process. During the first stage, all rows of an image block will be transformed using a 1D transform operator, resulting in an intermediate 2D block. In the second stage, the same transformation is applied to all the columns of the intermediate block to complete the transformation process. Mathematically, this transformation process can be illustrated as follows (Clarke, 1985) (Blinn, 1993):

Assuming,  $Y$ ,  $X$ , and  $T$  denote the transformed block, the input block and the orthogonal transformation matrix, respectively, a 2D transformation process based on separable transformation approach can be illustrated as follows:

1) Stage 1: Transform all rows of  $X$

$$B = TX^T$$

where  $B$  denotes an intermediate transform vector.

2) Stage 2: Transform all columns of  $B$  to complete the 2D transformation process:

$$Y = TB^T = T[TX^T]^T = T[XT^T] = TXT^T$$

### 2.2.1 Discrete Fourier transform (DFT):

Being a linear transformation, DFT can be described by adopting a general form for the transformation matrix:

$$T = \begin{bmatrix} a_{00} & a_{01} & \cdot & a_{0(M-1)} \\ a_{10} & a_{11} & \cdot & a_{1(M-1)} \\ \cdot & \cdot & \cdot & \cdot \\ a_{(M-1)0} & a_{(M-1)1} & \cdot & a_{(M-1)(M-1)} \end{bmatrix}$$

The basis functions of a transformation matrix can be defined as the columns of the matrix. For DFT, the basis functions are given by:

$$a_{nk} = \sqrt{\frac{1}{M}} \exp\left[j \frac{2\pi kn}{M}\right]$$

### 2.2.2 Discrete Hartley transform (DHT):

The basis functions for this transformation are given by:

$$a_{nk} = \sqrt{\frac{1}{M}} \text{cas}\left(\frac{2\pi nk}{M}\right)$$

where  $\text{cas}(\theta) = \cos(\theta) + \sin(\theta)$

Since DHT is a real transformation, no complex arithmetic is required during the transformation process. More importantly, since DHT transformation matrix is symmetrical, the direct and inverse transformation processes are identical.

### 2.2.3 Karhunen-Loeve transform (KLT):

The basis functions for KLT transformation are defined as the eigenvectors of the covariance matrix of the input signal source. For first order auto-regression sources,



such eigenvectors have the properties of being sinusoids whose frequencies are not evenly distributed in a unit circle.

From the statistical point of view, the KLT is considered as the optimal transform (Vetterli et al, 1992). This is because KLT is the only orthogonal transform capable of producing a set of uncorrelated coefficients from a colored source as well as maximising the energy compaction within transformed blocks. This is a highly desirable property for signal coding and signal compression applications.

For linear transformation, the relative coding gain is normally evaluated using the following expression (Malvar, 1992):

$$G_{TC} = \frac{\frac{1}{M} \sum_{k=1}^M \sigma_k^2}{\left( \prod_{k=1}^M \sigma_k^2 \right)^{1/M}}$$

This suggests that KLT maximises the coding gain by minimising the geometric mean of the variance of transformed coefficients in the denominator.

In spite of its desirable properties, applications of KLT to signal coding applications suffer several drawbacks. Firstly, KLT transformation is signal dependent. Therefore, successful applications of KLT are unlikely unless a good model for the source signal statistics can be established, the signal statistics does not change with time, and real-time computation of covariance matrices and eigenvectors is possible. Secondly, since KLT transformation matrix does not have any particular structure, its transformation process does not allow fast computation to be performed.

#### 2.2.4 Discrete Cosine transform (DCT):

The basis functions for this transformation are defined as :

$$a_{nk} = c(k) \sqrt{\frac{2}{M}} \cos \left[ \left( n + \frac{1}{2} \right) \frac{k\pi}{M} \right]$$

where

$$c(k) = \begin{cases} 1/\sqrt{2} & \text{if } k = 0 \\ 1 & \text{otherwise} \end{cases}$$

In DCT, when the frequency index  $k$  varies from 0 to  $(M-1)$ , there are  $M$  distinct frequencies in the interval  $[0,\pi]$ . As a result, spectral analysis performed with DCT will produce twice the number of frequency bands as opposed to DFT or DHT. In addition, if an input signal has a strong component at a particular DCT frequency but 90 degrees out of phase with respect to a DCT basis function, then the corresponding DCT coefficient will be zero (Malvar, 1992).

## 2.3 Subband and wavelet coding

As an alternative approach to image compression, subband coding can be considered as a frequency domain bit-rate reduction technique (Furukawa et al, 1992). This is because input signals are decomposed into various subbands before compression operation actually takes place. Its development is mainly motivated by the following observations:

- Signal power for high frequency components is generally 40 to 60dB less than the DC power.
- Power spectrum distribution in vertical and horizontal directions are similar.
- Subband coding does not suffer from blocking effects.
- Tree-structured subband decomposition processing facilitates hierarchical coding. It allows lower resolution images to be more readily accessible from the coding stream.
- The power spectrum of input signals in the logarithmic frequency scale is almost linear except in the very high frequency region.
- Band splitting filters can be implemented using various filter types. However, quadrature mirror filters (QMF) are frequently employed for band splitting operations. In general, band splitting filters must possess such properties as aliasing cancellation, perfect reconstruction, linear phase and good low-pass characteristics in order to minimise coding distortions.

It has been shown that the performance of subband coding is the best when there is a wide variation in the power levels of different subbands. Also, highly correlated signals with high prediction gains are amenable to efficient redundancy removal by

subband coding with the same ultimate performance as being achieved in transform coding. In fact, it can be shown that the worst-case behaviour in terms of reconstruction errors of subband coding is similar to that of DCT due to the close relationship between subband filter banks and unitary transforms. As with DCT coding, ordering subbands according to their energies also helps minimise mean squared errors when a subset of the subbands is used for the reconstruction. This is because the lower bands usually carry basic visual information while the higher ones only adds more detail to the reconstruction (Vetterli et al, 1992).

The extension of subband coding to 2D and 3D signals is generally facilitated by the adoption of separable filter banks. For example, in the case of still images where signals are associated with 2D representation over the horizontal and vertical dimensions, a simple subband decomposition can be implemented using separable filter banks as illustrated in Figure 2.3 (Furukawa et al, 1992). These separable filters can be efficiently implemented as row-column filters, which are in turn constructed using 1D QMF filters (Thyagarajan et al, 1987).

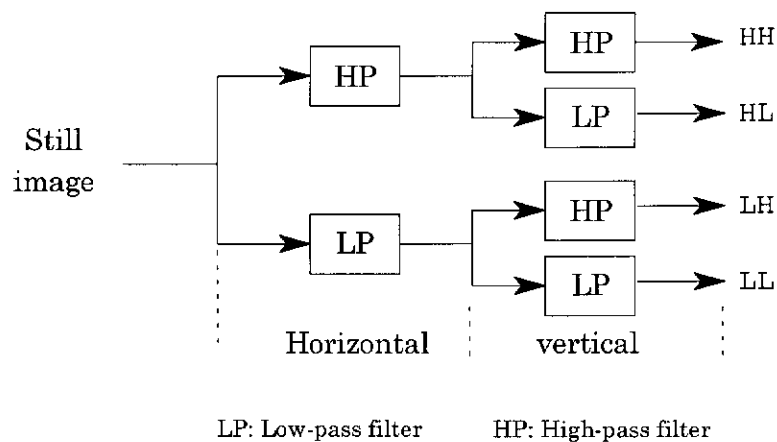


Figure 2.3: A typical subband decomposition of still images

However, one of the difficulties in implementing subband coding is that heavily constrained operators have to be used during subband decomposition process. This suggests that a compatible LP channel with reasonable complexity usually cannot produce adequate quality for subband coding operation. What is more, it has been found that maximum reconstruction errors in subband coding are weakly bounded (Vetterli et al, 1992).

For practical reasons, subband coding systems tend to maintain the constraint of critical sampling while relaxing the requirement for an ideal BP filter bank. As such, the filter bank in subband coding is usually replaced with physically realisable filters whose transfer functions are rational functions of  $\exp(j2\pi f/f_s)$  where  $f_s$  represents the sampling frequency. The consequence of this compromise is that some interband interference may occur, thus creating aliasing problems. In order to eliminate this effect, short kernel filters and QMF filters are frequently used for band splitting during subband decomposition process (Coppisitti et al, 1993).

Apart from filter type considerations, there are two other major issues concerning the operation of subband decomposition (Malvar, 1992). Firstly, if incoming signals are simply passed through a set of bandpass filters and their outputs are used directly, the amount of information to be processed would grow proportionally to the number of subbands. This observation suggests that any coding gain achievable by subband coding would be offset by the significant increase in subband information. Fortunately, this is not the case in most situations. In fact, following the band splitting process at the filter bank, subband signals are often sub-sampled or decimated in both horizontal and vertical dimensions, thereby resulting in a 4:1 data compression in each of the subband channels (Malvar, 1992). Secondly, the design of subband filters must be capable of recovering the original signal from its subbands. To meet this requirement, it is necessary that aliasing problems, due to sub-sampling or decimation process primarily, should be eliminated. Malvar (1992) has shown that the frequency spectrum of decimated signals expands with respect to the input signal spectrum by a factor equal to the decimation ratio  $M$ . Therefore, if the input signal spectrum is larger than  $\pi/M$ , then as the decimation process takes place, there will be overlapping regions in the spectral replicas, leading to aliasing problems. For this reason, it is a common practice to precede a decimator with a low-pass or band-pass filter with a bandwidth of  $\pi/M$  in order to minimise the possibility of aliasing.

The reverse of the decimation process is known as the interpolation process. Contrary to the decimation process where every  $M$ -th sample is retained from its original sequence, the interpolation process is implemented by replacing missing samples with zero values. In the frequency domain, this is equivalent to shrinking the

frequency spectrum of the input signal at the interpolator by a factor of  $M$ , and replicating this shrunken spectrum  $(M-1)$  times (Malvar, 1992). This observation suggests that aliasing is not an issue during the interpolation process. Instead, the output spectrum of the interpolator contains multiple replicas of its input signal spectrum, normally referred to as “imaging”. As a result, it is necessary to pass the output signal of the interpolator to a LP or BP filter having a bandwidth of  $\pi/M$  so that only one of the spectral images is retained, thereby enabling a full recovery of the original signal. A typical arrangement of a subband processing system is shown in Figure 2.4.

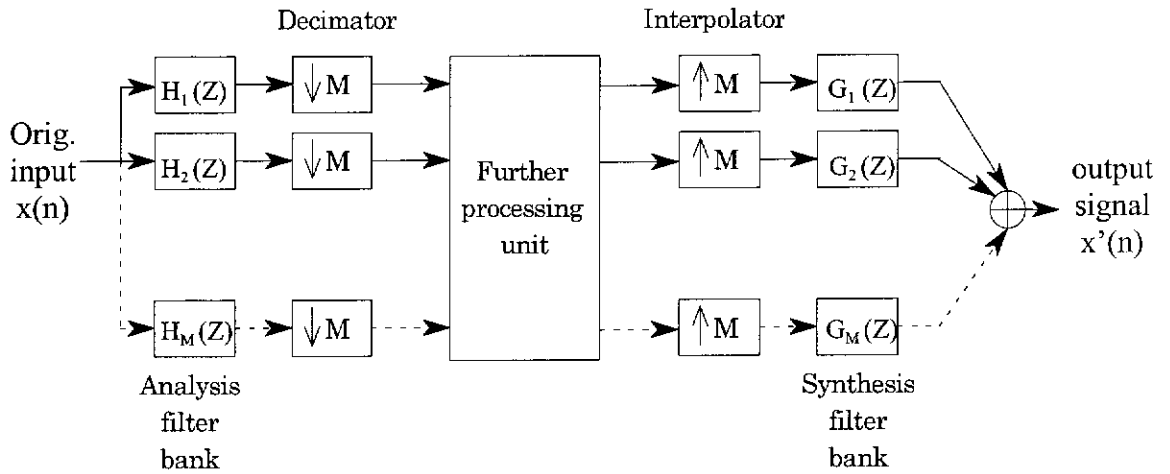


Figure 2.4: Block diagram of a subband coding system

According to the convolution principle, the output of the  $k$ -th analysis filter in Figure 2.4 is given by:

$$X_k(i) = \sum_{j=-\infty}^{\infty} x(j)h_k(iM - j)$$

and the reconstructed signal is expressed as:

$$x'(n) = \sum_{k=1}^M \sum_{i=-\infty}^{\infty} X'_k(i)g_k(n - iM)$$

If no further processing is applied to the subband signals at the further processing unit, it follows that:

$$X'_k(i) = X_k(i), \quad \forall k$$

Therefore,

$$x'(n) = \sum_{k=1}^M \sum_{i=-\infty}^{\infty} \left[ \sum_{j=-\infty}^{\infty} x(j)h_k(iM - j) \right] g_k(n - iM)$$

Re-organising the summation terms gives:

$$x'(n) = \sum_{j=-\infty}^{\infty} x(j) \sum_{k=1}^M \sum_{i=-\infty}^{\infty} h_k(iM - j) g_k(n - iM)$$

In order to achieve perfect reconstruction, Malvar (1992) has showed that the filter banks  $\{H_k(Z)\}$  and  $\{G_k(Z)\}$  must satisfy the following condition:

$$\sum_{k=1}^M \sum_{i=-\infty}^{\infty} h_k(iM - j) g_k(n - iM) = \delta(n - j - D)$$

so that,

$$x'(n) = \sum_{j=-\infty}^{\infty} x(j) \delta(n - j - D) = x(n - D)$$

This suggests that when the above condition is satisfied, the overall effect of the subband coding process is simply equivalent to introducing a delay into the original signal. This result is in fact consistent with those presented by Kim (1993), where a different analytical approach has been undertaken.

As mentioned earlier, the decimation process in subband coding plays a key role in reducing subband information associated with individual subband signals. However, the ultimate compression gain of subband coding is in fact achieved by compressing each individual subband signal with an appropriate bit allocation scheme which takes into account not only statistical redundancy in subband signals but also psycho-visual deficiency of human visual system. As such, it allows more bits to be allocated to the lower subbands because of their perceptually importance and less to the higher ones.

Recent studies of human visual characteristics in perceiving visual images have revealed the possibility of achieving high image compression with minimal image impairment using subband coding (Kim, 1993). Specifically, it has been found that human visual system seems to exhibit LP filtering in the temporal domain and BP in the spatial domain. Therefore, subband coding presents a sensible approach because it offers an efficient way to exploit this result, in which more emphasis is placed on lower bands while components of higher bands can be progressively eliminated.

As an alternative approach, discrete wavelet coding can be applied equally well to image compression applications. From the theoretical point of view, discrete wavelet

coding can be considered as a special form of subband coding with a logarithmic decomposition tree structure. The application of the logarithmic tree structure in discrete wavelet coding enables the doubling of image resolution every time a wavelet channel is added. Basically, the development of discrete wavelet coding has been motivated by recent studies of human visual system modeling. Various psycho-visual experiments have suggested that the retinal processing in human eyes uses independent BP filters which are almost linear and have a constant relative bandwidth of about 1 octave among them. Therefore, if one assigns a roughly constant number of bits per octave, it is then possible to maintain equally perceived quality across filtered channels (Vetterli et al, 1992).

The filter bank used in wavelet coding is usually constructed using a special type of LP filter called regular filters. These regular LP filters possess the same properties of orthogonality and linear phase as normally exhibited by subband filters. In addition, when connected in a cascade filter/sub-sampling configuration, these filters tend to produce a smooth equivalent impulse response. It is the smoothing feature of these regular LP filters that results in very little energy being retained in the high bands during the wavelet analysis process, hence significantly enhancing the performance of subsequent compression operations. However, if the above regularity property is not satisfied, this may result in equivalent filters with fractal impulse response, which in turn affects the reconstruction of the signal (Vetterli et al, 1992).

## **2.4 Vector quantisation**

### **2.4.1 Overview**

Vector quantisation (VQ) is typically a lossy compression technique for signal compression applications. A general coding structure of vector quantisation is presented in Figure 2.5. Conceptually, vector quantisation can be regarded as Shannon's model of source coding. Its central idea is to encode signal samples as a group instead of one at a time as in the case of scalar quantisation, resulting in a more efficient way to exploit the redundancy among signal samples.

As Gersho & Gray (1992) have demonstrated, a regular VQ can be visualised as having a polytopal structure in which each cell in the partition is a convex polytope.

For a  $k$ -dimension polytopal structure, the  $(k-1)$  dimensional faces of the cells constitute hyperplane segments. These hyperplane segments serve as boundaries among polytopal regions in the entire structure. In general, the number of hyperplanes typically bounding a polytope increases very rapidly with respect to its space dimension. This result partially explains why the complexity of VQ encoders grows dramatically with increased vector dimension.

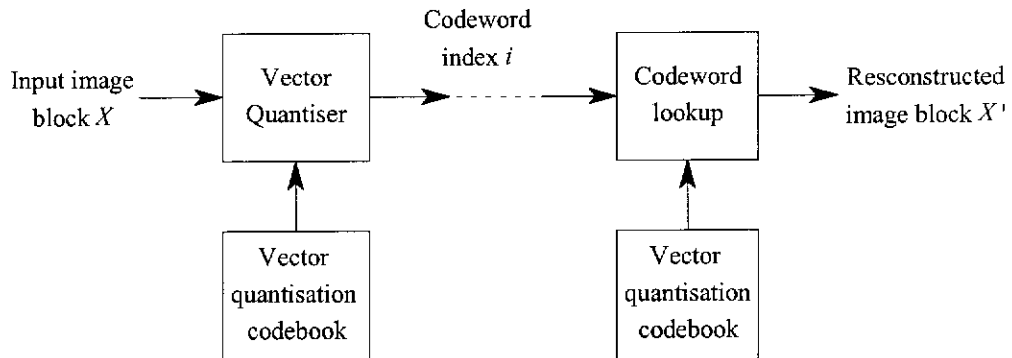


Figure 2.5: Block diagram of vector quantisation coding systems

Statistically, for the same total bit allocation, it has been proven that VQ indeed offers better performance than scalar quantisation because its coding algorithm allows it to exploit the redundancy among vector samples more efficiently. In addition, it has been found that VQ is capable of offering compression gain not only when correlation exists among vector components but also in the case where little correlation is anticipated. Other advantages of VQ include (Gray et al, 1992):

- The ability to exploit both linear and non-linear dependency among vector components;
- The extra freedom in choosing multi-dimensional quantiser cell shapes;
- Offering fixed coding rates in the case of conventional vector quantisation;
- The flexibility in setting codebook size and choosing arbitrary partitions;
- The simplicity of the decoding process because its decoding algorithm is basically based on table look-up operations and requires very little computation;
- The ease of incorporating other compression techniques into its coding algorithm to form more effective hybrid coding schemes such as transform VQ;



- The ease of incorporating clustering techniques into its coder design process to provide good quality codes;
- Fractional bit allocation as coding bits are allocated on a vector basis; and
- Image enhancement and local classification capability can be incorporated into the vector quantisation process in a natural way by making simple modification to its design technique. This capability can lead to significant simplification to subsequent processing tasks.

In spite of the significant advantages that unconstrained VQ promises to deliver, its applications in practice are somewhat limited due mainly to its implementation difficulties. In particular, its computational complexity and storage requirement grow exponentially with increased vector dimension and coding rate. And yet, for applications requiring good picture quality, it is often necessary to use high vector dimension and high bit rates. Therefore, improvements including those imposing some constraints on VQ code structures are all necessary to make VQ a viable approach. These improvements generally include (Chee et al, 1994) (Vetterli et al, 1992):

- Developing more effective codebook generation algorithms;
- Using more efficient codebooks such as those used by classified VQ or transform VQ;
- Employing faster encoding algorithm;
- Achieving higher compression rate by using variable block size coding, higher vector dimension, smaller codebooks and entropy-constrained VQ;
- Making use of adaptive VQ such as mean adaptive VQ, switched codebook adaptation, variable bit-rate VQ and adaptive codebook VQ;
- Reducing the blocking effects such as using side-matched classified VQ;
- Provision for progressive image transmission which can be achieved by applying progressive image transform, multistage VQ or pyramidal VQ; and
- Using other structurally constrained VQ such as lattice-based VQ, classified VQ, TSVQ, transform VQ, predictive VQ, finite state VQ and so on.

### 2.4.2 Vector quantisation

In order to illustrate the operation principle of VQ, let us first assume  $d(X, X') \geq 0$  denote a general form of distortion measure between an input vector  $X$  and its reproduction  $X'$ . It follows that an optimal vector quantiser with reference to a given codebook shall select the code vector  $Y_i$  if and only if:

$$d(X, Y_i) \leq d(X, Y_j); \quad \forall j \neq i$$

If the codebook is fixed for all input vectors during the vector quantisation process, then such a VQ scheme is regarded as operating in a memoryless fashion. This is in contrast to predictive and finite state VQs which operate with a predetermined memory model.

As a result of vector quantisation, reconstruction distortion during the decoding process is inevitable. However, for most image compression applications, such distortion is considered as being acceptable in exchange for high compression gain. For this reason, the average distortion between the original input vector and its reproduction has been commonly used as a means to evaluate the performance of various VQ (Gray et al, 1992). Mathematically, this is expressed as:

$$D = E[d(X, X')]$$

If the vector process is assumed to be stationary and ergodic, then the expectation in the above expression can be replaced by the time average, which is defined as:

$$D = \bar{d} = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n d(X_i, X'_i)$$

where  $n$  denotes the total number of input vectors.

In order to provide a quantitative distortion measure between a  $k$ -dimension input vector  $X$  and its quantised reproduction vector  $X'$ , the sum of squared errors is often used due to its mathematical simplicity and is defined as:

$$d(X, X') = \|X - X'\|^2 = (X - X')'(X - X') = \sum_{j=1}^k (x_j - x'_j)^2$$

Therefore,

$$D = E[d(X, X')] = E[\|X - X'\|^2]$$

The latter expression suggests that the average distortion defined in the above manner can be interpreted as a measure of energy or power associated with the error signal between the original input vector and its reproduction resulted from the vector quantisation process.

For generality, the weighted squared errors may be used and it is given by:

$$d_w(X, X') = (X - X')' W (X - X')$$

where  $W$  denotes a symmetric and positive definite weighting matrix.

Two special cases of  $W$  are of great interest. If  $W$  equals to an identity matrix, the later expression reduces to the normal sum of squared errors. However, if  $W$  is a diagonal matrix with positive diagonal element  $w_j$ , the above expression describes the simplest form of weighted squared errors and can be rewritten as:

$$d_w(X, X') = (X - X')' W (X - X') = \sum_{j=1}^k w_j (x_j - x'_j)^2$$

The computation of distortion measures generally falls into two categories, namely, design distortion and test distortion. In general, design distortion is associated with that incurred in quantising the vector training set with the resulting codebook. This performance measure is sometimes called the “inside the training set” distortion. By contrast, test distortion refers to that incurred in quantising vectors which do not belong to the training set using the same codebook. This measure is referred to as the “outside training set” distortion.

Due to mathematical convenience, the peak signal-to-noise ratio ( $PSNR$ ) is commonly quoted as a distortion measure for evaluating reconstructed image quality and is defined as:

$$PSNR = 10 \log_{10} \left[ \frac{255^2}{MSE} \right]$$

where  $MSE$  denotes the mean squared error of the quantisation process and is given by:

$$MSE = E \left[ \frac{1}{k} d(X, X') \right] = E \left[ \frac{1}{k} \sum_{j=1}^k (x_j - x'_j)^2 \right]$$

It should be noticed that the above expression is only applicable to images with 8-bit pixel representation because it assumes image pixel values vary in the range from 0 to 255. For this reason, a peak signal value of 255 has been used in the calculation of PSNR.

### 2.4.3 Shannon's rate-distortion theorem

Let  $\{x_n\}$  denote a sequence of random variables generated by a discrete time source and  $\mu$  define the probability property of  $\{x_n\}$ . It follows that for block-based compression systems, consecutive source blocks can be represented as (Gray et al, 1976):

$$X_k^M = (x_k, x_{k+1}, \dots, x_{k+M-1})$$

where  $k$  refers to the index of random variables in the sequence  $\{x_n\}$  and  $M$  denotes the dimension of the source block.

Due to the vector quantisation process, consecutive source blocks are mapped into consecutive reconstructed blocks  $X_k'^M$ . In order to ascertain compression gains, it is necessary that the allowed set for the reconstructed blocks be much smaller than that for the source blocks

Furthermore, Shannon assumed that a tractable mathematical distortion measure such as mean squared error or Hamming distance exists for assessing the distortion of the vector quantisation process, which is represented by:

$$\rho(X_k^M, X_k'^M) = \frac{1}{M} \sum_{i=0}^{M-1} \rho(x_i, x'_i)$$

In this way, the average distortion of a code can be determined by evaluating the expectation of statistical average over  $\mu$ , that is,

$$\rho_\mu(C) = E_\mu[\rho(X_k^M, X_k'^M)]$$

and its coding rate is determined by:

$$R(C) = \frac{1}{M} \log_2 \|C\|$$

where  $C$  represents the vector quantisation codebook and  $\|C\|$  denotes the size of the codebook in terms of a finite number of codewords.

Having established the distortion and coding rate for vector quantisation process, Shannon suggested that two approaches are possible for determining the optimal attainable performance of a vector quantiser. They are generally referred to as the rate-distortion and the distortion-rate principles. In the former case, the maximum allowable average distortion of the vector quantisation process is constrained while the coding rate is minimised. Mathematically, this condition can be expressed as:

$$r_{\mu}(d) = \inf_M r_{\mu}(d, M)$$

where,

$$r_{\mu}(d, M) = \inf_{C: \rho_{\mu}(C) \leq d} R(C)$$

and inf is a Greek abbreviation for infimum, meaning the lower bound of a function.

Similarly, in the latter approach, the coding rate is constrained while the resulting average distortion of the vector quantisation process is minimised. This relationship can be expressed as:

$$\delta_{\mu}(R) = \inf_M \delta_{\mu}(R, M)$$

where,

$$\delta_{\mu}(R, M) = \inf_{C: R(C) \leq R} \rho_{\mu}(C)$$

In general, the distortion-rate approach is more favourable because in most communications systems the rate constraints are well defined, whereas the required average distortion is rarely precisely known.

Shannon and others have shown that a rate-distortion function  $R_{\mu}(d)$  and its inverse or distortion-rate function  $D_{\mu}(R)$  exist such that if the source is stationary and ergodic then the following relationship is satisfied:

$$\delta_{\mu}(R) = D_{\mu}(R) \quad \text{and} \quad r_{\mu}(d) = R_{\mu}(d)$$

These functions are collectively referred to as Shannon's rate distortion theorem. For a given source and error fidelity criterion, the minimum possible transmission rate is governed by Shannon's rate-distortion function. Conversely, it has been shown by Malvar (1992) that for a stationary and ergodic source with a Gaussian probability density function, its theoretical distortion-rate function is given by:

$$D(R) = 2^{-2R} \gamma_x^2 \sigma_x^2$$

where  $\gamma_x$  and  $\sigma_x$  refer to the spectral flatness and the variance of the source, respectively. For ideal vector quantisation operating at an average rate of  $R$  bits/sample, it has been found that its distortion-rate function is given by:

$$D_{VQ}(R) = 2^{-2R} \sigma_x^2 = \frac{D(R)}{\gamma_x^2}$$

#### 2.4.4 Optimality conditions for vector quantisation

One of the key issues concerning the design of a vector quantiser is its optimality. In order to achieve this, it is necessary that a design technique must satisfy at least three conditions of optimality (Gersho & Gray, 1992). These conditions are the nearest neighbour condition, centroid condition, and zero probability boundary condition. Mathematically, these conditions can be expressed as follows:

1. Nearest neighbour condition:

$$R_i \subset \{X: d(X, Y_i) \leq d(X, Y_j), \quad \forall j \neq i\}$$

Where  $d(X, Y_i)$  and  $d(X, Y_j)$  denote the distortion measures between the input vector  $X$  and code vectors  $Y_i$ ,  $Y_j$  of a reference codebook, respectively.

2. Centroid condition:

$$Y_i = Cent(R_i) = \min_Y^{-1} E[d(X, Y) | X \in R_i]$$

where  $Cent(R_i)$  denotes a centroid selection function. In the case of squared error distortion criterion, this expression can be further reduced to:

$$Y_i = Cent(R_i) = E[X | X \in R_i]$$

3. Zero probability boundary condition:

$$p(X: d(X, Y_i) = d(X, Y_j); \exists i \neq j) = 0$$

If  $X$  is a continuous random variable, this condition will automatically be satisfied because the boundary associated with a continuous random variable has zero volume, thus resulting in zero probability.

Although the above conditions are necessary, they are not sufficient to assure the optimality of a vector quantiser. What these conditions really suggest is that such a vector quantiser is guaranteed to be at least locally optimal. As a result, special care

should be taken when designing a vector quantiser to avoid this pitfall since locally optimality can sometimes be very sub-optimal.

#### 2.4.5 Nearest Neighbour Quantisation

Being an important class of VQ, a nearest neighbour quantiser (NNQ) has a unique feature that the partition of its vector space is completely determined by its codebook and its distortion measure criterion. Therefore, a vector encoder using NNQ does not require any explicit storage for geometrical description of its nearest neighbour cells in the vector space. Instead, a conceptually simple algorithm based on the nearest neighbour condition can be efficiently used to partition the vector space, that is,

$$R_i \subset \{X: d(X, Y_i) \leq d(X, Y_j), \quad \forall j \neq i\}$$

In order for the cells  $R_i$  to constitute a partition, each boundary point in the vector space must be uniquely assigned to only one cell.

A general encoding rule of NNQ is to determine a single code vector in a reference codebook that best matches the input vector in the sense of minimum distortion. This encoding process can be illustrated as follows:

1. Initially, set  $d = d_0$ ,  $j = 1$ , and  $i = 1$ , where  $d_0$  must be larger than any anticipated distortion value and is typically set to the largest positive number that a processor's arithmetic unit can represent;
2. For each input vector  $X$ , compute the corresponding distortion  $D_j = d(X, Y_j)$ ;
3. If  $D_j < d$ , then update  $d = D_j$  and set  $i = j$ ;
4. If  $j < N$ , increment  $j$  by 1 and go back to step 2; otherwise, the encoding process is completed. The index of the best representative code vector and the resulting minimum distortion are given by the values in  $i$  and  $d$ , respectively.

Since the distortion computation is performed sequentially for every code vector in the reference codebook, the above encoding rule is referred to in literature as the exhaustive or full search algorithm. As a consequence of this, the exhaustive search strategy requires a fixed time to search through an entire codebook. More

importantly, the exhaustive search strategy is considered to be the only scheme capable of producing optimal results in the sense of squared error distortion. However, its high computation complexity has restricted its applications in practice. In fact, this setback has been so severe that it has motivated the development of various constrained VQ encoding schemes as well as the exploration of different fast search algorithms (Buhler et al, 1987) (Gray et al, 1992) (Kolagotla, 1993).

#### **2.4.6 Structurally constrained vector quantisers**

For normal vector quantisation with exhaustive search strategy, the required codebook storage space in words and the search complexity in terms of number of operations per input vector are both proportional to  $kN = k2^{rk}$ , where  $k$ ,  $N$  and  $r$  denote input vector dimension, total codebook size, and coding rate in bits/vector component, respectively. As such, this reveals that the search complexity and memory space requirement grows exponentially with the product of vector dimension and coding rate. For this reason, constrained VQ has become a very popular subject in recent research activities (Ngan et al, 1992) (Phamdo et al, 1993).

In general, constrained VQ compromises the performance achievable with unconstrained VQ in exchange for computation simplicity. Due to its lower complexity, constrained VQ is usually deployed for applications involving high vector dimension and coding rate. This is because both of these parameters are crucial for the realisation of better performance VQ. In particular, increasing vector dimension enables more effective exploitation of statistical redundancy among vector components as well as the extra freedom to choose higher dimension quantisation cell shapes. Among the most prominent groups of constrained VQs are structurally constrained VQs. This group typically includes tree structured vector quantisers, classified vector quantisers, transform vector quantisers, multistage vector quantisers, hierarchical and multi-resolution vector quantisers, lattice vector quantisers, and predictive vector quantisers (Gersho & Gray, 1992).

##### **1. Tree structured vector quantisers (TSVQ)**

TSVQ are generally classified as balanced and unbalanced TSVQ. As the names imply, balanced TSVQ refers to a particular encoding tree structure whose



branches all have equal lengths, whereas unbalanced TSVQ corresponds to those whose branches have various lengths. Accordingly, this structural difference leads to the development of fixed length and variable length codes, respectively. A potential benefit of variable length codes is that it is possible for such codes to outperform even full search VQs at the same coding rate. This is because full search VQs must allocate the same number of bits to every vector, whereas unbalanced TSVQs only devote more bits to important vectors and fewer to insignificant ones as a result of the variable length code approach.

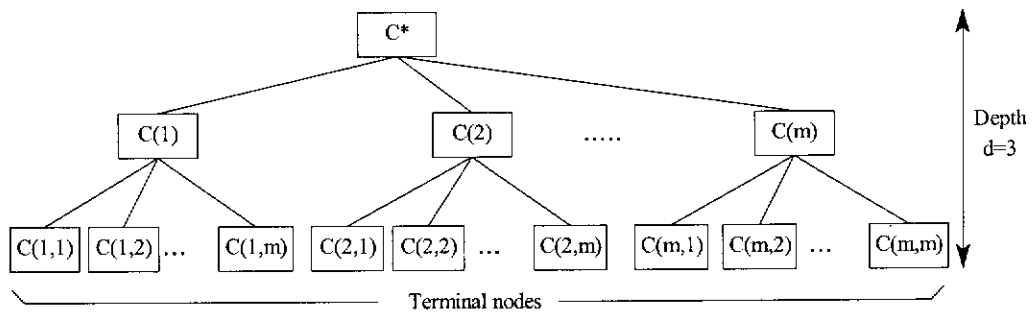


Figure 2.6: Typical codebook structure of TSVQs

In the case of balanced  $m$ -ary TSVQs as shown in Figure 2.6, a search at each stage of the tree structure effectively reduces the number of code vectors from the previous set of candidates by a factor of  $m$ . If the codebook size  $N$  equals  $m^d$ , then a total of  $d$   $m$ -ary search stages are required to locate the best matched code vector from the terminal nodes of the  $m$ -ary tree. Therefore, the search complexity of balanced  $m$ -ary TSVQs is proportional to  $md$  instead of  $m^d$  as in the case of exhaustive search VQs, hence offering a significant reduction in computation complexity and latency time. In addition, the searching process of TSVQs can be further accelerated by adopting VLSI implementations (Kolagotla, 1993).

However, the storage requirement of TSVQ is usually higher than that of unconstrained VQ. This is because in addition to storing actual code vectors of the overall codebook at terminal nodes, TSVQs need to maintain node codebooks at intermediate nodes of the tree. These node codebooks, normally consisting of a set of  $m$  test vectors each, are needed to provide branching information at the intermediate nodes during the encoding process.

As a special case of TSVQs, 2-ary or binary TSVQs are of particular interest in practical implementations. This is because binary TSVQs have the lowest search complexity among all other TSVQs. However, a major drawback of binary TSVQs is that they require the largest storage space for their operation.

Regardless of various approaches, codeword search in TSVQs is considered as being sub-optimal because it does not guarantee that nearest neighbour code vectors are always chosen from their codebook. However, in practice, it has been found that the performance loss due to the tree search constraint is relatively small, whereas the gain in speed and computation simplicity is very significant (Gersho & Gray, 1992).

Another advantage of TSVQs is that they have a built-in successive approximation capability so that a coding process can be scaled easily to match communications channel capacity. For example, if for some reasons the available rate is reduced, the coding process simply proceeds less deeply into the tree, and vice versa (Kolagotla, 1993).

## 2. Classified vector quantisers

The adoption of image classes in classified VQs generally serves to delimit features which could not be found in a certain codebook. This is because it would be a lot more difficult to attain good quality images of mixed types than it is for one or more classes having similar features. Moreover, classified VQ is noticed for its ability to preserve the integrity of reconstructed image blocks. This property is very important for image compression applications because it has a significant impact on the overall perceptual quality of reconstructed images.

Conceptually, classified VQs can be regarded as one-stage TSVQs where the branching decision is made based on a heuristic set of criteria. This heuristic set of criteria serves to identify different attributes among input vectors. So, instead of using a node codebook for determining the branching decision, classified VQs rely on an arbitrary classifier for identifying a subset of the overall codebook, from which code vectors are found to best represent the input vector. This operation of classified VQ is illustrated Figure 2.7.

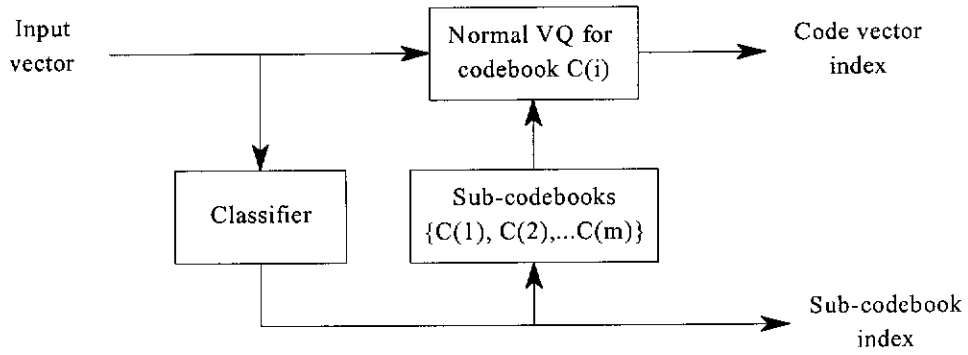


Figure 2.7: Block diagram of classified VQ

Various algorithms can generally be used at the classifier for image block classification. One approach is to use the gradients in the horizontal and vertical directions as a means to determine the nature of image blocks. This classification process typically consists of three separate test procedures relating to block activity, orientation and position, thereby allowing various image block attributes such as uniformity, medium activity, and horizontal/vertical/diagonal contour or edge features to be identified (Buhler et al, 1987). Alternatively, block means, energy levels and other relevant statistics of image blocks can also be used as effective measures for classification purposes.

Apart from the issue of classifier, sub-codebook design represents another significant challenge for the design of classifier VQ. An optimal codebook design algorithm for classified VQ can be illustrated as follows:

Let  $C$  represent the overall codebook of a classified vector quantiser, that is,

$$C = \bigcup_{i=1}^M C_i$$

where  $M$  and  $C_i$  denote the total number of vector classes and the sub-codebook for the  $i$ -th class, respectively.

This implies that if  $N_i$  denotes the size of each sub-codebook  $C_i$ , then the size of the overall codebook  $C$  can be determined by:

$$N = \sum_{i=1}^M N_i$$

It follows that the overall average distortion  $D$  of classified VQ is given by:

$$D = \sum_{i=1}^M p_i D_i$$

where  $p_i$  is the probability of classifying an input vector into the  $i$ -th class and  $D_i$  denotes the average distortion of the  $i$ -th class.

As long as the vector source is stationary and ergodic, the average distortion of individual vector classes can be computed as:

$$D_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d_j(X, Y) \quad , \quad \forall X \in \text{the } i\text{-th class}, Y \in C_i$$

where  $n_i$  denotes the number of training vectors for the  $i$ -th class.

It has been shown by Ramamurthi et al (1986) that in order to obtain an optimal set of sub-codebook sizes  $\{N_i^*\}$  so that the overall average distortion can be minimised, the following expression must be satisfied:

$$A_i = \frac{p_i D_i^*}{N_i^*} = \text{const}$$

where  $A_i$  represents the partial distortion per code vector of the  $i$ -th vector class and  $D_i^*$  denotes the average distortion of the  $i$ -th vector class with sub-codebook size  $N_i^*$ .

In effect, the above expression attempts to equalise the partial average distortion per code vector across all the sub-codebooks of classified VQ as a means to minimise the overall distortion resulting from the classified vector quantisation process. In proposing this strategy, it is assumed that distortion caused by various image classes is equally noticeable. Statistically, this expression also suggests that image classes with lower classification probability tends to be allocated smaller sub-codebook sizes and hence subject to higher average distortion, and vice versa.

### 3. Transform vector quantisers

A typical coding arrangement of transform VQs is presented in Figure 2.8. Theoretically, since linear transformation is a linear operation, it follows that optimal code vectors for the above vector quantiser would correspond to the transforms of optimal code vectors in a codebook that would otherwise be used

for direct vector quantisation of the input vector  $X$  without any transformation. This implies that the average distortion of transform VQ will be exactly the same as that of direct VQ. Therefore, this may seem to suggest that transform VQ would offer no apparent advantages over direct VQ while adding some extra complexity.

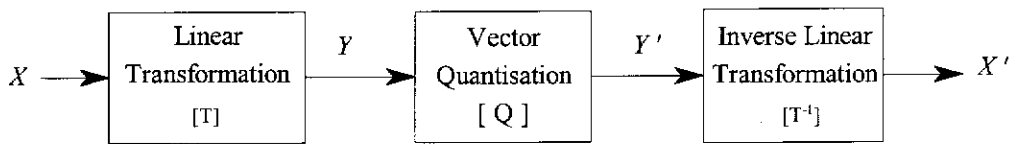


Figure 2.8: Block diagram of transform VQ

However, since linear transformation has the property of compacting information in original vectors into a subset of transformed coefficients and natural images do not contain all frequency components, it follows that a large portion of those coefficients are likely to be so small that they can be neglected altogether without any significant impact on picture reconstruction quality. As a result, vector dimension in transform VQs is virtually reduced, thereby lowering its overall complexity. In addition, because human visual perception seems to be more sensitive in the transform domain, the application of transform VQs will facilitate the implementation of adaptive bit allocation to transformed vectors according to their perceptual significance.

Instead of ignoring insignificant transformed coefficients altogether as a means to reduce vector dimension, it is also quite possible to partition the coefficients into several groups, and then allow separate vector quantisation to be applied to them. In this way, low energy transform coefficients can still be separately coded at a very low bit-rate, without taking the risk of ignoring them entirely. In addition, since these smaller groups are normally associated with a collection of features representing the original signal, it follows that if the transformed coefficients are judiciously partitioned into subsets of features, called feature vectors, separate vector quantisation can be applied with even higher efficiency. Furthermore, partitioning transformed coefficients into smaller groups enables transform VQs to cope well with high vector dimension applications (Plansky, 1992) (Gersho & Gray, 1992).

#### 4. Multistage vector quantisers

As the name implies, multistage VQs divide the encoding process into several vector quantisation stages as shown in Figure 2.9. While the first stage performs a coarse quantisation of input vectors using a relatively small codebook, vector quantisation at subsequent stages only deals with residual vectors produced from previous stages. As such, the coarsely quantised vector together with the quantised residual vectors constitutes successive approximations of the original input vectors, thereby allowing progressive reconstruction of the input. Apart from the property of successive approximation, another key feature of multistage VQ is that the overall distortion is constrained by the distortion committed at the final stage.

In general, since the codebook size at each subsequent stage is rapidly reduced, the overall search complexity as well as the storage requirement of multistage VQs is generally much lower than that of a single direct VQ subject to the same coding rate. However, as elements of residual vectors become less and less correlated, subsequent stages of vector quantisation tend to have rapidly diminishing coding gains. For this reason, practical implementations of multistage VQs are usually restricted to a few stages.

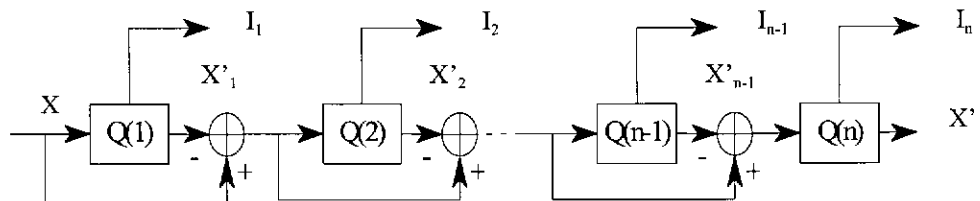


Figure 2.9: Block diagram of multi-stage VQ

#### 5. Hierarchical and multi-resolution vector quantisers

Hierarchical VQs are generally suitable for coding vectors with very high dimensionality due to its coding efficiency. The basic idea of hierarchical VQs is to extract feature vectors, normally of lower dimension, from the original vectors. These feature vectors are judiciously selected so that they contain all important attributes of the original vectors. When the feature vectors are vector quantised, the quantised feature vectors represent an approximate description of the original

vectors and result in the decomposition of the original vectors into a set of lower dimension sub-vectors, thereby increasing the efficiency of the encoding process.

Since the above decomposition process can be applied recursively to each of the sub-vectors, this approach has led to the so-called hierarchical VQs. If, however, the feature vectors represent sub-sampled versions of the original vectors during the decomposition process, this variant of hierarchical VQ is called multi-dimensional VQ because different degrees of sub-sampling are involved at various stages during the encoding process.

## 6. Lattice vector quantisers

A special class of VQs which attracts great interest is lattice VQs due to its regular structure. According to Gersho & Gray (1992), a lattice can be visualised as a regular arrangement of points in a  $k$ -dimensional space that includes the origin as a zero vector. The term regular means that every point actually sees the same geometrical environment as any other points in the lattice. In other words, any translation of a set of points by adding or subtracting one lattice point from all other points will result in the same lattice. Mathematically, a lattice in a  $k$ -dimensional space  $R^k$  is a collection of all vectors of the form:

$$X = \sum_{i=1}^n m_i u_i$$

where,  $m_i$  denotes arbitrary integers and  $\{u_i\}$  represents a linearly independent set of  $n$  vectors (usually  $n \leq k$ ) and is normally called a basic or generating set of the lattice. In the case where  $n$  equals  $k$ ,  $\{u_i\}$  becomes a non-degenerate set which is by far the most common case of interest.

A lattice VQ is simply a VQ whose codebook structure is constrained either by a lattice or a coset of a lattice, or by a truncated version of a lattice or its coset so that its codebook size is finite. As such, lattice quantisers are naturally chosen when uniform quantisation is required. An important parameter of lattice VQs is lattice density because a high density lattice generally leads to high bit rate and small average distortion.

In general, lattice VQs are most suited for applications where a vector source is memoryless with a smooth probability distribution function; quantisation resolution is high; and overload noise is negligible in comparison with granular noise. However, it should be noted that lattice VQs also have some setbacks. Firstly, their highly constrained structure may compromise their optimality in the sense of minimising average distortion for a given rate, especially when the input signal source is not uniformly distributed. Secondly, the implicit assumption of very high resolution makes it primarily suited for applications with high bit rate and small distortion. And lastly, lattice VQs generally cannot be optimised using optimising algorithms without sacrificing their regular structure (Gray, 1984).

## 7. Predictive vector quantisers

In predictive VQ, the predictor usually takes the form of a closed-loop prediction of  $X_n$ , that is,

$$\tilde{X}_n = P(X'_{n-1}, X'_{n-2}, \dots, X'_{n-m})$$

The difference vector is then formed by:

$$e_n = X_n - \tilde{X}_n$$

As the difference vector is vector quantised, the approximation of the original input vector  $X_n$  can be determined by:

$$X'_n = e'_n + \tilde{X}_n$$

where  $e'_n$  denotes the quantised difference vector.

As a fundamental theorem, it has been proven that the overall reproduction error of predictive VQs is exactly the same as the error resulting from the quantisation of the difference signal at the vector quantiser. It is interesting to note that this theorem holds regardless the specific nature of the predictor, that is, whether it is linear or non-linear (Gersho & Gray, 1992).

Generally, the achievable coding gain of VQs operating on the difference signal  $e_n$  tends to be less than that operating on the original input signal. The reason to account for this is that vector components of an error vector are in general less correlated than those of an original vector due to the de-correlation effect of the



vector prediction process. However, since the vector prediction process exploits the correlation among successive input vectors efficiently, the prediction gain tends to more than compensate the reduction in coding gain in relation to quantising the difference signal, resulting in an increase in the overall performance.

#### **2.4.7 Fast nearest neighbour encoding techniques**

Apart from structurally constrained VQs, the choice of search strategy has a strong impact on the encoding performance of the vector quantisation process, especially in case of single codebook. While exhaustive or full search strategy is only feasible when the codebook is small enough, sub-optimal search strategies with faster search algorithms are better suited for most applications because of their reduced search complexity and shorter search time (Buhler et al, 1987). In fact, the application of fast nearest neighbour encoding techniques has been widely regarded as a sensible approach to further accelerate the encoding speed of VQs. However, it should be noted that the relative advantages of various fast nearest neighbour encoding techniques tend to be implementation dependent. For instance, a technique well suited for implementation on a programmable signal processor may not be so suitable for implementation using application specific integrated circuits. Likewise, a technique which seems to be very effective on a general-purpose microprocessor actually achieves only marginal improvement on a programmable signal processor. In addition, unlike exhaustive search which has a fixed search time, most fast nearest neighbour encoding techniques experience a variable search time. Typical fast nearest neighbour encoding techniques include partial distance coding algorithm, the projection method, the  $k-d$  tree approach, and triangle inequality method.

##### **1. Partial distance coding algorithm**

Being the simplest algorithm among fast nearest neighbour encoding techniques, partial distance coding has been developed by making slight modification to the exhaustive search technique. Typically, the average search time of the partial distance coding algorithm can be reduced by a factor of four. Its algorithm can be briefly described as follows:

Given an input vector  $X$  of  $k$  dimensions, then:

- 1) As in the case of exhaustive search, codebook search using partial distance coding algorithm is initially performed sequentially throughout the codebook;
- 2) As the  $m$ -th code vector has been tested, the minimum distortion  $D$  and the corresponding code word index  $I$  would be retained;
- 3) To test the  $(m+1)$ -th code vector  $Y_{m+1}$ , a counter  $r$  is initialised to count from 1 to  $k$  for squared error distortion computation purposes. Within the counting loop, the partial distance is progressively calculated as:

$$D_r = \sum_{i=1}^r (x_i - y_i)^2$$

In this way, the calculation of the partial distance can be prematurely terminated as soon as the condition warrants that the current code vector is not the best reproduction of the input vector, that is,

- If  $D_r \geq D$ , the  $(m+1)$ -th code vector can be safely rejected without having to compute the squared error distortion in full and the whole process can be re-started for testing the next code vector.
- If  $D_r < D$  and  $r < k$ , then one more iteration of partial distance computation is performed and  $D_r$  is tested again.
- If  $D_r < D$  and  $r = k$ , then the minimum squared error distortion associated with the current code vector and its index are recorded, that is,  $D=D_r$  and  $I=m+1$ , before proceeding to the next code vector.

The above encoding procedure indicates that the worst case search time of the partial distance coding algorithm is no better than exhaustive full search although it is unlikely that such a case would occur. In addition, the partial distance coding algorithm is only beneficial to implementations where the time incurred in comparison operation is significantly shorter than that in multiplication. Otherwise, it will be better off to calculate the overall squared error distortion in its entirety.

## 2. The projection method

In the projection method, a data structure marking various subsets of candidate code vectors to be searched is used to accelerate the codebook searching process. This data structure is obtained by establishing for each nearest neighbour cell  $R_i$  the boundaries of the smallest hyper-rectangular region in a  $k$ -dimension space that contains it. The minimum and maximum values of points within cell  $R_i$  in the  $j$ -th coordinate are found by projecting the cell onto the  $j$ -th coordinate axis. As such, each cell will consist of a set of  $2k$  hyperplanes that specify the faces of the boundary hyper-rectangular region and each coordinate axis is partitioned into  $2N-1$  intervals, where  $N$  denotes the number of cells.

A table is then created for each coordinate axis indicating which nearest neighbour cells have a projection that lies in each interval. This means that a search procedure based on scalar quantisation can be applied to individual input vector component one at a time, thereby enabling a set of candidate cells for the input vector to be identified. The intersection of various sets of candidate cells then determines the final set of candidates from which a code vector can be found to best represent the input vector. At this point, since the size of the final set is much smaller in number than the original codebook, an exhaustive search procedure can be efficiently applied to identify the best code vector.

The main problem with the projection method is that the task of finding the exact projections of nearest neighbour cells are in general computationally infeasible. Therefore, in practice, these projections are usually found by partitioning a large training set into individual clusters and then taking the projection of each cluster as an approximation for the actual projection of the cell itself.

## 3. The $k$ - $d$ tree approach

The  $k$ - $d$  tree approach generally relies on a binary tree structured codebook for its fast search algorithm. As such, a  $k$ - $d$  tree can be regarded as a binary decision tree of  $d$  stages depth with a hyperplane decision test at each node and a set of terminal nodes. Each hyperplane is chosen to be orthogonal to one of the coordinate axis in

the  $k$ -dimension space so that each hyperplane test effectively eliminates the candidate terminal nodes into half.

The simplicity of this search algorithm comes from the fact that each node is required to examine only a single vector component of the input vector so as to make a branching decision. In most cases, this examination is carried out in the form of comparison against a threshold value.

#### 4. Triangle inequality method

The key idea of this technique is to use some anchor points in a  $k$ -dimensional space as reference points from which the distances to each code vector in the codebook are pre-computed and stored for future references. The encoder then computes the distances between the input vector and each anchor point so that a large number of candidate code vectors can be eliminated from subsequent nearest neighbour search by making some simple comparisons in conjunction with the pre-computed data. To illustrate this idea, let us examine a system with a single anchor point  $A$  as shown in Figure 2.10.

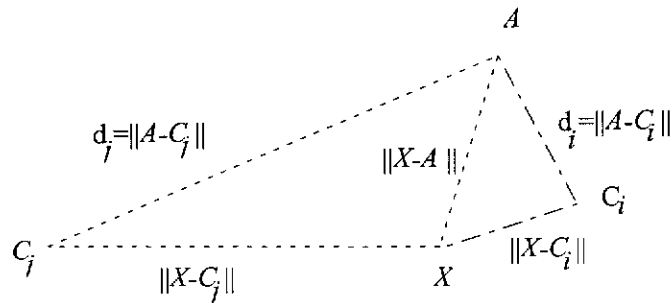


Figure 2.10: Triangle inequality codebook search method

Using the triangular inequality property, it follows that,

$$\|X - C_j\| + \|X - A\| \geq d_j$$

So, for every code vector search, says  $C_i$ , a portion of the codebook can be eliminated without having to calculate their actual distances. For instance, assume that the following inequality is satisfied for the code vector  $C_i$ :

$$d_j > \|X - C_i\| + \|X - A\|$$

it follows that,

$$\|X - C_j\| > \|X - C_i\|$$

This suggests that  $C_j$  can be safely eliminated from subsequent nearest neighbour search without having to actually calculate  $\|X - C_j\|$ .

#### 2.4.8 Vector quantisation codebook generation

Apart from the issues of structurally constrained vector quantisation and fast encoding algorithms, the construction of vector quantisation codebooks has a significant influence on the perceptual quality of reconstructed images (Buhler et al, 1987). In general, vector quantisation codebooks are generated using a two stage approach. In the first stage, an initial codebook is created, followed by a codebook optimisation process in the second stage. For initial codebook generation, various algorithms are available such as random coding, pruning technique, pair-wise nearest neighbour design algorithm and splitting design technique (Gray, 1984) (Gersho & Gray, 1992):

##### 1. Random coding

As the name implies, the operation principle of random coding technique is based on a random process in which code vectors are randomly selected from a training sequence or training set. As such, this simple codebook generation technique is best suited for applications where vector source has large vector dimension and experiences noise-like behaviour. However, because of its simple approach, random coding technique tends to suffer a major setback, that is, the resulting codebook is highly unstructured, thus leading to increased encoding complexity.

##### 2. Pruning technique

In pruning technique, training vectors of a training set are selectively eliminated as being candidates for the final codebook. This process is carried out iteratively until a final set of the training vectors are retained for the generation of the codebook. A typical codebook generation algorithm based on this technique can be briefly described as follows:

- 1) Initially, the 1st training vector from the training sequence is added to the codebook by default.

- 2) Subsequently, the minimum distortion between the next training vector from the training sequence and those already in the codebook so far is evaluated.
- 3) If the resulting minimum distortion is less than a pre-determined threshold, go back to step 2.
- 4) Otherwise, add that vector to the codebook and then go back to step 2.

For the *MSE* distortion measure criterion, the pre-determined threshold value is typically chosen to be proportional to  $N^{-2/k}$  or  $2^{-2r}$  where  $N$ ,  $k$  and  $r$  denote the codebook size, vector dimension and the rate of the code, respectively. This result is actually consistent with Shannon's distortion-rate theorem.

### 3. Pair-wise nearest neighbour design algorithm

Similar to the pruning technique is the pair-wise nearest neighbour design algorithm. However, the final codebook generated using this algorithm may contain code vectors which are not directly extracted from the training sequence. Basically, this process tries to merge vectors, which are assumed to belong to separate clusters containing a single vector, into bigger clusters until the desired codebook size is achieved. The codebook is then formed as a set of centroids of these clusters, resulting in an improved initial codebook. Typically, the pair-wise nearest neighbour design algorithm proceeds as follows:

- 1) Initially, the distortion of all vector pairs from a training set containing  $L$  vectors is computed.
- 2) The training vector pair with the lowest distortion is then merged, thereby reducing the number of initial clusters to  $L-1$ .
- 3) Thereafter, the merging of clusters is determined by evaluating the relative increase in average distortion when two clusters are merged. For the case of squared error criteria, the best pair of clusters to be merged is determined as follows:

For each pair of vector clusters, denoted as

$$R_i = \{X_i(l); l = 1, 2, \dots, L_i\} \text{ and } R_j = \{X_j(l); l = 1, 2, \dots, L_j\}$$

their contribution to the average distortion before and after the merging is:

$$\Delta_{i,j} = \sum_{l=1}^{L_i} d(X_i(l), \text{cent}(R_i)) + \sum_{l=1}^{L_j} d(X_j(l), \text{cent}(R_j))$$

$$\Delta'_{i,j} = \sum_{l=1}^{L_i} d(X_i(l), \text{cent}(R_i \cup R_j)) + \sum_{l=1}^{L_j} d(X_j(l), \text{cent}(R_i \cup R_j)) \geq \Delta_{i,j}$$

Naturally, the cluster pair that results in the minimum relative increase in average distortion ( $\Delta'_{i,j} - \Delta_{i,j}$ ) shall be merged. This process continues until the total number of clusters corresponding to  $N$  code vectors are obtained for the final codebook.

It should be noted that although each merging operation is optimal, there is no guarantee that the overall process is optimal. In addition, for a large training set, this technique is generally not feasible because of excessive computation complexity. In particular, during the initial stage of this codebook generation process, the requirement that the distortion of every vector pair from the training set has to be evaluated to determine a single vector pair for the merging operation causes substantial implementation problems. This is because the number of vector pairs for which the distortion needs to be evaluated will grows very rapidly as a function of  $(L-1)!$ , therefore potentially causing the algorithm to breakdown.

#### 4. Splitting design technique

Contrary to the pair-wise nearest neighbour design algorithm, the splitting design technique expands a codebook to a desired codebook size by splitting its vector clusters. Basically, the splitting process is applied selectively to individual clusters of a training set which result in a maximum reduction in the average distortion. This process is repeated iteratively until the desired codebook size is achieved.

As already mentioned, after an initial codebook has been created, it is often necessary that codebook optimisation is performed on the initial codebook in order to improve its optimality. In general, codebook optimisation can be performed using the following techniques (Buhler et al, 1987) (Gersho & Gray, 1992):

## 1. Clustering technique

Being based on the simplest clustering technique, the Generalised Lloyd or the Linde, Buzo and Gray (LBG) algorithm iteratively improves a codebook by alternatively optimising the encoder for the decoder using minimum distortion or nearest neighbour mapping criterion and the decoder for the encoder by replacing the old codebook with a better set of generalised centroids (Linde et al, 1980). As a result, its operation indeed satisfies the necessary conditions for optimality. In addition, a typical feature of the Generalised Lloyd algorithm is that its operation is based entirely on a representative set of training data, rather than a mathematical model of the vector source. For this reason, it is essential that the training set used in the optimisation process be judiciously selected to ensure the generality of the final codebook.

2. Other codebook optimisation algorithms are based on simulated annealing, deterministic annealing and stochastic relaxation techniques. The operation of these techniques is generally very slow because of a large amount of computation involved. However, these techniques are well noticed for their ability to generate globally optimised codebooks.

### 2.4.9 Generalised Lloyd algorithm

Being one of the well-known codebook optimisation algorithms, the Generalised Lloyd algorithm (GLA) optimises a vector codebook by modifying its code vectors iteratively (Buhler et al, 1987). The resulting codebook is guaranteed to better satisfy the necessary conditions for optimality, namely, nearest neighbour condition, centroid condition, and zero probability boundary condition. As the name suggests, this algorithm in fact generalises Lloyd's iteration algorithm for scalar quantisation. The generalisation for the vector case can be described as follows:

1. Given a codebook  $C_m = \{Y_i; i = 1, \dots, N\}$ , find the optimal partition among quantisation cells to satisfy the nearest neighbour condition. If a training vector yields a tie in its distortion measure with respect to several cells, that vector shall be assigned to the cell having the smallest index.



2. Based on the centroid condition, determine an updated codebook  $C_{m+1} = \{Cent(R_i); i = 1, \dots, N\}$ , which contains optimal code vectors for the cells just formed.

In many practical situations, a sample distribution based on empirical observations of the input vector source is used for optimising an existing codebook. This is mainly because analytical methods for evaluating the centroids using multiple integrals over a complicated  $k$ -dimension region are generally impossible even if the probability distribution of the input vector source has a relative simple and tractable expression. In addition, it has been pointed out that if one designs an optimal  $N$ -point quantiser with a sufficiently large training ratio  $M/N$ , where  $M$  denotes the total number of training vectors in the training set, it is reasonable to expect that the quantiser is in fact very close to optimal for the true distribution of the input random vector source (Vaisey et al, 1992).

During the optimisation process, it is possible that empty cells are formed as a result of satisfying the nearest neighbour condition, thus causing the computation of centroids to break down. To deal with this computation problem, various approaches can be adopted. For example, the largest quantisation cell can be divided into two smaller cells to replace the empty cell. Alternatively, the cell with the highest partial distortion is the one to be divided for empty cell replacement.

In general, the Generalised Lloyd algorithm can be briefly described as follows:

1. Firstly, let  $N$  and  $C_0$  denote the number of representative or code vectors and initial codebook, respectively. Also, a distortion threshold  $\varepsilon \geq 0$  is chosen for controlling the iteration process. The training sequence is denoted as  $SEQ = \{X_j; j = 1 \dots n\}$ . And last of all, initialise the iteration parameters  $m = 0$  &  $D_{-1} = \infty$ .
2. Subsequently, apply Lloyd's iteration algorithm:

Let  $C_m = \{Y_k; k = 1 \dots N\}$  denote an intermediate codebook after  $m$  iterations.

Find an optimal partition  $P(C_m)$  for the training sequence  $SEQ$  so that the distortion is minimised, that is,

$$P(C_m) = \{S_i; i = 1 \dots N\}$$

where,

$$X_j \in S_i \text{ if } d(X_j, Y_i) \leq d(X_j, Y_k); \quad \forall k \neq i$$

And then compute the mean distortion after  $m$  iterations:

$$D_m = D(C_m, P(C_m)) = \frac{1}{n} \sum_{j=1}^n \min\{d(X_j, Y_i)\}; \quad Y_i \in C_m$$

3. If the rate of distortion reduction  $\frac{D_{m-1} - D_m}{D_m} < \varepsilon$ , the iteration process is completed and  $C_m$  becomes the final optimised codebook. Otherwise, the iteration process continues.
4. Find the optimal reproduction vectors for the partition just formed and update the intermediate codebook  $C_m$ , that is,

$$C_m = X' \{P(C_m)\} = \{\text{cent}(S_i); i = 1 \dots N\}$$

Then increment the iteration counter  $m$  by 1 and go back to step 2.

An important result of the GLA is that each application of the Lloyd iteration must reduce the average distortion of a codebook or leave it unchanged. Practical implementations of the algorithm have already confirmed its effectiveness. However, one should be aware that the GLA does not have the ability to locate a globally optimal codebook because distortion measures such as the mean squared error generally have many local minima, and the monotonically decreasing nature of the algorithm may lead to the nearest local minimum associated with the initial codebook. For this reason, stochastic relaxation techniques are sometimes applied during the codebook optimisation process in an effort to obtain a globally optimal codebook.

The aim of incorporating stochastic relaxation techniques into the codebook optimisation process is to avoid the pitfall of possible local minima as sometimes experienced by the GLA. The key idea in applying stochastic relaxation techniques is that during each iteration process, a set of independent parameters associated with the

cost function of an optimisation process is perturbed in a random fashion. However, in order to ascertain the convergence of these techniques, it is necessary that the magnitude of such perturbation be gradually reduced with time. In the case of codebook optimisation problems, for example, noise can be added to the training vectors prior to each application of the Lloyd iteration, and subsequently the variance of the noise source is gradually reduced to zero. Alternatively, after each computation of centroid vectors, a zero-mean noise is added to each of them. The variance of the noise is also reduced with successive iterations according to a predetermined scheme.

## **2.5 Proposed block adaptive classified vector quantisation**

### **2.5.1 Variable block-size coding with adaptive block segmentation**

A vector quantiser is generally considered to be adaptive if either its codebook or its encoding arrangement changes in time in response to any variations in observed local statistics of its input vector source (Gersho & Gray, 1992). As such, variable block-size vector quantisation is an adaptive coding algorithm due to its block-size adaptation. Similarly, classified VQs can be regarded as adaptive VQs due to its switched codebook adaptation.

For most block-based compression applications, it has been found that an effective way to further exploit the statistical dependency among image pixels is to use variable block-size coding with adaptive block segmentation. This approach is necessary because images generally experience different local statistical properties. While image blocks with slow intensity variations can be adequately coded using larger block sizes so as to increase compression efficiency, those with significant visual features should be dealt with using smaller block sizes to reduce visual artefacts (Buhler et al, 1987).

Being one of the highly efficient segmentation algorithms, a quadtree segmentation strategy has been adopted in this study for the implementation of adaptive block segmentation. As such, input images are adaptively segmented into variable sized image blocks based on a quadtree segmentation structure. That is to say, for every quadtree segmentation operation being applied to an image block, four image sub-blocks with smaller block size are produced. In theory, this segmentation process can

be applied recursively as many times as necessary to meet segmentation requirements. However, in order to facilitate the implementation of subsequent vector quantisation process, only two levels of quadtree segmentation were used, resulting in two different image block sizes of 8x8 and 4x4 as shown in Figure 2.11.

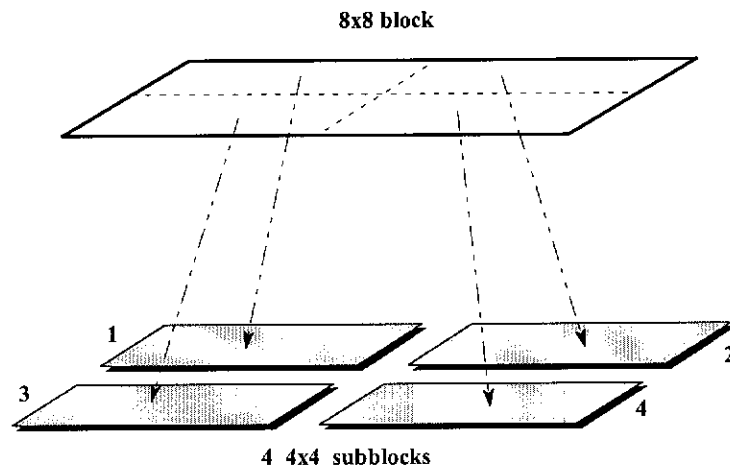


Figure 2.11: Adaptive block segmentation structure

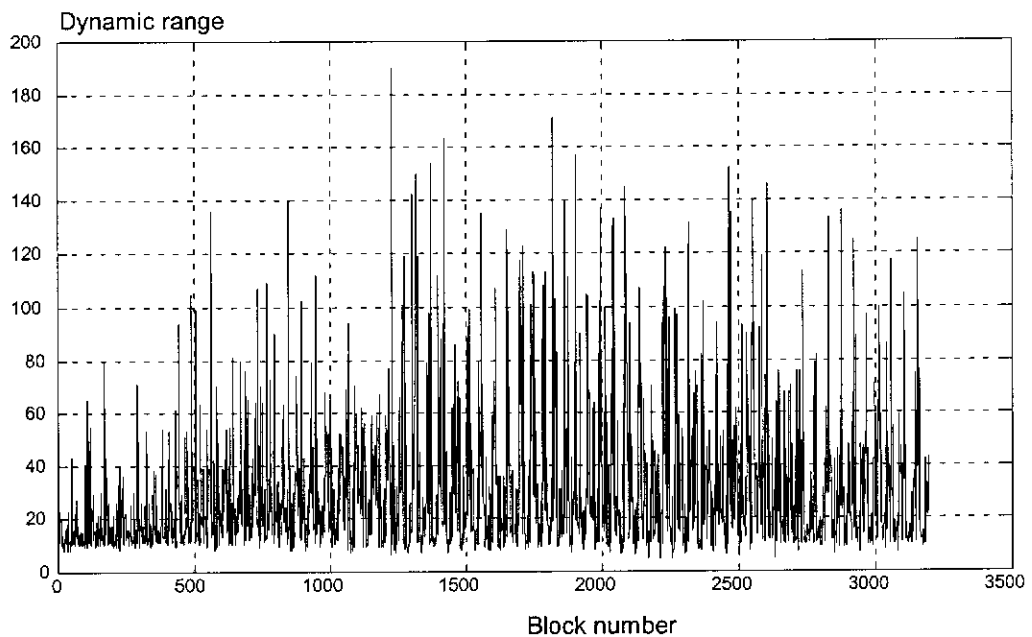


Figure 2.12: Dynamic range variation of 8x8 smooth blocks for image 'Lenna'

To implement the proposed adaptive block segmentation process, local activity of individual image blocks is selected as a segmentation criterion for controlling adaptive segmentation operations. Initially, an input image is segmented into image blocks with a uniform block size of 8x8. Local activity evaluation based on both spatial and transform domains is then performed on these blocks to determine

whether quadtree segmentation is required. According to Wen et al (1993), the two lowest-ordered AC coefficients  $X_{01}$  and  $X_{10}$  of a DCT transformed block generally provide a good indication of the nature of an original image block and therefore can be used for local activity evaluation. Their study indicates that a threshold value of 85 is quite adequate for local activity evaluation purposes and therefore has been adopted in this study.

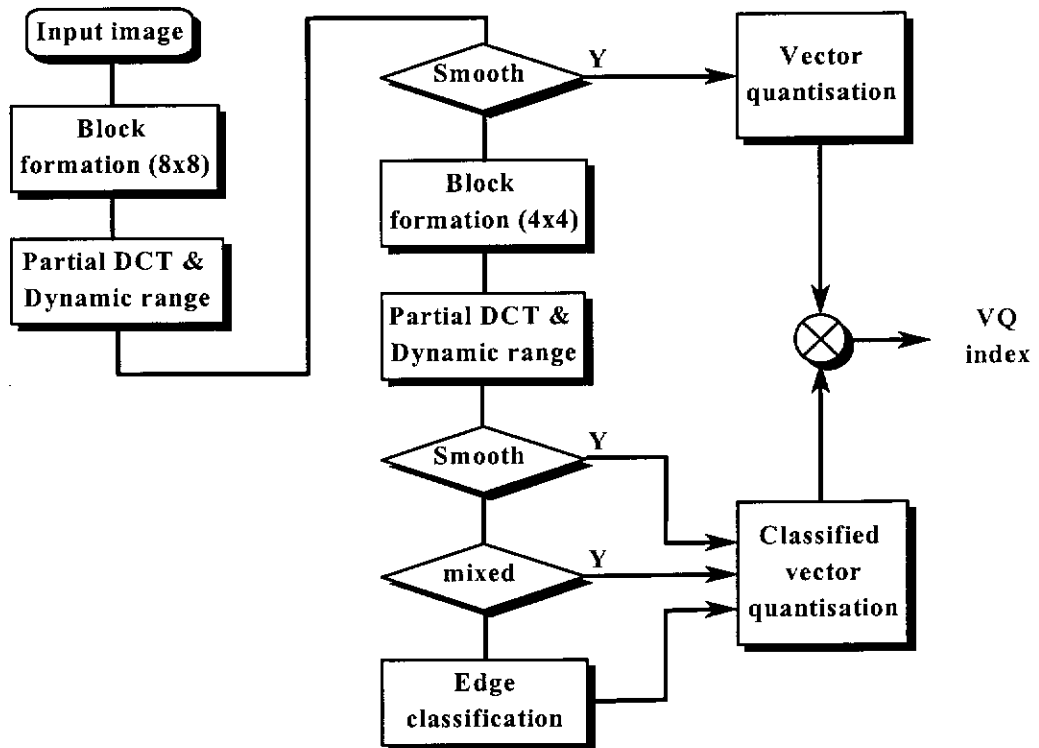


Figure 2.13: Illustration diagram of BACVQ

However, based on our observation, a dynamic range, which is a measure of the difference between the minimum and maximum pixel values within an image block, should be used as an additional measure for local activity assessment to improve the consistency of the adaptive segmentation process. In particular, this improvement enables low activity image blocks to be better characterised because in order to be qualified as low activity, not only should image blocks have small values in their two lowest-ordered AC coefficients, but they also need to have a low dynamic range. These requirements have been observed to better reflect common visual attributes among low activity image blocks. As shown in Figure 2.12, because a majority of low activity blocks were observed to have a relatively low dynamic range, a

threshold value of 120 can be safely applied without adversely affecting the coding efficiency of the proposed algorithm. So, for every 8x8 input image block, if both of its lowest-ordered AC coefficients and its dynamic range are lower than their corresponding threshold values, it will be evaluated as a low activity or smooth block and can be coded adequately using direct VQ. Otherwise, it will be characterised as a high activity block and will be subject to further segmentation into four smaller sub-blocks of 4x4 block size. Each of these sub-blocks is sequentially coded using classified VQ in order to enhance the perceptual quality of reconstructed images, especially in terms of edge integrity. A flow chart illustrating the operation of the proposed adaptive block segmentation algorithm is presented in Figure 2.13.

### **2.5.2 Classified VQ**

Early studies of VQs for image compression applications have revealed that image coding using a single optimal codebook constructed based on a large training set is only effective for image blocks containing shade and non-edge features (Ngan et al, 1992). The reasoning for this is two fold. Firstly, for a typical image, since there is a large proportion of shade and non-edge areas, a normal codebook design procedure based on a large training set will allocate more code words to shade and non-edge vectors and fewer to edge vectors, thus resulting in edge vectors being coded with excessively higher distortion. Secondly, a more serious problem facing the single codebook approach is that the widely adopted distortion measure MSE for VQ encoding operations does not have edge preservation properties. This means that a codebook search based on the MSE criterion does not guarantee an edge vector will always be coded with an edge code word, hence severely affecting the perceptual quality of reconstructed images. Fortunately, these weaknesses of VQs can be effectively overcome by using classified VQs (Ramamurthi et al, 1986). A block diagram illustrating the operation principle of classified VQ is given in Figure 2.14.

To address the former problem, normal codebook generation techniques for classified VQs generally allow more flexible control over how sub-codebook sizes should be allocated to different sub-codebooks in order to minimise the overall distortion. To solve the latter problem, classified VQs indeed offer a more radical solution. During the encoding process, since input image blocks are always classified into a specific

image class prior to vector quantisation using a corresponding codebook, they are guaranteed to be coded with those codewords in the same image class. A detailed discussion of classified VQ has already been given in Section 2.4.6.

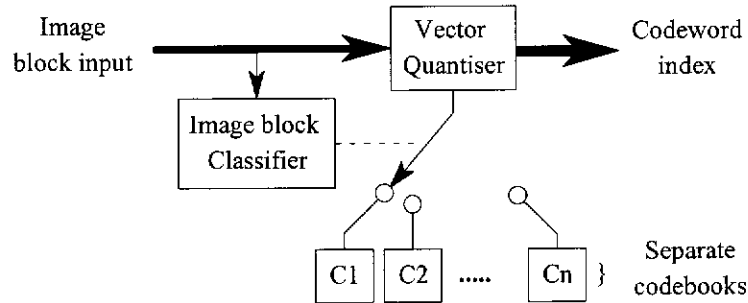


Figure 2.14: Classified VQ operation

Because of the significant advantages offered by classified VQ, it has been adopted in the implementation of the proposed BACVQ algorithm. Following the adaptive segmentation process, classified VQ is applied to code all 4x4 image sub-blocks to take into account a higher level of detail associated with these sub-blocks, especially those containing edges. For practical reasons, a total of 16 image classes have been established in this study to cover major visually distinct features. These include smooth patches, mixed texture, and edges with various orientations and positions among image sub-blocks. Accordingly, 16 sub-codebooks have been generated to support the operation of classified VQ.

In addition to the determination of the number of sub-codebooks, image sub-block classification plays a crucial role in classified VQ. Not only is it a necessary process during the operation of classified VQ, but it also has substantial influence on the coding performance of classified VQ. Basically, what is required from the image sub-block classification process is that it should be capable of classifying image sub-blocks into the predefined image classes with reasonable accuracy so that they can be coded appropriately with classified VQ using correct sub-codebooks.

### 2.5.3 Image sub-block classification

The classification for smooth image sub-blocks is achieved by evaluating the dynamic range of pixel values within image sub-blocks. Based on Wen et al's results

(1993), a similar threshold value of 18 was adopted in this study for this classification. However, in order to exploit the contrast sensitivity of the human visual system (HVS), this threshold value is allowed to vary according to the average intensity  $I_0$  of image sub-blocks. For low intensity blocks, it has been observed that the HVS tends to be more sensitive to the variation of pixel values, and therefore necessitating lowering the dynamic range threshold for smooth block detection. As such, three difference threshold values have been empirically determined for this classification process as shown below:

- If  $I_0 \leq 165$ , threshold = 18;
- If  $165 < I_0 \leq 220$ , threshold = 23; and
- If  $I_0 > 220$ , threshold = 28.

In addition, transform domain techniques were also adopted in this study for the classification of other image sub-blocks. First of all, let  $X_{00}$ ,  $X_{01}$ ,  $X_{10}$  and  $X_{11}$  denote the four lowest-ordered DCT coefficients of a transformed image sub-block. It then follows that a mixed texture sub-block can be detected if both ratios  $(X_{01} / X_{11})$  and  $(X_{10} / X_{11})$  are below a certain threshold value. This threshold value should be judiciously selected in order to give the best indication of a mixed texture sub-block. Specifically, our experimental results seem to indicate that a threshold value of 0.7 is quite adequate for the detection of mixed texture sub-blocks, that is,

- if  $\left| \frac{X_{01}}{X_{11}} \right|$  and  $\left| \frac{X_{10}}{X_{11}} \right| < 0.7$  then a mixed texture block; otherwise an edge-like block.

The adoption of the above ratios is justified because they reflect the relative strength of the vertical and horizon to the mixed frequency contents, thereby allowing the detection of high texture sub-blocks. Experimentally, it has been observed that for a typical high texture sub-block shown in Figure 2.15, coefficient  $X_{11}$  of the transformed block tends to have a much higher value than the vertical and horizontal coefficients  $X_{01}$ ,  $X_{10}$ . Therefore, in order to detect this condition, thereby allowing the detection of high texture sub-blocks, the above ratios have been adopted due to its simplicity.



100	50	100	50
50	100	50	100
100	50	100	50
50	100	50	100

300 (X00)	0 (X01)	0	0
0 (X10)	14.6 (X11)	0	35.3
0	0	0	0
0	35.3	0	85.3

Original block x

Transformed block X

Figure 2.15: An original and transformed high texture block

As long as image sub-blocks fail to satisfy both smooth and high texture classification criteria, they will be treated as edge sub-blocks. These sub-blocks are subject to further classification into one of the 14 edge-like classes as shown in Figure 2.16. The allocation of these edge-like classes is necessary to take into account typical edge orientations and positions within image sub-blocks.

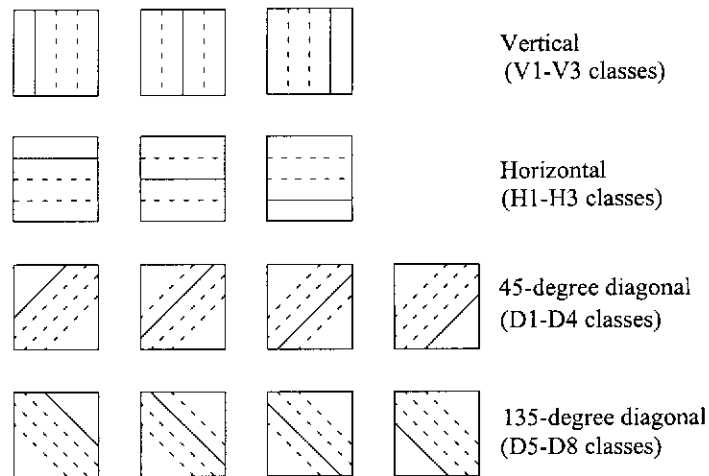


Figure 2.16: Various classes for blocks containing edges

During the edge classification process, the orientation of the dominant edge within an edge sub-block is first evaluated based on transform domain techniques. Experimentally, it has been found that the relative difference between the coefficients  $X_{01}$  and  $X_{10}$  serves as a good indication for the dominant edge orientation within an image sub-block. For simplicity, this relative difference is compared against an empirically chosen threshold value so that an appropriate decision about edge orientation can be made. If the relative difference exceeds the threshold value, then the image sub-block will be associated with either vertical or horizontal class, due to

the dominance of its vertical or horizontal coefficients. Otherwise, it is assumed to contain a diagonal edge, that is,

- For  $X_{01} > X_{10}$  , if  $\frac{X_{01} - X_{10}}{X_{10}} > 0.67$  then it is treated as a vertical image sub-block; otherwise, a diagonal block results.
- For  $X_{01} \leq X_{10}$  , if  $\frac{X_{10} - X_{01}}{X_{01}} > 0.67$  then it is classified into a horizontal image sub-block; otherwise, a diagonal block results.

As the final stage of edge orientation detection process, a sub-block with a diagonal edge is analysed to determine whether it contains a 45-degree or 135-degree diagonal edge. It has been found that this classification is most efficiently implemented by taking into account the polarity of the coefficients  $X_{01}$  and  $X_{10}$ . If both of them have the same sign, then a 45-degree diagonal sub-block is detected; otherwise, a 135-degree sub-block results, that is,

- if  $(X_{01}X_{10}) > 0$  then it is treated as a 45-deg diagonal sub-block; otherwise, a 135-deg diagonal block results.

Once the edge orientation has been determined, the edge position within an image sub-block shall be evaluated based on the contrast sensitivity of the HVS in the spatial domain. Such an evaluation enables the dominant transition in intensity across the sub-block to be detected, thereby revealing the corresponding position of an edge. The contrast sensitivity in this case is given by,

$$\text{Contrast sensitivity} = \frac{2 \times (I_1 - I_2)}{(I_1 + I_2)}$$

where,  $I_1$  and  $I_2$  represent the average intensities of pixels in adjacent columns (for vertical edge sub-blocks), rows (for horizontal edge sub-blocks) or diagonal segments (for diagonal edge sub-blocks).

As shown in Figure 2.16, three possible edge positions are assigned for image sub-blocks with either vertical or horizontal edge orientations, hence resulting in three vertical and three horizontal edge classes V1, V2, V3, H1, H2 and H3, respectively. For image sub-blocks with either 45-degree or 135-degree diagonal edges, however,

four edge positions are allocated each, thus leading to eight diagonal classes starting from D1 to D8.

#### 2.5.4 BACVQ Codebook design

As shown in Figure 2.17, two main codebooks of equal size are empirically allocated for the operation of BACVQ. These codebooks contains 4096 code words each. Due to the high sensitivity of the HVS to visual artefacts in image regions containing smooth 8x8 blocks, all code words in the first codebook are dedicated for coding those blocks. In addition, this coding arrangement is favourable because all the code words in this codebook have the same dimension of 8x8, therefore facilitating the implementation of the vector quantisation process.

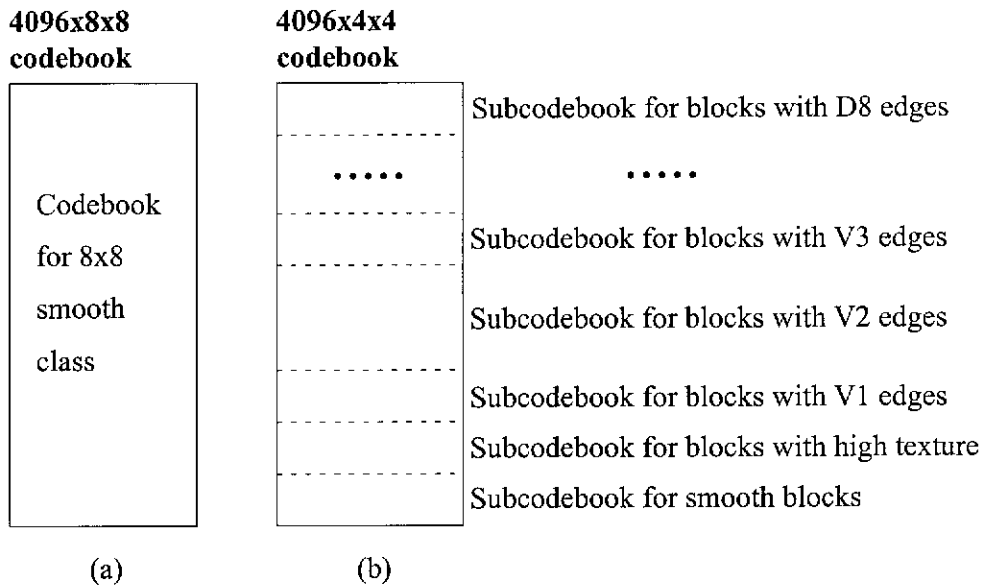


Figure 2.17: Codebook structures for BACVQ

The second codebook, however, is shared among all the possible image classes and shall only be used for encoding 4x4 image sub-blocks. The first two sub-codebooks in the second codebook are associated with the smooth and high-texture classes while the remaining 14 sub-codebooks are dedicated for the edge classes. All of these sub-codebooks are concatenated to one another to form the second codebook. This specific arrangement of the codebook was adopted because it was anticipated that different sub-codebooks are likely to have different sizes as a result of the optimal sub-codebook size determination process. A well-known codebook size optimisation algorithm has already been discussed in Section 2.4.6. In addition, because all sub-

codebooks in this experiment have the same vector dimension, this arrangement enables the classified VQ process to be implemented more consistently and efficiently.

Since the visual characteristics of the smooth and high-texture classes are quite distinct from those of the edge classes, the sub-codebook size for these two classes needs to be determined separately (Lee et al, 1994a). Specifically, sub-codebook sizes of 128 and 64 were empirically allocated for encoding smooth and high-texture image sub-blocks, respectively. These sub-codebook sizes were adopted on the grounds that high-texture image sub-blocks can generally be coded with higher distortion than smooth sub-blocks due to the masking effects of the HVS. In addition, as far as reconstruction distortion is concerned, these small sub-codebook sizes were observed to have minimal impact on the perceptual quality of reconstructed images.

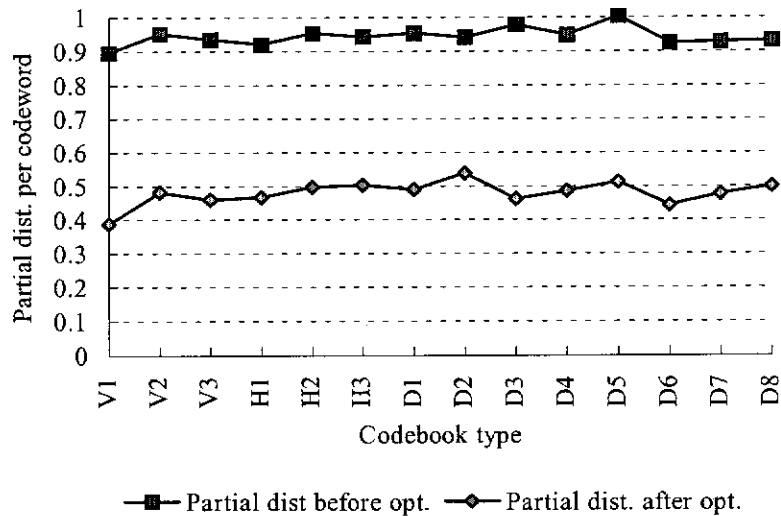


Figure 2.18: Partial distortion among various edge-like codebooks

For the edge classes, their sub-codebook sizes have to be determined in such a way that the overall average distortion resulting from encoding edge sub-blocks be minimised. A highly effective principle that can be used to meet this requirement is based on Ramamurthi et al's work (1986) and has already been discussed in Section 2.4.6. Based on this principle, the Classified Nearest Neighbour Clustering algorithm proposed by Kubrick et al (1990) offers a systematic approach to determine the optimum codebook sizes for image classes containing edges. However, because its codebook design is based on the pair-wise nearest neighbour design algorithm

discussed in Section 2.4.8, its computation complexity soon becomes a problem, particularly when large codebook training sets are involved. Therefore, an alternative approach has been proposed in this study to overcome this complexity problem. Based on the experimental results shown in Figure 2.18, it was observed that the average distortion resulting from using initial codebooks generally has a strong correlation with that using optimised ones. As such, in our implementation, the calculation of the average distortions  $D_i$  was intentionally based on the initial sub-codebooks rather than the optimised ones. This way of calculating the average distortions was adopted throughout the codebook size optimisation process, hence resulting in a significant reduction in computation complexity. The major steps involved in this optimisation algorithm are briefly outlined as follows:

1. Classify all image sub-blocks extracted from a training set into corresponding classes to form sub-training sets for sub-codebook generation purposes.
2. Set all sub-codebook sizes  $N_i$  and current step sizes  $H_i$  to half the size of the corresponding sub-training sets.
3. Construct all initial sub-codebooks using the pruning technique.
4. Compute the average distortions  $D_i$  and the partial distortions per codeword  $A_i$ .
5. Check the overall codebook size  $\sum N_i$  :
  - If  $\sum N_i$  is greater than the target overall codebook size  $N$ , then reduce the size  $N_i$  of the sub-codebook that has the lowest partial distortion by half of its current step size. Update the current step sizes  $H_i$  and then go back to step 3.
  - If  $\sum N_i$  is less than  $N$ , then expand the size  $N_j$  of the sub-codebook that has the maximum partial distortion by half of its current step size. Update the current step sizes  $H_j$  and then go back to step 3.
  - If  $\sum N_i$  is actually equal to  $N$  and the minimum current step size is greater than 5, then reduce the size  $N_i$  of the sub-codebook having the minimum partial distortion and expand the size  $N_j$  of the sub-codebook having the maximum partial distortion simultaneously by half of their current step sizes. Update the current step sizes  $H_i$  and  $H_j$  and then go back to step 3.

6. Otherwise, the search for the optimal sub-codebook sizes is considered to be complete.

As discussed earlier, VQ codebook generation generally consists of two phases: initial codebook generation followed by codebook optimisation. In this study, all initial codebooks are generated based on the pruning technique. This approach of generating initial codebooks offers two key advantages. Firstly, since only representative vectors with most significant distinction are chosen from the training set for inclusion into a codebook, this approach effectively rules out the possibility that similar code vectors be added into the codebook. In addition, it is believed that this result also contributes favourably to the generation of optimised codebooks during the codebook optimisation process, although this requirement of non-repeating code word is not explicitly specified in most codebook optimisation algorithms. Secondly, the operation of the pruning technique for initial codebook generation is relatively simple with moderate computation and memory requirements, therefore facilitating the overall implementation. In particular, given the fact the codebook sizes of classified VQ sub-codebooks need to be iteratively optimised, the computation simplicity of the pruning technique allows the sub-codebook size optimisation process to be carried out more efficiently.

During BACVQ coding operation, the processing of 4x4 image sub-blocks is considered to be highly efficient due to the adoption of classified VQ, whereas that of 8x8 image blocks is computationally intensive because of the involvement of higher vector dimension and larger codebook. However, in processing 8x8 image blocks, it was found that the processing time could be significantly reduced by adopting an appropriate partial search strategy. In particular, since all 8x8 image blocks are characterised as having relatively low intensity variations, a partial search strategy based on the average block intensity is indeed very effective. As such, following the process of codebook optimisation, the code vectors in the resulting 8x8 codebook were sorted in the order of ascending average intensity level in order to facilitate the implementation of the proposed partial search algorithm. However, because codebook search based solely on the average intensity could not guarantee to produce best matched code vectors, a small number of adjacent code vectors with lower and

higher average intensity are also included during codebook search as illustrated in Figure 2.19. These lower and upper bounds are separately recorded in an index table to accelerate the partial codebook search operation. For a reasonable compromise between search results and complexity, a threshold of 30 additional code vectors was empirically adopted for determining the lower and upper bounds of the partial code book search operation. Compared with a full search implementation, the processing time to compress a typical 512x512 image on a PC/486DX33 using this partial search approach was reduced approximately from 5 minutes to 1 minute with a small *PSNR* degradation of 0.03dB.

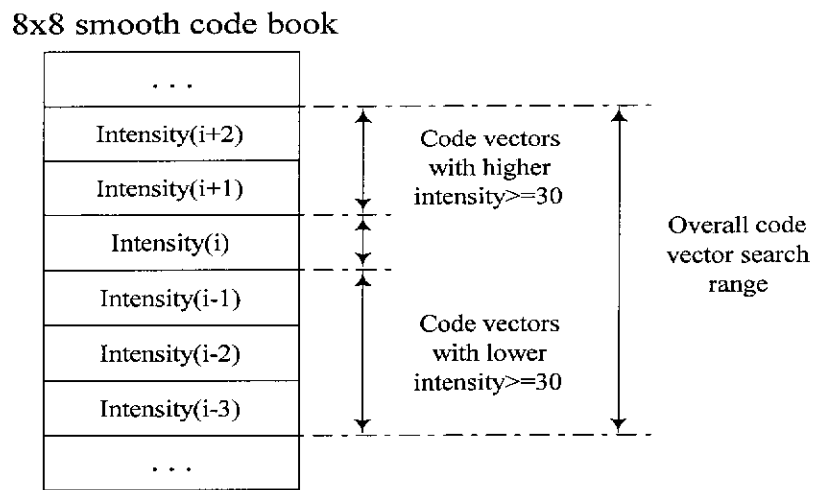


Figure 2.19: Partial codebook search range

### 2.5.5 Bit allocation

Due to the adaptive segmentation scheme and the specific codebook arrangement of BACVQ, a flexible bit allocation scheme for controlling bit allocation to codeword indices was adopted as illustrated in Figure 2.20. For 8x8 image block coding, it can be seen that 12 bits need to be allocated in order to uniquely identify any codeword from the first code book which consists of 4096 codewords. In addition, one more additional bit is required for codebook identification. As such, a total of 13 bits is necessary for coding every 8x8 image block.

For 4x4 image sub-blocks, however, because they are always coded in groups of four consecutive sub-blocks in a coding stream, only the codeword associated with the first 4x4 sub-block needs to be coded explicitly. This implies that in coding the first

image sub-block, one more additional bit has to be allocated for the second codebook identification purposes, in addition to the 12 bits used to uniquely identify the corresponding codeword within the second codebook during classified VQ. For the remaining three image sub-blocks, only 12 bits are needed for each of them. In other words, a total of 49 bits shall be required to encode every group of four consecutive image sub-blocks.

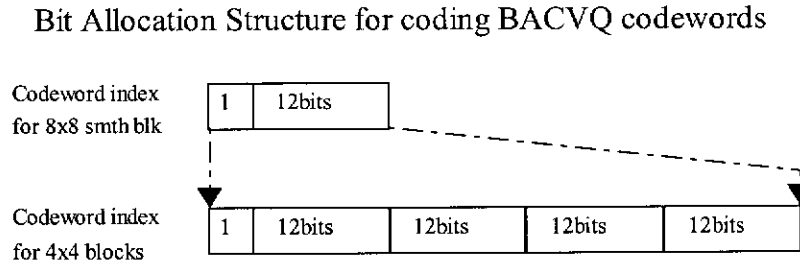


Figure 2.20: Bit allocation for BACVQ code words

### 2.5.6 Simulation results

Being an efficient still image compression algorithm, BACVQ has indeed demonstrated its great potential for still image compression applications in terms of high compression ratio, low computational complexity and high perceptual reconstructed picture quality. Our simulation results indicate that reconstructed images with *PSNR* ranging from 28.2 to 35.3dB can be achieved at a coding rate between 0.2784 to 0.461bpp. These results are tabulated in Table 2.2 and Table 2.3 for the case of non-training and training test images, respectively. In addition, the results from these tables confirm that a large portion of image region is characterised as smooth areas, even the most complex one like the ‘Baboon’ containing more than 50% of smooth region. Therefore, the application of adaptive block segmentation helps increasing the compression performance of BACVQ substantially because potential bit savings can be easily achieved in smooth regions where larger block sizes can be used in conjunction with VQ. In addition, a large number of 4x4 edge sub-blocks in those tables makes it necessary to adopt adaptive block segmentation because they are coded more efficiently using smaller block sizes.

Even though samples of original and reconstructed images shown in Figure 2.22 and Figure 2.23 may not reveal any significant differences in terms of subjective quality



between the training and non-training test images, the actual grey-scale images as observed from computer screen do confirm that higher degradation is encountered for the case of the non-training images. This is mainly because smooth 8x8 blocks in the non-training images are coded with significantly higher distortion than those in the training images as indicated in Table 2.2 and Table 2.3. As a result, the blockiness among the non-training images becomes much more noticeable, hence affecting their perceptual quality to a great extent. In addition, the higher distortion experienced by the smooth 8x8 blocks may suggest that there must be a significant difference in the statistical properties associated with the smooth 8x8 blocks between the non-training and training images. Therefore, in order to improve the robustness of BACVQ coding algorithm, it is necessary that a more general training image sequence should be used during the codebook training process. Apart from the blocking effects, it is also observed that fine details in the reproduced images are not so well preserved. This is because part of the region with fine details such as around the hat and hair areas of image ‘Lenna’ has been misclassified as smooth 8x8 blocks instead of high texture or edge ones by the image block classifier, hence causing the fine details in those misclassified blocks to be lost. However, it is believed that this problem can be fixed by fine-tuning the threshold parameters of the image block classifier and modifying the adaptive segmentation algorithm slightly if necessary.

Apart from a few coding problems associated with 8x8 smooth block coding, consistent results in coding 4x4 image sub-blocks have been achieved for both training and non-training images. Table 2.2 and Table 2.3 show roughly the same level of distortion across all the 4x4 image sub-blocks. As a result, the difference in subjective quality in those regions containing 4x4 image sub-blocks are generally not noticeable for both training and non-training images. In addition, although the quantitative distortion in coding edge-like and high texture sub-blocks is significantly higher than that of smooth 4x4 sub-blocks, the difference in subjective quality among those sub-blocks is not well perceived mainly due to the ‘masking effect’ of the HVS. This result reconfirms that high detailed sub-blocks can generally be coded with higher distortion, and that MSE is not always adequate for measuring subjective quality of reconstructed images.

Edge codebook	V1	V2	V3	H1	H2	H3
Training vectors	3701	2995	2992	3370	3396	3501
Codebook size Nj	470	442	386	478	537	478
Classifying probability	0.1371	0.1110	0.1109	0.1249	0.1258	0.1297
Avg dist before opt.	3068.4	3791.9	3256.5	3519.9	4072.9	3470.0
Partial dist before opt.	0.8953	0.9520	0.9353	0.9195	0.9544	0.9417
Avg dist. after opt.	1327.4	1924.3	1599.1	1791.4	2124.4	1849.8
Partial dist. after opt.	0.3873	0.4831	0.4593	0.4680	0.4978	0.5020

Edge codebook	D1	D2	D3	D4	D5	D6	D7	D8
Training vectors	1118	712	743	1084	1003	652	664	1057
Codebook size Nj	162	120	133	157	148	120	122	151
Classifying probability	0.0414	0.0264	0.0275	0.0402	0.0372	0.0242	0.0246	0.0392
Avg dist before opt.	3725.9	4279.8	4720.9	3704.6	3998.3	4586.7	4597.8	3588.2
Partial dist before opt.	0.9528	0.9409	0.9772	0.9478	1.0040	0.9234	0.9272	0.9307
Avg dist. after opt.	1911.9	2450.5	2229.2	1898.4	2035.8	2200.0	2361.1	1924.2
Partial dist. after opt.	0.4889	0.5387	0.4614	0.4857	0.5112	0.4429	0.4762	0.4991

Table 2.1: Adaptive allocation of codebook sizes to various edge codebooks

Images	Overall results					8x8 smooth		4x4 smooth		4x4 edge		4x4 mixed	
	MSE	PSNR	bpp	CR	Time	Blocks	MSE	Blocks	MSE	Blocks	MSE	Blocks	MSE
Lenna	62.69	30.16	0.3308	24.18	1'11"	3166	46.80	543	17.20	2989	125.03	188	273.35
Man	97.98	28.22	0.3492	22.91	1'21"	3032	80.17	710	14.70	3313	167.59	233	288.88

Table 2.2: Results of coding non-training images with BACVQ

Images	Overall results					8x8 smooth		4x4 smooth		4x4 edge		4x4 mixed	
	MSE	PSNR	bpp	CR	Time	Blocks	MSE	Blocks	MSE	Blocks	MSE	Blocks	MSE
Airplane	33.88	32.83	0.3524	22.70	1'03"	3009	12.74	813	11.60	3401	107.23	134	207.02
Baboon	89.77	28.60	0.4613	17.34	1'28"	2216	16.94	125	22.66	6387	138.31	1008	431.05
Peppers	37.46	32.40	0.3392	23.58	1'05"	3105	21.80	724	15.47	3118	97.66	122	223.90
Sailboat	55.36	30.70	0.3940	20.30	1'15"	2706	20.47	524	15.18	4792	129.71	244	229.43
Spash	19.14	35.31	0.2865	27.92	0'58"	3489	15.17	1091	8.04	1325	68.43	12	200.15
Tiffany	29.02	33.50	0.2784	28.74	0'56"	3548	19.23	620	14.23	1496	112.48	76	334.84

Table 2.3: Results of coding training images with BACVQ

BACVQ	MSE	PSNR	BPP	CR	Notes on various implementations of BACVQ
#1	66.87	29.87	0.3324	24.06	Codebook 1 (2048): Smooth 8x8 block Codebook 2 (256+256): Smooth 4x4 and mixed blocks Codebook 3 (4096): Blocks with edges
#2	64.15	30.05	0.3414	23.42	Codebook 1 (4096): Smooth 8x8, smooth 4x4 and mixed blocks Codebook 2 (4096): Blocks with edges
#3	62.69	30.16	0.3308	24.18	Codebook 1 (4096): Smooth 8x8 blocks Codebook 2 (4096): Smooth 4x4, mixed and edge blocks

Table 2.4: Results obtained from various BACVQ for non-training image "Lenna"

Table 2.1 presents some statistical results relating to the process of determining optimal sub-codebook sizes for the edge classes. It is noticed that the number of training vectors for both vertical and horizontal edge classes is significantly higher than those for diagonal edge classes. Therefore, more codewords are allocated to the

former classes in order to minimise the overall average distortion. Table 2.1 also shows that the partial distortion per codeword in each edge class is reduced roughly by half when initial and optimised codebooks are used in the evaluation respectively. Such a correlation has enabled us to determine optimal sub-codebook sizes with significantly lower computation complexity by using initial codebooks.

In this study, the operation of the BACVQ coding algorithm with various codebook configurations has also been investigated for their effects on the overall perceptual quality of reconstructed images, thereby enabling us to optimise the coding performance of BACVQ. As indicated in Table 2.4, results for coding non-training image ‘Lenna’ using three different codebook configurations are presented for comparison purposes. It can be seen that the last codebook configuration offers the best coding performance in terms of both coding rate and average distortion.

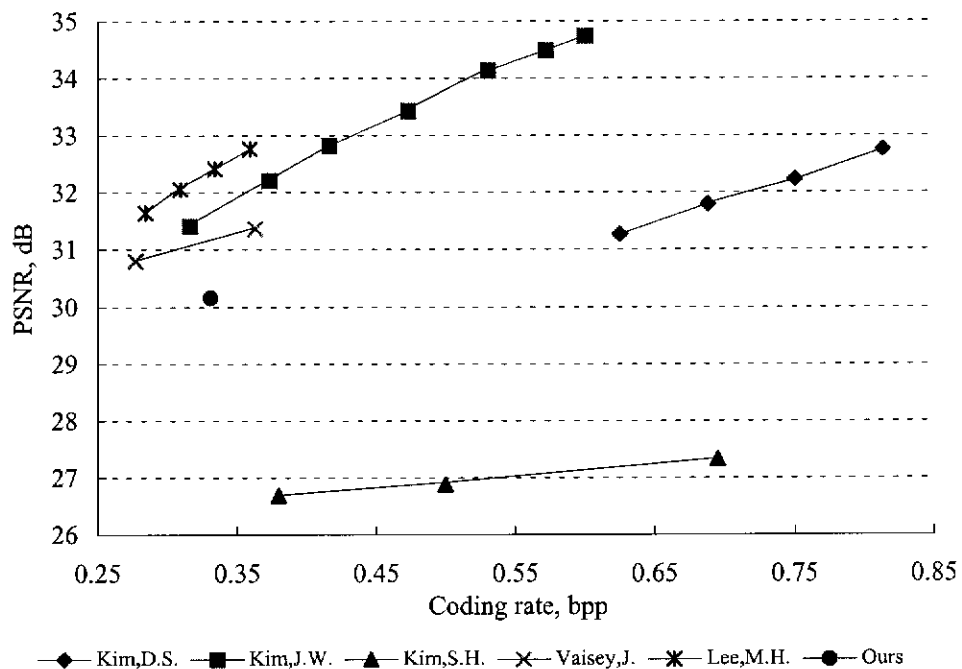


Figure 2.21: Various coding results for ‘Lenna’ image

Finally, in order to evaluate the merit of BACVQ, results from Lee et al’s work (1994a) are also quoted for comparison purposes. As shown in Figure 2.21, the proposed BACVQ algorithm indeed achieves very encouraging results. Its performance in terms of coding rate and distortion is only slightly lower than other high performance coding schemes. However, given the fact that the largest block size

employed in BACVQ is only 8x8 as opposed to 16x16 in other schemes, slightly inferior in its coding performance is inevitable. In addition, since the coding algorithm of BACVQ does not allow the coding rate to be changed so easily during the coding process, this has somewhat restricted our experiment and only allowed us to project a single result instead of a distortion curve on to the graph shown in Figure 2.21.



(a), (c) and (e): Original test images of “Lenna”, “Tiffany” and “Baboon”; (b), (d) and (f): Corresponding reconstructed images coded at coding rate/PSNR of 0.3308bpp/30.16dB, 0.3524bpp/32.83dB and 0.4613bpp/28.60dB, respectively. Only test image “Lenna” is not part of the training set.

Figure 2.22: Sample results of BACVQ coding process (set #1)



(a) and (c): Original test images of “Man” and “Airplane”; (b) and (d): Corresponding reconstructed images coded at coding rate/PSNR of 0.3492bpp/28.22dB and 0.3524bpp/32.83dB, respectively.

Figure 2.23: Sample results of BACVQ coding process (set #2)

## 2.6 Huffman coding and arithmetic coding

In general, entropy coding such as Huffman or arithmetic coding is applied as the final stage of the encoding process to remove any redundancy remaining from preceding stages without adding any errors (Vetterli et al, 1992). This is possible because entropy coding is basically a lossless coding algorithm. Therefore, given the fact that entropy coding is lossless in nature and relatively simple to implement, there would be no reason why entropy coding should not be employed in the final stage of an overall encoding process.

Depending on image contents, compression gain of lossless coding techniques for still image compression applications may range from unfavourable expansion to 4:1 optimal compression. In fact, it has been shown that the ultimate limit of the compression gain for lossless compression techniques is totally governed by Shannon's source entropy (Gray et al, 1992).

Being the most popular entropy coding algorithm, Huffman coding is generally considered as a powerful statistical encoding technique that reduces the amount of data needed to represent a given sample. As mentioned earlier, since this is a lossless compression technique, original data can be exactly recovered from encoded data. Generally, a compression ratio of 2:1 can be expected from Huffman coding (Fedele et al, 1988).

Another popular entropy coding algorithm is arithmetic coding. In arithmetic coding, it has been shown that the average number of code symbols allocated to individual input sample tends to converge to the entropy of the input source if an encoded sequence is sufficiently long (Vetterli et al, 1992). As a result of this, arithmetic coding is a coding technique that effectively matches the code symbol allocation process to the source entropy, thereby maximising the overall compression efficiency. Compared to Huffman coding, arithmetic coding is generally more complicated to implement but in return offers greater compression.

## **2.7 Summary**

Among various coding techniques, transform coding has been regarded as the longest established coding algorithm for still image compression applications. Its advantages in these applications have been well perceived, including the effectiveness in de-correlating image elements, the regularity of its coding arrangements, and the ability to exploit the deficiency of the HVS. However, transform coding also suffers some major drawbacks, most noticeably, the blocking and ringing effects. Various forms of linear transformation are available for transform coding implementations. While KLT is considered as the optimal linear transformation statistically, DCT is regarded to be the most efficient linear transformation for image compression applications. A common feature among these forms of transformation is that they are all based on

orthogonal linear transformations. This particular class of linear transformation possesses many properties which are highly desirable in image compression applications. The adoption of separable linear transformation significantly reduces the complexity of transform coding, especially when a multi-dimensional signal is involved. Apart from the issue of linear transformation, the order in which transform coefficients are quantised also has a great impact on the overall performance of transform coding.

Another potential coding technique for image compression applications is subband coding. In essence, this technique can be considered as a typical frequency domain compression algorithm. Its operation principle is to decompose original input signals into various subbands before compression process actually takes place. As such, a filter bank is generally required for subband decomposition purposes. However, in order to ease implementation problems and to improve reconstruction quality, heavily constrained filter banks are to be used. As in the case of transform coding, the adoption of separable filter banks greatly facilitates the implementation of subband coding. Another important aspect of subband coding is concerned with the decimation and interpolation process. While the role of the decimation process is to overcome the problem of increased subband information as a result of the subband decomposition process, that of the interpolation process is to facilitate the recovery of the original signal.

From the theoretical point of view, wavelet coding can be regarded as a special subband coding technique in which a logarithmic decomposition tree structure is used for decomposing input signals. Its development is primarily motivated by recent studies of human visual system modeling.

As an alternative compression technique, vector quantisation have revealed its great potential for image compression applications. Being considered as Shannon's model of source coding, VQ operates on the principle of encoding a group of signal samples together instead of one at a time as in the case of scalar quantisation, thereby allowing the redundancy among signal samples to be exploited more efficiently. In addition, Shannon's rate-distortion theorem has provided a strong foundation for the



studying of VQ. In fact, the advantages of VQ are well established not only by analytical results but also by practical applications. Unfortunately, practical implementations of VQ usually suffer some significant setbacks unless constraints are imposed on the structure of its codes and fast search algorithms are adopted.

Another key issue concerning the design of a vector quantiser is that its operation needs to satisfy the conditions of optimality, namely, nearest neighbour condition, centroid condition and zero probability boundary condition. Although these conditions are necessary, they are not sufficient to assure the optimality of a vector quantiser. At worst, they can only guarantee it to be locally optimal. Nevertheless, these conditions have already had significant influence on the operation of nearest neighbour vector quantisers, VQ codebook design techniques and codebook optimisation process.

Optimum VQ usually insists on the use of large vector dimension and high coding rate. However, the problem of this is that its codebook size and codebook search complexity increase exponentially with the product of vector dimension and coding rate. Therefore, for most practical applications, coding constraints are frequently imposed on the coding structure of VQ with little sub-optimality to improve its coding performance. In fact, this has motivated the development of a wide range of constraint VQs and the exploration of various fast nearest neighbour search algorithms.

As an alternative approach to improve the coding performance of VQ, BACVQ has been proposed for still image compression applications. While its coding efficiency is achieved mainly by applying variable block-size coding with adaptive block segmentation, its performance in terms of the perceptual quality of reconstructed image is significantly improved by applying classified VQ. During the adaptive segmentation process, local activities in both the transform and spatial domains are evaluated together in order to improve the consistency of the segmentation process. This results in larger block size being associated with low activity blocks and smaller one being assigned to those with high activities. For low activity blocks, the application of VQ together with high vector dimension accounts for the high coding

efficiency of this algorithm because real-world images usually contains a large portion of low activity regions. For high activity blocks, the application of classified VQ together with lower vector dimension ensures that their visually important features are well preserved, especially in terms of edge integrity. Apart from studying the coding structure of BACVQ, an alternative approach has been developed to determine the optimal codebook sizes for classified VQ sub-codebooks with edge features.

Simulation results for BACVQ coding have indicated that reconstructed images with *PSNR* ranging from 28.2 to 35.3dB have been obtained at a coding rate between 0.27 and 0.46 bit-per-pixel. Although the proposed compression algorithm offers slightly inferior coding performance than other high performance coding schemes quoted by Lee et al (1994a), this is likely because a smaller block size has been adopted for low activity block coding in this study.

Now that various aspects of still image compression techniques in general and the proposed BACVQ in particular have been studied, the next chapter will cover the coding concept of image sequence compression. The transition of focus from still image compression to image sequence compression is embraced as a natural course of investigation because still image compression can basically be considered as part of image sequence compression operation. Apart from discussing some fundamental differences between still image and image sequence compression, which arise from the need to remove temporal redundancy in image sequences, the next chapter will elaborate the idea of incorporating still image compression into image sequence compression operation and examine its role in the image sequence compression process. And lastly, the operation principle of the proposed adaptive spatial/temporal coding algorithm for image sequence compression will be presented in conjunction with some computer simulation results.

## **Chapter 3: Image Sequence Compression**

In this chapter, the topic of image sequence compression is presented. Unlike still image compression, image sequence compression needs to apply motion compensated predictive coding to achieve greater compression efficiency. Various adaptive coding algorithms have also been developed aiming at improving their compression performance. In spite of these efforts, further improvements in coding performance are still possible because of the complexity of the entire compression process. In particular, scene segmentation, motion compensation and estimation strategy and adaptive coding strategy are among the main areas attracting significant attention due to their significance in the image sequence compression process.

In this chapter, an adaptive spatial/temporal image sequence compression algorithm is presented as an alternative algorithm for image sequence compression. In this algorithm, an adaptive scene segmentation scheme with significantly better scene segmentation consistency has been developed. Improved image-block classification and adaptive algorithms have also been proposed for variable block-size motion compensation. An alternative motion vector estimation strategy has been established to improve motion estimation consistency and to ease implementation issues associated with motion estimation operation. The application of BACVQ algorithm, developed in the previous chapter, to the intra-frame and residual frame compression process is also discussed.

### **3.1 Introduction**

In recent years, although the rapid advance of digital technology has already benefited many applications in highly diversified areas, its deployment in the field of digital television is somewhat limited due mainly to its implementation difficulties in dealing with a huge amount of visual data. In particular, while digital video compression has been regarded as a key technology to reduce transmission bandwidth and mass storage requirements, its practical implementation generally encounters enormous difficulties, especially when real-time operation is involved. For this reason, many recent research activities have been directed to improving the performance of image sequence compression techniques. While intra-frame coding

techniques such as transform coding, subband and wavelet compression, vector quantisation, and other hybrid coding schemes discussed in the previous chapter can be applied to remove spatial redundancy, inter-frame compression based on motion compensated predictive coding generally offers a more efficient technique for removing both spatial and temporal redundancy within video sequences, thereby significantly enhancing their compression performance (AT&T, 1993) (Li et al, 1994) (Zafar et al, 1991).

For image sequence compression applications, scene segmentation is usually considered as an essential operation not only to facilitate implementation but also to improve overall compression performance (Lee & Dickinson, 1994). Although fixed scene segmentation scheme like the MPEG standard is frequently adopted due to its simplicity, better coding performance can normally be achieved by adopting an adaptive scene segmentation approach (Puri & Haskell, 1992). This is because under fixed scene segmentation, it is highly possible that scene transition picture frames may be included within a group of pictures (GOP) structure, resulting in uncorrelated picture segments, which in turn affects severely the efficiency of subsequent motion compensated coding process. By contrast, adaptive scene segmentation can effectively eliminate this problem by ensuring that scene changes are always placed at the boundary of GOP structures. It accomplishes this by using temporal correlation as its segmentation criterion. Recently, Lee & Dickinson (1994) have reviewed five different schemes for evaluating temporal correlation between consecutive image frames within an image sequence. These schemes are referred to as Difference of Histogram (DOH), Histogram of Difference (HOD), Block Histogram Difference, Block Variance Difference and Motion Estimation Errors. In general, the main features that differentiate these schemes include their sensitivity to local/global motion and computation complexity.

Although the HOD scheme offers a reasonably good measure for temporal correlation evaluation, its application in adaptive scene segmentation operation sometimes suffers some setbacks such as false scene transition detection and missed scene change detection. For this reason, an improved scene segmentation criterion called "Magnitude of HOD variations" has been proposed for the adaptive

segmentation process. Our experimental results have indicated that scene segmentation based on this criterion indeed offers highly accurate scene transition detection with high tolerance to spurious image activities, and yet requires virtually the same level of computation complexity as the HOD scheme.

In addition, as a result of adopting adaptive scene segmentation with relatively long inter-reference frame intervals, an alternative approach of progressive motion estimation technique has been proposed for the implementation of motion compensated predictive coding. The main advantages of this technique include offering fixed motion search range regardless the length of inter-reference frame intervals, allowing the motion vector field to be coded more efficiently, and most importantly the ability in dealing with overlapping motion among moving objects in an image sequence more effectively.

Basically, for image sequence compression applications, an intuitive approach to reduce the amount of visual data in a video sequence is to apply frame skipping techniques. These techniques are effective because successive frames within video sequences usually differ slightly from one frame to another (Rocca & Zanoletti, 1972). Conceptually, this approach is equivalent to applying sub-sampling principle to the temporal direction. For example, if every other frame of an image sequence is discarded, then a compression gain of 2:1 shall be achieved. However, such a simple approach usually leads to poor perceptual quality during picture reconstruction. On the one hand, if no temporal averaging is applied during the reconstruction process, this 2:1 temporal decimation operation will cause jerkiness appearance among moving objects. On the other hand, if temporal averaging is applied, image blurs then become a problem (Fedele et al, 1988) (Musmann et al, 1985).

Apart from frame skipping techniques, temporal DPCM coding can also be applied to image sequence compression as illustrated in Figure 3.1 (Fedele et al, 1988). Being based on the same operation principle as the conventional DPCM, temporal DPCM coding also operates on the difference signal resulting from the prediction process rather than on the original signal. Because the difference signal generally has much lower energy than the original one, the quantisation process applied to the

difference signal is normally more efficient, hence leading to an overall improvement in the performance of the image sequence compression process (Musmann et al, 1985). However, without additional modifications, it is unlikely that a simple temporal DPCM coding would be able to function satisfactorily in compressing image sequences, especially when those containing high motion are involved. The ineffectiveness of simple temporal DPCM is due to the fact that the brightness of picture elements in an image sequence is generally independent in moving areas although the information content of images remains more or less the same. In addition, the ineffectiveness can also be accounted for by the fact that simple temporal DPCM coding does not take into account object movements within an image sequence during its coding process (Rocca et al, 1972). It is for these reasons that motion compensated DPCM coding has become increasingly popular among image sequence compression techniques (Zafar et al, 1991).

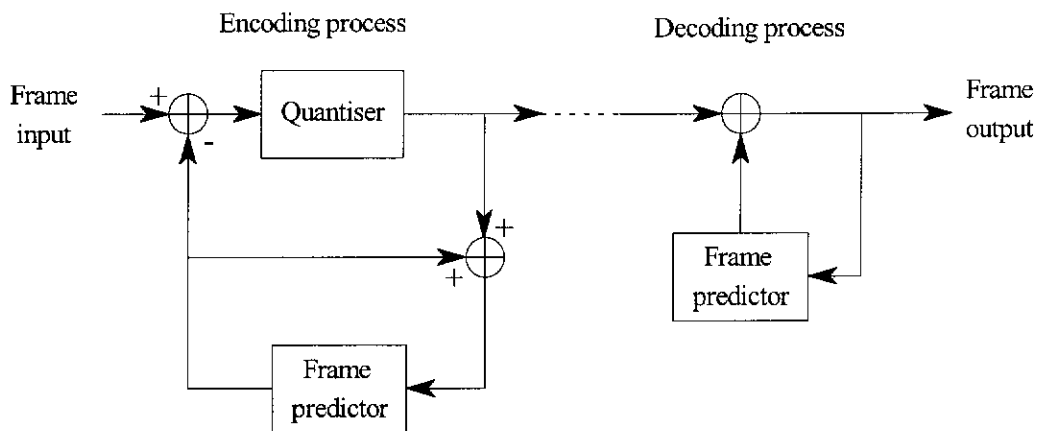


Figure 3.1: Typical temporal DPCM coding

Being a special case of predictive coding techniques, motion compensated DPCM coding is expected to offer significant bit-rate reduction by quantising motion compensated prediction errors with an optimised quantiser. For image sequence compression applications, it has been found that subjectively optimised quantisers are in general more efficient than statistically optimised quantisers because the latter tends to produce too many levels for small prediction errors. As a result, the application of some forms of visual weighting functions in quantiser design has been considered to be important, even though the optimum choice of such functions has not been well established yet. Nevertheless, for broadcast television applications, an optimum quantiser design shall consider visibility thresholds and provide a smallest

number of levels with minimal noticeable visible impairment. Apart from subjectively optimised quantisers, adaptive quantisers are regarded as an alternative approach for quantising motion compensated prediction errors. In this approach, a suitable quantiser is dynamically selected based on the merit of a local activity indicator (Musmann et al, 1985).

Due to the significance of motion compensated predictive coding in image sequence coding, intensive studies have been conducted in an effort to further improve its coding performance. Specifically, since the effectiveness of applying motion compensated predictive coding to image sequence compression depends largely on the accuracy of motion estimation strategy, considerable research work has been undertaken to improve its efficiency (Zafar et al, 1991). In general, motion estimation strategy can be classified into two broad categories, namely, pel-recursive algorithms and block-based techniques. Although pel-recursive algorithms inherently offer a more representative approach for estimating motion in natural scenes, their application in practice has not been very successful due to their high computation complexity and overheads. For block-based approach, although deformable block-based motion estimation techniques such as Hierarchical Grid Interpolation proposed by Huang & Hsu (1994) or Generalised Block-Matching by Seferidis & Ghanbari (1994) are capable of handling more general types of motion such as translation, deformation, rotation and zooming, their application is also highly restricted because of their computation complexity. Therefore, a more common approach is to assume that motion in video sequences is of translational types, thereby enabling the development of various block matching algorithms (BMA).

As the name implies, the operation principle of BMA is based on a block matching process among image blocks in order to determine an underlying motion vector. This implies that for every input image block for which motion needs to be estimated, it is assumed that there exists a single motion vector that describes the translational motion between the current image block and a corresponding one from an adjacent picture frame. As such, the application of BMA during motion estimation process generally enable a considerable reduction in computation. However, due to the assumption of uniform translational motion within individual image blocks, BMA is

also known to have a relatively poor performance when dealing with image sequences experiencing complex motion. Nevertheless, it has been found that as image block size gets smaller, the adverse effects of the uniform motion assumption of BMA become subdued. It is for this reason that variable block size motion compensation has become increasingly popular among image sequence compression algorithms (Chan et al, 1990) (Lee & Crebbin, 1994b). Specifically, an improved variable block size motion compensation scheme has been investigated in this study. In order to achieve better adaptivity and improve coding performance, motion estimation error has been used as a segmentation criterion during the variable block size motion compensation process because it gives a direct indication of how effective the motion estimation and motion compensation process is going to be. In addition, provisions have been made to allow the motion compensated predictive coding process to switch to intraframe coding mode in case of unsatisfactory motion compensation results. This condition typically occurs when overlapping motion among moving objects is involved.

In implementing BMA, although optimal search results can only be obtained with exhaustive search, other sub-optimal search strategies usually offer a very attractive trade-off between search outcome and computation efficiency. Their computation efficiency is generally achieved by means of matching criteria simplification, search point reduction or a combination of both. In particular, an improved motion estimation (ME) strategy has been proposed in this study. Its implementation is motivated by our observation that motion estimation errors are more noticeable in low motion activity regions. Therefore, in order to take into account this factor, the proposed ME strategy has been implemented based on the modified 2D search algorithm with additional search points in the central search region. In this way, not only does the proposed ME strategy inherit the good performance of the modified 2D search algorithm, but it also has the capability to estimate accurately any small motion with negligible increase in computation complexity, especially when large block size is used.

For image sequence compression applications, it is important to notice that reconstructed image quality can be significantly improved by taking into account the



deficiency of human visual system (Fedele et al, 1988). For instance, in regions of motion, it has been found that spatial resolution can generally be lowered since human eyes have difficulties in resolving high spatial-frequency details in motion areas. Similarly, temporal resolution can be greatly reduced in static image regions because only picture information in motion areas needs to be transmitted.

### **3.2 Human Visual System**

Recent studies of human visual system (HVS) have revealed that judicious exploitation of HVS can lead to substantial improvement in coding performance of many still image and image sequence compression techniques. By taking advantages of the deficiency of HVS, these compression techniques have been able to achieve not only substantial compression gain but also significantly less noticeable degradation in reconstructed picture quality. In general, all of these compression techniques operate based on the same principle that visual data which are less sensitive to or even could not be perceived by HVS are maximally compressed with greater distortion in order to maximise compression efficiency, whereas those with visual importance are coded with minimum distortion to enhance the overall perceptual quality of reconstructed pictures.

In regions of motion, it has been observed that spatial resolution can be greatly reduced since human eyes have difficulties in resolving high spatial frequency details in motion areas (Fedele et al, 1988). Similarly, temporal resolution can be reduced in static image areas because these areas can be replenished using previously stored visual data and hence only picture information in motion areas needs to be transmitted. For image sequence compression applications, these observations have enabled image sequence compression algorithms to dramatically improve their compression gains while at the same time enhancing reconstructed picture quality. In particular, in compressing image sequences, image regions involving high motion activities can be coarsely compressed, whereas those with static appearance can be replenished using previous scene information.

In addition, it has been found that HVS generally does not perceive equal changes in luminance equally (Jain, 1981). Instead, the visual sensitivity of HVS seems to have

a nearly uniform response to changes in contrast, which is given as:

$$C = \log L \quad \therefore \quad \Delta C = \frac{\Delta L}{L} \approx \text{const}$$

where  $C$  and  $L$  denotes the contrast and luminance levels, respectively.

This result suggests that distortion in image regions with higher luminance intensity is less noticeable than that with lower luminance intensity. As such, for better overall compression performance, image regions with high luminance intensity shall be compressed with higher distortion whereas those with low luminance intensity should be coded with higher fidelity.

Furthermore, it has been observed that when a major scene change occurs, the subjective sensitivity of HVS to visual artefacts in an image is substantially reduced until HVS adapts itself to the new scene. This phenomenon is generally referred to as temporal masking effects (Rocca et al, 1972). In general, temporal masking can be classified into two broad categories, namely, forward and backward masking effects (Lee & Dickinson, 1994). In the former case, temporal masking occurs when an arriving stimulus acts forward in time. Whereas, in the latter case, temporal masking takes place when subsequent stimulus affects those already happened. For convenience, it is customary to refer the forward temporal masking phenomenon as temporal masking effects.

One of the well established results from the study of temporal masking effects is the principle of temporal summation, or sometimes known as Bloch's law (Lee & Dickinson, 1994). It states that below a certain critical duration, luminance perception of human eyes is found to be constant, provided that the product of stimulus duration and stimulus intensity is maintained at a constant level. As such, this phenomenon implies that it takes time for human eyes to actually interpret the content of images, and therefore suggesting that HVS indeed has limited temporal resolution. This result is of particular importance to image sequence compression applications because it enables compression algorithms to exploit temporal redundancy among consecutive image frames of an image sequence more effectively. In particular, it has been found that in coding image sequences, the immediate next

frame following a scene transition can be coarsely compressed to lower the overall bit rate with little impact on the perceptual quality of reconstructed pictures.

Apart from forward temporal masking effects, backward temporal masking effects of HVS can also be exploited to enhance the coding performance of image sequence compression algorithms. Specifically, it has been found that the immediate past frame at the instance of scene changes can also be coarsely coded. In fact, according to Lee & Dickinson (1994), both immediate past and next frames at the instance of scene transition can be coded at just 20% and 5% of the normal bit rate respectively with little perceptual impairment.

### **3.3 Scene Segmentation**

In image sequence compression, one of the key operations that needs to be performed initially on input image sequences is scene segmentation. Its prime purpose is to divide lengthy image sequences into GOPs of manageable lengths for subsequent processing. This segmentation process is necessary because not only does it greatly facilitate hardware implementation, but it also provides a mechanism allowing correlated picture frames of an image sequence to be grouped together, thereby enabling temporal redundancy among picture frames to be exploited more efficiently.

In order to fully eliminate temporal redundancy within an image sequence, it is necessary that its temporal correlation be evaluated and used as a criterion for scene segmentation operations. As such, this adaptive segmentation process would ensure that not only shall correlated image frames be partitioned into the same GOPs, but boundaries of GOPs shall also be placed at the instance of scene transitions where temporal correlation experiences a sudden disruption. Depending on the duration of temporal correlation, this approach generally results in GOPs of variable lengths.

However, applying the above adaptive scene segmentation scheme to image sequence compression also suffers some drawbacks. In particular, its unconstrained scene segmentation structure causes not only implementation problems in terms of processing hardware, but also unnecessary processing delay. This is because during an unconstrained segmentation process, exceptionally long intervals between inter-

reference frames may be asserted when the duration of temporally correlated frames is sufficiently sustained. On the one hand, processing hardware would require much more data storage for accommodating larger GOPs. On the other hand, the processing of these GOPs could not commence until image data for a whole GOP have been completely acquired. It is for these reasons that unconstrained adaptive scene segmentation schemes based solely on temporal correlation segmentation criteria are rarely adopted in practical applications. Instead, hybrid adaptive scene segmentation schemes based on temporal correlation segmentation criteria and subject to a maximum segmentation length are generally more favourable.

Apart from the capability of segmenting image sequences temporally, the computation complexity of scene segmentation schemes also needs to be taken into account. Ideally, an efficient scene segmentation scheme should be capable of producing consistent segmentation results with minimal computation complexity. Recently, Lee & Dickinson (1994) have reviewed a number of scene segmentation schemes. These schemes, with increasing computation complexity and sensitivity to local activities, includes difference of histogram (DOH), histogram of difference image (HOD), block histogram difference (BH), block variance difference (BV), and motion compensation error (MCE).

In the DOH scheme, the temporal correlation or the distance between two image frames  $fn$  and  $fm$  can be evaluated using the absolute sum of their histogram difference, that is,

$$D(fn, fm) = \sum_{i=0}^{L-1} |h_n(i) - h_m(i)|$$

where,  $D(fn, fm)$  : The distance between image frames  $fn$  and  $fm$ ;

$L$  : The number of histogram levels; and

$h_n(i), h_m(i)$  : The  $i$ -th level histograms of image frames  $fn$  and  $fm$

A typical example of applying the DOH scheme to temporal segmentation is to use luminance histogram as a segmentation criterion. In general, the luminance histogram is considered to be a very efficient index for image content. However, because the luminance histogram approach only examines the statistics of luminance distribution across an entire image as a whole, its application usually suffers poor

sensitivity to local motion activities. Therefore, its application to temporal segmentation has the tendency not to take into account local motion activities when assessing the level of temporal correlation among picture frames. As such, this may adversely affect the segmentation consistency and the coding efficiency of subsequent coding stages.

In the case of the HOD, the distance between two image frames is calculated as:

$$D(fn, fm) = \frac{\sum_{i \in [-\alpha, \alpha]} hod(i)}{\sum_{i=-L+1}^{L-1} hod(i)} = \frac{\sum_{i \in [-\alpha, \alpha]} hod(i)}{N}$$

Where,  $\alpha$  : The threshold for determining the closeness of a pair of pixels;  
 $N$  : The total number of pixels in an image frame; and  
 $hod(i)$  : The histogram of frame difference ( $fn-fm$ )

Comparing to the DOH scheme, the adoption of the HOD scheme for temporal segmentation generally provides more sensitivity to local motion. In addition, a useful feature of applying the HOD scheme to temporal segmentation is that its frame distance  $D(fn, fm)$  has a strong tendency to increase monotonically when two image frames become further apart. As such, its application is expected to improve the consistency of the temporal segmentation process.

Being a variant of the DOH, the BH scheme is also expected to offer improved capability in local motion detection. In the BH scheme, the distance between image frames is defined as follows:

$$D(fn, fm) = \sum_b \sum_i |h_n(b, i) - h_m(b, i)|$$

where,  $h_n(b, i)$  and  $h_m(b, i)$  denote the  $i$ -th level block histograms of the  $b$ -th blocks in frames  $fn$  and  $fm$ , respectively.

From the above expression, it can be seen that the improvement in local motion detection of the BH scheme comes about as a result of evaluating the DOH on a basis of individual image blocks rather than over an entire image. As such, this effectively eliminates the averaging effect of the DOH computation from affecting the detection of local motion.

Similarly, the BV scheme can also be used to achieve improved sensitivity to local motion and it is given by:

$$D(fn, fm) = \sum_b |\text{var}_n(b) - \text{var}_m(b)|$$

where  $\text{var}_n(b)$  and  $\text{var}_m(b)$  denotes the  $b$ -th block variances in frames  $fn$  and  $fm$ , respectively.

As in the case of BH, the BV scheme also improves its local motion sensitivity by confining its computation to individual image blocks. This is necessary to overcome the averaging effect of the variance computation.

Being the best distance measurement criterion, the MCE scheme can be used to achieve optimal temporal segmentation. Its superior performance can be attributed to the fact that MCE actually serves as an indirect measure of temporal correlation between image frames. Therefore, its application to temporal segmentation process is expected to produce optimal segmentation results. However, a major drawback in applying this measurement criterion to scene segmentation is due to its computation complexity. This is because motion compensation by itself is a computationally intensive process and its application just for scene segmentation purposes is hardly justified. Mathematically, the distance measurement criterion used in the MCE scheme is defined as follows:

$$D(fn, fm) = \sum_{\forall i,j} |fm(i,j) - fm'(i,j)|$$

where  $fm'$  denotes the corresponding motion compensated predicted frame of  $fm$  and is derived from the reference frame  $fn$ ; and  $(i,j)$  refers to pixel coordinates within image frames.

### 3.4 Motion compensated coding

#### 3.4.1 Motion compensated predictive coding

Following the scene segmentation process, various image sequence compression techniques are normally applied to compress picture frames in GOPs. In particular, being a highly effective compression technique for image sequence compression, motion compensated predictive coding is usually adopted to remove temporal redundancy among picture frames, thereby enabling image sequence compression

process to achieve a very high compress gain. The application of motion compensated predictive coding to image sequence compression is motivated by the fact that successive picture frames in an image sequence generally differ slightly from one frame to another (Rocca et al, 1972). More importantly, for motion pictures, temporal redundancy usually occurs in the form of physical motion experienced by moving objects in an image sequence.

Basically, motion compensated predictive coding operates in a similar principle as conventional DPCM. Rather than dealing with the original signal directly, a conventional DPCM coder operates on the difference signal as a by-product of the prediction process. Because the difference signal usually has much lower energy than the original one, the process of compressing the difference signal is generally much more efficient. Similarly, motion compensated predictive coding also operates on the difference image instead of the original one so as to enhance its compression performance. However, for motion pictures, since motion is a prominent feature in all image sequences, motion compensated predictive coding has to rely heavily on motion compensation techniques in order to improve its efficiency in image prediction, thereby enabling the energy of the difference image to be significantly reduced and subsequent compression operations on the difference image to be highly efficient. A typical structure of motion compensated prediction coding process is illustrated in the Figure 3.2.

Although it is not shown explicitly in the figure, it should be noted that in forward motion compensation, the predicted frame of the original is obtained based on an estimate between a previously reconstructed frame and the original one. Therefore, in order to enable the original frame to be reconstructed, motion compensated predictive coding needs to include both the displacement vector information and the prediction error resulting from the prediction process in its coded data stream (Naveen & Woods, 1994).

There are several important aspects regarding the operation of motion compensation (Li et al, 1994). Firstly, the choice of motion compensation modes determines the capability of the motion compensation process. For block matching algorithms,

search strategy, block size and motion vector accuracy are among the most important parameters that need to be taken into account because they have significant influence on the overall performance of the motion compensation process. Secondly, in order to eliminate the "blocking effects" as well as increase the coding efficiency, it is necessary to model the underlying motion field accurately. In general, projection modeling such as orthogonal projection and perspective projection provides an effective means to model 2D temporal changes. And yet, for motion pictures, modeling the underlying motion field could be further improved if camera motion and object motion models can be incorporated into its operation. And lastly, although various motion estimation techniques can be chosen depending on the specific motion model in use, their operation basically involves very similar global concepts. In particular, in implementing motion estimation, it is necessary to define an error measurement criterion so that an optimal motion field can be determined.

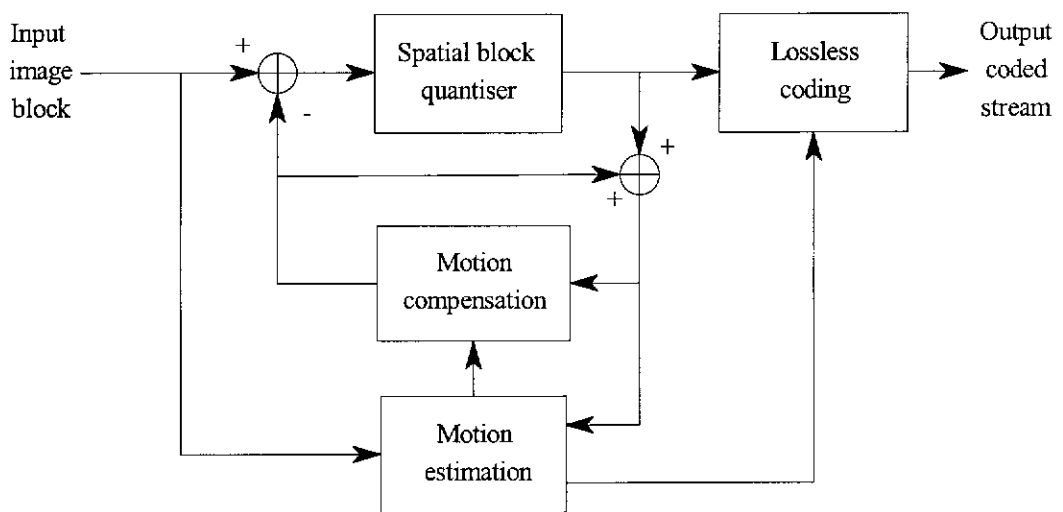


Figure 3.2: General coding structure of a motion compensated DPCM coder

Basically, the establishment of an optimal motion field should minimise motion estimation errors, be of real physical meaning, and be coded with minimum overheads. For local motion estimation, fixed region based motion estimation and adaptive-region based motion estimation techniques can both be used. However, recent experimental results have indicated that the latter generally offers better performance. In addition, the efficiency of motion compensation process can be further improved by separating local activities from global motion. Furthermore, it is desirable that motion compensation process is capable of dealing not only with



geometric transformation such as translation, zooming, panning, and shape deformation, but also with the so-called massive transformation due to illuminance effects on moving objects. And lastly, motion compensation process should take into account overlapping and non-overlapping motion in order to optimise its operation (Li et al, 1994).

In general, motion compensation process can be implemented based on block-based forward motion estimation techniques or pel recursive algorithms (Musmann et al, 1985). In a conventional block-based forward motion estimation approach, motion is usually estimated using block matching algorithms. As such, conventional block-based forward motion estimation typically operates with an assumption that a single uniform motion exists for each individual image block. Apart from block matching algorithms, techniques based on spatial luminance gradient can also be used in motion estimation process. In addition, integer motion displacement is frequently adopted in block-based motion estimation techniques so as to reduce overheads in coding motion vector field as well as to ease implementation difficulties.

By contrast, in the pel recursive approach, motion estimation is performed on a pel-by-pel basis rather than on a block-to-block basis. Because the operation principle of the pel recursive approach does not have to commit itself to the assumption of uniform motion, its application is inherently more accurate, especially when complex motion is involved in high detailed regions. Therefore, its results are generally more representative to real motion in video sequences (Zafar et al, 1991). However, this flexibility of the pel recursive approach does come at a high price. Because motion is estimated down to the pixel level, the computation complexity of the pel recursive approach is significantly higher than that of the block-based counterpart. Also, as motion estimation is performed independently among individual pixels, this strategy hinders itself from taking advantages of spatial correlation during motion estimation process. Furthermore, this approach tends to involve excessive data overheads, thus adversely affecting the overall compression efficiency.

Unlike conventional block-based motion estimation techniques, deformable block-based motion estimation techniques offer a more effective approach for motion

compensation. In general, these techniques are capable of dealing not only with translation motion in planes parallel to the image plane, but also with deformation, rotation and zooming motions. However, a major setback in applying this approach to real-time applications is due to its increased computation complexity. Being a typical deformable block-based motion estimation technique, Hierarchical Grid Interpolation (HGI) has been studied recently by Huang et al (1994). Its operation basically relies on the manipulation of quadrangles and quad-tree segmentation. As such, during the motion estimation process, it tries to displace grid points of the quadrangles so that the underlying motion can be better formulated. Another effective motion estimation scheme is the so-called Generalised Block Matching technique (Seferidis et al, 1994). Its operation principle is based on the use of affine, bilinear or perspective transforms in order to deal with complex motion activities in natural scenes. Algorithmically, the Generalised Block-Matching motion estimation process consists of two stages. Initially, the translational component of motion vectors is determined by applying the traditional full-search block-matching technique with a search range of  $\pm 16$  pixels/frame. Subsequently, transformation parameters are determined by independently displacing the four corner coordinates of the quadrilaterals within a predetermined search range.

In addition to deformable block-based motion estimation techniques, variable block-size motion estimation techniques offer an alternative approach to motion compensation with significantly improved performance. Due to its segmentation efficiency, quadtree segmentation is frequently employed for spatial decomposition during the motion compensation process. For simple implementations, quadtree segmentation decision can be made based on the absolute temporal difference. However, this segmentation strategy is considered as sub-optimum because the segmentation process is directed by the inter-frame difference rather than by the homogeneity of moving objects (Seferidis et al, 1994).

For image sequence compression applications, it is recognised that the more accuracy a motion compensation algorithm is capable of, the lower bit rate a coding process is likely to achieve (Huang et al, 1994) (Chen & Pang, 1992) (Musmann et al, 1985). This is because residual frames resulting from an efficient motion compensated

prediction process are in general more uniform and therefore requiring less bit allocation to them. In fact, it has been shown that the number of bits required to encode residual frames is proportional to the magnitude of motion estimation errors (Wu & Gersho, 1994).

In motion compensated predictive coding, as the main objective of motion compensation is to minimise the variance of prediction errors, this may seem to suggest that obtaining a true motion field for underlying motion is not necessary (Hoang et al, 1994). If for one reason or another a few prediction errors do occur in the motion estimation process, this simply increases the magnitude of the prediction errors. The result is that a few more bits are then required to transmit the prediction errors, which to some extent offset their impact on the quality of reconstructed pictures. It is for this reason that less accurate block-based motion estimation techniques are frequently employed in motion compensated predictive coding (Dubois, 1992). However, for applications involving bidirectional motion interpolation, it is desirable that motion estimation be carried out as accurately as possible. This is because without accurate motion information, temporal interpolation operations would be adversely affected.

#### **3.4.2 Motion compensated interpolative coding**

In addition to motion compensated predictive coding, motion compensated interpolative coding represents another approach for implementing motion compensated coding. Unlike motion compensated predictive coding in which image samples are predicted from a single reference frame, motion compensated interpolative coding reconstructs missing image samples by applying temporal interpolation along motion trajectories established by the two enclosing reference frames at both ends as illustrated in Figure 3.3. It is for this reason that this coding technique is also known as bidirectional interpolative coding or bidirectional predictive coding.

Figure 3.3 reveals that the missing sample represented by an asterisk (\*) at position  $x$  at time  $t$  can be reconstructed from the two enclosing reference frames using temporal interpolation (Dubois, 1992), that is,

$$\hat{u}(x,t) = a\tilde{u}(x - (1-a)dx, t_1) + (1-a)\tilde{u}(x + adx, t_2)$$

where,

$\hat{u}(x,t)$ : a prediction of the missing sample at position  $x$  at time  $t$ ;

$\tilde{u}(x - (1-a)dx, t_1)$ : an estimate of  $\hat{u}(x,t)$  from previous reference frame at time  $t_1$ ;

$\tilde{u}(x + adx, t_2)$ : an estimate of  $\hat{u}(x,t)$  from next reference frame at time  $t_2$ ; and

$a$ : normalised distance between current and next reference frames and is given by:

$$a = \frac{t_2 - t}{t_2 - t_1}$$

In deriving the above expression, it is assumed that image objects experiences uniform motion within the interval from  $t_1$  to  $t_2$ . Also, only one dimension of motion is examined so as to demonstrate the operation principle of motion compensated interpolative coding techniques. Nevertheless, these results can be easily extended to cover two dimensional motion as normally encountered in image sequences as illustrated in Figure 3.4.

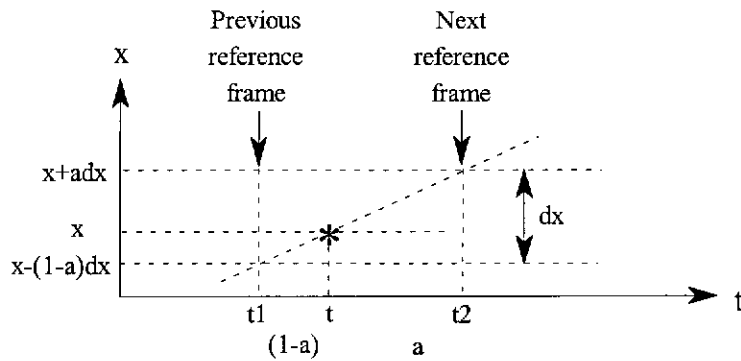


Figure 3.3: Operation principle of temporal interpolation

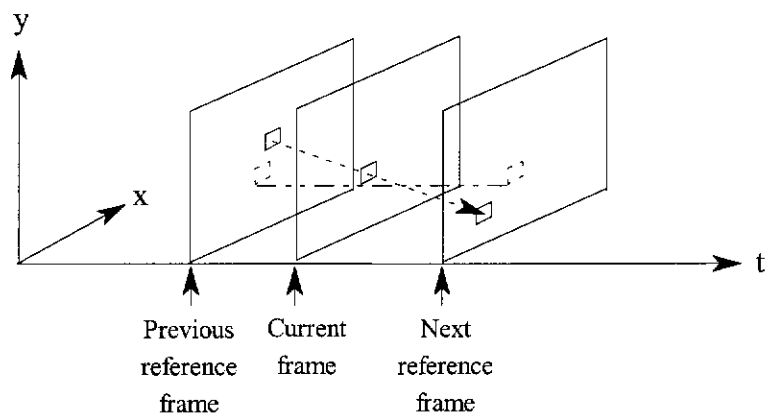


Figure 3.4: Predicting missing image samples using bidirectional temporal interpolation.

Furthermore, it should be noticed that a simple scaling scheme has been incorporated into the above expression in order to improve the accuracy of the temporal interpolation process. Its function is to give more weight to predictions derived from a closer reference frame than to those from a further one. This strategy is justified because predictions from a closer reference frame are in general more accurate and reliable.

Although the operation principle of temporal interpolation suggests that an interpolated picture frame can be fully reconstructed without needing any other information except the two ending reference frames, practical implementations of motion compensated interpolative coding usually require additional information to avoid excessive visual artefacts (Dubois, 1992). This is because during temporal interpolation process, interpolation errors caused by erroneous motion vectors, moving object occlusions and so on are not corrected. Therefore, in order to improve the perceptual quality of reconstructed pictures, it is necessary to include both motion compensated interpolation errors and forward/backward motion vector fields in coded data streams. However, if motion vector fields are sufficiently accurate, motion compensated interpolation errors tend to be negligible and therefore could be excluded from coded data streams altogether.

### **3.4.3 Motion estimation strategy**

As mentioned earlier, motion estimation techniques can be classified into either pel recursive algorithms, in which motion estimation is performed on a pel-by-pel basis, or block-based algorithms where the estimation is carried out on a block-by-block basis. Naturally, pel-recursive algorithms offer a more representative approach in dealing with real motion in image sequences. While the pel recursive approach may produce better results than the block-matching counterpart, the latter often requires much simpler computation because it operates with the assumption that all image blocks of an image sequence exhibit uniform motion. It is for this reason that block matching algorithms are more favourable in practical implementations.

In spite of conceptual simplicity, practical implementation of block matching algorithms using exhaustive search strategy still requires massive computational

power during its operation, especially when sub-pixel or fractional motion estimation accuracy is involved. As a result, various sub-optimal block-based motion estimation techniques have been proposed to improve the efficiency of the motion estimation process. Also, to improve the accuracy of block matching operations, additional techniques such as interpolation can be applied during the motion estimation process (Musmann et al, 1985) (Srinivasan & Rao, 1985). In addition, it should be noticed that the problem of estimating motion vectors based on minimising some distortion function is in fact an optimisation problem. Therefore, in order to improve the performance of motion estimation process, joint estimation of forward and backward motion vectors techniques such as the one proposed by Wu et al (1994) could also be employed. This joint optimisation can be applied to the estimation process of forward and backward motion vectors in an iterative manner and serves as an effective mechanism to minimise the overall motion interpolation errors. However, a major drawback of such a scheme is its increased computation complexity.

As far as matching criteria are concerned, the so-called Normalised 2D Cross-Correlation offers an effective estimation criterion for block displacement estimation. Under this scheme, the displacement of an image block can be determined by the position within a reference window, which coincides with the peak of the Normalised 2D Cross-Correlation function. This reference window is derived from a reference frame from which motion needs to be estimated. Mathematically, the Normalised 2D Cross-Correlation function is defined as (Musmann et al, 1985) (Kappagantula & Rao, 1985):

$$M(p, q) = \frac{\sum_{k=1}^m \sum_{l=1}^n u(k, l) u_r(k + p, l + q)}{\sqrt{\left[ \sum_{k=1}^m \sum_{l=1}^n u^2(k, l) \right] \left[ \sum_{k=1}^m \sum_{l=1}^n u_r^2(k + p, l + q) \right]}}$$

where,  $m, n$ : Block dimension in term of number of pels;

$p, q$ : Shifting coordinate of motion vector;

$u(k, l)$ : Pixel value in the current image block; and

$u_r(k + p, l + q)$ : Corresponding pixel value in the reference block

In addition, the number of search points is another important parameter that determines the capability and computation complexity of the motion estimation

process. Because the search for the underlying motion vector is normally performed within the reference search window with a dimension of  $W$  pixels on each side as indicated in Figure 3.5, an exhaustive search strategy would require a total of  $N$  search operations given by the following expression:

$$N = (W - m + 1)(W - n + 1)$$

This expression suggests that the total number of search operations would grow exponentially with respect to the search window size. Therefore, given the high complexity of the Normalised 2D Cross Correlation computation, a large number of search operations required in the exhaustive search strategy indeed imposes a tremendous computational load on processing hardware and causes excessive processing delay.

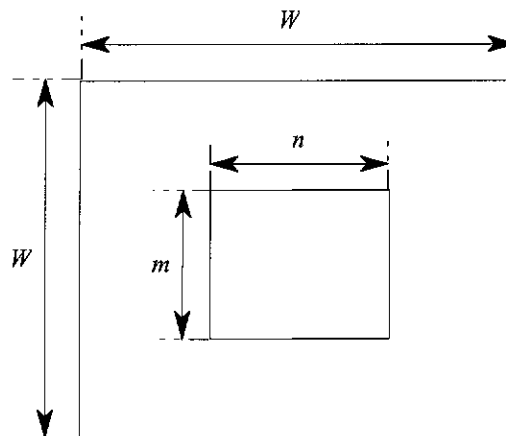


Figure 3.5: Window search range for motion estimation

As an attempt to speed up the motion estimation process, various sub-optimum motion estimation schemes have been proposed. In general, the development of these sub-optimum schemes falls into two broad categories, namely, matching criteria simplification and search point reduction. The former involves replacing the Normalised 2D Cross Correlation function by simpler matching criteria such as the mean squared error or the mean absolute error. The latter is concerned with fast search schemes such as 2D logarithmic search proposed by Jain & Jain (1981), modified 2D search by Kappagantula et al (1985), three-step search by Koga et al. (1981), one-at-a-time search by Srinivasan et al (1985), and so on. In spite of apparent differences, all of these fast search schemes are based on the same

assumption that matching errors shall increase monotonically with respect to motion estimation errors.

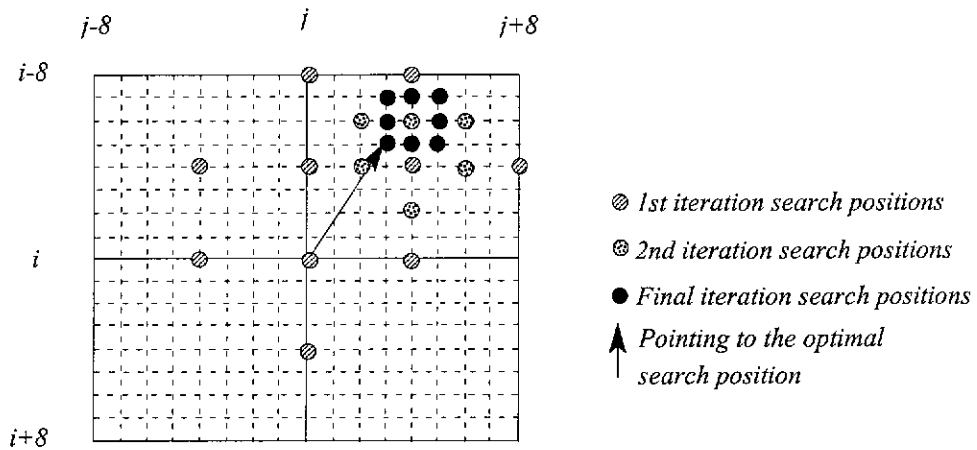


Figure 3.6: Example of 2D logarithmic search strategy

Among the fast search schemes, the 2D logarithmic search strategy can be regarded as a direct extension of the well-known 1D binary or logarithmic search. As such, for every iteration of the 2D logarithmic search strategy, its step size in both directions is reduced by half. When the step size is ultimately reduced to one, all surrounding locations are then searched for the best representative motion vector. In effect, this searching process proceeds coarsely initially and then refines its search progressively as it approaches the end. A typical example illustrating the 2D logarithmic searching process is given in Figure 3.6.

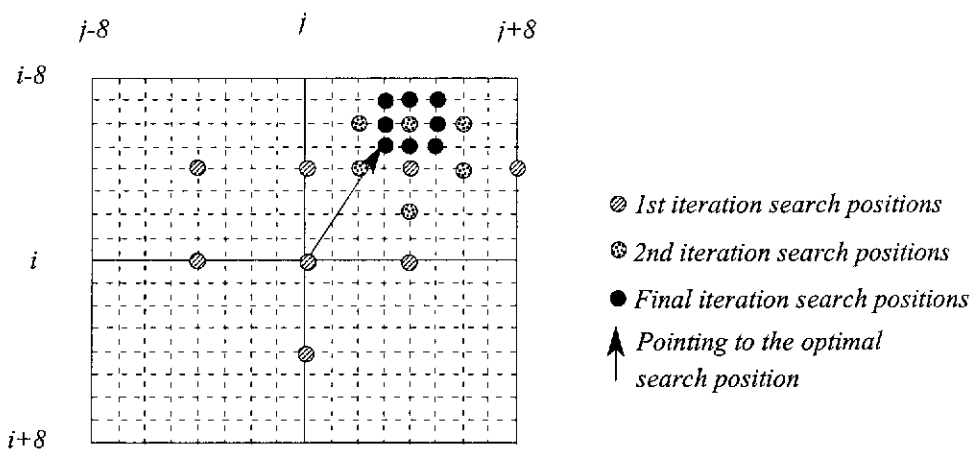


Figure 3.7: Example of modified 2D search strategy

In order to reduce the number of search points of the motion estimation process further, a modified 2D search technique has also been proposed. As the name



implies, this technique is derived directly from the original 2D algorithmic search strategy with some modifications to its search pattern as illustrated in Figure 3.7.

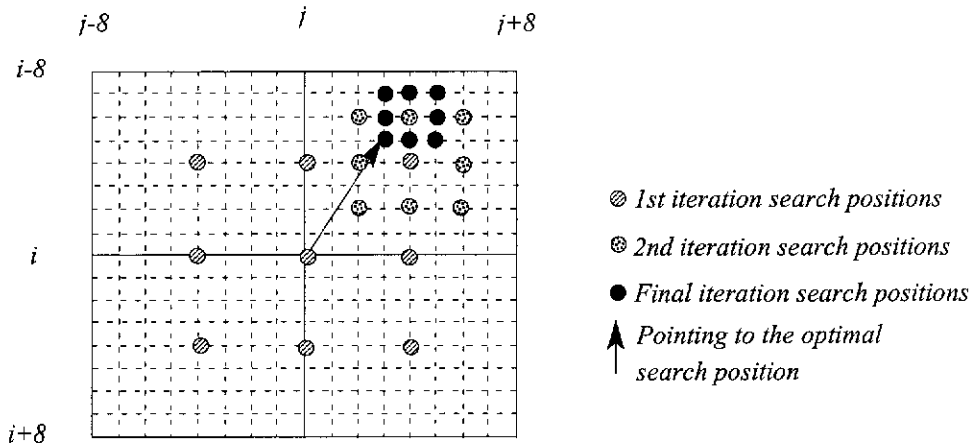


Figure 3.8: Example of 3-step search strategy

The operation principle of the 3-step search can be described briefly as follows: at each iteration, eight search points around the neighbourhood are examined for the direction of minimum distortion. Also, like the already mentioned 2D search strategies, the step size in both directions of the 3-step search strategy is progressively reduced by half following each iteration. A typical example demonstrating this search strategy is presented in Figure 3.8.

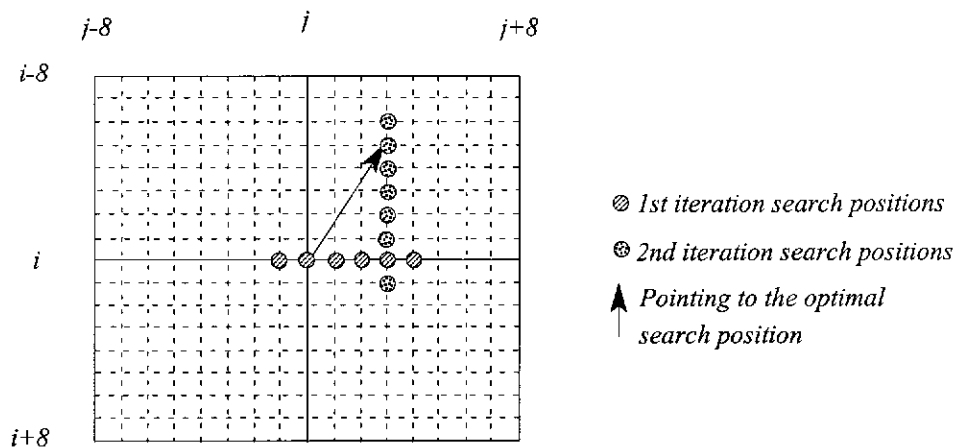


Figure 3.9: Example of modified one-at-a-time search strategy

In the one-at-a-time search scheme, the minimum in both  $i$ - and  $j$ -directions are found in a sequential manner. Within each search, three points consisting of the two adjacent points and a centre are examined progressively until the location of the

minimum direction is located. This direction is found when the location of the smallest distortion coincides with the centre position. Once the minimum in one direction is identified, the minimum in the other is searched in a similar fashion. This search strategy is illustrated in Figure 3.9.

It should be noted that not every fast search algorithm would be able to locate optimal motion vectors at the boundary of a pre-determined search window due to its specific search pattern. For example, given a search window of  $\pm 8$  pixels in both directions, while the 2D algorithmic and one-at-a-time search schemes are able to locate optimal motion vectors up to  $\pm 8$  pels/frame, the modified 2D and 3-step search schemes are only capable of searching up to  $\pm 7$  pels/frames. In addition, there are large variations in the number of searching steps and search points among various fast search schemes as shown in Table 3.1. However, for real-time hardware implementations, the number of required sequential searching steps can be of greater importance than the number of search points. This is because the former directly relates to the latency time of the motion estimation process, and more importantly it determines how effective parallel processing could be deployed to improve its search performance.

Search strategy	Search range (pels/frame)	Maximum number of search steps	Maximum number of search points
Exhaustive or full search	$\pm 8$	1	289
2D logarithm search	$\pm 8$	8	20
Modified 2D search	$\pm 7$	5	21
Three-step search	$\pm 7$	3	25
One-at-a-time search	$\pm 8$	16	19

Table 3.1: Maximum search steps and search points for a  $\pm 8$ -pixel search window.

Although the objective of the above sub-optimum fast search algorithms is to obtain an optimal motion vector that results in minimum distortion or block mismatch, they all have tendency to converge to local distortion minima. This is because motion vector field of image blocks may exhibit a complex contour with several local

minima. For this reason, alternative approaches have been proposed, which attempt to incorporate inter-block correlation into the motion estimation process (Hsieh et al, 1990) (Zafar et al, 1991) (Zhang & Zafar, 1991).

In these alternative approaches, inter-block correlation, which can be derived from motion vectors of adjacent blocks in the same image frame or from those of the past frames, is used to provide useful information on the likelihood of where the optimal motion vector resides. Apart from the benefit of slight complexity reduction, motion vectors estimated in this way seem to be more realistic. In addition, as motion vector fields tend to exhibit strong spatial and temporal correlation, potentially better results can be achieved by judiciously exploiting the correlation among motion vectors in image sequences. Not only does this lead to faster motion estimation operation, but the consistency of the resulting motion vector field also enables it to be coded more efficiently. In fact, it has been reported that a quadtree decomposition could be employed to code motion vector field information, thereby achieving further bit-rate reduction (Hoang et al, 1994).

### **3.5 Adaptive coding techniques**

Adaptive coding techniques using quadtree segmentation have become increasingly popular in image sequence compression applications because of its high segmentation efficiency. In Lee and Crebbin's implementation (1994b), for example, input image frames under intraframe coding mode are segmented into variable-sized blocks using quadtree segmentation. All of these variable-sized blocks are subsequently coded with classified VQ. For inter-frame coding mode, a similar quadtree segmentation is applied to residual frames in order to separate motion regions from stationary background. This approach results in larger block sizes being associated with stationary background and smaller ones with motion regions. Since the texture in stationary background areas tends to change very slowly, image blocks in those areas can generally be replenished from a previously reconstructed frame, thereby enabling a high bit-rate reduction. For motion regions, however, a more elaborate coding technique is used in order to maintain satisfactory reconstructed picture quality. Image blocks in these regions are coded using motion compensated predictive coding technique and classified VQ.

The application of classified VQ to residual image block coding in motion regions is justified because the majority of residual image blocks in these regions are observed to be of highly impulsive nature, and hence favouring VQ coding technique to be used because of its non-linearity property. Transform coding techniques generally could not achieve the same level of performance as VQ under this circumstance. In addition, as residual blocks are segmented into smaller ones following the quadtree segmentation process, they become increasingly similar to those image blocks containing actual edges, thus making classified VQ a highly favourable choice (Lee & Crebbin, 1994b).

In Chan et al's implementation (1990), however, adaptive inter-frame coding structure is implemented using a binary segmentation scheme followed by a merging process which is applied to neighbouring blocks at the same segmentation level. During the segmentation process, binary segmentation decisions are made based on the sum of square errors resulting from the motion estimation process, as opposed to the simple frame difference in Lee and Crebbin's implementation (1994b). This binary segmentation process is applied recursively until motion compensation errors fall below a certain threshold or a minimum block size has been reached.

After the segmentation process, merging operation is invoked to merge those neighbouring blocks at the same segmentation level, thereby enabling further improvement in compression gains. In this operation, if it finds that a merged block can be adequately approximated by a corresponding block in a previous frame, the merging operation shall be granted. The main reason for incorporating the merging operation into this adaptive coding algorithm is due to the fact that the segmentation process does not normally start from the top level which is equivalent to an entire picture. Clearly, estimating motion of an image block as large as the whole picture is both impractical and a waste of effort. However, segmenting an image into smaller image blocks does not necessarily result in better coding performance either. On the one hand, while the application of smaller block size provides a higher level of adaptivity for the coding scheme, its usage may also lower the compression efficiency because spatial correlation among image elements is exploited less efficiently. On the other hand, although the use of larger block size enables the

spatial correlation to be better exploited, its usage may violate the assumption that image blocks shall experience uniform motion.

Apart from block size adaptation, an inter-frame coding algorithm can incorporate both intra- and inter-frame coding modes into its coding structure in order to improve its adaptability. As such, image blocks can be coded using either intra- or inter-frame coding techniques during inter-frame coding process. The choice obviously depends on which technique offers better results or lower distortion. The operation principle of this particular coding technique can be illustrated as follows (Chan et al, 1990):

For intraframe coding mode, assuming that the simple block average is taken as an approximation of an input image block, that is,

$$f_i(x, y) \approx \hat{f}_i(x, y) = \frac{1}{M} \sum_{(x,y) \in S_i} f_i(x, y)$$

where  $M$  is the total number of pixels in an image block;

$S_i$  refers to the  $i$ -th block in an image; and

$f_i(x, y)$  and  $\hat{f}_i(x, y)$  denote the actual and predicted pixel values at  $(x, y)$ .

Also, the sum of square errors ( $SSE$ ) is employed as a performance measure for this coding process and is given by:

$$SSE_1(S_i) = \sum_{(x,y) \in S_i} \left[ f_i(x, y) - \hat{f}_i(x, y) \right]^2$$

For inter-frame coding mode, assuming that motion compensated predictive coding technique is applied, that is,

$$f_i(x, y) \approx \hat{f}_i(x - \tilde{d}x, y - \tilde{d}y)$$

where  $(\tilde{d}x, \tilde{d}y)$  designates the estimated motion vector for the current block.

For comparison, the SSE in this case also needs to be evaluated and is given by:

$$SSE_2(S_i) = \sum_{(x,y) \in S_i} \left[ f_i(x, y) - \hat{f}_i(x - \tilde{d}x, y - \tilde{d}y) \right]^2$$

Naturally, with the aid of the above performance measure, the coding process of the adaptive intra- and inter-frame coding algorithm should be guided by the coding mode which leads to the least SSE. In addition, it should be noted that although the  $SSE$  serves as an effective criterion for the above operation, a simpler measure such

as the mean absolute error (*MAE*) can also be used. In fact, it has been reported that the *MAE* may produce slightly better results than the *SSE*.

### 3.6 Basic coding structures of MPEG

As a well established coding standard for motion pictures, the coding structure of Motion Picture Experts Group (MPEG) coding algorithm can generally be divided into two broad categories, namely, with and without the provision of bidirectionally interpolated or B-frames (Puri et al, 1992). When B-frames are not provided in the coding structure, all picture frames in a GOP are coded as forward motion compensated predicted frames (P-frame), except for the first one which is always coded as an intra-coded frame (I-frame) using intra-frame coding techniques. Because both I- and P-frames can equally be used as reference frames in this coding arrangement, all P-frames shall be predicted progressively from their preceding frames. This coding structure is illustrated in Figure 3.10.

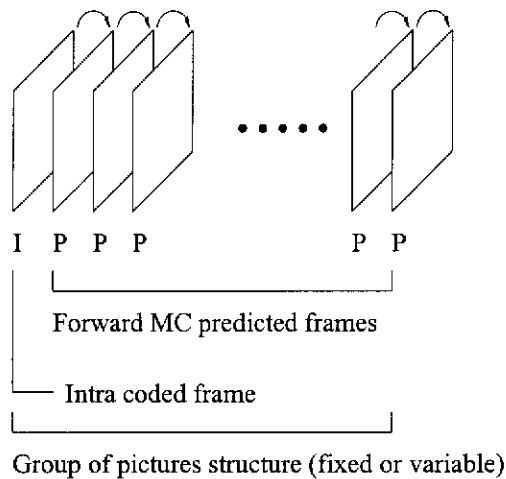


Figure 3.10: Basic GOP frame structure for MPEG coding without B-frame inclusion

On the other hand, when B-frames are used during the coding process, the structure of GOPs is altered to accommodate three different frame types (AT&T Technical Journal, 1993). As in the previous category, the first frame of a GOP will be coded as an I-frame using intra-frame coding techniques. Depending on implementations, a number of P-frames are then marked against the remaining frames at regular intervals. The rest are designated as bidirectionally interpolative frames (B-frame). While P-frames can be processed using forward motion compensated prediction

techniques, B-frames are normally coded using bidirectionally motion compensated interpolation techniques. As such, this particular coding arrangement enables B-frames to be predicted from two neighbouring reference frames at both ends. The frame structure for this coding arrangement is illustrated in Figure 3.11.

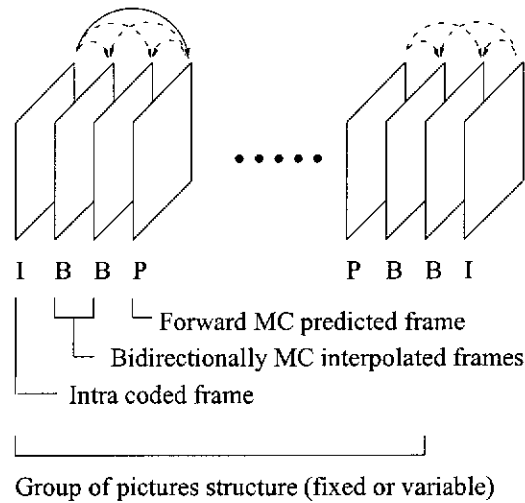


Figure 3.11: Basic GOP frame structure for MPEG coding with B-frame inclusion

### 3.7 Proposed adaptive image sequence compression algorithm

#### 3.7.1 Adaptive scene segmentation

As already mentioned, one of the key operations in dealing with lengthy image sequences is scene segmentation. The purpose of scene segmentation is to partition lengthy image sequences into GOPs of manageable length so that not only can temporal redundancy within image sequences be better exploited, but it also serves to ease hardware implementation. Also, for better segmentation results, adaptive scene segmentation should be employed during the segmentation process. Its operation is based on the evaluation of temporal correlation among consecutive picture frames in order to make its segmentation decision. In general, the performance of an adaptive scene segmentation scheme is primarily judged by how effective a scene segmentation criterion is capable of in assessing temporal correlation among picture frames and how efficient its computation is likely to be.

Although the HOD criterion may offer a better segmentation scheme for adaptive scene segmentation than the DOH due to its sensitivity to local activities, its

application unfortunately failed to detect all locations of scene changes in our test sequence. As listed in Table 3.2, this test sequence was constructed by cascading together 8 short video clips in CIF format with different levels of local activities, ranging from simple and low motion shots to complex and high motion scenes. The reason for the pitfall of the HOD scheme is primarily due to the fact that instantaneous HOD values at locations of scene changes are not necessarily higher than those within the same video clips as noted in Figure 3.12. Therefore, if adaptive scene segmentation is merely carried out based on a simple thresholding scheme against the HOD readings, inconsistent detection for locations of actual scene transitions would become inevitable. Such a segmentation scheme would be more likely to detect a false scene transition location or to miss out a correct one altogether.

Frame Number	Description
0-60	Salesman
61-120	Claire
121-180	Miss American
181-240	Susie
241-273	Caltrain
274-313	Table Tennis
314-373	Football
374-472	Flower garden

Table 3.2: Composition of test image sequence

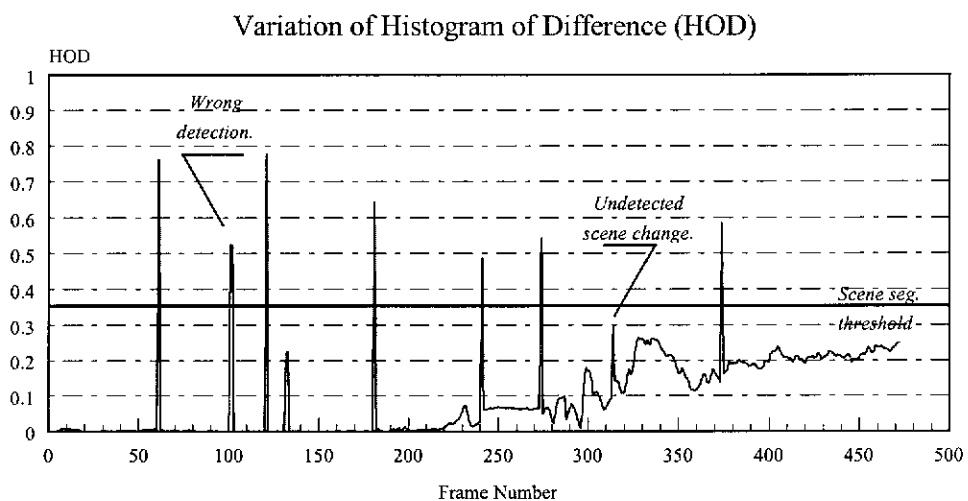


Figure 3.12: Histogram of difference for the test sequence



As an attempt to improve the consistency of the adaptive scene segmentation process, a major modification has been investigated in relation to the application of the HOD scheme. Based on our experimental data, it was observed that HOD readings for image frames at locations of scene transition are normally much higher than those of remaining frames. Therefore, a new scene segmentation scheme called the Magnitude of HOD Variation (MHOD) has been proposed for the implementation of adaptive scene segmentation and is defined as follows:

$$\begin{aligned} diff1 &= |HOD(i) - HOD(i - 1)| \\ diff2 &= |HOD(i) - HOD(i + 1)| \\ MHOD &= \min(diff1, diff2) \end{aligned}$$

where  $HOD(i-1)$ ,  $HOD(i)$  and  $HOD(i+1)$  denote the HOD readings for the previous, current and next frames, respectively. As it can be seen from its formulation, the computation of the MHOD measure is in fact very efficient because apart from normal HOD evaluation, the proposed scheme only requires a minimal amount of additional computation with scalar arithmetic.

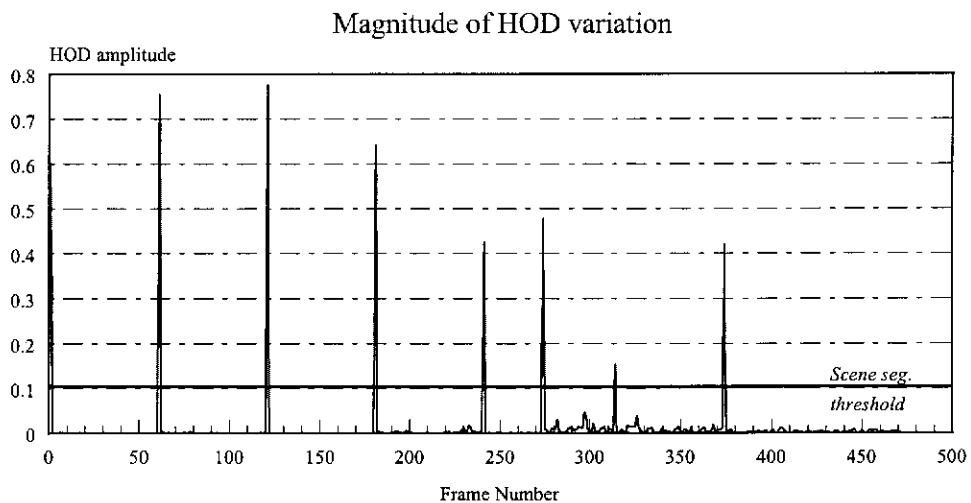


Figure 3.13: Magnitude of HOD variation of the test sequence

As shown in Figure 3.13, the proposed MHOD measure indeed offers an effective scheme for evaluating the level of temporal correlation among consecutive picture frames of an image sequence. Specifically, 40 and 0.1 have been empirically chosen as optimum threshold values for the computation of HOD data and for the scene transition detection process, respectively. Accordingly, this segmentation criterion has been able to locate all locations of scene transition correctly in the test sequence.

In addition, it has been found that this scheme offers a considerably wider margin, thereby enabling thresholding schemes to operate more reliably. Most importantly, the proposed segmentation criterion has demonstrated a high degree of tolerance to spurious image activities. This property is highly desirable because it prevents inconsistent scene transition detection from occurring, especially when high detail image sequences with high motion activities such as the 'Football' and 'Flower garden' are involved.

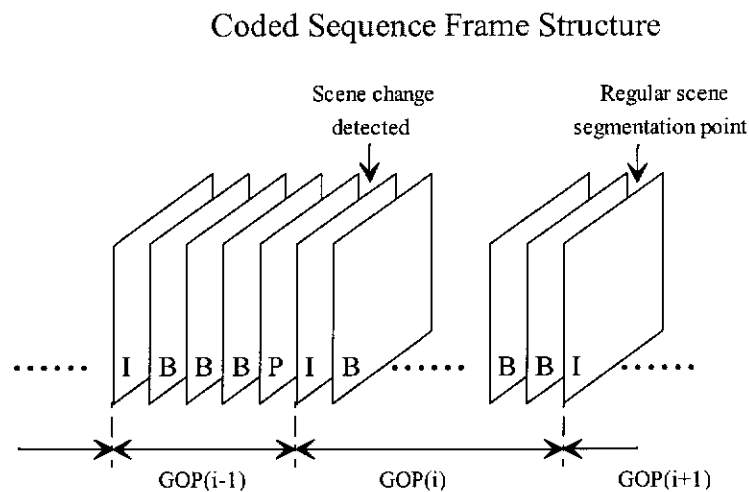


Figure 3.14: Dynamic GOP structure

As mentioned earlier, adaptive scene segmentation schemes are more capable of producing optimal temporal segmentation than those with fixed scene segmentation. Depending on the extent of temporal correlation among picture frames, the application of adaptive temporal segmentation would result in a dynamic GOP structure as illustrated in Figure 3.14. However, since excessively long GOPs usually cause processing difficulties as well as long processing delay, the use of lengthy GOPs should be avoided. This implies that constraints have to be imposed in the adaptive temporal segmentation process to ensure that the resulting dynamic GOP structure does not exceed a predetermined maximum length. In other words, the operation principle of the proposed adaptive scene segmentation scheme is based on adaptive temporal segmentation but subject to a maximum GOP length.

Because the first frame of a GOP is always coded as an I-frame, the maximum GOP length for this adaptive temporal segmentation process can be conveniently set to 9

to accommodate a maximum of 8 additional motion compensated predicted frames. Also, in the process of determining a specific structure for GOPs, two cases are anticipated as to whether an ending P-frame should be inserted. In the first case, if a scene transition occurs at the first frame of the following GOP, then an ending P-frame shall be inserted into the current GOP. All the remaining frames in between shall then be coded as B-frames. This arrangement is necessary because such a condition would indicate that the current and following GOPs are completely un-correlated. So, the insertion of the ending P-frame serves not only as a separating frame between un-correlated GOPs, but also as a reference frame for the processing of B-frames contained in the current GOP.

In the second case, if a scene transition does not occur at the beginning of the following GOP, the insertion of an ending P-frame becomes unnecessary because of the temporal correlation between the current and following GOPs. As such, no P-frame shall be inserted and all the remaining frames in the current GOP shall be processed as B-frames. However, this coding arrangement may seem to create a problem because normal processing of B-frames needs to refer to two reference frames and yet the structure of the current GOP contains only one of them. Fortunately, this problem can be easily resolved by taking into account the correlation between the current and following GOPs. As such, this enables the first I-frame of the following GOP to be deployed as the second reference frame for the purpose of B-frame processing.

### **3.7.2 Variable block-size motion compensation**

Following the adaptive scene segmentation process, image frames in GOPs shall be coded using adaptive motion compensated coding techniques. In implementing adaptive motion compensated coding, a variable block-size approach similar to that proposed by Lee & Crebbin (1994b) has been adopted. However, instead of using inter-frame difference as frame segmentation criterion, a more consistent measure based on motion compensation errors has been employed for controlling the quadtree segmentation process. In this way, regions with uniform motion can be handled using larger block-size for better compression efficiency without causing excessive motion compensation distortion. Conversely, regions experiencing complex motion activities

shall be dealt with using smaller block size to minimise motion compensation errors and to improve the perceptual quality of reconstructed pictures.

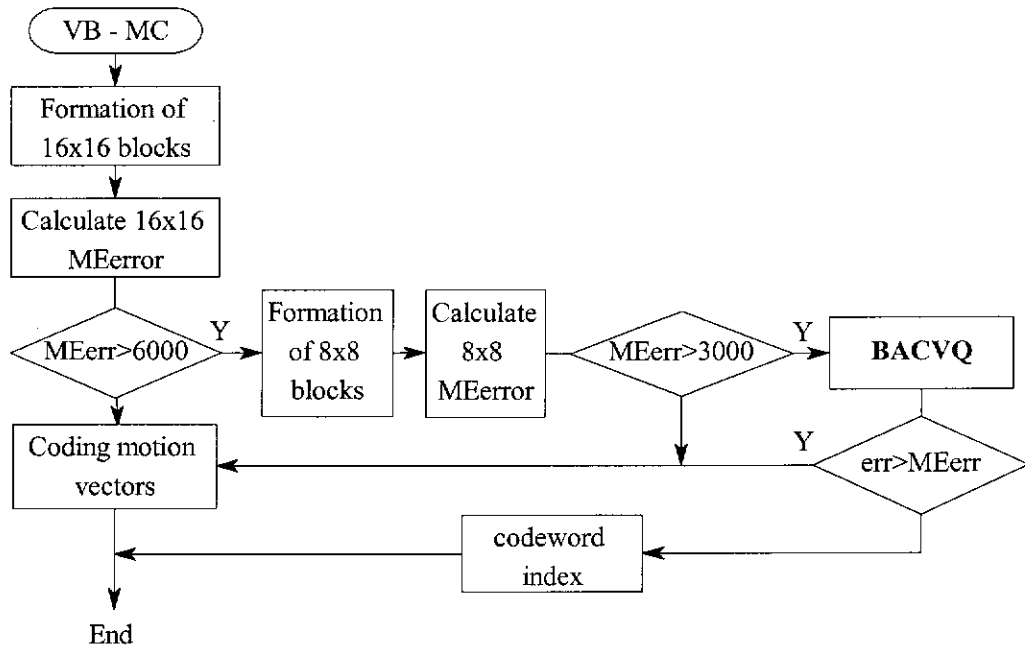


Figure 3.15: Flowchart for the proposed image sequence compression algorithm

As with most variable block-size motion compensation schemes, the proposed algorithm was implemented using a progressive segmentation structure. As such, an incoming image frame of an image sequence is initially segmented into uniform square blocks of 16x16 as illustrated by the flowchart in Figure 3.15. This block size was primarily selected as a compromise between computation complexity and motion compensation efficiency. During forward motion compensated prediction process, block-based motion estimation is performed sequentially on every 16x16 block with regard to a preceding reference frame. The resulting absolute motion estimation (ME) error is then compared against a predefined threshold to determine if further segmentation is required. If the value of this threshold is too low, more image blocks will be subject to quadtree segmentation to produce multiple image sub-blocks with smaller block size, thus adversely affecting the compression efficiency of the motion compensation process. Conversely, if it is too high, fewer image blocks will be subject to further segmentation, hence achieving better compression efficiency. However, the use of larger block size in motion compensation may affect the perceptual quality of reconstructed pictures because of its poorer adaptivity to

complex motion. For this study, this threshold value is empirically set to 6000, which enables the motion compensation process to achieve a reasonable level of adaptability to complex motion at the cost of a moderate reduction in compression gain.

When further segmentation becomes necessary, a quadtree segmentation scheme is applied to divide the current 16x16 image block into four image sub-blocks of 8x8. Again, a similar block-based motion estimation is applied to each of them. The resulting absolute ME error of each sub-block is then compared to a second threshold to determine whether intra-frame coding mode based on BACVQ should be attempted for coding it. Again, this threshold value is empirically set to 3000 so that a rapid increase in bit rate as a result of switching to the intra-frame coding mode could be avoided. However, even at this stage, if the absolute error resulting from the intra-frame coding mode is comparable to or worse than the current absolute ME error, it makes sense to reject the intra-frame coding result because its coding requires higher bit allocation, and to resume the 8x8 motion compensated coding process.

### **3.7.3 Progressive motion vector estimation**

#### ***3.7.3.1 Improved motion vector estimation strategy***

During the motion compensation process, motion estimation has to be performed on individual image blocks to identify the optimal motion vectors that minimise motion compensation errors and possibly best represent the underlying motion. For block matching algorithms, although the exhaustive search scheme offers an optimal search strategy for motion vector estimation, its computation complexity grows exponentially with increased motion vector search range (Musmann et al, 1985). Therefore, to overcome this computational problem, other sub-optimum search strategies such as 2D logarithmic, modified 2D, 3-step, conjugate direction and cross search have been proposed (Srinivasan et al, 1985) (Ghanbari, 1990). All of these search strategies offer a very attractive compromise between computation complexity and sub-optimality in performance, especially when large motion vector search range is involved.

In this study, the Modified 2D search strategy was adopted with a search range of  $\pm 15$  pixels/frame. This is because through preliminary study, it has been found that the modified 2D search strategy offers virtually the same level of performance as the 2D and 3-step search schemes in terms of motion estimation consistency but with an added advantage of less computation. In addition, it has been found that while smaller motion vector search range is generally acceptable for “head-and-shoulder” type of image sequences, larger search range of at least  $\pm 15$  pixels/frame proves to be more adequate for dealing with most motion pictures.

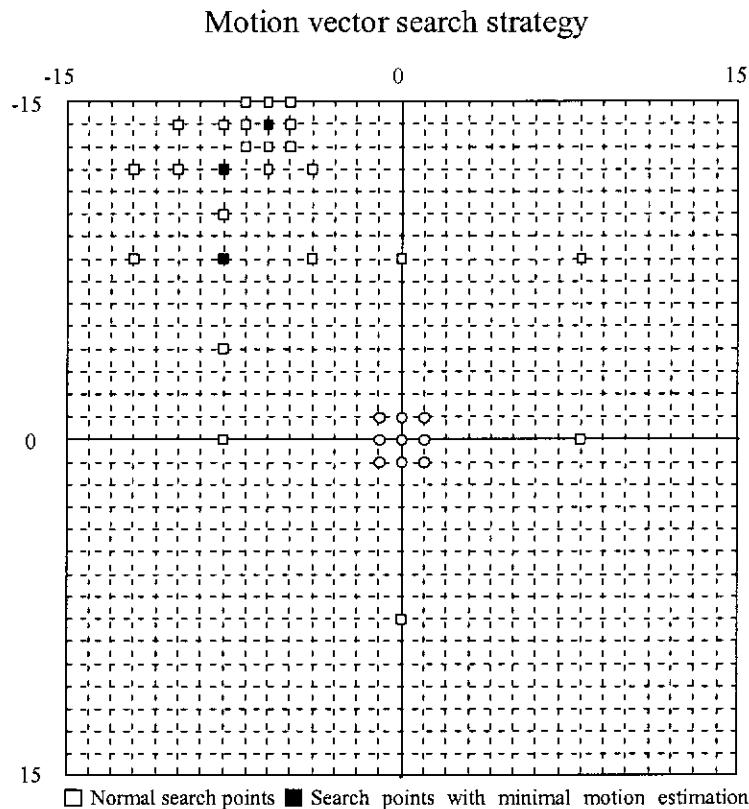


Figure 3.16: Typical search locations for the proposed motion estimation strategy

However, in order to enhance the motion estimation consistency of the modified 2D search strategy for small motion, additional computation is performed for pixels around the central region of the motion search domain as illustrated in Figure 3.16. This modification is primarily motivated by our observation that motion estimation errors in slow motion regions are generally more noticeable than those in fast motion areas. As a result, by taking additional search points at the central region of the search window, the proposed motion estimation strategy not only inherits the

performance of the modified 2D search algorithm, but also has the ability to accurately estimate any small motion of  $\pm 1$  pixel/frame.

In terms of computation complexity, the proposed motion estimation strategy only requires slightly more computation than the modified 2D algorithm, especially when large motion search range is involved. In this study where a  $\pm 15$ -pixel search window is used during the motion estimation process, the maximum number of search steps of the proposed motion estimation strategy remains exactly the same as that of the modified 2D algorithm, but the maximum number of search points does increase slightly from 27 to 35. Even so, this number remains significantly less than a total of 961 search points in the exhaustive search scheme.

### ***3.7.3.2 Progressive motion estimation***

In motion compensated predictive coding, motion is normally estimated with respect to a fixed reference frame. In MPEG implementation, for example, an I-frame is used as a fixed reference frame for the prediction of subsequent P-frames in a GOP. While this arrangement is possible in MPEG implementation, its application in this study has created some problems due to the adoption of significantly longer inter-reference frame intervals. Under this circumstance, normal motion estimation techniques would have to be performed using a significantly larger motion vector search window so that accumulated motion across those intervals can be accounted for. However, considering the limitation of processing hardware, motion estimation efficiency and computation complexity, any significant increase in the size of motion vector search windows will be impractical. Therefore, in order to overcome this, a progressive motion estimation technique has been proposed as illustrated in Figure 3.17.

During the progressive motion estimation process, motion estimation is carried out with respect to an immediate adjacent frame in a GOP. This implies that motion estimation for each image frame will always be fixed to a motion search range of  $\pm 15$  pixels/frame, regardless the length of inter-reference frame intervals. Not only does this simplify the implementation of motion estimation process considerably, but it also enables the resulting motion vector field to be coded more efficiently due to

fixed motion vector size. In addition, unlike the telescopic search technique proposed by Lee & Dickinson (1994), it was found that the proposed technique is less susceptible to overlapping movements among image objects in an image sequence. This is because in the telescopic search technique, motion estimation is based on a collection of successive motion vector fields from adjacent frames in order to determine the motion search window for the motion estimation process. By contrast, in the proposed technique, motion is estimated directly from an adjacent predicted frame whose overlapped regions, if any, would have been coded using intra-frame coding mode. However, because motion is estimated progressively from a series of adjacent frames, a major concern with the proposed technique is that its performance may deteriorate due to the propagation of motion estimation errors. Nonetheless, this concern is somewhat relieved by the provision of intra-frame coding mode. This mode of coding will be deployed should motion compensation errors exceed a predetermined threshold level.

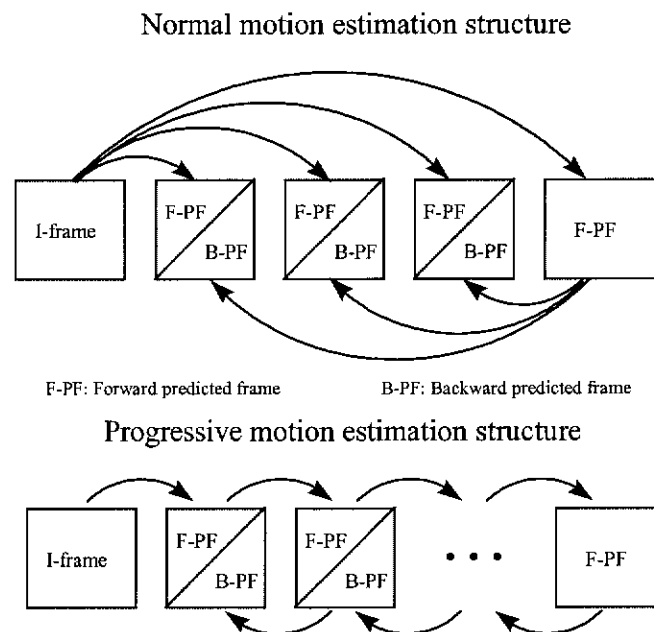


Figure 3.17: Conventional and progressive motion estimation strategies

### 3.7.4 Block adaptive classified vector quantisation (BACVQ)

As discussed in the previous chapter, BACVQ indeed offers an effective coding technique for compressing still images. In this technique, an image is initially segmented into square image blocks of 8x8 block size. Each of these blocks is then



subject to activity evaluation to determine whether further segmentation is required to achieve better coding results. For low activity blocks, there is no need for further segmentation because these blocks are characterised by relatively smooth variations in intensity. As such, they can be coded directly using normal VQ. By contrast, if an image block is detected to contain high activity features, a quadtree segmentation scheme is applied to segment the block into four 4x4 image sub-blocks. Each of these sub-blocks is then coded using classified VQ. Apart from computation efficiency, classified VQ is highly capable of preserving edge integrity in reconstructed images, thereby significantly improving the perceptual quality of reconstructed pictures.

Apart from being used for still image compression, BACVQ can also be applied to image sequence compression for I-frame and residual frame compression. In general, coding I-frames does not required any major variations to the coding algorithm of BACVQ because I-frames can be treated more or less like still images. However, since motion picture features such as motion blurs and interlace field scanning have different visual attributes as opposed to those normally encountered in still images, BACVQ code-books for intraframe coding need to be recreated using the same algorithm as described in the previous chapter in order to obtain better coding results. Specifically, a training set of 25 image frames has been extracted from the test sequence for codebook training purposes. Each of these training frames was selected at a regular frame interval of 18-frames apart starting from CIF#000 in order to ensure the generality of the training set. The results for this code-book training process are summarised in Table 3.3.

For residual frame compression, however, the application of BACVQ requires that not only should a separate set of code-books be recreated but its coding algorithm also needs to be altered slightly to take into account the differences in statistical characteristics of residual frames. As shown in Figure 3.18, since the distribution of pixel values in residual frames is concentrated at 127 due to scaling and shifting operations in the formation process of residual frames, the assumption that the variation in pixel values is less noticeable when the average intensity of image blocks is higher is not relevant any more. As such, the provision for dynamically varying the

threshold value according to the average intensity during the classification process for the detection of 4x4 smooth blocks should be removed. In addition, since the distribution of pixel values is highly concentrated at the vicinity of 127, the additional “dynamic range” test for the detection of 8x8 smooth blocks becomes redundant and hence has been eliminated. Furthermore, as the size of BACVQ codebooks for residual frame coding has been reduced substantially as indicated in Table 3.4, the number of classes allocated for the operation of classified VQ during residual frame processing needs to be reduced in order to maintain overall compression efficiency.

Codebook type	Codebook size	Number of iterations	Avg dist before opt.	Partial dist. before opt	Avg dist. after opt.	Avg dist. with cw replacement
8x8 smooth	4096	29	5771.2	N/A	1630.19	N/A
4x4 smooth	128	21	434.83	N/A	197.17	204.9
V1 edges	399	16	4434.03	1.2345	1967.85	2392.17
V2 edges	442	16	5603.49	1.3473	2495.61	3095.74
V3 edges	335	21	4996.36	1.288	2274.66	2835.05
H1 edges	496	22	4859.3	1.3969	2145.73	2661.71
H2 edges	641	16	5014.06	1.3299	2070.81	2595.6
H3 edges	531	14	4955.58	1.387	2045.1	2540.98
45-deg/D1	156	18	6023.77	1.3515	2454.52	3041.85
45-deg/D2	137	15	5815.71	1.2273	2787.07	3540.84
45-deg/D3	134	12	6217.28	1.338	2842.38	3672.64
45-deg/D4	159	14	5465.69	1.2204	2361.67	2960.12
135-deg/D5	136	15	5481.43	1.236	2766.64	3529
135-deg/D6	99	10	6498.44	1.4275	2940.99	3605.18
135-deg/D7	101	11	6065.45	1.3388	2949.85	3673.68
135-deg/D8	138	15	6003.44	1.3929	2515.18	3192.54
4x4 mixed	64	19	11713.41	N/A	5265.77	6140.23

Table 3.3: BACVQ codebook optimisation for intraframe coding

As far as the issue of codebook generation for residual frame coding is concerned, a group of 8 consecutive residual frames has been extracted from the test image sequence for codebook training purposes. These residual frames were obtained by performing motion compensated prediction operation against a GOP consisting of 9 test images CIF#000-CIF#008. As a result of this operation, 8 consecutive residual frames associated with CIF#001-CIF#008 were created and used as training images for the codebook generation process. The reason for using 8 consecutive residual frames instead of well separated residual frames during the codebook generation process is because all residual frames are expected to have similar visual features and

statistical distribution of pixel values. The results for the residual code-book training process are presented in Table 3.4.

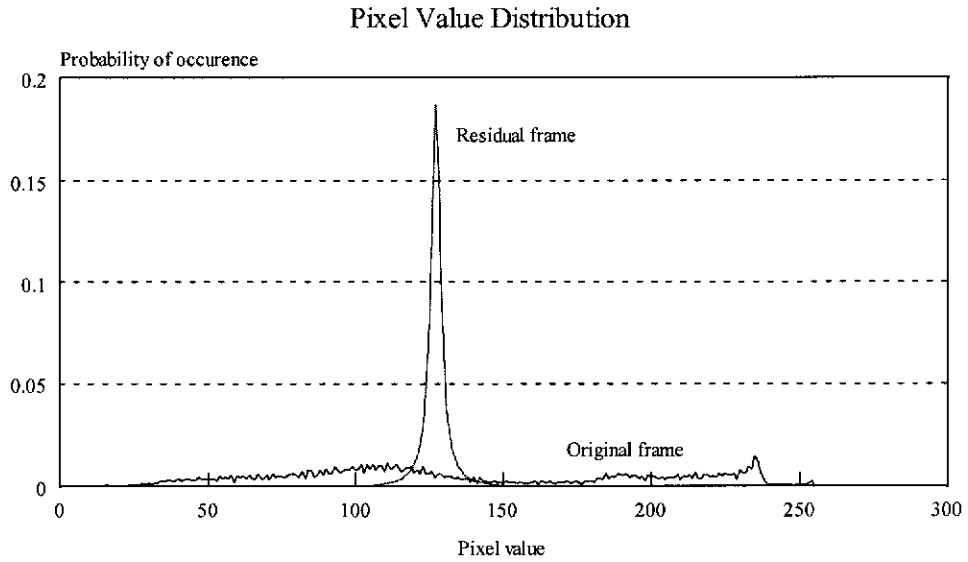


Figure 3.18: Typical pixel value distribution of original and residual frames

Codebook type	Codebook size	Number of iterations	Avg dist before opt.	Partial dist. before opt	Avg dist. after opt.	Avg dist. with cw replacement
8x8 smooth	128	39	4409.53	N/A	2721.9	N/A
4x4 smooth	32	3	56	N/A	32.88	51.19
Vert. edges	19	9	6600.5	85.27	3313.28	4200.16
Hor. edges	41	13	6048.8	72.87	2389.23	2956.69
45-deg	13	9	4918.7	50.62	2208.56	2885.44
135-deg	15	9	4098.1	34.59	2654.99	3421.59
4x4 mixed	8	5	9036.71	N/A	3594.83	4737.81

Table 3.4: BACVQ codebook optimisation for residual frame processing

### 3.7.5 Simulation results

Figure 3.19 shows the main functional blocks of the proposed image sequence coder. Among the most crucial operations in this coding algorithm are adaptive scene segmentation, variable block-size motion compensation and BACVQ. Although additional compression can normally be achieved by adding a lossless coding block

to the final stage of this coding structure, its omission does not affect the generality of this study because of its independent nature.

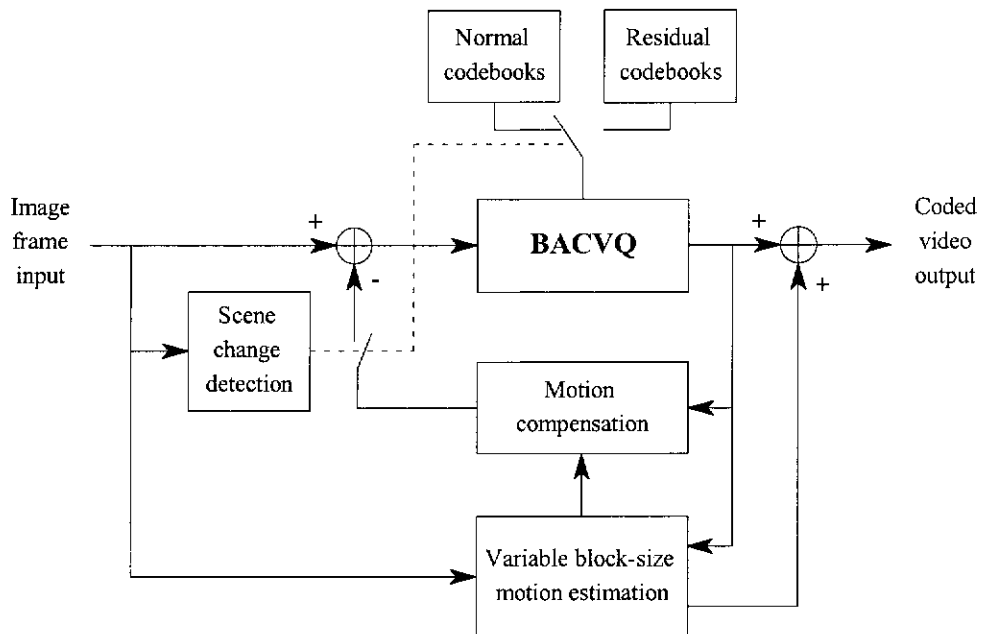


Figure 3.19: Block diagram of the proposed image sequence compress algorithm

#### Bit Allocation Structure for coding motion vector field

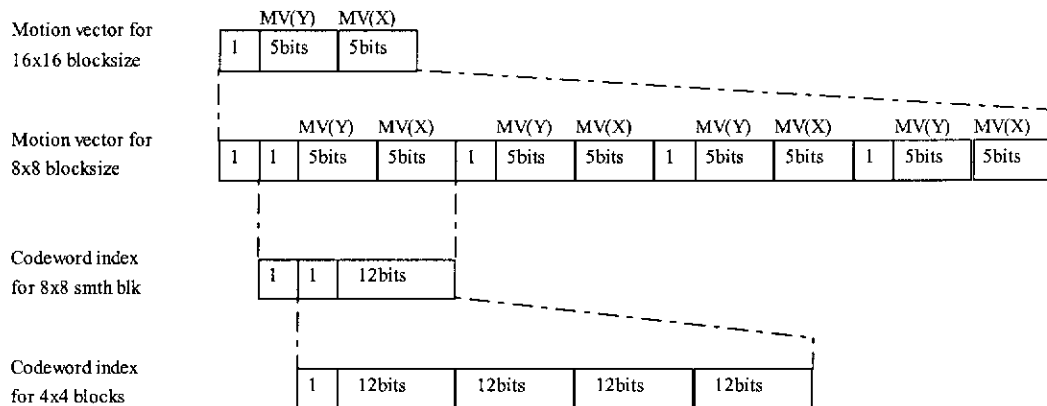


Figure 3.20: Bit allocation for coding motion vector fields

While the same bit allocation strategy as discussed in the previous chapter has been adopted for the I-frame and residual frame coding process, that for motion vector field coding is presented separately in Figure 3.20. Since motion is assumed to be within  $\pm 15$  pixels/frame, allocating 5 bits for each motion vector component will be sufficient for encoding any 2D motion vectors. However, because the provisions are made for variable block size motion estimation and intra-frame coding mode in the

proposed motion compensated predictive coding algorithm, additional bit allocation is required for embedding quadtree segmentation data and other information associated with intra-frame coding mode in the encoded data stream.

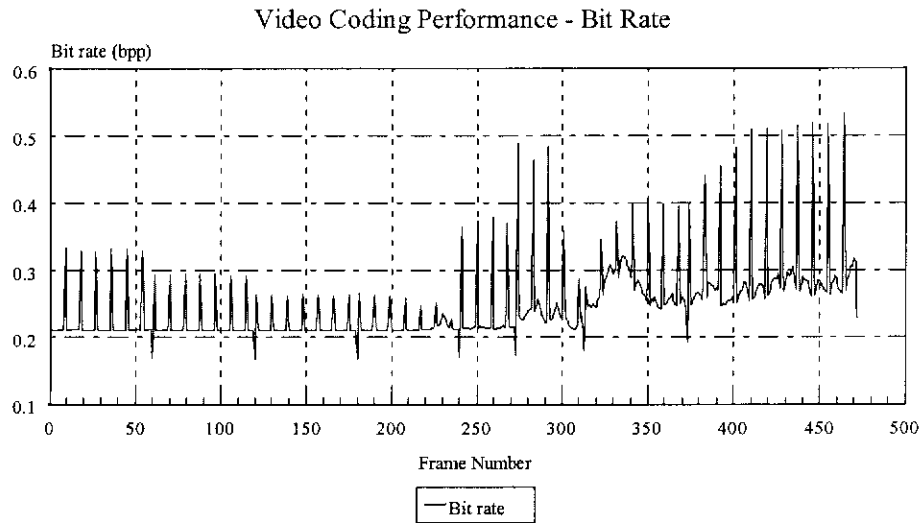


Figure 3.21: Bit rate performance on the test sequence

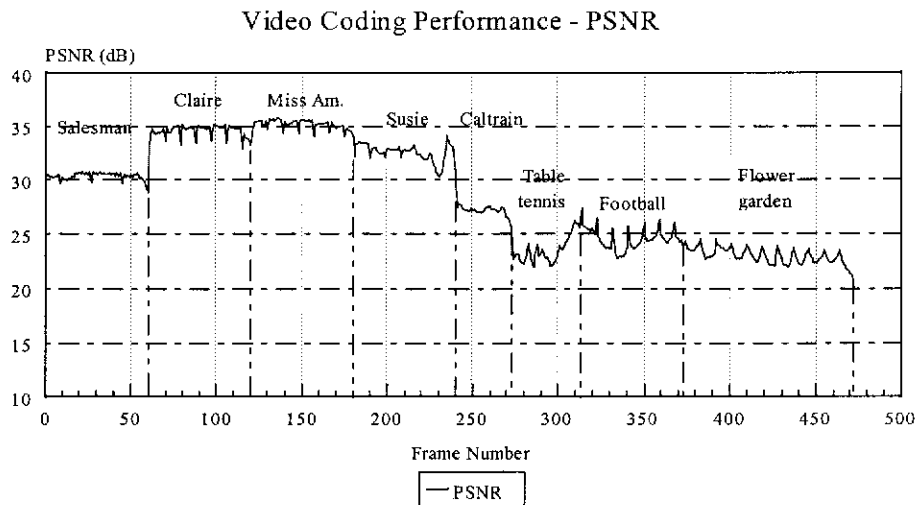


Figure 3.22: PSNR performance on the test sequence

Coding performance in terms of bit rate and PSNR of the proposed image sequence compression algorithm are presented in Figure 3.21 and 3.22, respectively. Because intraframe coding generally requires more bits for its coding operation, a sudden jump in the bit rate whenever an I-frame is being coded is what would be expected. However, it is noted that such a jump in the bit rate is not necessarily translated into higher *PSNR* as demonstrated by the coding results for the "Salesman", "Claire", "Missa" and "Susie" image sequence segments. The fact that all these image

segments are characterised as being “head-and-shoulder” types of sequences with relatively low motion activities has led us to believe that for image sequence compression, the performance of intraframe coding process will generally be inferior than that of motion compensated predictive coding process when low detailed image sequences with low motion activities are involved. Conversely, when high detailed image sequences with high motion activities are involved, motion compensated predictive coding tends to offer inferior performance than intraframe coding because of the drop in motion compensated prediction efficiency. Nevertheless, under this condition, both coding techniques do experience some coding difficulties, thus causing the overall bit-rate and *PSNR* performance of the proposed image sequence to deteriorate.

It should be noted that in order to examine the performance of image sequence compression process, it is generally required to evaluate the perceptual quality of reconstructed pictures in real-time. This is because in developing an image sequence compression algorithm, it is its aim to exploit the limitations of HVS in order to achieve higher compression performance while minimising noticeable picture degradation. However, because such a requirement normally involves expensive and dedicated hardware, we were unable to perform such a task in order to fully evaluate the performance of the proposed image sequence compression algorithm. Instead, we had to rely on the bit rate and *PSNR* performance as quantitative measures for the quality of reconstructed pictures. Nevertheless, as an attempt to assess reconstructed picture quality perceptually, selected individual picture frames were inspected statically for any major visual artefact. As shown in figures 3.23 through 3.30, samples of the original (Cif####), reconstructed (Enc####) and residual (Res####) picture frames for various test image sequences are presented for demonstration purposes. In most cases, it has been observed that the perceptual quality of reconstructed pictures are quite acceptable, given that a relatively low bit rate of around 0.3bpp has been achieved. In particular, the edge integrity of all reconstructed pictures is in fact very well preserved.

### 3.8 Summary

In this chapter, various aspects of image sequence compression have been discussed in detail. Being a key operation in image sequence compression, scene segmentation is frequently employed to divide a lengthy image sequence into groups of correlated pictures of manageable lengths for subsequent processing. Not only does this operation greatly facilitate hardware implementation, but it also enables temporal redundancy among picture frames to be exploited more efficiently. In adaptive scene segmentation, temporal correlation among picture frames within an image sequence is used as a segmentation criterion. Because lengthy GOPs resulting from adaptive scene segmentation processes tend to cause implementation problems and unnecessary processing delay, constrained adaptive scene segmentation in terms of maximum GOP length is regarded as a more favourable approach.

Apart from the consistency of temporal segmentation, complexity aspects of scene segmentation operations also contribute to the viability of a segmentation scheme. Various scene segmentation schemes have been studied, each of which offers different degrees of computation complexity and sensitivity to local activities. While the DOH scheme features the least computation complexity and sensitivity to local activities, the MCE approach offers the most optimal segmentation at the highest computational cost. For this reason, a new adaptive scene segmentation scheme based on the so-called “Magnitude of HOD variations” measure has been proposed. Experimental results show that this scheme is capable of detecting scene transitions accurately, therefore offering more consistent temporal segmentation results. Other benefits of the proposed scheme include high tolerance to spurious image activities and relatively low computation complexity.

For image sequence compression applications, inter-frame coding has been widely regarded as a fundamental coding technique. Unlike intra-frame coding which can only exploit the spatial redundancy within individual image frames, inter-frame coding allows both spatial and temporal redundancy in an image sequence to be effectively removed, thereby offering significantly improved compression performance.

As an effective inter-frame coding technique, motion compensated predictive coding operates on the same principle as conventional DPCM. However, in the case of motion compensated predictive coding, its operation relies heavily on both motion compensation and residual frame processing. Ideally, an effective motion compensation technique should be capable of handling both geometric and massic transformations. It should also take into account overlapping and non-overlapping motions among moving objects in an image sequence in order to optimise its operation.

Most motion compensation techniques are implemented based on pel recursive or block-based motion estimation algorithms. While pel recursive algorithms are more capable of handling real motion in image sequences, block-based motion estimation techniques can only operate efficiently when image blocks experience uniform motion. However, the former usually involves excessive overheads and high computation complexity, whereas the latter tends to suffer inferior performance, especially when image sequences containing complex motion are involved.

As an effort to improve the performance of block-based motion estimation techniques, various deformable block-based motion compensation techniques have been developed. Similarly, variable block-size motion compensation has become increasingly popular in inter-frame coding because of its potential in improving not only computation efficiency but also the perceptual quality of reconstructed pictures. For this reason, an alternative approach has been proposed for the implementation of variable block-size motion compensation. In this approach, a quadtree segmentation scheme has been adopted for the operation of variable block-size segmentation process and its segmentation criterion is based on motion estimation errors. Provisions have also been made to allow for the proposed motion compensation process to switch to intraframe coding mode should motion compensation results prove to be unsatisfactory. In implementing motion compensation, the modified 2D motion estimation strategy has been adopted with additional search points in the central region of search windows. Not only does this proposed motion estimation strategy inherit the good performance of the modified 2D search algorithm, but it



also has the capability to accurately estimate any small motion with a negligible increase in computation complexity.

In addition, an alternative progressive motion estimation technique has been proposed for dealing with large inter-reference frame intervals. Not only does this technique simplify the implementation of motion estimation process, but it also enables motion vector fields to be coded more efficiently. Furthermore, the proposed progressive motion estimation technique is considered to be less susceptible to overlapping motion among moving objects in an image sequence.

Apart from motion compensated predictive coding, motion compensated interpolative coding proves to be a very effective approach for reconstructing missing image samples. The operation principle of motion compensated interpolative coding is based on temporal interpolation along motion trajectories between two enclosing reference frames. As such, motion compensated interpolative coding offers a powerful motion compensated coding technique for image sequence compression.

While motion compensated predictive coding techniques have enabled temporal redundancy in image sequences to be effectively eliminated, spatial redundancy within intra-frames and residual frames can be efficiently removed by applying BACVQ. Although intra-frame coding does not required any major variations to the coding algorithm of BACVQ, its application to residual frame coding needs to be altered to take into account the significant differences in statistical characteristics between intra-frames and residual frames.

In addition, recent studies of HVS have also contributed significantly to the development of image sequence compression techniques. Not only have these studies enabled compression techniques to achieve outstanding compression performance, but they also helps them improve the perceptual quality of reconstructed pictures. The reasoning for this is that visual data which are less sensitive to or even could not be perceived by HVS can be safely compressed with higher distortion to maximise compression efficiency, whereas those with perceptual importance will be coded with higher fidelity in order to enhance the overall perceptual quality of reconstructed pictures.

Finally, it has been shown that the proposed image sequence compression algorithm has achieved good coding performance by adapting its coding structure to incoming image sequences, thereby enabling spatial and temporal redundancies within image sequences to be exploited more effectively. Such adaptability is collectively achieved with the aid of a new adaptive scene segmentation scheme, variable block-size motion compensation technique, and block adaptive classified vector quantisation. Depending on the nature of individual test image sequences, the *PSNR* and bit rate performance of the proposed image sequence compression algorithm may vary between 20 to 40dB and 0.15 to 0.55bpp, respectively.

Having established appropriate algorithms for still image and image sequence compression, efforts have been directed to studying parallel image sequence compression techniques. The objective of such a study is to evaluate the potential benefits offered by parallel processing in relation to reducing the overall processing time of an image sequence compression process. This aspect of image sequence compression is particularly important for practical implementations because a huge amount of raw visual data is usually involved and real-time processing is generally required. Due to the complex nature of parallel processing in general and parallel image sequence compression operation in particular, they shall be covered in the following two chapters. While Chapter 4 provides important background information about parallel processing techniques, Chapter 5 discusses in more detail some implementation aspects of parallel processing when it is incorporated into the image sequence compression process.

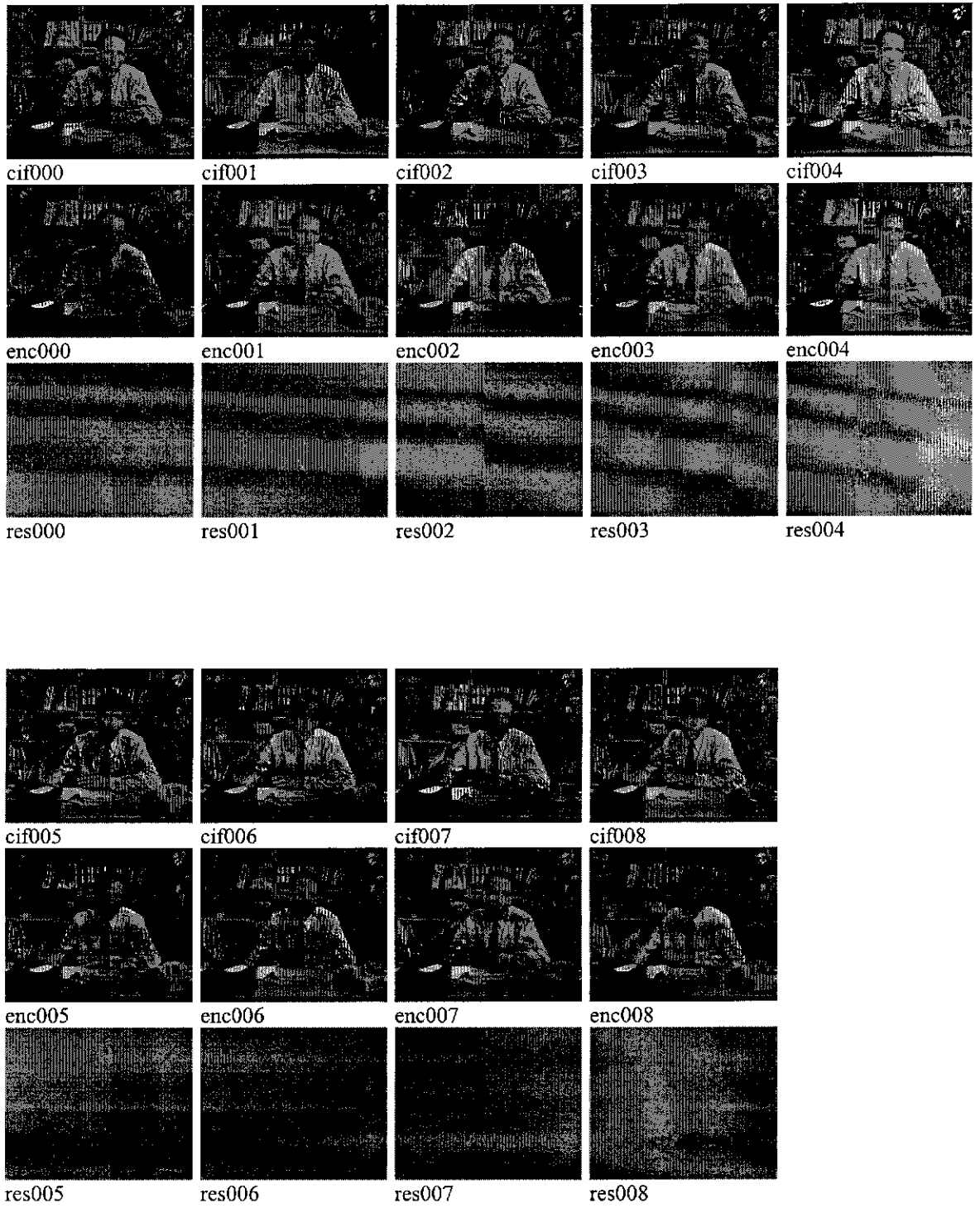


Figure 3.23: Sample pictures for the "Salesman" test image sequence

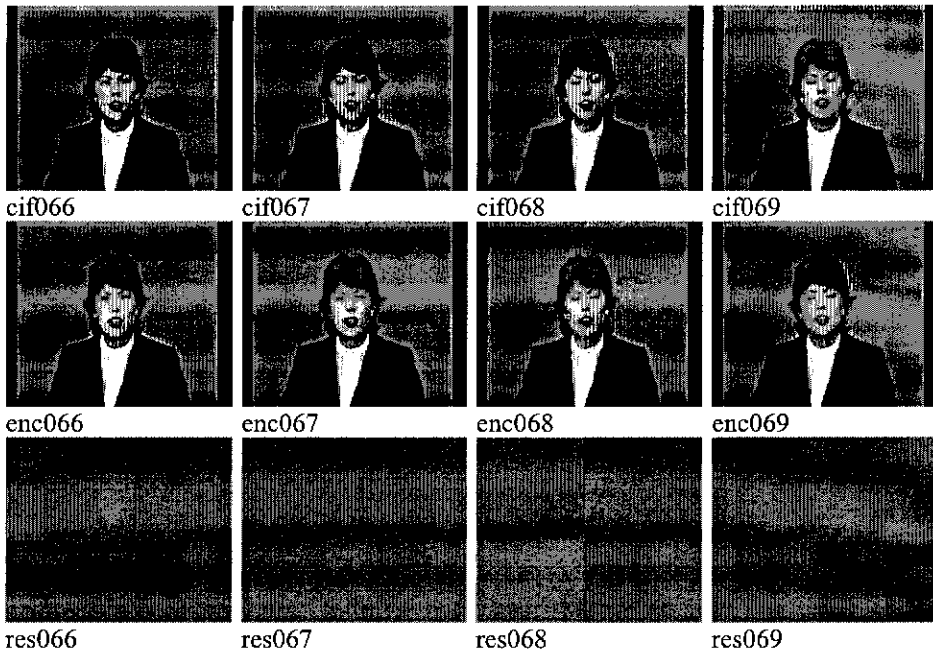
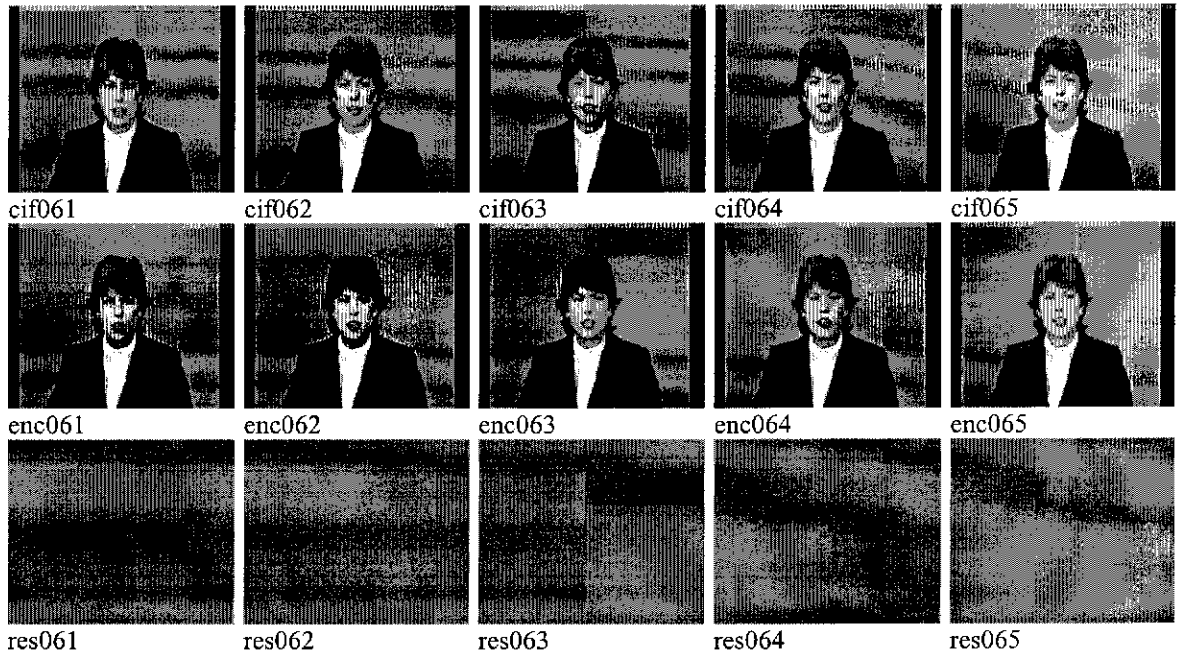


Figure 3.24: Sample pictures for the “Claire” test image sequence

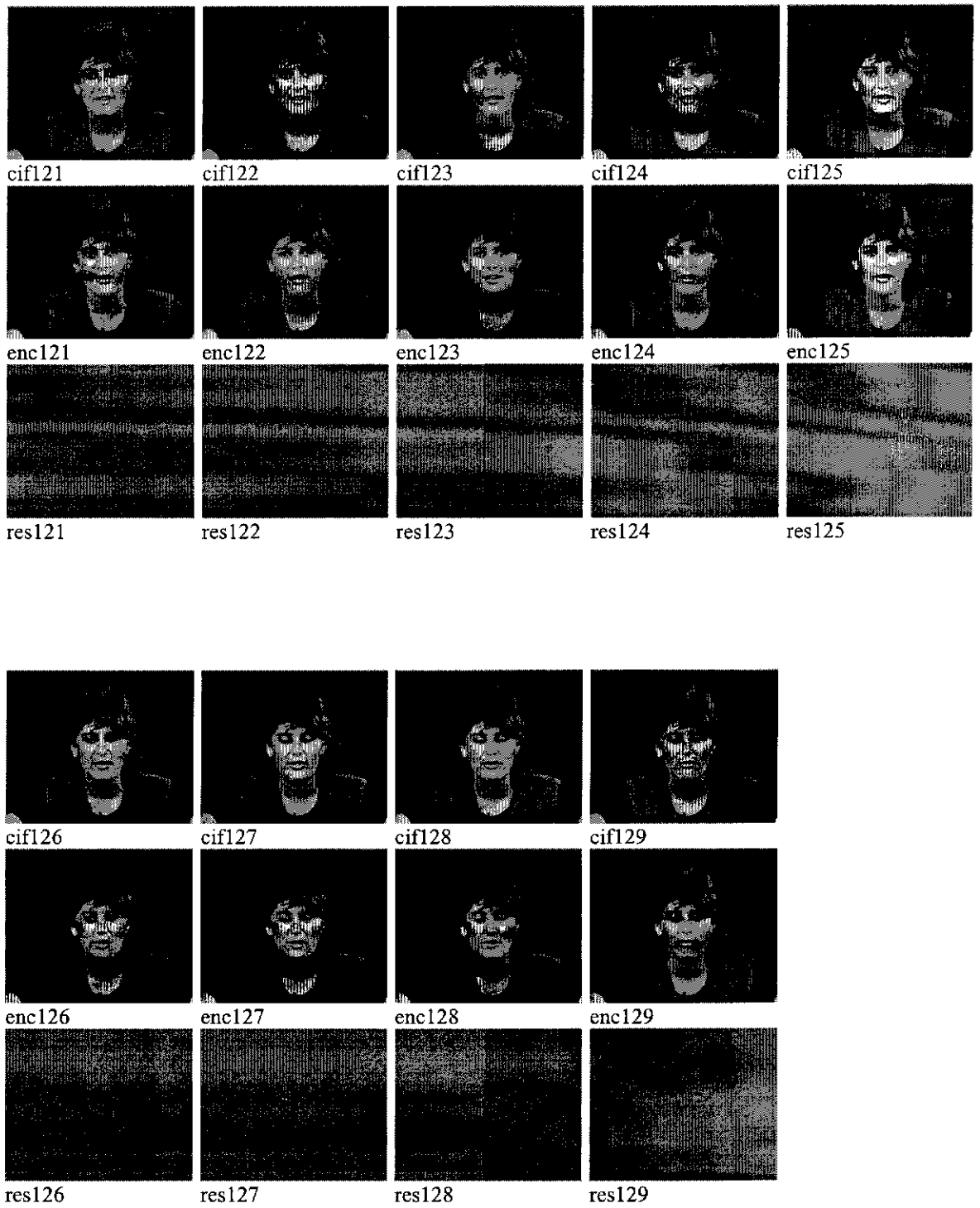


Figure 3.25: Sample pictures for the “Miss American” test image sequence

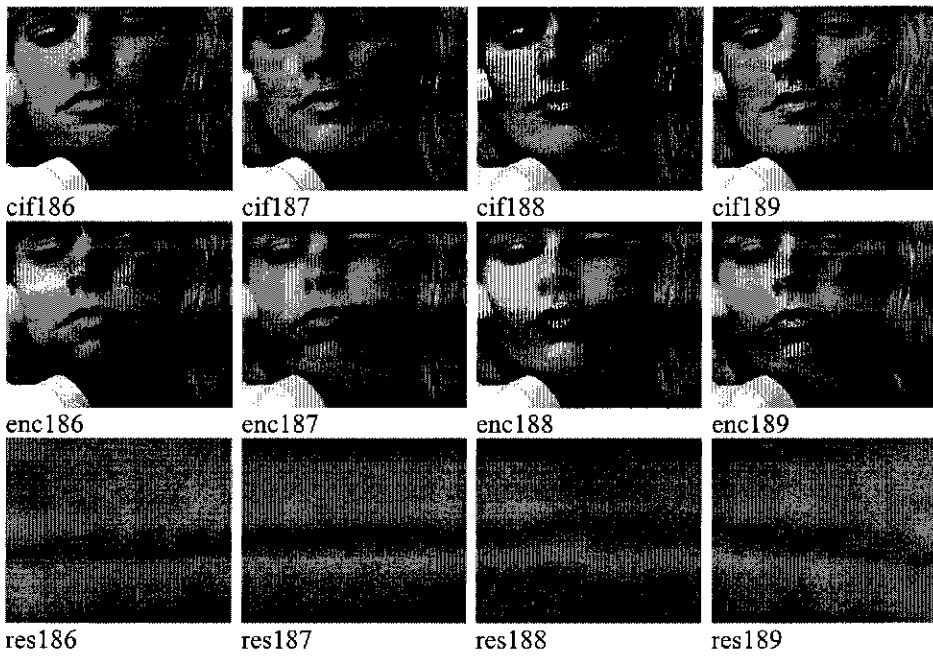
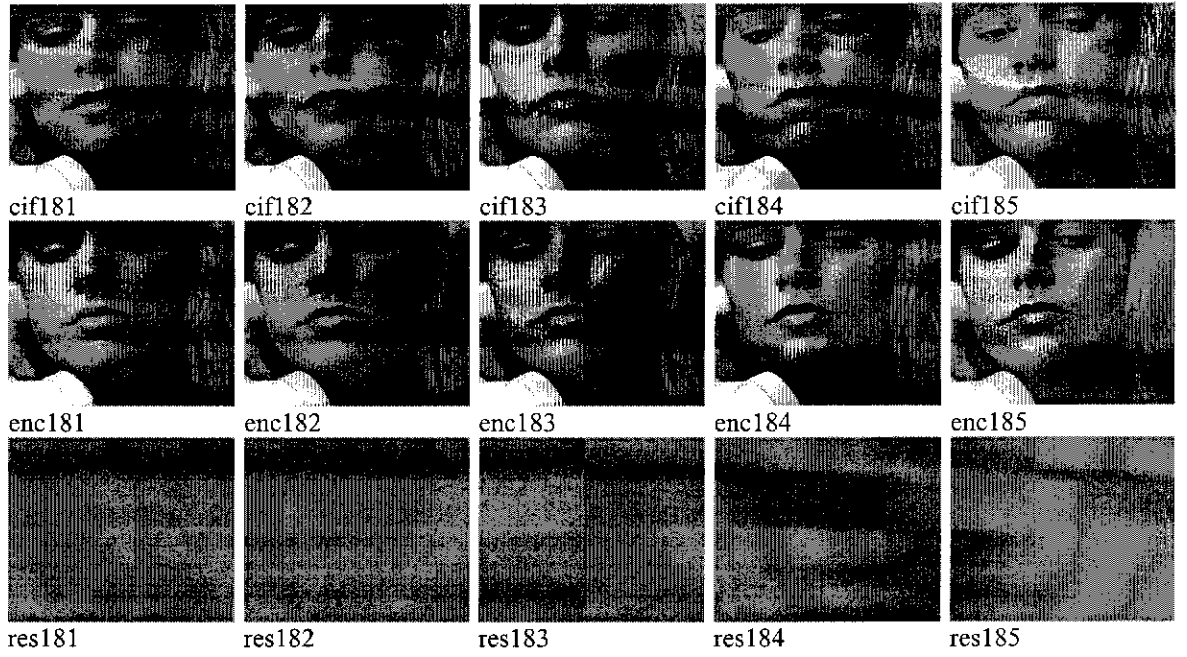


Figure 3.26: Sample pictures for the “Susie” test image sequence

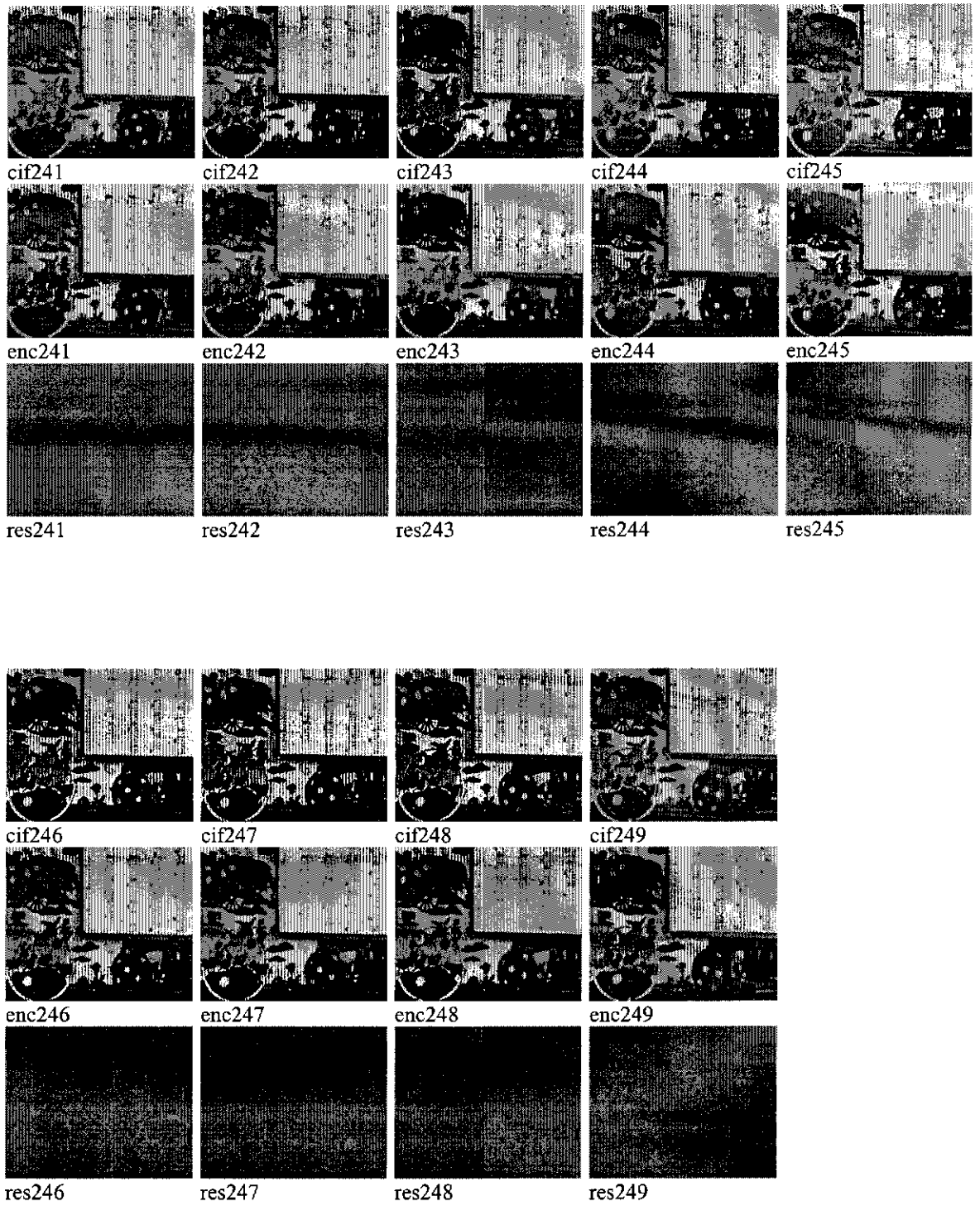


Figure 3.28: Sample pictures for the “Caltrain” test image sequence

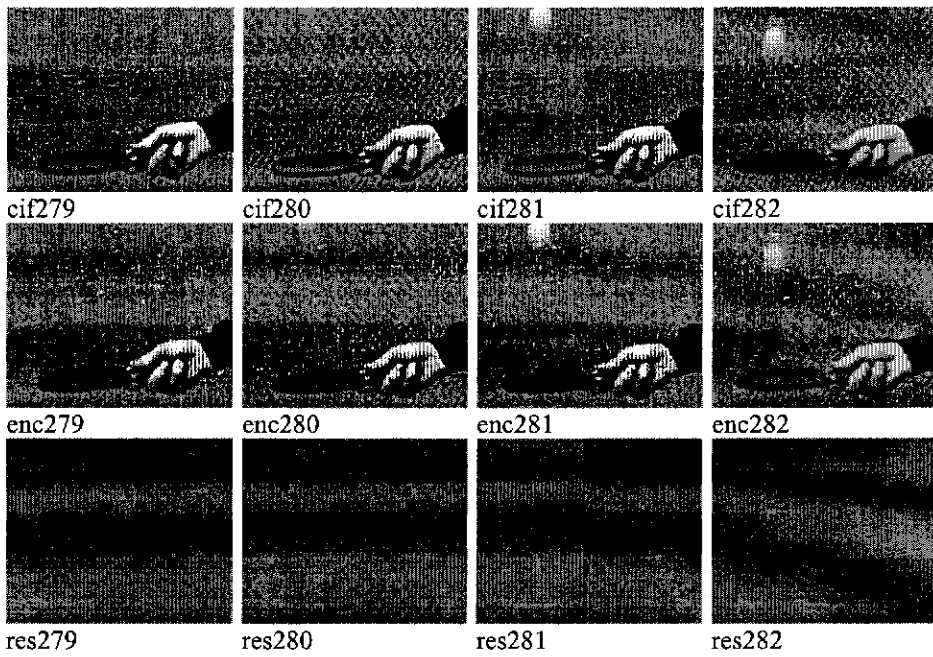
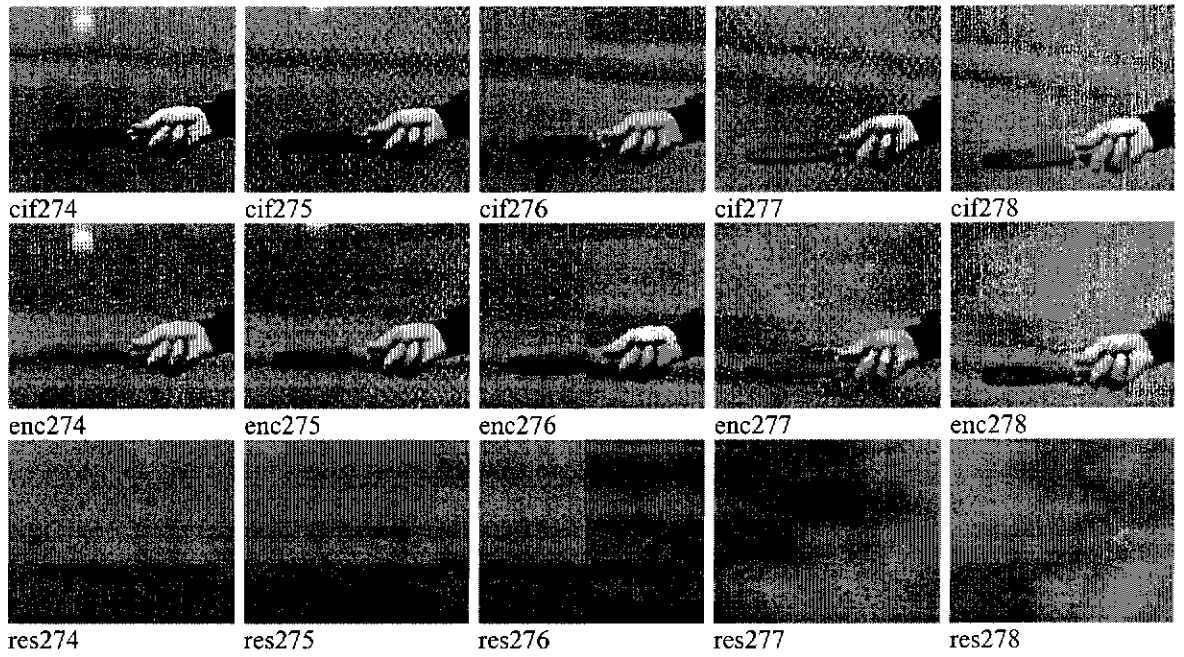


Figure 3.28: Sample pictures for the “Table Tennis” test image sequence



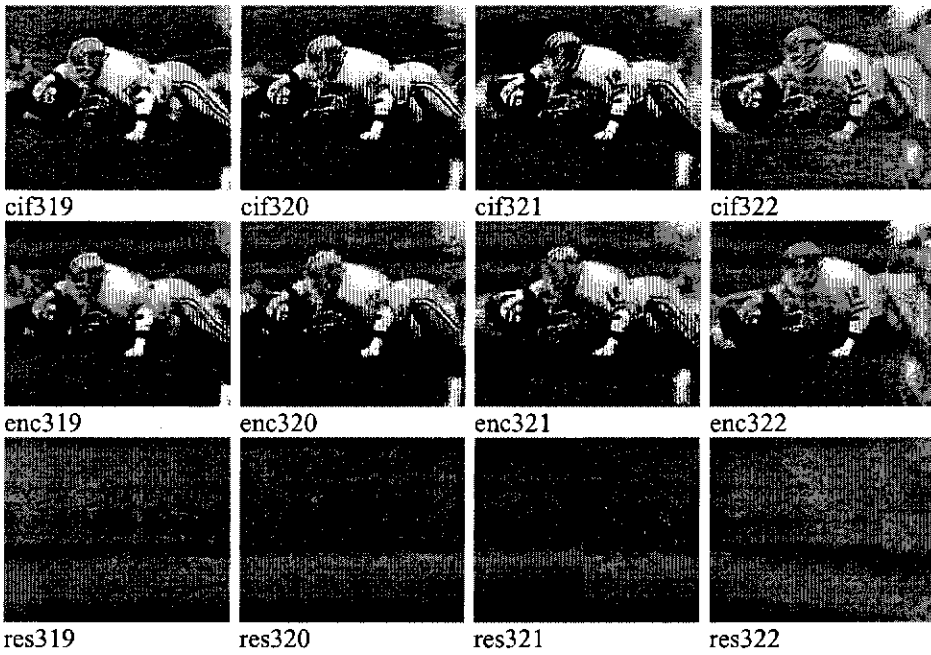
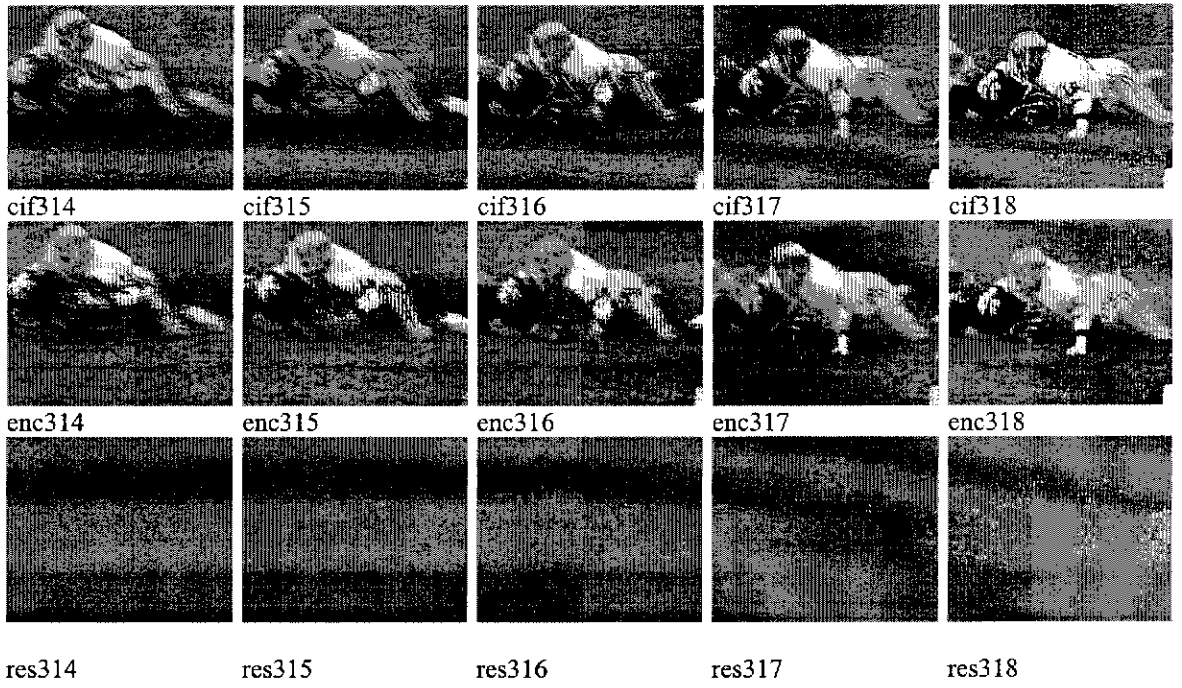


Figure 3.29: Sample pictures for the “Football” test image sequence

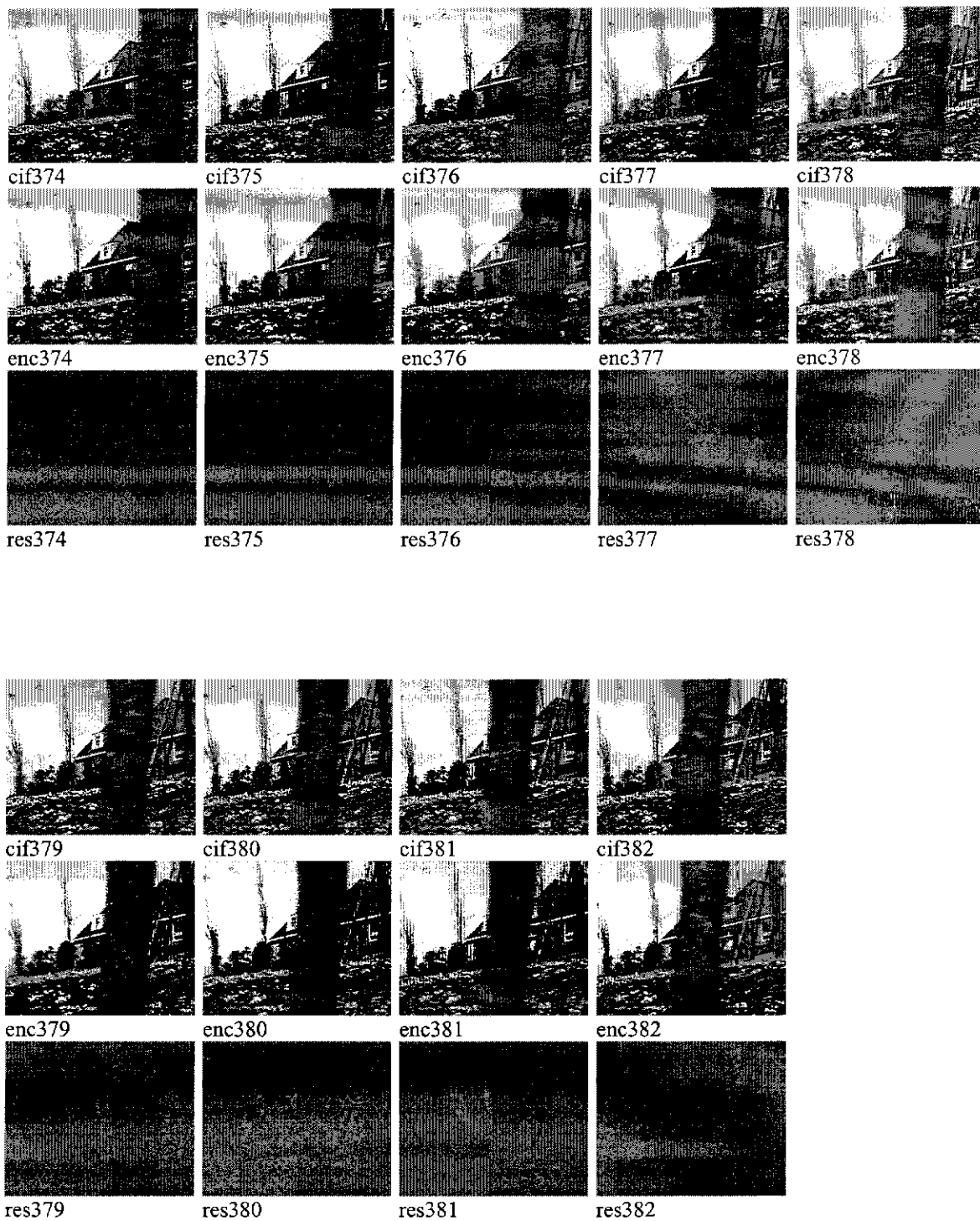


Figure 3.30: Sample pictures for the “Flower Garden” test image sequence

## **Chapter 4: Parallel Processing**

Having developed suitable algorithms for still image and image sequence compression, the issue of improving the processing efficiency of those algorithms is now investigated. For real-time applications, the lengthy processing time of the proposed algorithms using sequential processing is certainly unacceptable and hence parallel processing has been considered as the most effective way to accelerate their operation. However, before presenting the actual parallel processing algorithms for still image and image sequence compression, it is useful to discuss some fundamental aspects of parallel processing, thereby providing a better understanding of various issues involved in parallel processing.

In this chapter, a general classification of parallel processing systems is presented to facilitate the study of various parallel processing architectures and their capabilities. A discussion on the connectivity issue is then given to illustrate how parallel processors are connected to form a parallel processing system and what implications it may have on its data exchange capability. The issues of processing synchronisation and task partitioning and scheduling are examined in detail because they have significant impacts on the reliability and efficiency of parallel processing operations. Various performance measures are described as a means for assessing the processing performance of a computer system quantitatively. And finally, some important issues relating to message-passing and shared-memory multi-processor system design are discussed as they facilitate the development of the proposed parallel processing algorithms for still-image and image sequence compression presented in the next chapter.

### **4.1 Introduction**

In 1966, M. J. Flynn established a useful framework for studying parallel computer architectures by classifying digital processing into four main categories (Hobbs & Theis, 1970). The central idea of this classification is based on the fact that parallel processing could be realised in terms of data and instruction streams, each with singularity or multiplicity arrangement. As such, this classification scheme enables parallel computer architectures to be divided into four major categories, namely,

single instruction stream single data stream (SISD), single instruction stream multiple data stream (SIMD), multiple instruction stream single data stream (MISD), and multiple instruction stream multiple data stream (MIMD).

As a typical SISD computation architecture, the Von Neumann model operates with a single processor performing instruction execution tasks. It adopts a control scheme that continuously fetches instructions from its memory for the central processing unit (CPU) to execute and shuttles data between them one data unit at a time (Uffenbeck, 1987). As such, its memory is mainly used to store both instructions and data of an execution program with unique addresses. However, this model tends to suffer severe bottleneck problems because of the heavy involvement of the processor throughout its operation.

By contrast, SIMD and MIMD represent the two most prominent parallel computer architectures among parallel processing applications (Barnwell et al, 1993) (Jeschke et al, 1992) (Shen et al, 1994). A SIMD computer is typically constructed using a number of identical processors, each of which receives identical broadcast instructions to execute on their own data. Because SIMD computers require simple synchronisation mechanism and do not have to store individual programs for each processor locally, they tend to be constructed using a very large number of simple processors. In fact, this arrangement of processors presents the simplest conceptual model for the construction of vector or array processor machines. SIMD computers offer superior processing performance than SISD computers because they do not have to perform a separate instruction fetch for each data item while allowing data items to be processed in parallel.

In the case of the MIMD parallel computer architecture, each processing element (PE) gets its instructions from its own stored program instead of from a central broadcast source (Almasi & Gottlieb, 1994). A MIMD machine can also operate based on private-memory or shared-memory model. This depends on whether each PE has exclusive access to a separate block of memory or whether all PEs have direct access to a common memory region. Apart from data access mechanism, control mechanism of MIMD machines can be divided into such categories as control-

driven, pattern-driven, demand-driven and data-driven; with progressively less explicit control over the operation of individual PEs. Because MIMD machines are usually required to deal with a diverse range of applications, a more general design approach tends to be adopted for their design. As such, they are usually constructed using highly sophisticated PEs, and hence necessitating the deployment of a more conservative number of PEs.

Apart from parallel computer architectures, parallel algorithms also play an equally important role in parallel processing (Kumar et al, 1994). Basically, a parallel algorithm can be considered as a collection of independent task modules, some of which can be carried out in parallel. As such, the main criteria used to characterise parallel algorithms are often based on module granularity, concurrence control, data mechanism, communication geometry and algorithm size. Module granularity is considered as the amount of computation contained in a typical module. For optimal overall processing speed, it is necessary to compromise between module granularity and parallelism because fine granularity usually leads to increased communication overheads. Concurrence control, however, refers to the scheme in which modules are selected for execution. This control scheme must satisfy data and control dependencies to ensure proper parallel operations. While data mechanism specifies how instruction operands are being used, communication geometry reflects the interconnection pattern among computational modules. Algorithm size refers to the total amount of computation that a processing algorithm must perform. This criterion generally gives a good indication of how many processors and how much memory are required for a parallel processing system.

During parallel processing, events must happen in certain orders so that cooperative parallel processes can produce proper results (Almasi et al, 1994). In particular, synchronising concurrent activities based on access and sequence controls is of great importance in parallel processing. The need for access control arises because cooperating activities may compete for the same shared resources during parallel operations. An effective form of access control is based on the principle of mutual exclusion, whereby access to a shared resource is controlled in such a way that only

one competing process may access it at any time. Other important aspects of access control include deadlock-free operation and fairness in accessing shared resources.

While some problems can be decomposed into almost completely independent sub-tasks and thus allowing parallel processing to be applied with minimum effort, partitioning of more general problems into sub-tasks usually features a certain level of dependency. Therefore, under these circumstances, communications among sub-tasks are very critical because this enables them to exchange data and to coordinate their activities. In general, communications based on shared memory and message passing schemes are frequently deployed in parallel processing because of their flexibility. However, it is important that resource allocation for such operations should be minimised in order to achieve good processing efficiency.

For parallel processing systems lacking shared memory supports, inter-processor communications can only be accomplished via message passing. Shared memory machines, on the other hand, can provide efficient message passing supports, in addition to utilising the common memory space to hold shared state information for task cooperation. Nevertheless, a major drawback in utilising shared memory machines is their hardware complexity. For a given machine size, systems with shared memory tend to offer noticeably less peak computational power than those without (Almasi et al, 1994). This is because shared memory systems offer increased flexibility and ease of programming at the cost of additional hardware complexity, and therefore lowering their peak computational power.

Successful implementation of a parallel program also depends largely on how well its data are partitioned, especially when distributed-memory computers are involved. Basically, in order to execute a parallel program effectively, a set of sub-tasks needs to be defined for parallel execution. As such, it is important to maximise the parallelism among parallel sub-tasks. However, it is also desirable to be able to scale the amount of parallelism depending on specific hardware configuration and other run-time conditions.

Although optimal parallel processing insists on parallel architecture be designed specifically for a particular algorithm, a common bus architecture is usually adopted

in parallel processing due to its ease of construction and low cost (Parker et al, 1990) (Simar et al, 1992). However, in order to reduce bus contention in relation to shared memory access, it is a common practice to provide parallel processors with some local memory. In this way, bus contention is reduced by reducing traffic and control transfers from a central location. In addition, a centrally controlled bus scheme may also be used to reduce bus contention even further. In this case, a master processor is responsible for distributing data from its data memory to local data memory of each processor for parallel processing. When a processor completes its processing task, it will notify the master processor to retrieve processed results before another processing cycle is re-initiated.

Due to the complexity of parallel processing, various performance indices have been established for measuring the processing performance of multi-processing systems. These indices typically consist of execution rate, speed-up ratio, efficiency, and so on. Mathematical models are also available for studying the limiting behaviour of parallel processing systems. These include Amdahl's law, Weighted Harmonic Mean Principle, and limits of parallel computation (Moldovan, 1993).

## **4.2 Parallel processing architectures**

### **4.2.1 Single instruction stream-single data stream (SISD)**

A typical example for this computer architecture is the traditional Von Neumann computer model. Due to its simplicity, many computers today are still designed based on this basic architecture. In its original form, a Von Neumann computer generally consists of a single processing unit and a single memory unit for holding both instruction and data. As such, this computational model is only capable of dealing with a single sequence of instructions and operating on a single sequence of data, hence resulting in the so-called SISD computer (Kumar et al, 1994).

Strictly speaking, SISD computers do not involve any parallel processing in their normal operation. Moreover, the fact that a single memory unit is allocated for both instruction and data storage implies that only a single memory location can be accessed at any instance, resulting in high processor idle time. For this reason, attempts have been made to separate the instruction and data memory spaces, thereby

allowing instructions and data to be fetched simultaneously. This improved architecture is frequently referred to as the Harvard architecture (Hussain, 1991).

The execution speed of SISD computers is generally limited by their instruction execution rate and the speed at which information can be exchanged between memory and the CPU. Therefore, to improve the performance of SISD computers, it is necessary to optimise these two factors simultaneously. Most current optimisation techniques tend to confine to the utilisation of faster clock rate, instruction pipelining, memory interleaving and cache memory. However, these optimisation techniques also have their own limitations, and therefore, for many applications, multi-processing is regarded as a more natural and sensible approach for increasing computation performance (Kumar et al, 1994).

#### **4.2.2 Single instruction stream-multiple data stream (SIMD)**

In this parallel computer architecture, all parallel processors will execute the same instructions synchronously on their own local data. As such, this architecture represents a basic approach of data parallelism because independent data elements are processed concurrently on multiple processors. A parallel SIMD computer typically consists of a single control unit which broadcasts identical instructions to all PEs and an interconnection network which provides processor-to-processor and processor-to-memory connectivity supports. In order to maintain execution synchronisation among PEs, broadcast instructions from the control unit must be executed by each PE in lock steps (Hussain, 1991).

The key advantage of the SIMD architecture is its sheer amount of parallelism (Fountain, 1994). This feature of parallelism is realised because of the high degree of parallelism in its target data sets and the scalability of the system itself. Additionally, no new concept is required, no additional overhead is involved, and performance improvement is proportional to the number of additional processing elements available to a processing system regardless of its size. However, in practice, as SIMD systems grow beyond a certain limit, problems would arise because of the requirement for maintaining data distribution to those systems at a high rate.



A typical implementation of the SIMD architecture is to employ an array of parallel processors. This approach itself has its own advantages and disadvantages (Hussain, 1991). On the one hand, data access from the nearest neighbours in the processor-array implementation is implicitly implied due to its specific structure. The emphasis on local processing makes the processor-array implementation a highly favourable approach because local memory access is usually very efficient. Furthermore, the utilisation of an array of parallel processors makes parallelism inherently visible. And yet, another advantage of data parallelism in the processor-array implementation is that it requires a very simple flow control. Most importantly, processed results produced by processor-array systems are generally deterministic and independent of the number of physical processors. On the other hand, because a large number of processing elements are used in the construction of processor-array systems, these processing elements are usually characterised as having relatively simple structure with very limited memory space. In addition, although operation involving neighbourhoods is very efficient in the processor-array implementation, inter-processor communication over larger distances is generally difficult.

#### **4.2.3 Multiple-instruction stream-Single data stream (MISD)**

In this parallel computer architecture, multiple processors achieve parallel operation by applying different instructions to the same datum. A typical example of this computer architecture is a pipeline computer in which parallel processors are connected in a pipeline structure. Each of these processors executes different instructions on a single input data stream and outputs its processed data stream to the next processor. There are a number of important aspects relating to pipeline processing operation (Fountain, 1994). These are:

1. The efficiency of pipeline processing depends critically on the nature of problems to be solved.
2. There is always a latency in pipeline processing. Its magnitude is directly proportional to the number of pipeline stages.
3. Since pipelining itself is a sequentially organised process, programming techniques used for pipeline computer programming generally require little or no

modification from the user's perspective. Also, system optimisation is relatively easy to achieve.

4. Performance improvement in pipeline processing is directly proportional to the level of additional resource allocation. For example, the speed-up factor of a pipeline machine is generally equal to the number of stages in the pipeline.
5. A major disadvantage of applying pipelining to parallel processing is that a specific pipeline configuration is often required for each problem.

Two areas of applications receiving widespread use of pipelining are at the machine instruction level and at the algorithm level. The former is due to the fact that at the lowest machine level, many computer instructions can be broken down into a common sequence of operations, therefore allowing pipelining to be applied with maximum efficiency. At the algorithm level, pipelining can be effectively utilised in the construction of special-purposed computers to take advantage of both data and functional parallelism inherent in certain applications.

#### **4.2.4 Multiple-instruction stream-Multiple data stream (MIMD)**

A MIMD machine is generally constructed using an assembly of PEs, each of which can carry out any task either independently of or cooperatively with other PEs. As such, this parallel computer architecture can be interpreted as a basic approach to task or function parallelism. In general, parallel processors of MIMD machines are capable of acting autonomously, each executing different instruction streams together with their own local data stream. Also, synchronisation among parallel processors is usually achieved by passing messages among themselves via an interconnection network or accessing data in shared memory space. In addition, PEs of MIMD computers are considerably more complex than those of SIMD machines because they are usually designed for handling a diverse range of applications. In fact, PEs of MIMD computers are usually constructed as stand-alone Von Neumann engines with additional supports for inter-processor communications and input/output operations (Hussain, 1991).

Among the most important aspects of the MIMD architecture are memory resources and processor inter-connectivity. For example, in a shared-memory MIMD system,

shared-memory access needs to be properly arbitrated among multiple processors to avoid memory contention problems. More importantly, processor interconnections and global memory should be carefully planned so as to minimise shared memory access. This requirement is very important because it directly affects the performance of MIMD machines. Generally, PEs of MIMD computers can be arranged in the following configurations (Fountain, 1994):

1. Organising PEs in a regular structure, such as a 2-D grid, and providing direct point-to-point communication links between neighbouring PEs.
2. Providing multiple buses among groups of processors in order to alleviate the problems caused by single bus bottleneck.
3. Connecting parallel processors by means of a multi-dimensional hypercube network.
4. Supplying a full crossbar switch among parallel processors. This arrangement assures that a direct communication channel exists between any two processors, but it does not necessarily mean that such a channel will always be available since crossbar switching resources are normally shared among parallel processors.

In implementing MIMD parallel computers, special attention should also be paid to three key issues. Firstly, if a processing task cannot be broken up adequately into parallel elements, then parallel processing resources available on MIMD computers will not be fully utilised. Secondly, if too much inter-processor communication is required during parallel processing operations, then the interconnection network may become overloaded. And thirdly, if the amount of time spent to keep track of parallel processing operations is too high, then the benefits of applying parallelism may quickly diminish.

In addition, software and hardware also play an important part in improving the efficiency of MIMD parallel computers. While software needs to support the idea of functional parallelism, hardware should provide an assembly of PEs capable of operating autonomously and an adequate inter-processor communication network. However, because PEs of most MIMD computers are considerably more complex, a smaller number of them will generally be used in their construction.

One of the key factors affecting the performance of MIMD multi-processor systems is the granularity of program execution. The granularity of a parallel program can be defined as the average size of a sequential computation unit in a program, without involving inter-processor synchronisations or communications, that is, the average sequential task size. In general, because processing granularity has opposite effects on both parallelism and processing overhead, it is possible to achieve optimal processing performance by selecting an optimal level of processing granularity (Sarkar, 1989).

### 4.3 Connectivity in parallel processing systems

Depending on whether shared memory or message passing is required, connectivity among parallel processors in a parallel processing system can be implemented in various forms such as bus approach, crossbar switching, multistage networks, near neighbour connectivity, tree structure, pyramid, and hypercube (Fountain, 1994). For demonstration purposes, connectivity of a parallel processing system consisting of three processing elements is presented.

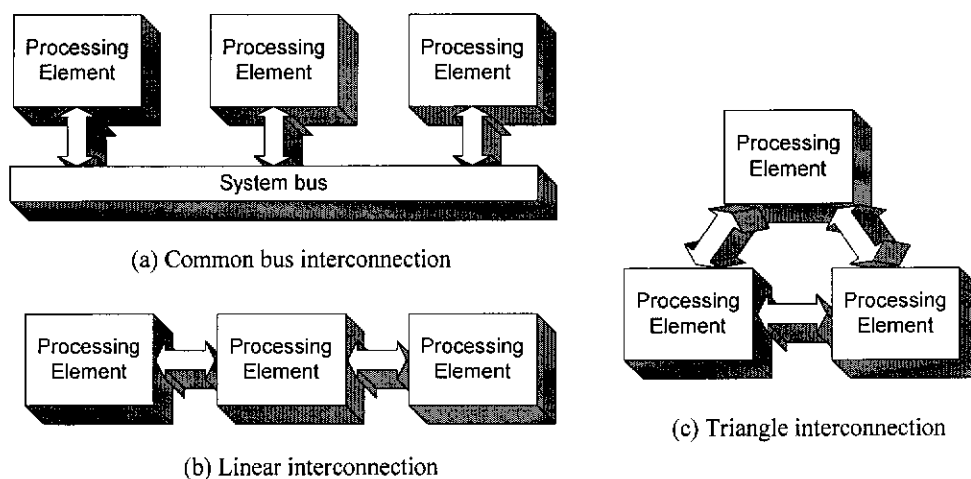


Figure 4.1: Typical connectivity for a three processing element system

As shown in Figure 4.1, a three processing element system can be arranged in a number of configurations. In a common bus approach, any pair of the three can communicate in one step, thus allowing one data item to be passed in a time unit. In a linear arrangement with near neighbour connectivity, there are two connection channels, thus allowing up to two data items to be passed in a single time unit.

However, the outer pair of PEs can only communicate by passing data via the third, thus requiring two time units for a single data item to be transferred. In a triangle arrangement also with neighbour connectivity, there are three connection channels available, thus allowing three data items to be passed in a single time unit. Any pair of the processing elements can communicate in a single step, and therefore the connectivity can be considered as completely symmetrical.

It should be noted that given a small number of processing elements, the differences in connectivity are unlikely to have any significant impact on the overall system performance. However, as far as parallel programming is concerned, parallel processing systems with unsymmetrical connectivity are generally more difficult to work with (Fountain, 1994).

## **4.4 Processing synchronisation**

### **4.4.1 Synchronisation in shared-memory processing**

Processing synchronisation always plays a critical role in ensuring reliable and efficient parallel operation in a shared-memory parallel processing environment (Almasi et al, 1994). In order to facilitate processing synchronisation among parallel processes in a shared-memory system, special low level primitives such as test-and-set instructions are frequently provided for access arbitration purposes. Normally, access arbitration using the test-and-set instructions is implemented in the form of semaphore whose usage is to enforce mutual exclusion access to a shared resource. The key feature of such an arbitration process is the indivisible nature of the test-and-set instructions. What this really means is that these instructions can read in the value of a special binary shared variable  $S$  and alter its value in one indivisible step. Therefore, by forcing all parallel processes to inspect the value of this special shared variable prior to any shared-memory access, it can be assured that only one process is ever allowed to access the shared memory while all the others would be suspended. It is for this reason that the test-and-set instructions are also known as blocking synchronisation primitives.

Apart from the test-and-set instructions, a more versatile protocol for coordinating parallel processes is based on both the concept of semaphore and a pair of higher-

level primitives which provide at least mutual exclusion access and are deadlock free. These primitives, designated as  $P$  and  $V$ , operate on a special non-negative integer semaphore variable  $S$ . In general, the  $P$  and  $V$  primitives shall operate as follows (Dijkstra, 1968):

1. During the course of  $P(S)$  execution, the process invoking the  $P$  operation will be suspended if  $S$  is equal to 0. However, as long as  $S$  is found to be positive, it will be decremented by 1 and the invoking process is allowed to proceed. As in the case of the test-and-set instructions, the successful testing and decrementing of the semaphore variable  $S$  in the  $P$  operation must be carried out in one indivisible operation to ensure coherent access arbitration.
2. During the course of  $V(S)$  execution, however,  $S$  is simply incremented by 1 in a single indivisible operation regardless its current content. In effect, this operation is equivalent to relinquishing a previously granted semaphore back to the parallel processing system so that other parallel processors are given a fair chance to access the shared resources via the access arbitration process.

With the aid of this protocol, a normal procedure for initiating parallel processes on a parallel processing system can be illustrated by the following pseudo code:

```
begin
  Integer S;      /* S is a semaphore variable */
  S = 1;
  parbegin      /* Initiate n parallel processes*/
    Process #1: begin ..... end;
    .....
    Process #n: begin ..... end;
  parend
end
```

where each of the #n parallel processes is implemented in the form:

```
Process #i:
begin
  Li: ...      /* Local processing */
  P(S);      /* For mutual exclusion execution */
  ...      /* Critical section of #i process */
  V(S);
  ...      /* Remaining local processing */
  goto Li;
end
```

In order to ensure proper parallel operations, it is required that the above semaphore variable  $S$  must not be accessed simultaneously by any other parallel processes

during the  $P$  and  $V$  operations. This requirement is particularly important because if multiple processes were able to access the semaphore variable  $S$  simultaneously, then they would all see the same value of  $S$ . This means that, for any positive value of  $S$ , multiple processes would proceed to access the same shared resource simultaneously, thus causing resource contention problems. For this reason, both the  $P$  and  $V$  operations are required to execute in a single indivisible operation.

Being a special case, a binary semaphore scheme allows the semaphore variable  $S$  to take on only two distinct values, namely, logic 0 and 1. Alternatively, if the semaphore variable  $S$  is allowed to take on additional values, then the scheme will be referred to as “counting semaphore” and the  $P$  and  $V$  operations can be used for condition synchronisation.

Fetch-and-add operation is another simple but very effective multi-processor coordination scheme. It provides critical supports for highly concurrent execution of operating system primitives as well as application programs. Although its operation principle is very similar to the test-and-set primitive, there is a fundamental distinction between them. On receiving simultaneous access from multiple processes, the test-and-set primitive only results in one process to go ahead, whereas the fetch-and-add scheme allows multiple processes to proceed, each with a unique identification number.

While the concept of semaphore has been applied extensively in access control, barrier synchronisation is generally used for sequence control in multi-processor systems (Kumar et al, 1994). In this concept, a synchronisation point known as a barrier is arbitrarily established so that no process can proceed any further until all other processes have reached this barrier. This implies that faster processes will have to wait at the barrier for slower processes to complete before they can proceed to the next processing stage. However, it is a common practice not to terminate a process at the barrier, even though it has completed its current stage of computation. Instead, it is forced to check a shared variable constantly so as to tell whether the last process has indeed reached the barrier. From then on, any process checking the shared variable will be able to proceed to the next processing stage, thus passing the barrier.

There are a number of issues concerning the operation of barrier synchronisation. Because all processes must wait for the slowest process to complete, each process at the barrier will have to force itself into an idle state if barrier synchronisation is implemented using a busy-waiting scheme. In addition, a deadlock situation may also occur at a busy-waiting barrier if the number of parallel processes exceeds the total number of PEs. This is because a busy-waiting scheme would prevent exceeding processes from making their way to the barrier indefinitely. Alternatively, although task-switching schemes can prevent PEs from entering an idle state and eliminate potential deadlock situations at the barrier, its application is generally much more complicated, involves higher overheads and often requires operating system supports.

#### **4.4.2 Synchronisation in message passing computation**

As already mentioned, both access and sequence controls play a crucial role in ensuring proper parallel operation in a multiple-processor environment. These aspects apply not only to shared memory machines but also to message passing systems. In general, message passing is a more self-synchronising form of communications than shared memory because of the nature of the message passing process itself. Nevertheless, problems such as lost or overwritten messages may still occur in message passing, and therefore necessitate processing synchronisation.

Most programming notations developed for message passing computer systems support the concept of communication channels and provide a set of primitives for facilitating message exchange operation among processing nodes (Almasi et al, 1994). In addition, if communication channels allow messages to be queued for subsequent transfers, a more flexible and efficient operation would result because a processing node can send a message and then continue its operation without having to wait for another node to acknowledge it. This mode of operation is therefore regarded as non-blocking and this form of communications is said to be asynchronous. By contrast, synchronous message passing requires that a direct communication channel be established between two processes before messages are exchanged and the operation of the sending process be suspended until all messages have been successfully received by the recipient. This implies that synchronous message passing by its own nature offers an implicit synchronisation mechanism



between processes. In addition, although synchronous message passing can alleviate the need for dynamic buffer allocation, its application needs to take into account a potential deadlock situation in which a sending process may wait hopelessly for an acknowledgment from its recipient.

#### **4.5 Partitioning and scheduling**

In parallel processing, task partitioning and process scheduling have strong influence on the overall processing efficiency of a parallel processing system (Sarkar, 1989). While partitioning is necessary to ensure that the granularity of a parallel program is coarse enough for running on a target multi-processor system without losing too much parallelism, scheduling is required to achieve good processor utilisation and to optimise inter-processor communications in a multiple processor environment. The influence of partitioning and scheduling on parallel program execution can normally be attributed to their parallelism and overheads contributions. These contributions always have opposite effects on the processing performance of a parallel processing system. In fact, the presence of parallel processing overheads makes it impossible for parallel programs to achieve theoretical speed-up factors. Nevertheless, parallel execution time can generally be minimised using optimal intermediate granularity corresponding to optimal intermediate partition. However, because programs may contain discrete structures, it may not always be possible to partition them into tasks of equal size for optimal parallel execution.

Although static partitioning offers a simple strategy for task partition and distribution, its implementation may yield a relatively poor performance (Kumar et al., 1994). This is because any imbalance in work load caused by static partitioning would lead to a situation in which some processors may become overloaded while others are under utilised. This problem is particularly severe when there is a substantial variation in the size of processing tasks. Under these circumstances, it may be more beneficial to adopt dynamic partitioning so as to balance work load among processors. A typical approach for achieving dynamic partitioning is to allow a parallel processor to obtain work from other processors with escalated work load once it runs out of work. However, it should be noted that its application usually

involves additional communication overheads because of work requests and work transfers.

Unlike task partitioning, a major concern in task scheduling is to distribute processing tasks of a partitioned program to a group of processors in such a way that the overall parallel execution time is minimised (Sarkar, 1989). Basically, the task of a scheduler is to examine the current state of process allocation among parallel processors and, when appropriate, to perform a re-allocation according to a certain scheduling scheme. In addition, it is generally desirable to apply different scheduling strategies for different types of processes such as batch versus interactive processes, system versus user processes, and so on (Bic and Shaw, 1988). As in the case of task partitioning, task scheduling also involves trade-offs between parallelism and overheads because of their contradicting nature.

As an important class of scheduling strategies, time-based scheduling is a special scheduling scheme in which all arguments of its priority function are related to time (Bic et al, 1988). Because of its flexibility and efficiency, this type of scheduling algorithms has been widely used in interactive systems where priori knowledge of process timing is not normally available. Among the most common time-based scheduling algorithms are First-In/First-Out (FIFO), Last-In/First-Out (LIFO), Shortest Job Next, Shortest Remaining Time, Round Robin and Multilevel Feedback scheduling.

#### **4.6 Processing performance**

Generally, processing performance of a computer system can be evaluated based on clock rate, instruction rate, computation rate, data precision, memory size, and addressing capability (Fountain, 1994). Being the simplest performance measure, clock rate is considered as being the least effective criterion because it gives no indication about what operations actually occur within a single clock cycle. As a result, even though two processing units may have the same nominal clock rate, it is still not sufficient to judge their relative performance. However, for evaluating processors from the same family, it is indeed a very effective indicator for their relative performance.

Instruction rate in general provides a more useful performance measure even though it is usually quoted specifically for a processor rather than a processing system as a whole. Because instruction rate offers a more direct measure of processor execution speed, it can be used more convincingly for assessing the average performance of a general-purpose computer. Similarly, computation rate also provides a fairly reliable criterion for evaluating the processing performance of processors offering high-precision arithmetic capability.

Data precision is another criterion that can be used for determining the performance of a processing system. This is because processing systems with high data precision alleviate the need for high overhead software implementations, and hence offer an effective means to accelerate computationally intensive processing tasks. However, applications for which these systems are designed should also be taken into account when data precision is used as a criterion for performance evaluation. For example, given that image information is generally represented in byte quantities, an image processing system can achieve very high processing performance by optimising its architecture specifically for byte operations.

Memory size is perhaps a particularly significant factor among parallel systems. This is because processing overhead associated with secondary memory access is extremely high and its access is often very slow as opposed to system primary memory. Similarly, addressing capability also has significant impacts on the performance of a computer system. This is mainly because any restriction imposed on its operation is likely to affect the efficiency of its program execution.

Apart from the above basic performance measures, other criteria can also be used to evaluate the processing performance of parallel computers. In particular, information communication among parallel processors is of paramount importance to the operation of parallel processing systems. For data parallelism, information exchange is mainly confined to data, whereas for function parallelism, both data and programs will be involved. In addition, due to the complexity of parallel processing, other factors such as transfer rate, transfer distance, and the presence of bottlenecks would

also have significant impacts on the overall performance of parallel processing systems (Fountain, 1994).

Being an effective measure of parallel processing performance, speed-up ratio is defined as the ratio between the time taken to solve a problem on a single processor to the time required to solve the same problem on a parallel computer with  $N$  identical processors (Kumar et al, 1994). Analytically, this relationship can be expressed as (Hussain, 1991):

$$speedup = \frac{(s + p)}{\left(s + \frac{p}{N}\right)} = \frac{1}{\left(s + \frac{p}{N}\right)}$$

where  $s$  and  $p$  denote the sequential and parallel parts of a processing algorithm, respectively; and  $(s + p) = 1$  represents the normalised total execution time required to solve the same problem on a single processor.

Although it is not possible for the theoretical speed-up ratio of any parallel processing systems to exceed the number of parallel processors, such an assertion may seem to be violated in some practical situations. However, it is believed that this violation can only occur as a result of the utilisation of a non-optimal sequential algorithm or due to specific hardware characteristics that put the sequential algorithm at a disadvantage.

In addition, it has been found that for a given problem size, the speed-up ratio does not increase linearly with the number of parallel processors (Moldovan, 1993). In fact, as the number of parallel processors keeps increasing beyond a certain threshold, it is not uncommon to discover that the performance improvement of a parallel processing system may level off and eventually decrease due mainly to memory contention and the limitation of the underlying inter-processor communication network. This phenomenon is frequently referred to as the 'saturation' effect and is governed by Amdahl's law (Hussain, 1991).

## 4.7 Parallel processing system design issues

### 4.7.1 Message-passing MIMD

Most message-passing MIMD designs can be categorised based on their interconnection networks (Almasi et al, 1994). As such, these designs consist of clusters-of-workstations (COW), hypercubes, meshes, bus-connected designs, trees, and dataflow machines. As the name implies, the COW approach achieves parallel processing by clustering a number of workstations together in a local area network (LAN), each responsible for the execution of a portion of a parallel processing algorithm. Since communication within LAN-connected clusters is relatively slow, the COW approach is generally suitable for those applications requiring modest communications among parallel processes.

In a hypercube-connected distributed-memory MIMD machine, each node has a direct point-to-point connection to its neighbouring nodes according to the structure of the underlying hypercube network. For example, in a 3-dimension hypercube network, each processing node is connected to three similar neighbouring nodes using direct point-to-point connections as illustrated in Figure 4.2.

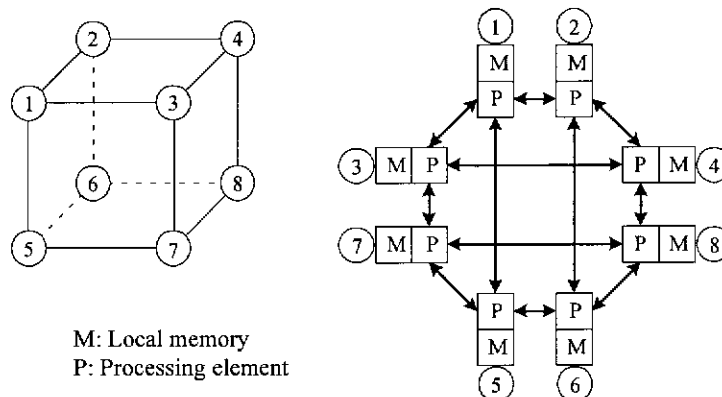


Figure 4.2: Physical connections of a 3-D hypercube machine

It is noticed that hypercube machines generally do not have any shared memory. Therefore, if a processor needs to access memory data of another processor in the network, it must send a message to that processor for such a request. This suggests that an adequate message routing mechanism must be incorporated into hypercube machine design to support message passing operations. Routing of messages in

hypercube machines can be handled by software, operating kernel or embedded hardware.

In a mesh parallel processing system, parallel processors are connected in a rectangular mesh pattern. As such, each parallel processor in a mesh connected network has a direct point-to-point connection to its four neighbouring nodes. Alternatively, a tree interconnection approach offers simple connectivity and its structure well matches the layout of VLSI chips and circuit boards. However, because PEs located at 'leaf' nodes do not have direct communication links among themselves, the tree approach is usually susceptible to communication traffic overloads at root nodes if there is too much interaction among the 'leaf' PEs. For this reason, a key issue in designing tree-connected machines is to maximise the efficiency of task partitioning and data distribution among PEs. The bus-connected approach offers another effective strategy for message-passing MIMD parallel processing design. Bus-connected MIMD designs are generally used in distributed on-line transaction processing systems due to its high performance.

Dataflow parallel processing systems normally consist of PEs which are themselves dataflow computers. A basic feature of dataflow processing model is that processing data flow from one instruction to another directly instead of via shared variables. As such, a typical operation sequence of a dataflow PE is to receive a data token, wait for remaining data tokens to arrive, then use the collected information to compute and generate new tokens, and finally pass them on to their next destination according to a predetermined dataflow graph. In addition, instead of having to rely on a program counter or other central control mechanism to schedule its processing operations, instruction execution in dataflow machines can proceed at any time following the arrival of valid data.

In addition, message-passing protocol is another important issue in designing message-passing parallel processing systems (Kumar et al, 1994) (Bic et al, 1988). Because this protocol governs all aspects of message-passing operation throughout a network of parallel PEs, its implementation has a strong influence on the overall performance of message-passing parallel processing systems. Also, in order to

facilitate process cooperation, message-passing protocol should provide adequate supports for message-based synchronisation using a set of primitives such as “send” and “receive”. In fact, these primitives provide complementary supports for message exchange operation among parallel processors as illustrated by the pseudo code in Figure 4.3.

<pre> Process #1:   loop   ...   produce_data();   send(Process #2, data);   ... end /*loop*/ </pre>	<pre> Process #2:   loop   ...   receive(Process #1, data);   process_data();   ... end /*loop*/ </pre>
--	---

Figure 4.3: Pseudo code for message based synchronisation

#### 4.7.2 Shared-memory MIMD

In shared-memory multi-processor systems design, two key issues that require special attention are the relatively long latency of shared-memory access and the provision of unconstrained, yet synchronised, access to shared data (Almasi et al, 1994). A common solution to the former issue is to use n-port memory modules. However, even under ideal circumstances, increasing the number of memory ports would also lengthen the access time to some extent. The latter issue can be resolved by adopting an appropriate sequence and access control strategy. While sequence control is necessary in parallel processing for coordinating cooperative processes, access control is required for dealing with parallel processes competing for shared resources simultaneously. In both cases, unnecessary serialisation should be avoided so that parallelism among processes can be maximised.

Apart from the utilisation of n-port memory modules, a bus-based architecture offers another effective solution for reducing the latency time of shared memory access. Basically, the development of the bus-based approach is motivated by the fact that a microprocessor does not normally saturate the available bandwidth of a typical bus system. As such, it is possible to have multiple processors sharing the same bus system for shared memory access without overloading it. However, the bandwidth available from a single bus would ultimately impose an upper limit on the total

system performance. Therefore, in order to conserve the bus bandwidth, caching has been used extensively in most bus-based shared-memory parallel systems (Sarkar, 1989). This allows the amount of shared memory access as well as the average bandwidth requirement for remaining shared memory access activities to be reduced quite dramatically.

For sequence and access control, the semaphore principle has been widely adopted in shared memory MIMD design to provide a reliable and effective means for coordinating multiple sequential processes and arbitrating simultaneous access to shared resources in a shared-memory parallel processing environment (Dijkstra, 1968). Although the semaphore principle can be implemented quite easily by software, dedicated hardware synchronisation mechanism is generally preferred due to its greater efficiency. In fact, implementation based on dedicated semaphore registers and special-purpose primitives called P-operation and V-operation offers a very efficient means to achieve mutual exclusion operation during sequence and access control.

## **4.8 Summary**

In this chapter an overview of parallel processing has been given and various issues relating to parallel processing have been discussed. The broad classification of parallel processing systems in Section 4.2 provides a useful framework for studying various parallel computer architectures. While SISD offers no parallel processing capability, SIMD and MIMD have been identified as the two most prominent parallel computer architectures because of their capability in accelerating computationally intensive operations. They achieve this by offering data parallelism and function parallelism. In Section 4.3, various arrangements for connecting parallel processors in parallel processing systems have been examined. For a typical parallel processing system with three processing elements, a common bus, linear arrangement and triangle arrangement can be used with slightly difference in data exchange capability.

The issue of processing synchronisation has been explored in Section 4.4. For shared-memory systems, processing synchronisation plays a crucial role in ensuring reliable and efficient parallel operation and can be achieved by means of access and



sequence controls. For access control, special primitives based on semaphore principle have been discussed. Their application is to enforce mutual exclusive access to a shared resource. The most important feature of these primitives is the indivisible nature. Depending on the type of semaphore variables, semaphore processes can be classified as binary semaphore or counting semaphore schemes. For sequence control, barrier synchronisation based on busy-waiting or task-switching schemes can be used to coordinate cooperative processes.

For message-passing systems, access and sequence controls also play an important role in ensuring proper parallel operations although message passing is inherently a more self-synchronising form of communications itself. Processing synchronisation is needed in a message-passing system to deal with abnormal conditions arising from lost or overwritten messages. Depending on implementation, processing synchronisation can be categorised as asynchronous versus synchronous and blocking versus non-blocking algorithms.

Task partitioning and scheduling have strong influence on the overall processing efficiency of a parallel processing system. While task partitioning is needed to ensure the granularity of a parallel program is coarse enough for running on a target multi-processor system without losing too much parallelism, task scheduling is required to ensure good processor utilisation and to optimise inter-processor communications. Static partitioning offers a simple strategy for task partition, whereas dynamic partitioning is more effective in reducing work load imbalance at the cost of extra overheads. Being an important class of task scheduling strategies, time-based scheduling has been widely used in interactive multi-processor systems.

To evaluate the processing performance of a computer system, various performance criteria have been discussed in Section 4.6. Apart from the general criteria, speed-up ratio has also been used specifically to measure the relative performance of parallel processing systems. For a parallel processing system, increasing the number of processors does not necessarily improve the speed-up ratio proportionally because of the saturation effects. This problem often occurs because of memory contention and the limitation of the underlying inter-processor communication networks.

As discussed in Section 4.7.1, message-passing MIMD designs are generally categorised based on their interconnection networks. However, most of these designs rely on point-to-point connections to provide various degrees of neighbour connectivity. Another important design issue relates to message-passing protocol. This protocol should provide adequate supports for message-based synchronisation to ensure reliable and efficient parallel operations.

For shared-memory MIMD systems, the relatively long latency of shared-memory access and the requirements for unconstrained and yet synchronised access to shared-memory data represent the two major design issues. As discussed in Section 4.7.2, while the former issue can be effectively dealt with using multi-port memory devices or bus-based architectures, the latter can be resolved using proper sequence and access controls. Their implementation is often based on the semaphore principle because it offers a reliable and effective means for coordinating multiple sequential processes and arbitrating simultaneous access to shared resources.

Now that parallel processing and various parallel processing issues have been discussed, it is the aim of the next chapter to explore how parallel processing can be applied to still image and image sequence compression applications and to evaluate how effective parallel processing can be deployed to accelerate image sequence compression operations. Practical applications of the semaphore principle for sequence and access controls are discussed together with the partitioning and scheduling operation of image processing tasks. Because of the spatial and independent nature of most image processing tasks, the application of parallel processing to still image and image sequence compression has proved to be very effective, thereby allowing it to take better advantages of parallel processing to accelerate its operation.

## **Chapter 5: Parallel Image Sequence Compression**

With an insight of various parallel processing issues discussed in the preceding chapter, it is the aim of this chapter to explore how parallel processing can be applied to still image and image sequence compression to accelerate its operation. Various experiments conducted in this study have proved that parallel processing indeed offers a very powerful processing technique for accelerating the process of image compression because of the independent and spatial nature of most image processing operations. The proposed parallel image sequence compression algorithm presented in this chapter was able to reduce the average processing time to compress four consecutive image frames in CIF format from 3'35" on a 486DX33 PC using sequential processing to 1'32" with parallel processing.

### **5.1 Introduction**

In recent years, the rapid advance of digital technology has led to a swift transition from analogue to digital in many areas. For control applications, it becomes increasingly common to design basic PID control loops with digital signal processing techniques. In telecommunications, similar transition has also occurred so rapidly that many analog systems such as conventional telephony systems and mobile phone networks will soon become obsolete. For audio applications, the introduction of compact disc in the 80s has quickly established itself as a new recording medium in the audio industry and has opened a new era for digital audio (Ely, 1996). Even in the field of television broadcasting, there is little doubt that digital television has a lot to offer in terms of flexibility, reliability and high channel capacity. In fact, it is believed that digital television has the potential to enhance human experience to such an extent that has never been experienced before (Petajan, 1992).

In spite of its benefits, the promotion for digital television has been somewhat limited mainly due to technical difficulties in processing a huge amount of raw visual data in real-time. For example, the raw data rate of a video channel capable of delivering colour pictures at a resolution of 720-by-486 pixels and a frame refresh rate of 30 frames/second is in the order of 100 Mbps. Therefore, in order to deal with such a huge amount of raw visual information, many image sequence compression

techniques have been proposed with encouraging results. In fact, some of these techniques such as H.261, MPEG-1 and MPEG-2 have already been adopted as international standards (AT&T, 1993) (Privat & Petajan, 1992). As mentioned in Chapter 3, the operation principle behind almost all image sequence compression techniques is based on the removal of statistical and psychological redundancy in both spatial and temporal domains in order to achieve high compression gain. In addition, adaptive spatial/temporal compression algorithms are generally more capable of removing statistical and psychological redundancy, hence offering superior performance.

Apart from selecting compression algorithms with high efficiency, the issue of real-time processing is also of great importance to the operation of a video compression system. If the processing speed of a video compression system cannot keep up with the rate of incoming video signals, some picture frames will have to be dropped out to keep processing synchronisation. Unfortunately, such an action often causes severe degradation to reconstructed picture quality (Musmann et al, 1985). For this reason, high performance processing hardware is always required in most video processing applications. In fact, its processing speed should at least match, if not exceed, the incoming raw visual data rate in order to satisfy real-time processing operation.

There are many alternative design strategies that can be used to enhance the processing performance of a computer system to whatever level necessary for real-time operation. However, it is believed that meeting such requirements by means of parallel processing would offer a better cost/performance ratio than just relying on high performance uni-processor systems (Hussain, 1991), (Kumar et al, 1994), (Sarkar, 1989). Accordingly, a more sensible approach to increase processing power of a processing system is to apply parallel processing with multiple processors. In fact, this approach has become increasingly viable due to the advent of VLSI technology. Such technological advances have allowed microprocessors in general and digital signal processors in particular to be designed not only with a greater level of sophistication suitable for parallel processing but also at relatively low cost.

Being one of the most computationally intensive applications, image processing has benefited significantly from parallel processing. For still image and image sequence compression, parallel processing offers a very attractive solution for accelerating many low level image processing tasks. In addition, implementation based on spatially-orientated parallel processing architecture is generally more favourable for image processing applications because of the spatial nature of image processing operations (Hussain, 1991) (Parker & Ingoldsbey, 1990). In this way, a large number of independent data elements can be efficiently processed by distributing them to corresponding processors for parallel processing. Moreover, parallel digital signal processing techniques can be applied to enhance the execution of many image processing operations even further because of the extremely high processing speed of many advanced digital signal processors.

## **5.2 Parallel digital signal processing platforms**

### **5.2.1 Overview**

When digital signal processors were first introduced in the mid 1980s, they were designed to operate primarily in a batch processing environment. As such, their normal operation required that input data were copied from external mass storage memory into their on-chip or internal memory before actual processing operation was carried out. When computation finished, they copied processed results back to external memory and the whole process started all over again for the next batch of input data. It is for this reason that many early digital signal processing applications were developed specifically for off-line processing tasks (Fountain, 1994).

However, the rapid advance of VLSI technology has fuelled the development of numerous advanced digital signal processor (DSP) architectures, thereby enabling them to work more efficiently in real-time processing environments. Basically, the fast execution speed of advanced DSPs is accomplished by applying both the so-called "Harvard architecture" and extensive pipelining. In addition, due to the nature of most digital signal processing applications, all advanced DSP designs are specifically optimised for executing frequent multiplication and accumulation instructions in a single clock cycle. Specifically, the TMS320 DSP family offered by

Texas Instruments achieves fast arithmetic operations and high throughput by applying the following techniques (Hussain, 1991):

1. Harvard architecture: Basically, the Harvard architecture can be realised by separating program and data memory banks into two separate spaces, thereby allowing fully overlapping operation during instruction fetch and instruction execution.
2. Extensive pipelining: This is a very effective technique for reducing instruction cycle time, and thus increasing the throughput of a processor. In general, processing pipelines are optimally implemented with 2 to 4 levels deep, hence enabling processors to perform 2 to 4 instructions in parallel.
3. Dedicated hardware multiplier: Since multiplication is a very common arithmetic operation to most digital signal processing applications, an effective approach to enhance the processing performance of DSPs is to use a dedicated or hard-wired multiplier which is capable of operating at much higher speed than that of a general purpose microprocessor.
4. Special DSP instructions: These special instructions are provided specifically for executing multiple instructions in a single instruction cycle. Because the processing arrangement of these instructions can be regarded as a form of parallel processing, their provision helps enhance the processing performance of DSPs. For the TMS320C3x family, these include all the parallel instructions in its instruction set.
5. Fast instruction cycle time: Another popular solution to increase the throughput of DSPs is to operate them in an ever increasing clock frequency. This results in faster instruction cycles and directly enhances the processing speed of DSPs.

In spite of significant improvement in DSP capability, processing systems relying on single DSP design may not be able to handle certain computationally intensive applications in a timely fashion such as those involving real-time video compression. Therefore, parallel digital signal processing has been widely regarded as a more favourable approach to satisfy those applications with exceptionally high processing

demand. In general, parallel digital signal processing can be accomplished in the following forms:

1. Making use of a general-purpose microprocessor as a host processor and a DSP as a coprocessor which is dedicated solely for computational intensive operations.
2. Combining a DSP and a general-purpose processor into a single chip. Also, an interactive mechanism such as those based on mailbox messaging schemes is usually provided to facilitate communications between them. Nevertheless, in this approach, programmers need to create code separately for each processor.
3. Constructing a parallel processing network and providing point-to-point links between DSPs. This approach is simple, inexpensive and easily expandable. Furthermore, to facilitate parallel program development on these systems, development tools such as the VIPER programming package offered by Spectrum Signal Processing are usually built on top of these point-to-point port connections.
4. Adopting single chip approach with massive on-chip parallel processing capability such as the TMS320C8x offered by Texas Instruments (TI). These systems are normally supported by a real-time kernel, whose function is to synchronise and coordinate parallel operations among individual on-chip processors. In the case of the TMS320C80, while the four on-chip DSPs can be dedicated for single tasking operation, the remaining on-chip RISC processor usually operates in multi-tasking configuration to provide critical supports for real-time kernel operation.
5. Making use of parallel operating system software to achieve parallel processing operations. For example, the VCOS DSP operating system offered by AT&T offers automatic task partitioning and dynamic load balancing among parallel processors. However, because VCOS needs to download application code to individual DSPs at runtime, high overheads are generally involved. Nevertheless, the ability to share low cost memory with the host processor is a major advantage of such an approach. In particular, the availability of directly addressable data space and large global memory allows processing data to be shared efficiently among the host processor and individual DSPs, thereby minimising unnecessary data transfers.

### **5.2.2 Texas Instruments TMS320C80 MVP DSP system**

Following the trend of advanced DSP design, TI has recently introduced the TMS320C80, the first member of the new 'C8x DSP family. It was claimed that the 'C80 is the highest performance and most highly integrated DSP ever produced by the company (Guttag, 1994). Basically, this DSP device contains five powerful and fully programmable processors on a single chip and is capable of delivering a combined processing power of up to 2 billion operations per second (BOPS). While four of the on-chip processors are identical fixed-point advance DSPs (ADSPs), the other is a RISC processor with a built-in floating point processing unit. In addition, the 'C80 also features a highly sophisticated DMA controller with built-in DRAM, SRAM and VRAM interface for external memory expansion purposes. While the utilisation of the sophisticated DMA controller capable of handling up to 400 MB per second helps increase the memory transfer capability of the 'C80 substantially, the built-in support for various types of memory helps simplify memory interfacing tasks and offers a cost-effective means to support mixed memory configuration for optimal system performance. Furthermore, a unique cross-bar switch architecture has been incorporated into the 'C80 design to support multiple independent parallel access to the 50 KB of on-chip SRAM memory. Its application offers substantial advantages over the conventional bus-based approach in terms of memory access capability.

Being primarily developed for multimedia and video processing applications, the 'C80 has a flexible built-in video timing control unit to facilitate interfacing to various video sources (Bursky, 1994). This video control unit has two independent programmable frame timers, hence offering the ability to simultaneously capture and display video images in horizontal and vertical presentation formats. In fact, the independent nature of these frame timers enables virtually any capture/display combination to be supported. Moreover, the fact that these timers can operate asynchronously and synchronously suggests that images can be captured at a different rate from those being displayed.

In terms of processing capability, the 'C80 possesses the following key attributes:



- Capable of handling multiplication-intensive, pixel and bit-field processing applications;
- Offering high precision floating point computations; and
- Featuring high data transfer bandwidth and flexible inter-processor communication capability.

Apart from the hardware aspects, software development tools for the ‘C80 typically consist of optimised ANSI C compiler, assembler/linker, parallel processing in-circuit emulator, and software simulator. In addition, to facilitate parallel program development, these development tools may also include a multi-tasking executive running on the RISC processor to handle basic functionality of a master processor. Its main role is to act as a kernel offering efficient interface to software libraries of user-callable functions, basic program controls and inter-task communications; and to serve as application program interface between user software and the on-chip ADSPs. This processing arrangement is illustrated Figure 5.1.

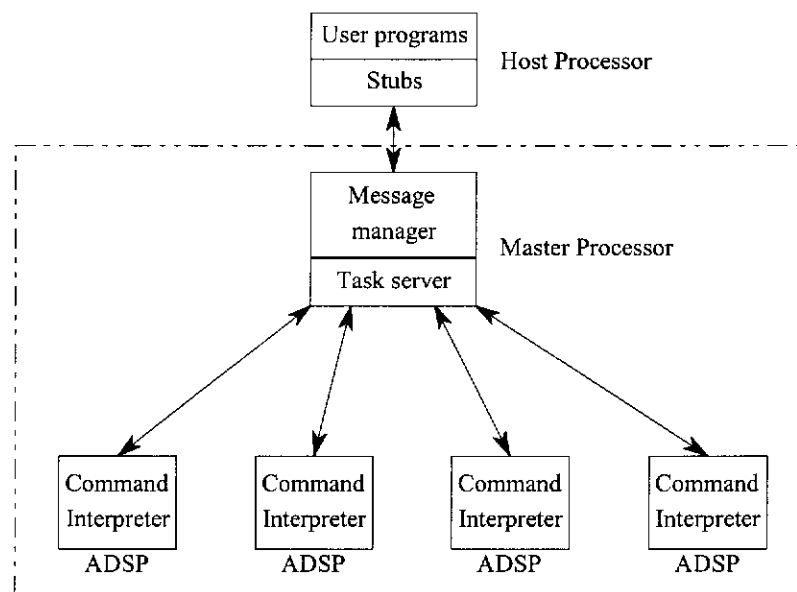


Figure 5.1: Typical processing arrangement of C80-based development system.

In spite of its advanced architecture and huge processing power, the ‘C80 development system unfortunately failed to meet other selection criteria, and therefore was not adopted for this study of parallel image sequence compression. Basically, the fact that the host processor for the TI’s ‘C80 development system was

based on a Sun SPARC work-station at the time when various parallel DSP platforms were evaluated has had a major impact on the selection process due to the lack of internal supports for it. In addition, the excessive introductory cost of the 'C80 development system has also contributed to the decision not to use it for the study of parallel image sequence compression.

### **5.2.3 Sonitech SPIRIT-40 system**

Among the key features of the Spirit-40 system is the fact that it is powered by two high performance TMS320C40 DSP chips from TI and it interfaces with an IBM/PC via the PC AT/ISA bus. As such, the Spirit-40 system already offers dual DSP processing capabilities with several additional desirable features such as flexible architecture and PC/AT based development environment. However, because the total cost of this development system at the time of inquiry exceeded the available budget for this project, this has led to a decision not to use the Spirit-40 system in the study of parallel image sequence compression.

### **5.2.4 Atlanta Signal Processors ELP DSP platform**

The ELP DSP platform offered by Atlanta Signal Processors Inc. contains a number of attractive features. In general, because the ELP DSP card was designed to be compatible with the PC/AT ISA-bus architecture, it indeed offers a relatively low cost DSP development system and ease of use. In addition, upgrading to a multi-DSP system can be easily achieved by adding up to two additional DSP add-on boards onto the base ELP DSP card. This suggests that for each AT expansion slot of an IBM/PC, up to three TMS320C31 DSPs can be supported. Moreover, the ELP DSP card has a high quality on-board 16-bit stereo ADC/DAC converter and telephone line interfacing circuitry incorporated into its design, thus making it highly suitable for audio and telephony signal processing applications.

As far as software development is concerned, the ELP DSP platform offers three different development packages: the DSP application evaluation toolkit, the DSP application developer's toolkit, and the ELF algorithm development package. As the names imply, these separate packages provide increasing levels of support for various DSP application development requirements.

### **5.2.5 Innovative Integration PC32 DSP system**

Similar to the ELP platform, the PC32 is a very high performance, low cost and IBM/PC ISA-bus compatible plug-in coprocessor card capable of delivering 30 million instructions per second (MIPS) or 60 million floating point operations per second (MFLOPS) processing power. Central to the PC32 design is the TMS320C32 DSP, the latest member of the well-known TMS320C3x 32-bit floating point DSP family offered by TI. Specifically, the 'C32 features a hardware floating point multiplier/accumulator, two separate DMA controllers, a single on-chip high speed synchronous serial port, a fully programmable interrupt controller, two independent timers, and an enhanced memory interface with dynamic bus sizing capability. Apart from the ease of use and performance advantage of 32-bit DSPs, the 'C32 also offers a cost advantage over 16-bit DSPs. In fact, the 'C32 is the first floating point DSP that broke the price barrier of less than \$US10 per unit.

Although the PC32 is a single-chip DSP card, a limited form of parallel DSP platform can be realised by deploying multiple PC32 cards within a single PC system. This approach of constructing a parallel DSP system is possible thanks to specific features in the PC32 design. Firstly, because the PC32 interfaces with the host PC via a relocatable dualport memory bank, this implies that not only high speed data transfer can be achieved but also multiple dualport memory banks can be supported, thereby allowing the construction of an efficient parallel processing system. Secondly, the PC32 features a set of on-board hardware semaphores dedicated for dualport memory access arbitration as well as parallel processing synchronisation purposes. Apart from the ease of use, these hardware semaphores offer a very effective mechanism for access arbitration and process synchronisation, thereby significantly improving the overall performance of a parallel processing system as well as its reliability during parallel operations.

Because the PC32 offers a very low cost solution with decent performance, it has been used in the construction of the prototype parallel DSP platform for the studying of parallel image sequence compression. At the time of inquiry, a basic PC32 DSP card with 32k x 32-bit on-board SRAM and depopulated on-board analog sub-system was priced at \$A739. However, for applications requiring more memory space, larger

memory options were also available at extra cost. In addition, a development package, which includes a proprietary full-featured software-debugger resident monitor and high-performance library routines for accessing on-board peripherals and supporting interprocessor communications between the PC32 and the host PC via the dualport memory, was priced separately at \$A1803. While the resident monitor provides a very critical debugging environment for debugging application software running on multiple PC32 cards, the accompanying library source code offers the user invaluable resources for application program development.

### **5.2.6 Other parallel DSP systems**

The HEPC2-PKG1 offered by TRAIQUAIR consists of a PC/AT compatible HEPC2 TIM-40 motherboard with 2MB SRAM on board. Also, included in this package are two separate HETWIN TIM-40 processing modules, each of which is driven by a well-known TMS320C40 DSP from TI. At the time of inquiry, this package was priced at \$US7995. Additional options and accessories are also available at extra cost. These include Texas Instrument compiler/assembler/linker priced at \$US1500, AT-compatible C4x emulator system at \$US3495, parallel processing debugger option at \$US495, and TI TMS320C4x C-source debugger at \$US1450.

The SLALOM-50 PC/AT compatible C5x development system also from TRAIQUAIR consists of TI C/Assembly source debugger, two TMS320C50 fixed-point DSPs, two 64KB program memory and two 64KB data memory. The pricing for this system was around \$US4495. Other accessories available at extra cost include SLALOM-50-POD remote JTAG POD assembly and software priced at \$US495, TI C2x/5x compiler/assembler/linker at \$US1500, PC/AT compatible C5x emulator system at \$US3495, and parallel processing debugger option at \$US495. Given that the SLALOM-50 system only offers fixed point processing capability, its selection would certainly not be justified at such a cost.

## **5.3 Hardware configuration of the prototype parallel DSP platform**

The basic hardware configuration of the prototype parallel DSP platform used in this study consists of a 486DX33 IBM-compatible personal computer (IBM/PC) and two additional PC32 DSP cards made by Innovative Integration Incorporation. These

DSP cards were designed in the form of 16-bit half-size IBM/PC/ISA bus compatible expansion cards, and hence must be installed inside IBM/PC machines for proper operations.

A simplified block diagram illustrating the key components of the prototype parallel DSP processing system is presented in Figure 5.2. As indicated, the interconnection between the PC and two individual DSP cards is provided primarily via a dualport memory device to cater for high speed inter-processor data communications. This useful feature of the dualport memory is realised because data exchange over the dualport memory region can take advantage of 16-bit data transfers at a much higher transfer rate. In fact, the actual data transfer rate of the dualport memory device on the PC32 cards is only limited by the effective bandwidth of the IBM/PC/ISA bus system which typically operates at a rated speed of 8.33 MHz.

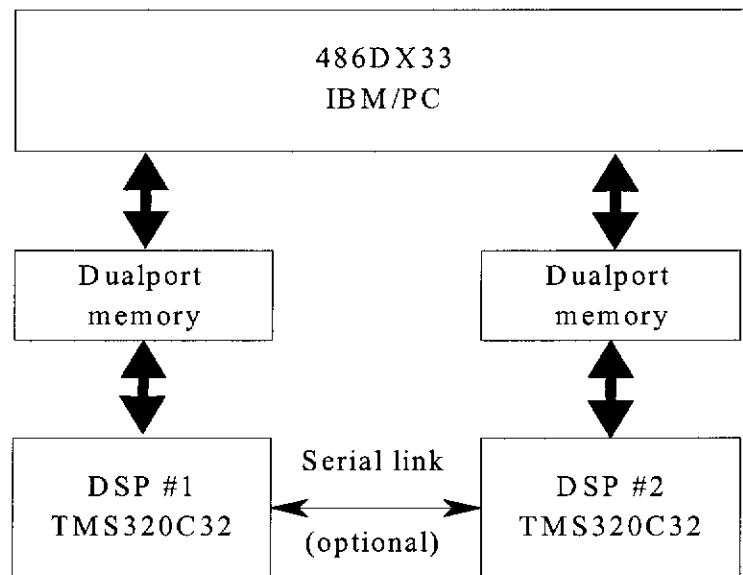


Figure 5.2: Block diagram of the prototype parallel DSP platform

Basically, the dualport memory of the PC32 is a special type of memory that allows a common block of memory to be mapped transparently to both the PC32 and the host PC memory space as if it is part of their own local memory. This suggests that during normal operation, both the PC32 and the host PC would have equal access to the same memory block, hence leading to the term “dualport memory”. Accordingly, the provision for the dualport memory enables a limited form of shared-memory to be

emulated for a parallel processing system in which there are only two processing elements, namely, the microprocessor of the host PC and the corresponding DSP of the PC32.

Because the dualport memory of the prototype parallel DSP system is regarded as a special form of shared resources, it follows that an access arbitration scheme must be established between the host PC and individual PC32 DSP cards for administering all dualport memory access. In essence, the role of the arbitration scheme is to resolve simultaneous access to identical dualport memory regions so that possible corruption of shared memory data could be prevented. As in most implementations, a popular access arbitration scheme based on the semaphore principle has been adopted in the PC32 design. Also, in order to achieve higher performance and lower overheads, a set of dedicated hardware semaphores have been provided to facilitate access arbitration process.

Specifically, each of the PC32 DSP cards has four hardware semaphore registers for access arbitration purposes. With the aid of these hardware semaphore registers, an efficient arbitration scheme can be adopted, in which a processor must possess one of the hardware semaphores prior to accessing an associated region of the dualport memory. This approach suggests that the operating software running on both the PC and the PC32 DSP cards must be able to pass access control over a region of the dualport memory by passing the ownership of a corresponding semaphore. However, it is important to note that while accessing the hardware semaphores is fully arbitrated by the hardware on the PC32 DSP cards, it is the software running on the host PC and the PC32 that is responsible for ensuring fully arbitrated access to the dualport memory. In fact, failing to possess hardware semaphores does not prevent a processor from accessing the dualport memory at all. Therefore, it is imperative that the operating software running on both the PC and individual PC32 DSP cards be fully cooperative in order to assure appropriate parallel operations.

Apart from dualport memory interface, the PC32 DSP cards can also communicate with the host PC via its I/O bus space. However, due to its lower capacity and slower transfer speed as opposed to the dualport memory interface, it is mainly provided for

the purpose of initialising the hardware settings on the DSP cards as well as exercising various control functions in relation to hardware semaphore operations.

In addition, each of the DSP cards is also equipped with a fully programmable high-speed full-duplex synchronous serial communication port. This serial port interface can be conveniently utilised to provide a point-to-point communication channel between the two DSPs on the PC32 cards, should inter-processor communications be required such as during task cooperation.

As far as processing capability is concerned, the PC32 is a single board DSP system built around the TI TMS320C32 DSP chip capable of delivering 30 MIPS sustained performance at 60 MHz clock rate. In addition, being a latter member of the TMS320C3x 32-bit high performance floating point DSP family, the TMS320C32 also offers a very high performance processing engine as normally expected from TMS320C3x DSPs. Its major data processing strengths are outlined below:

- Integer and floating-point number multiplications are both single cycle operations, thus making it highly suitable for computational intensive applications.
- Both integer and floating-point multiplications can be executed concurrently with other arithmetic-logical unit (ALU) operations using special parallel instructions.
- The provision for separate internal buses CPU1/CPU2 and REG1/REG2 enables two operands from memory and two operands from the internal register file to be fetched concurrently, thereby allowing parallel multiplication and addition/subtraction with four integer or floating-point operands to be executed in a single cycle.
- The provision of two independent DMA channels offers even better supports for performing concurrent data transfer operations, thus extending data handling capability of the TMS320C32.

Apart from the issue of DSP type, another important aspect of the PC32 cards relates to the configuration of their on-board memory. Basically, the TMS320C32 on the PC32 cards is a 32-bit DSP capable of supporting 24-bit address and 32-bit data buses. Hence in theory the TMS320C32 would be able to accommodate a total of

16Mx32-bit words within its directly addressable memory space. However, because such an exceptionally large memory size is rarely required by most DSP applications, the actual memory space available to the PC32 is significantly less and it is divided into separate memory regions for various purposes. These memory regions are designated as boot-loader memory, dualport memory, external strobe-0 and strobe-1 program memory, external data memory, IOSTROBE region, internal RAM memory and internal memory-mapped peripherals.

Because of the flexible design of the PC32, various program memory options with different block sizes up to 512kwords x 32-bit of zero-wait-state SRAM memory are available on request. In particular, only 128kwords are actually requested for the first PC-32 card and 32kwords for the second one. The main reasons for having two different memory configurations are due to budget constraints as well as anticipated research requirements. In fact, the availability of larger program memory size would offer extra flexibility in studying various parallel processing algorithms.

#### **5.4 Basic operations of the prototype parallel DSP platform**

Because of the specific connectivity arrangement and characteristics of the prototype parallel DSP platform, it is believed that this parallel platform would operate most efficiently when the PC is configured as a host processor and the two DSP cards as coprocessors. Accordingly, the main functions of the host PC are to handle those processing tasks of the proposed image sequence compression algorithm that prove to be difficult or inefficient to execute on the DSP cards, and to distribute independent tasks to the DSP cards for parallel processing. This arrangement of parallel processing is justified in this study because the host PC has the advantage of having huge system memory of its own and has ready access to image sequence data in secondary mass storage. By contrast, the DSP cards are only equipped with very limited on-board memory but are capable of processing digital data at much higher speed. In addition, in order to take advantage of the processing capability of the parallel platform, it is essential that the granularity of processing tasks distributed to the DSP cards for parallel processing be large enough to encapsulate reasonably intensive processing operations during the task distribution process. In this way, not only will the overhead incurred due to task distribution be kept to a minimum, but it



will also help improve the overall processing efficiency of the prototype parallel DSP platform because the significant processing power of the PC32 DSP cards is more fully exploited.

In addition, another important operational aspect of the prototype parallel DSP platform is concerned with its dualport memory operation. Basically, because the dualport memory of the PC32 cards can handle direct 16-bit data transfers over the IBM/PC/ISA bus, its provision indeed offers a very efficient medium for inter-processor communications during parallel processing. However, because the dualport memory can be regarded as a special form of shared memory and is subject to simultaneous access from the host PC and individual PC32 cards, it follows that some form of access arbitration must be put in place in order to ensure proper parallel operation. In fact, any non-arbitrated access to shared memory regions would likely cause their memory content to be corrupted. In addition, non-arbitrated access to shared memory data may also cause data integrity problems. For instance, a typical situation in which this problem usually occurs is when one processor tries to reference the content of a shared memory structure before it is completely updated by another processor. This means the former processor may receive partially updated information from the share structure, and hence affecting its operation.

During the operation of the prototype parallel DSP platform, all dualport memory access is arbitrated using the dedicated hardware semaphores provided by the PC32. As such, these hardware semaphores have been used extensively by the software running on both the host PC and the PC32 cards for access arbitration purposes, and have provided an efficient mechanism for controlling bidirectional data flow through the dualport memory. In particular, arrangements have been made so that operating software running on the host PC and the PC32 can easily pass access controls over a predetermined region of the dualport memory to a peer processor by simply passing the ownership of a corresponding semaphore to it. Once the ownership of a semaphore is successfully acquired by a competing processor, it will become the only processor that is deemed to have full access to the predetermined region of the dualport memory, hence effectively eliminating the chance of having multiple simultaneous access to the same dualport memory regions.

Apart from the issue of access arbitration, data packing is regarded to be another important operational aspect of the prototype parallel DSP platform. Firstly, the difference in physical memory structures between the host PC and the PC32 makes it necessary to incorporate data packing operation into dualport memory data transfer process. While dualport memory access from the host PC is performed on a 16-bit word basis, that from the PC32 only supports 32-bit long word transfers. Therefore, by imposing dualport memory data transfers in a packed format, the consistency of dualport memory access from both the host PC and the PC32 can be well maintained. Secondly, since a large portion of data transferred over the dualport memory in this study consists of visual information with byte representations, it follows that applying data packing would help improving data transfer efficiency. For example, instead of copying visual data directly into individual long word locations in the dualport memory of the PC32, it would be more efficient to pack every four bytes of visual data into a long word before transferring them to the dualport memory. In this way, the actual number of dualport memory access would be reduced by a factor of four. Lastly, applying data packing also helps conserving dualport memory usage substantially. This aspect is particularly important in this study because on the one hand the dualport memory of the PC32 cards is limited to a small memory size of 4Kbytes each, on the other hand there is a constant need to transfer a large amount of visual data to individual PC32 cards via the dualport memory for parallel processing during parallel image sequence compression process.

## **5.5 Dualport memory operation**

As depicted in Figure 5.3, the dualport memory used in this study is a special type of memory which is mapped directly to the local memory space of individual processors via both the IBM/PC/ISA bus and PC32 local bus simultaneously, hence offering a highly efficient medium for exchanging large blocks of data between the two buses. As far as the DSP on the PC32 is concerned, its dualport memory is mapped to a fixed memory region within its local memory space between 0x1000 and 0x13FF inclusive. However, on the PC side, the mapping of the dualport memory can be aligned to any 16Kbyte memory block between C000:0000 and E000:C000 memory region within the PC expanded memory space. This feature of memory interfacing

was designed to facilitate the installation process and more importantly to support multiple PC32 card implementations.

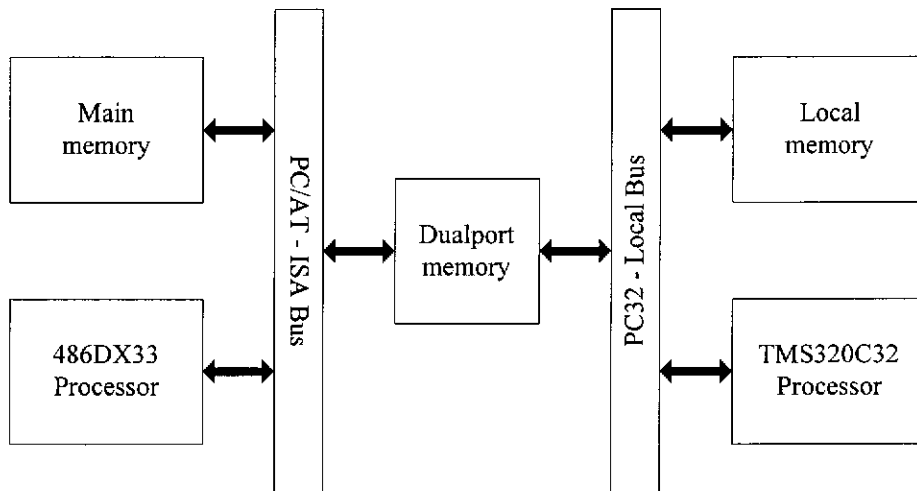


Figure 5.3: Dualport memory mapping between the host PC and the PC32

Specifically, the on-board 4KByte dualport memory of the two PC32 cards was mapped to two separate PC expanded memory regions D000:0000-D000:1000 and D000:8000-D000:9000, respectively. These memory regions were chosen to avoid hardware conflicts between the PC32 cards and other devices already installed on the host PC. Apart from the differences in the mapping addresses, the dualport memory is accessed virtually in the same way by both the PC and the PC32 as if it is part of their own local memory. However, it should be noted that because the dualport memory does not have any arbitration hardware to arbitrate simultaneous dualport memory access from both the PC and the PC32, the responsibility must rest on their operating software to provide proper arbitration.

Basically, the dualport memory of the prototype parallel DSP platform is divided into two separate regions, namely, the mailbox and common dualport memory regions. While the mailbox region is mainly used for passing short messages between the host PC and the PC32, the common dualport memory region constitutes the actual space through which large blocks of data are exchanged between them. As shown in Figure 5.4, the mailbox region is allocated enough space for holding up to four separate mailboxes, hence allowing four independent messaging channels to be supported. The reason for allocating four separate mailboxes is to match the same number of hardware semaphores available on the PC32. These hardware semaphores are

normally used for mailbox access arbitration. To further facilitate software development, a common data structure has been defined to support mailbox messaging operation. This mailbox structure basically consists of four long word members designated as mailbox receive (RCV), mailbox acknowledge (ACK), mailbox transmit (XMT) and mailbox request (REQ). As such, these long words serve as a buffer through which incoming messages are received and acknowledged, and outgoing messages are sent and flagged.

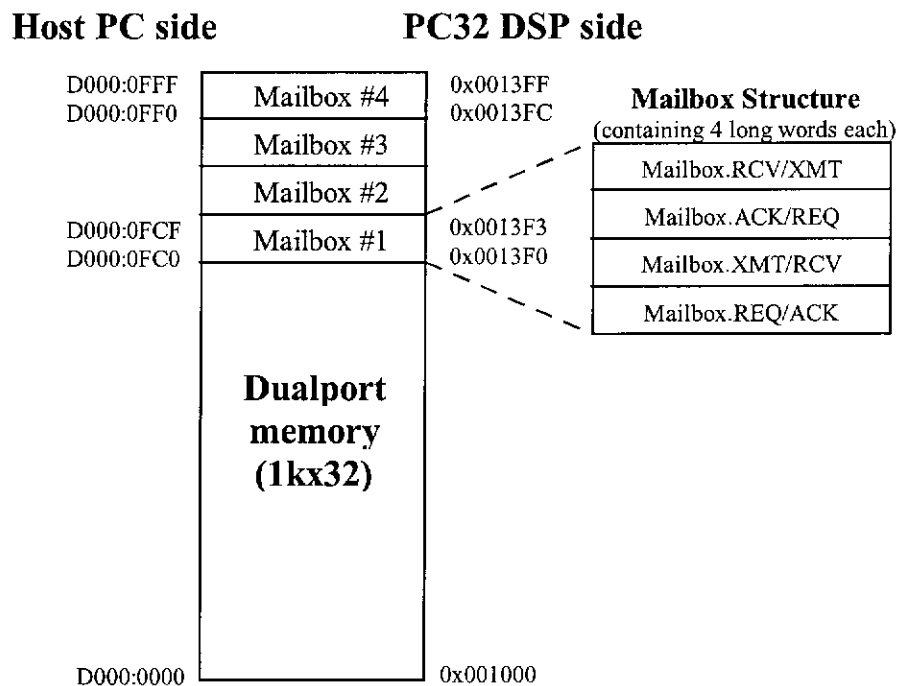


Figure 5.4: Mailbox data structure and its relative position in dualport memory.

Depending on whether a mailbox is being accessed from the PC or the PC32, the order in which mailbox data structure is organised will be different to enable full hand-shake operation. For example, a data member may be accessed as a XMT from the PC32 but as a RCV from the host PC. This is because both the XMT and the RCV in this example actually refer to the same memory location in the mailbox region. Therefore, as the PC32 sends a message to its XMT, the same message will immediately appear in the host PC RCV mailbox. Likewise, the same reasoning applies for the operation of the other pair of mailbox data members ACK and REQ.

During the operation of the prototype parallel DSP platform, an efficient mailbox messaging scheme has been adopted to facilitate interprocessor communications

between the host PC and the PC32. Specifically, all outgoing messages are required to be copied to the XMT followed by writing a long word literal of (-1) to the REQ. As this happens, the same messages and long word literal will also appear in the RCV and ACK of the peer processor due to the special arrangement of its mailbox data structure. Therefore, as soon as a long word literal of (-1) is detected in its ACK, the peer processor will be able to retrieve the messages from its RCV correctly. What happens next is that the sending processor must be notified that its outgoing messages have already been received by the peer processor so that new messages can be sent. This requirement can be satisfied by having the peer processor acknowledge the incoming messages by clearing both the RCV and ACK once it has retrieved the messages from its RCV mailbox.

In addition to passing relatively short messages, it is often necessary that large blocks of visual data are to be transferred from the host PC to the PC32 for parallel processing. However, because mailbox messaging is generally suited for exchanging short messages, the common dualport memory region has to be used instead for visual data transfers. This approach is more appropriate because the common dualport memory region occupies a much larger portion of the dualport memory, hence offering much higher data transfer capacity. In addition, the common dualport memory region could be further divided into a number of sub-regions to enable multiple channel data transfers between the host PC and the PC32 if required. This is possible because software access to these sub-regions can be easily isolated from one another. Nevertheless, for this application, the number of sub-regions being allocated is actually determined by the nature of existing processing tasks. For example, the common dualport memory of the PC32 could have been divided into four separate sub-regions to take advantages of all the mailboxes in the mailbox region. However, because the proposed parallel image sequence compression algorithm does not use multiple channel data transfers, it makes sense not to divide the common dualport memory region at all. In fact, the entire common dualport memory has been allocated for single sub-region operation.

Basically, the allocation of the entire common dualport memory to a single sub-region was adopted for two reasons. First of all, the need for transferring relatively

large blocks of visual data to the PC32 makes it necessary to trade off the number of sub-regions for their size. For example, in performing motion estimation for a 16x16 image block, a 46x46 search block will be required to cover a motion search window of +/-15 pixels/frame, hence resulting in a total of 2372 bytes of visual data that needs to be transferred to the PC32 for parallel motion estimation operation. Therefore, if the size of a sub-region is not sufficiently allocated, then transferring large blocks of data over it would have to be performed incrementally, hence leading to higher overheads and slower operations. Secondly, due to the specific processing arrangements on the prototype parallel DSP platform, the host PC only needs a single data channel for sending and retrieving data to and from the PC32. In general, the interaction between the host PC and the PC32 commences when the host PC copies a block of data to the common dualport memory region and then sends a message or command to a corresponding mailbox to initiate processing on the PC32. When processing is complete, the host PC will be notified by the PC32 via its mailbox, hence allowing it to retrieve processing results correctly from the common dualport memory region.

## **5.6 Semaphore operation**

During the operation of the prototype parallel DSP platform, semaphore has been used as an efficient and reliable arbitration mechanism for arbitrating dualport memory access and synchronising parallel operations among the host PC and the PC32 cards. Central to the semaphore arbitration process is the on-board semaphore control logic on the PC32 cards, which offers fully hardware arbitrated access to a set of dedicated semaphore registers. As such, semaphore arbitration based on these semaphore registers is inherently more efficient and reliable because at the hardware level it is a simple matter for the semaphore control logic to ensure that these semaphores can never be owned by more than one processor at any time. In addition, because access to the semaphore registers is fully arbitrated by the semaphore control logic, its operation generally does not suffer from access contention problems.

Basically, the semaphore registers of the PC32 cards are accessible not only to their own DSP via their memory bus space, but also to the host PC processor via its I/O bus space. For the host PC, the four semaphore registers of each PC32 card are

mapped to contiguous 16-bit word locations relative to a base address within the host PC I/O space. This I/O base address shall be chosen by altering the DIP switch settings on the PC32 cards so that hardware conflicts do not exist. In this application, the I/O base address of the two PC32 cards was set to 0x0280 and 0x0290, respectively. For the PC32, the four semaphore registers are separately mapped to 32-bit long word locations with fixed memory addresses of 0x81B000, 0x81B800, 0x81C000 and 0x81C800. In addition, for both cases, only the least significant bit (LSB) of the semaphore registers is significant and it is actually used to arbitrate semaphore ownership requests.

In general, access to the hardware semaphore registers is guaranteed to be arbitrated on a first-come first-served basis. In a rare instance when simultaneous access from the host PC and the PC32 actually occurs, the semaphore control logic on the PC32 cards shall resolve the situation by producing a single arbitrary winner, thereby preventing a semaphore from being owned by two processors simultaneously. However, because dead-lock situations may arise from the above arbitration process, some precautions need to be taken when operating on the hardware semaphore registers. In particular, because unsuccessful semaphore ownership requests are latched by the semaphore control logic internally, it means that these requests will be eventually granted to the requesting processors when the requested semaphores become available. Therefore, if the requesting processors do not explicitly cancel their unsuccessful requests and do not expect the semaphores to be granted, a deadlock situation would arise because it is unlikely that belatedly granted semaphores would be recognised and hence released by the requesting processors. For this reason it is essential for the operating software running on both the host PC and the PC32 to explicitly cancel any unsuccessful semaphore requests if the running software does not prepare to wait for them to be granted and intends to proceed to other tasks. In general, the procedure for acquiring semaphore ownership would involve the following steps:

1. A processor first sets the LSB of a corresponding semaphore register in order to indicate to the semaphore control logic of a semaphore request.

2. Subsequently, the requesting processor will have to examine the status of its request by polling the LSB. If the LSB is asserted, it means that the ownership request has been granted and hence full access to the associated shared resources is now given to the requesting processor. Conversely, if the LSB remains de-asserted, this implies that the requested semaphore is not currently available. As this happens, the requesting processor should explicitly cancel the request to avoid a possible deadlock situation by clearing the semaphore register before proceeding to another task. Alternatively, if it prepares to wait for its turn, it shall keep on polling the LSB until the request is granted.
3. When associated shared resources of granted semaphores are no longer in need, the requesting processor shall relinquish the associated shared resources as well as the semaphores. This action should always be taken so that another processor is given a chance to own the semaphores, hence allowing fair access to the shared resources.

## **5.7 Parallel processing algorithm and experimental results**

### **5.7.1 Parallel HOD computation**

The first experiment conducted during the study of parallel image sequence compression was to evaluate the relative benefits of applying parallel processing to the computation process of HOD. As discussed in Chapter 3, HOD computation is normally performed on a pair of consecutive frames of an image sequence. Its result is then used to evaluate MHOD, a critical parameter for the adaptive scene segmentation process. However, for experimental purposes, the computation of HOD was deliberately performed on a pair of completely different image frames instead of consecutive ones so as to simulate a worst case scenario. Normally, because consecutive frames of an image sequence tend to experience strong correlation, this suggests the computation complexity of the HOD computation process is somewhat reduced, and hence requiring shorter processing time to complete. Therefore, by using a pair of non-correlated frames, not only will this allow the worst case behaviour of the parallel HOD computation algorithm to be investigated, but this will also facilitate the study of relative performance gains due to the extra processing time involved. For convenience, the HOD computation was applied to frames #000 and



#061 of the test image sequence and this serves as a case study for the relative processing benefits between sequential and parallel HOD computation algorithms.

Basically, the relative processing benefits between the sequential and parallel HOD computation algorithms were studied based on the execution time of three different processing arrangements. These processing arrangements were chosen to represent normal sequential processing on the host PC (1PC), parallel processing using both the host PC and a single PC32 DSP card (1PC+1DSP), and parallel processing using the host PC and two PC32 DSP cards (1PC+2DSP). It is intended that by adopting these processing arrangements, it would allow parallel processing to be studied in a progressive manner and give an insight of various issues arising from parallel processing as opposed to conventional sequential processing.

Unlike the conventional sequential processing approach, the application of parallel processing in the HOD computation requires that image data be evenly distributed among parallel processors in order to achieve efficient parallel processing operation. Therefore, during the parallel HOD computation process, not only does the host PC have to undertake its own share of processing tasks, but it is also responsible for distributing image data to the PC32 DSP cards for parallel processing. This specific processing arrangement is also necessary because on this prototype parallel DSP platform, only the host PC has direct access to image data stored on its own local hard disk.

In general, since a large amount of image data is involved in the parallel HOD computation, the common dualport memory region will have to be used for transferring image data to the PC32 DSP cards for better efficiency. However, because the size of the common dualport memory region is still very limited, image data transfer to the PC32 cards over this memory region will need to be performed incrementally using smaller data packets to ease implementation. In particular, as a compromise between packet size and transfer overheads, these data packets were chosen to contain two scan lines of image data from the two corresponding image frames and served as basic data units in the parallel HOD computation process. For example, in the case of moderate parallel processing arrangement, the odd scan lines

from both image frames are processed by the PC32 DSP card while the even ones are handled directly by the host PC as illustrated in Figure 5.5. When the entire HOD computation process is complete, the results from the DSP card are combined with those on the host PC to produce the overall HOD measure for the two image frames. Similarly, in the case of maximum parallel processing arrangement, the same processing arrangement was also adopted. However, because parallel processing resources in this case have been increased, it is now possible for the two PC32 DSP cards and the host PC to process every first, second, and third scan lines of image data from the corresponding image frames, respectively.

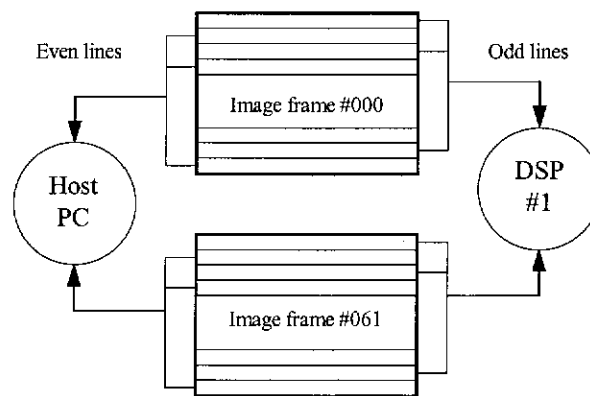


Figure 5.5: Image data distribution for the moderate parallel processing condition.

In order to facilitate data transfer operations, a portion of the common dualport memory region at least equal to the total number of pixels in two scan lines has been allocated and used as a data channel for transferring image data to the PC32 DSP cards. Because all the test image frames in this study are specified in the standard CIF frame format of 352x288 pixels, this portion of the common dualport memory equates to 704 bytes. In addition, in order to deal with possible simultaneous access to the common dualport memory region during parallel HOD computation process, a mailbox semaphore arbitration scheme has been adopted during the development of the parallel HOD computation algorithm. Accordingly, any operation involving the common dualport memory region will need to be arbitrated using a pair of mailbox semaphore directives `get_mailbox()` and `release_mailbox()` so that simultaneous access to the same common dualport memory region can be avoided.

Basically, the function of the `get_mailbox()` directive is to obtain the ownership of a semaphore that has been pre-assigned to a currently active mailbox. Also, in order to

simplify its operation and take advantage of the on-board semaphore registers of the PC32 DSP cards, a simple polling scheme has been adopted in its implementation. As such, this directive will only return controls back to the calling routine when the ownership of the currently active semaphore has been successfully acquired. When this happens, the requesting processor shall be granted full access to not only the corresponding mailbox data structure but also the common dualport memory region. By contrast, the function of the `release_mailbox()` directive is simply to relinquish the ownership of a previously granted semaphore/mailbox back to the parallel processing system, thereby giving other processors in the system a chance to own the semaphore/mailbox.

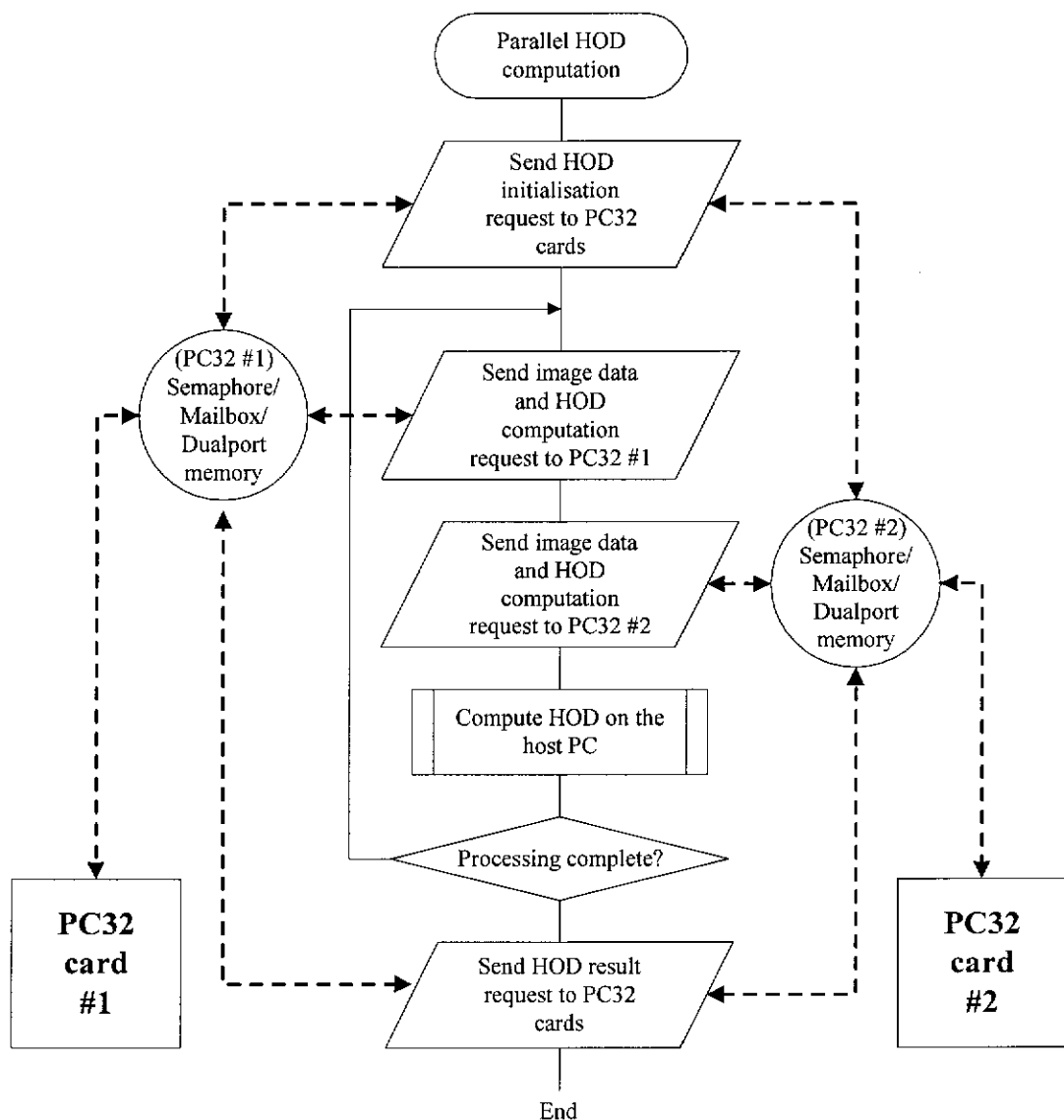


Figure 5.6: Parallel HOD computation algorithm

As far as parallel HOD computation is concerned, this arbitration scheme offers a reliable means for the host PC and the PC32 cards to coordinate their operations. The host PC uses this arbitration mechanism to distribute processing tasks to the PC32 cards for parallel processing, whereas the PC32 cards use it to service processing requests from the host PC. Normally, following a successful invocation of the `get_mailbox()` directive on the host PC side, the host PC shall transfer two scan lines of image data to an idle PC32 card via the common dualport memory region, followed by writing HOD computation command and service request (-1) codes to the corresponding XMT and REQ mailboxes before executing the `release_mailbox()` directive. On the PC32 side, the PC32 also uses the same arbitration mechanism to monitor its ACK mailbox for service requests from the host PC. As soon as a service request from the host PC is detected, the PC32 shall interpret the processing command code received via its RCV mailbox. It is important that the PC32 must maintain its ownership of the semaphore/mailbox while it is executing the HOD computation command. In doing so, the host PC shall be prevented from issuing any new HOD computation requests until the current one has been completed.

In addition, for greater processing efficiency, the parallel HOD computation algorithm shall not require the PC32 cards to return any processing results immediately following HOD computation requests. Instead, the PC32 cards shall accumulate their intermediate processing results locally while HOD computation requests are executed. This implies that the host PC must explicitly issue HOD initialisation and HOD result requests to the PC32 cards at the beginning and end of the parallel HOD computation process as illustrated in Figure 5.6. While the HOD initialisation request is needed for global variable initialisation prior to the HOD computation process on the PC32 cards, the HOD result request is used to retrieve partial HOD computation results from the PC32 cards. The host PC shall finally compute the overall HOD by combining its partial HOD results with those returned from the PC32 cards.

Experimental results for the three different HOD computation arrangements are shown in Table 5.1. As it can be seen, the results presented in this table for parallel HOD computation only reveal a marginal processing improvement over the

sequential HOD computation approach. This is mainly because the proposed parallel HOD computation algorithm actually involves a fair amount of overheads in relation to image data transfer operation from the host PC to the PC32 DSP cards. Therefore, given the simplicity of the HOD computation, such overheads would become so significant that they outweigh almost all the benefits of parallel HOD computation as illustrated in Figure 5.7. For this reason, parallel HOD computation has been removed from subsequent experiments of parallel image sequence compression.

	1 PC	1PC+1DSP	1PC+2DSP
Clock ticks	38	37	36
Execution Time (s)	2.087912	2.032967	1.978022

Table 5.1: Various HOD computation time for image frame #001 and #061.

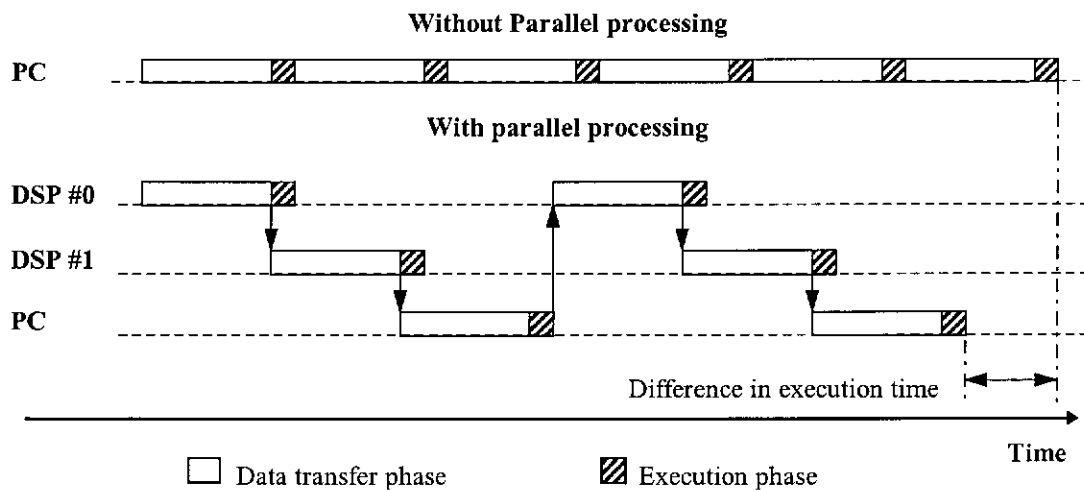


Figure 5.7: The effect of image data transfer overheads on parallel HOD computation

### 5.7.2 Parallel still image compression

As in the case of parallel HOD computation, the study of parallel still image compression was also conducted in a progressive manner in order to compare its relative processing performance to that of the sequential processing approach. In general, the application of parallel processing to still image compression based on the proposed BACVQ algorithm is expected to offer significant processing advantages over the sequential approach mainly because of the independent and spatial nature of its processing arrangement.

As discussed in Chapter 2, BACVQ is a still image compression algorithm which relies primarily on the principle of variable sized-block classified vector quantisation for its compression performance. During the operation of BACVQ, two different image block sizes of 8x8 and 4x4 are chosen adaptively depending on the nature of image blocks and they are both subject to vector quantisation processing for better compression efficiency. Specifically, following the initial segmentation of an image frame into 8x8 blocks, they are pre-processed by a classifier to determine their visual content. For those being qualified as 8x8 smooth blocks, normal vector quantisation with this block size shall be applied to maximise the overall compression gain. The rest shall be subject to a quadtree segmentation process to produce 4x4 image sub-blocks before CVQ is applied. The latter processing approach has been proven to significantly enhance the subjective quality of reconstructed images with marginal degradation in compression gain.

Basically, the application of parallel processing to BACVQ is facilitated by the fact that processing of both 8x8 and 4x4 image blocks is carried out independently from one another in the spatial domain. As such, processing these image blocks in parallel can be accomplished with little concern about the issue of data dependency, at least for those with 4x4 block size. For 4x4 image sub-block processing, an efficient parallel processing arrangement has been investigated, in which the host PC is given the task of distributing image sub-blocks to the two PC32 DSP cards for parallel processing. This processing arrangement was adopted partly because in this prototype parallel processing platform, only the host PC has direct access to image data on its own hard disk. More importantly, the reason for assigning the host PC to this simple data distribution task is to enable it to keep pace with the high processing speed of the DSP cards more closely, thereby enhancing the overall processing efficiency of the prototype parallel processing platform.

In addition, in order to facilitate parallel 4x4 image sub-block processing, a copy of the 4x4 optimised codebook for the CVQ process shall be stored locally on each of the DSP cards. A major benefit of keeping a copy of the codebook locally is that the host PC does not have to deal with the issue of dynamic codeword distribution, a process requiring substantial overheads. However, its implementation does require

32Kbytes of additional memory space for codebook storage since the CVQ codebook used in this study contains 2048 codewords. In packed data format, this additional memory space can be reduced to 8Kwords of the PC32 on-board memory as every four bytes of the codebook data are packed into a single 32-bit long word.

As far as processing interaction is concerned, parallel 4x4 image sub-block processing is always initiated by the host PC due to the host/client processing arrangement. In addition, in order to ensure proper parallel operations, a general arbitration scheme has been established between the host PC and the PC32 cards and its flowchart is referred to as “Data-Command processing arrangement #2” in appendix F. Basically, before the host PC initiates 4x4 image sub-block processing tasks on the PC32 cards, the host PC needs to identify an idle DSP card before requesting for the ownership of the corresponding semaphore mailbox using the `get_mailbox()` directive. Once the ownership is granted, the host PC will have to check if it needs to upload processed results from the previous 4x4 sub-block processing operation prior to transferring current image sub-block data to the idle DSP card via the common dualport region. And lastly, it shall issue a 4x4 sub-block processing request to its XMT and REQ mailboxes and then relinquish the semaphore mailbox ownership.

Since the PC32 cards at idle constantly monitor their mailboxes for host service requests, processing will commence almost immediately following the release of the semaphore mailbox ownership from the host PC without further host PC intervention. It is recommended that the PC32 cards should maintain its ownership of the semaphore mailbox while it is busy processing the current 4x4 image sub-block so that the host PC cannot interfere with its operation. This process of distributing processing tasks to the PC32 cards for parallel processing is set to continue until there is no more idle PC32 card or all the four image sub-blocks of a segmented 8x8 image block have been processed. When a PC32 card finishes its processing operation, it shall place the processed results back to the common dualport region and its XMT mailbox. It shall also notify the host PC of its completion status via its REQ mailbox so that the host PC can up-load the processed results from it afterwards. The processed results that are of interest include the minimum sum of

squared errors, the block type to which the 4x4 image sub-block belongs, the index of the best match codeword from the codebook and the codeword itself. Except for the minimum sum of squared errors which is sent back to the host PC via its RCV mailbox, the rest are returned via the common dualport memory region.

	Without parallel processing	Applied parallel processing to 4x4 blocks	Applied parallel processing to both 4x4 and 8x8 blocks
Execution time (s)	49	18	40

Table 5.2: Execution time for processing frame CIF061 as a still image.

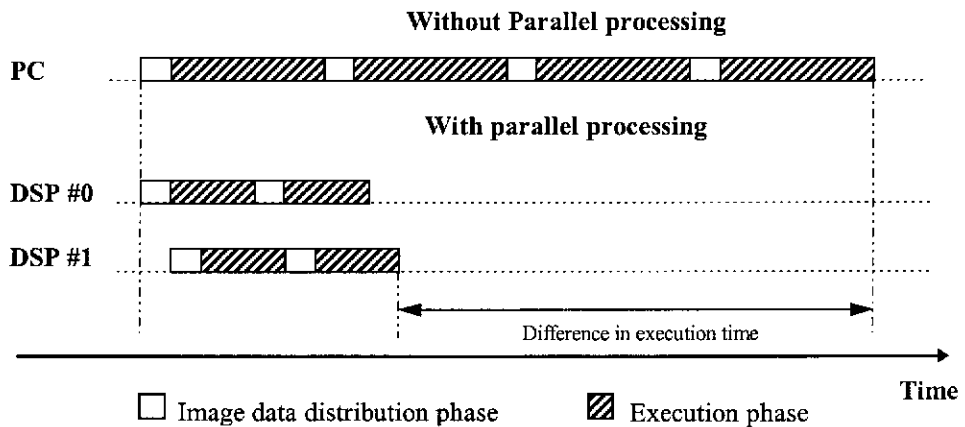


Figure 5.8: Execution timing for 4x4 image sub-block intraframe processing

As shown in Table 5.2, 4x4 image sub-blocks processing operation indeed benefits significantly from parallel processing. The intraframe processing time for image frame CIF#061 has been reduced from 49 to 18 seconds when processing was performed on the host PC using sequential processing approach and on the prototype parallel processing platform using parallel 4x4 image sub-blocks processing algorithm, respectively. This result represents a speed-up ratio of more than 2.7 for a parallel processing platform on which there are at most two active processing elements at any one time, that is, either the host PC alone or the two DSP cards. Such a processing improvement has been achieved as a result of having the host PC taking care of simple data distribution tasks while leaving more computationally intensive tasks to the PC32 DSP cards. Even though additional overheads have been involved in distributing image data to the PC32 cards for parallel processing, they are more than compensated for by the computationally intensive nature of 4x4 image sub-



block processing and the high speed processing capability of the PC32 cards as illustrated in Figure 5.8.

For 8x8 image block processing, the application of parallel processing unfortunately failed to deliver any processing gain. In fact, when the proposed parallel processing algorithm for 8x8 image block processing was incorporated into the parallel intraframe coding process, not only did it fail to offer any processing improvement, but it also offset all the processing benefits offered by the parallel 4x4 image sub-block processing algorithm. As indicated in Table 5.2, the intraframe processing time for compressing image frame CIF#061 has increased instead of decreased from 18 to 40 seconds when the parallel processing algorithm for 8x8 image block processing was applied. Such an adverse result has subsequently led to a decision not to apply parallel 8x8 image block processing during the intraframe coding process.

Basically, the shortcoming in applying parallel processing to 8x8 image block processing is caused by hardware limitation rather than the deficiency in the concept of parallel processing. Because the PC32 cards have very limited on-board memory, it follows that they cannot possibly store the entire codebook of 256Kbytes locally for 8x8 image block processing. Therefore, a compromise had to be adopted in the parallel 8x8 image block processing algorithm in which a corresponding portion of the codebook had to be down-loaded to the PC32 cards dynamically for 8x8 image block processing operation. However, given that the process of dynamic codebook data transfer has to be performed for every 8x8 image block, such a compromise would result in a significant increase in parallel processing overheads, and hence rapidly diminishing the benefits offered by parallel processing.

In addition, a rather different approach has been deployed to implement the parallel 8x8 image block processing algorithm due to dynamic codebook data distribution requirements. Basically, because the size of codebook data to be transferred to the PC32 cards is not fixed and may exceed the size of the common dualport memory region, this suggests that it would be very difficult to implement a simple parallel processing strategy in which separate image blocks and codebook data were sent to the PC32 cards for parallel processing. So, instead of having the PC32 cards process

individual image blocks independently as in the case of parallel 4x4 image sub-block processing, they can be set up to process identical image blocks in a cooperative fashion. As such, for every 8x8 image block, the same image data block shall be transferred or broadcasted initially to both PC32 cards. Subsequently, parallel processing can be achieved by dynamically distributing codewords to an idle DSP card for processing. When all the codewords have been distributed, the host PC shall retrieve and correlate the processed results from the two PC32 cards to determine the codeword that best represents the 8x8 image block.

### **5.7.3 Parallel image sequence compression**

As in previous sections, this group of experiments was also conducted in a progressive manner to facilitate the study of parallel image sequence compression algorithms. Also, a simple hypothetical GOP structure consisting of only two consecutive image frames was adopted initially for the preliminary study of a parallel motion compensated predictive coding process. While the first frame of this GOP is always processed as an I-frame using intraframe or still image compression techniques such as BACVQ, the other will be coded as a P-frame using predictive coding techniques such as forward motion compensated predictive coding.

As discussed in Chapter 3, image sequence compression generally involves both intraframe and interframe coding processes. While the first frame of a GOP, known as the anchor or reference frame, is always coded using intraframe coding, the rest are typically processed using interframe coding. Due to the similarities between intraframe coding and still image compression, the same processing algorithm based on BACVQ could be applied without any significant changes. By contrast, interframe processing generally involves not only motion compensated predictive coding to remove temporal redundancies among adjacent image frames but also residual frame processing to further eliminate spatial redundancies remaining in residual frames. In particular, forward motion compensated predictive coding and bidirectional interpolative coding are the key interframe processing techniques for P-frames and B-frames processing, respectively. Unlike forward motion compensated predictive coding which is based solely on forward motion prediction, bidirectional interpolative coding uses both forward and backward motion prediction in its

operation. To improve the prediction efficiency, a variable-sized block matching approach using 16x16 and 8x8 block sizes has been adaptively adopted in both forward and backward motion prediction. Being a by-product of the motion compensated predictive coding process, residual frames are coded using the same BACVQ algorithm as being adopted in intraframe compression due to its coding efficiency. However, because of the statistical differences between intraframes and residual frames, a separate set of optimised codebooks shall be used for residual frame processing.

Frame number	Sequence description	Execution time without parallel processing	Execution time for category A vs relative % improvement	Execution time for category B vs relative % improvement	Execution time for category C vs relative % improvement
000-001	Salesman	1'47"	0'53" vs 50.5%	0'46" vs 13.2%	0'39" vs 15.2%
061-062	Claire	1'13"	0'45" vs 38.4%	0'37" vs 17.8%	0'32" vs 13.5%
121-122	Missa	1'04"	0'41" vs 35.9%	0'32" vs 22%	0'28" vs 12.5%
181-182	Susie	1'19"	0'55" vs 30.4%	0'47" vs 14.5%	0'41" vs 12.8%
241-242	Caltrain	1'58"	1'00" vs 49.1%	0'52" vs 13.3%	0'44" vs 15.4%
274-275	Table tennis	3'21"	1'23" vs 58.7%	1'16" vs 8.4%	1'02" vs 18.4%
314-315	Football	1'51"	1'16" vs 31.5%	1'07" vs 11.8%	0'58" vs 13.4%
374-375	Flower garden	2'37"	1'06" vs 58%	0'57" vs 13.6%	0'46" vs 19.3%

Category A: Parallel processing applied to I-frame 4x4 and P-frame 8x8 motion compensated blocks.

Category B: Parallel processing applied to P-frame 16x16 motion compensated blocks plus those in category A.

Category C: Parallel processing applied to entire residual frame plus those in category A and B.

Table 5.3: Relative parallel performance gain of various processing stages.

As discussed in the previous section, intraframe processing has indeed benefited quite significantly from parallel processing due to its independent and spatial processing nature. Similarly, it has been found that parallel processing could be applied to interframe coding with promising results. As shown in Table 5.3, a number of experiments have been conducted for evaluating the proposed parallel interframe coding algorithm. This parallel algorithm was developed mainly based on parallel motion compensated coding for both 8x8 and 16x16 inter-block processing, and parallel residual frame processing. Also it can be seen that all the experimental results in this table reveal a consistent reduction in processing time although the relative processing gain may vary from one GOP to another depending on the nature of pictures being processed. For example, when parallel processing was

progressively applied to motion compensated coding of 8x8 blocks (category A) and 16x16 blocks (category B), and lastly to residual frame processing (category C), the relative processing time was reduced by 22% and 12.5% for the test “Missa” GOP. The 35.9% processing time reduction in category A, however, is aggregately achieved by applying both parallel intraframe processing using 4x4 block size and parallel motion compensated coding using 8x8 block size. By contrast, the relative processing gain for the test “Table tennis” GOP in category B and C accomplished 8.4% and 18.4% reduction, respectively. Such a variation is attributed mainly to the difference in computation complexity of 16x16 block size motion compensated coding and residual frame processing. In fact, it has been shown that as the computation complexity of a processing task becomes less significant, the achievable gain in applying parallel processing to such a task is likely to be more subdued because of the relative increase of parallel processing overheads.

Basically, parallel processing was incorporated into the interframe coding process mainly via parallel motion estimation and parallel residual frame processing. This is because both motion estimation and residual frame processing are regarded as the most crucial and time-consuming operations during the interframe coding process. Therefore, given the capability of parallel processing in accelerating computationally intensive tasks, its application to these crucial operations would offer sizeable improvements to the overall interframe coding process.

During parallel motion estimation processing, parallel processing was incorporated into variable sized block matching operation on both 8x8 and 16x16 image blocks. According to the proposed variable sized block motion estimation algorithm, motion estimation using 16x16 block size can be independently performed on every 16x16 image block. However, when any of the 16x16 motion estimation operations fails to deliver satisfactory results, motion estimation shall be performed using 8x8 blocks. As this happens, each of these 16x16 blocks shall be segmented into 4 smaller sub-blocks with 8x8 block size via quadtree segmentation before 8x8 motion estimation is carried out. For this reason, parallel 16x16 motion estimation operation is normally subject to runtime conditions, whereas parallel 8x8 motion estimation operation can proceed more independently once quadtree segmentation has been performed.

In addition, the development of a parallel 16x16 motion estimation algorithm needs to take a different approach as opposed to its sequential counterpart. This is because sequential motion estimation normally operates on each individual 16x16 image block at a time. Therefore, if the same processing algorithm were adopted for parallel 16x16 motion estimation, the extra processing power offered by parallel processors would be largely wasted because all parallel processors except the active one would be forced into idle. Alternatively, a more efficient parallel algorithm for 16x16 motion estimation is to process multiple 16x16 image blocks concurrently so that parallel processing hardware can be better utilised. However, because of implementation difficulties, this parallel 16x16 motion estimation algorithm shall be applied on a basis of individual image stripes. For CIF image format with 288x352 pixels, each of these image stripes contains 22 16x16 image blocks. Therefore, by confining this parallel processing algorithm to mere motion estimation on these 16x16 blocks, its parallel operation can be sufficiently sustained across all parallel processors, thereby improving the overall processing efficiency of the prototype parallel processing platform. However, this parallel processing algorithm does require that intermediate processing results for these 16x16 image blocks be stored temporarily for subsequent references to determine whether 8x8 motion estimation shall be applied.

In terms of implementation, the proposed parallel motion estimation algorithm was implemented on the prototype parallel platform with the host PC being dedicated to data distribution tasks and the two DSP cards acting as data coprocessors. As already discussed, this processing arrangement is more favourable because not only does the host PC have large addressable memory space needed for image data processing operation, but it also has direct access to test image sequence data on its own hard disk. So, when motion estimation needs to be performed on an 8x8 or 16x16 image block, the host PC simply has to transfer the image data block together with a corresponding search window data block, followed by an appropriate command word to an idle DSP card in order to initiate processing. When processing is complete, the host PC shall up-load processing results from the DSP cards and the whole process is repeated all over again for the next block of image data. For this experiment,

processing results returned from the DSP cards shall include the optimum motion vector and the corresponding minimum absolute motion estimation error (MEE).

Since a large amount of image data is involved in the motion estimation process, the common dualport memory region shall be used for transferring image data to the DSP cards for parallel processing. For example, given that 8x8 motion estimation is required to cover up to  $\pm 15$ -pixel movement per frame, the amount of image data to be transferred to a DSP card for each motion estimation request would consist of 64 bytes for the image sub-block itself plus 38x38 or 1444 bytes for the search window. These two sets of data are copied to the common dualport memory region in cascade in a packed format to facilitate data transfer operations. Also, since these blocks of data are fixed in size, the DSP cards would be able to extract them from the common dualport memory region quite easily.

Unlike image data transfer, command words for initiating 8x8 and 16x16 motion estimation operation on the DSP cards will be sent via the mailbox data structure. The main reason for this is two-fold. Firstly, since the processor on the DSP cards at idle always polls its mailbox for new commands from the host PC, this suggests that all host PC commands must be transferred via the mailbox data structure in order to get its response. Secondly, as image data needed for motion estimation has already been transferred to the common dualport memory region beforehand, a simple command word would be sufficient for initiating processing on the DSP cards. In fact, throughout this experiment, a unique integer literal shall be assigned to individual command for all parallel processing operations.

As mentioned earlier, motion estimation results are returned to the host PC via the common dualport memory region. These results mainly consist of the x- and y-components of the optimal 2-D motion vector and its corresponding absolute MEE. Because both motion vector components are limited to a range of  $\pm 15$  pixels/frame and the worst case minimum absolute MEE never exceeds 16320 ( $8 \times 8 \times 255$ ) and 65280 ( $16 \times 16 \times 255$ ) for 8x8 and 16x16 motion estimation operations respectively, it follows that both the motion vector components and the minimum absolute MEE can be packed nicely into a single 32-bit long word data structure to reduce the amount of

dualport memory access from the host PC. Specifically, a long word data structure consisting of two 8-bit fields for holding the two motion vector components and a 16-bit field for the minimum absolute MEE has been allocated for processing result retrieval purposes during parallel motion estimation process.

It should be noticed that although motion estimated blocks are required for the construction of residual frames, they are not returned as part of the processing results. This is firstly because the host PC can extract motion estimated blocks itself quite easily using returned motion vector information. Secondly, by not returning motion estimated blocks, parallel processing overheads can be partially reduced. This implies that the turn-around time for processing the next image block will be shortened, thereby enhancing parallel processing performance. In fact, the actual benefit of this arrangement is realised when motion estimation does not produce satisfactory results and quadtree segmentation or intra-coding has to be performed. Other considerations include data dependency issues, ease of processing and less memory usage on the host PC as it does not have to deal with returned motion estimated blocks immediately.

Apart from 8x8 and 16x16 parallel motion estimation, parallel residual frame processing plays an important role in accelerating interframe processing as indicated in Table 5.3. On average, parallel residual frame processing accounts for a further 15% reduction in the overall processing time of the parallel image sequence compression process. In spite of being based on the same BACVQ processing algorithm, the processing efficiency of parallel residual frame processing was observed to be much higher than that of parallel intraframe processing. This is mainly because residual frame processing generally requires much smaller codebooks. Therefore, apart from computational advantages associated with smaller codebook operation, residual frame processing can operate with significantly less memory requirements. This aspect of residual frame processing is particularly important in parallel processing environment where local memory size is a very influential factor to the overall processing efficiency of a parallel processing system.

The fact that all the codebooks for residual frame processing are small enough to be stored locally on the two DSP cards has enabled it to take full advantage of the processing capabilities of the prototype parallel processing platform. In particular, as all of its codebooks can be accessed by the DSP cards locally, there would be no need for performing dynamic codeword transfers during the parallel residual frame coding process, hence offering a significant reduction in parallel processing overheads. In addition, because of the availability of locally stored codebooks, the proposed parallel residual frame processing algorithm does not have to deal with 4x4 and 8x8 blocks separately as in the case of parallel intra-frame processing. Instead, individual DSP cards are allowed to make their own decision on whether 4x4 residual block processing needs to be performed.

Basically, the proposed parallel residual frame processing algorithm was developed using a similar parallel processing arrangement as adopted by the parallel motion estimation algorithm. In this arrangement, the host PC will serve as a host processor whose functions are mainly to distribute processing tasks to individual processors on the DSP cards for parallel processing and coordinate their operation. The DSP cards will act as parallel processors whose only task is to execute processing requests from the host PC. Accordingly, parallel residual frame processing will be initiated by the host PC and its operation can be sustained as the host PC constantly monitors the status of the two DSP cards. As soon as an idle DSP card is detected, the host PC will upload processing results from the previous processing task before transferring another 8x8 residual block followed by a residual block processing command to the idle DSP card to initiate residual block processing.

While residual block data are transferred to the DSP cards via the common dualport memory region, command words are always sent to them via the mailbox data structure as a mechanism to synchronise their processing operation. As soon as residual block processing commands are received, processing on the DSP cards will commence in accordance to the BACVQ coding algorithm for residual frame processing. When processing is complete, the DSP cards will return processing results containing the index of the best matched codeword and a corresponding reconstructed residual block back to the host PC via the mailbox and the common



dualport memory regions, respectively. If 4x4 residual block processing has been involved, four codeword indices instead of a single one will be returned to the host PC as illustrated in Figure 5.9. This condition can be detected by examining the value of codeword index 0.

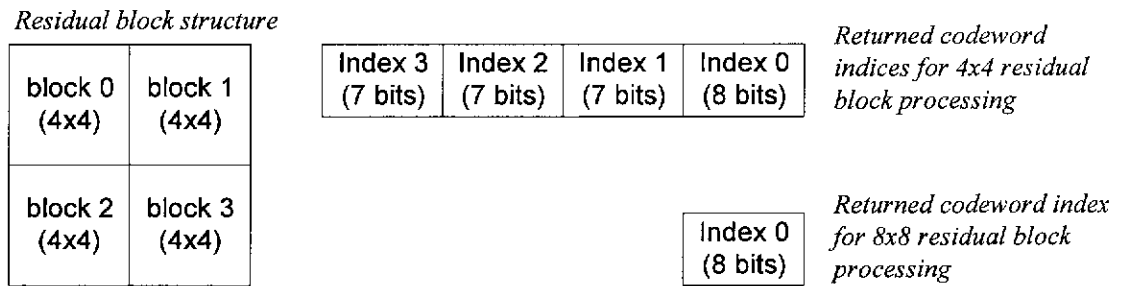


Figure 5.9: Residual block structure and returned codeword index arrangement.

Because the distribution of residual blocks to the DSP cards is performed dynamically, the idle time incurred while waiting for the next service request from the host PC can be largely reduced. Normally, the host PC shall keep on distributing residual blocks to the DSP cards for parallel processing until they all become busy with their processing tasks. As this happens, the host PC shall then poll its ACK mailboxes to determine which DSP card has actually completed its processing operation. Naturally, when processing completion acknowledgment is received, the host PC shall up-load processing results from the DSP cards before initiating another processing request. While processing results are being up-loaded and new processing requests are being set up, the DSP cards are in fact forced into an idle state. However, because the above processing overhead is considered to be less significant than the actual residual block processing operation, its effect is believed to be quite minimal.

Having investigated the relative benefits of incorporating parallel processing into various coding stages of a typical image sequence compression process, an optimised parallel image sequence compression algorithm was then developed and tested on the prototype parallel processing platform for its effectiveness. Specifically, this algorithm was tested against a number of relatively short representative GOPs extracted from the test image sequence. These GOPs, ranging from that of the “Salesman” to that of the “Flower garden”, represent a group of natural scenes that feature a very different level of visual details as well as motion activities. Such a

diversity is generally desirable because it allows the experiment to be conducted in a more realistic condition, thereby improving the generality of experimental results.

As indicated in Table 5.4, each of the representative GOPs was constructed using the first 4 consecutive image frames of individual video segments of the test image sequence. This specific configuration of GOP was adopted in this experiment so that the partial contribution from parallel I-frame processing, parallel P-frame processing and parallel B-frame processing to the overall image sequence compression process could be evaluated with shorter overall processing time. For example, while the first frame of these GOPs is coded as an I-frame via parallel I-frame processing, the second and third ones shall be coded as B-frames via parallel B-frame processing and the last one as a P-frame via parallel P-frame processing.

Frame number	Sequence description	Execution time without parallel processing	Execution time with parallel processing	Speed-up factor
000-003	Salesman	3'19"	1'25"	2.34
061-064	Claire	2'39"	1'18"	2.04
121-124	Missa	2'32"	1'12"	2.11
181-184	Susie	2'52"	1'27"	1.98
241-244	Caltrain	3'41"	1'32"	2.40
274-277	Table tennis	5'17"	1'57"	2.71
314-317	Football	3'47"	1'51"	2.05
374-377	Flower garden	4'37"	1'40"	2.77
	<b>Average</b>	<b>3'35"</b>	<b>1'32"</b>	<b>2.32</b>

Table 5.4: Performance gain of parallel image sequence compression.

Because of the different nature of the GOPs and the adaptability of the proposed image sequence compression process, the overall execution time required to process them generally varies quite substantially as indicated in Table 5.4. For example, before parallel processing was incorporated into the coding process, the overall execution time required to compress a sample GOP was observed to vary between 2'32" and 5'17". Similarly, after parallel processing was deployed, such a processing time variation was observed to drop to between 1'12" and 1'57". Although the exact mechanism affecting the overall processing time is difficult to establish analytically

due to its complexity, it will suffice to say that the overall processing time is generally affected by a number of factors such as the number of iteration steps required to determine the optimum code words for 8x8 and 4x4 image blocks during BACVQ coding process, the number of iteration steps involved in determining the optimum motion vectors for 16x16 and 8x8 image blocks during motion estimation process, and lastly the proportion of the above processing operations themselves.

Naturally, the more iteration steps are involved, the more computation cycles will be required, hence leading to increased processing time. In addition, because quadtree segmentation has been adopted in both BACVQ and motion estimation, processing tasks would increase in number by a factor of 4 whenever sub-block processing is required. That is, when quadtree segmentation is involved, processing will have to be extended to deal with 4 additional image sub-blocks for every image block being segmented. Although the number of processing tasks increases substantially when quadtree segmentation is involved, it does not necessarily mean that it would take significantly longer to process all four image sub-blocks than a single image block because sub-block processing is generally more efficient. Therefore, depending on the proportion of image block and image sub-block processing operations during BACVQ and motion estimation, the overall processing time of the proposed image sequence compression process will vary accordingly.

In spite of the relatively large variation in processing time, this study has clearly demonstrated that parallel processing indeed offers an effective means to accelerate image sequence compression operation. On average, the overall processing time required to compress a sample GOP consisting of 4 consecutive image frames in CIF format has been reduced from around 3'35" to 1'32", that is, a speed-up factor of 2.32. These encouraging outcomes have been achieved as a result of applying the proposed parallel image sequence compression algorithm which consists of parallel intraframe/residual frame processing, parallel B-frame processing and parallel P-frame processing. While parallel processing algorithm is a key component allowing a computational process to take advantages of the parallelism of the underlying parallel processing hardware, it is the parallel processing hardware that enables parallel processing to actually take place.

Although the combination of parallel processing algorithm and parallel processing hardware in this experiment only offers a moderate speed-up factor, it does not necessarily mean that this is the ultimate improvement that parallel processing could offer. In fact, the moderate processing improvement in this experiment can be attributed to a number of implementation factors. These include the limitations of the parallel processing hardware, the compromises that the proposed parallel image sequence compression algorithm has to adopt to overcome those limitations, and some inefficiencies in the existing parallel processing arrangement itself. As mentioned earlier, most of the limitations of the prototype parallel processing hardware are related to the small size of the on-board and dualport memory on the DSP cards. In particular, the shortage of on-board memory is conceivably the most influential factor in this study because it inhibits large VQ codebooks to be stored locally, thus jeopardising severely the efficiency of the proposed parallel processing algorithm. In addition, although the small dualport memory size does not manifest the same level of drawbacks as the on-board memory size does, its limitation has certainly constrained the implementation of the proposed parallel processing algorithm to some extent. For example, the shortage of the dualport memory makes it very difficult to incorporate a dual channel data transfer scheme into the proposed parallel processing algorithm. Its operation principle is similar to that of double buffering techniques, and therefore it could have been used to effectively reduce the idle time on the DSP cards.

Because of the above hardware limitations, some compromises have been made to the parallel image sequence compression algorithm in which the host PC has to take part in some processing operations instead of being dedicated to the task of coordinating parallel processing operations among the DSP cards as originally planned. For example, due to the on-board memory size limitation, the host PC is required to undertake 8x8 image block processing tasks during parallel intraframe processing operations. This suggests that while the host PC is busy analysing and processing 8x8 image blocks, the DSP cards virtually have nothing to do, thus adversely affecting the overall efficiency of the parallel processing algorithm.

Similarly, compromises have also been made during the development of parallel motion estimation algorithm for B- and P-frame processing. These compromises are needed mainly because of the provision for the coding process to switch to intraframe coding mode should motion compensation produce unsatisfactory results. For example, instead of just distributing 16x16 image blocks and search window blocks to the DSP cards for parallel motion estimation operation and then expecting processing results to be returned once processing is complete, the host PC now has to adopt a two-stage parallel processing approach as a compromise. In the first stage, parallel motion estimation is performed on all 16x16 image blocks of an image stripe so that parallel processing can be sustained. In the second stage, the host PC will analyse the returned motion estimation results in the first stage to determine whether further segmentation is needed. If further segmentation is indeed required for any of the 16x16 image blocks, a similar parallel motion estimation process will be applied to each of them at a time, each containing 4 8x8 image sub-blocks. The returned results are once again analysed to determine whether intraframe coding mode should be attempted for any sub-blocks. In other words, the implementation of the parallel motion estimation algorithm has been severely compromised and the two-stage processing approach has been adopted to sustain parallel processing operation and to reduce DSP card idle time at the cost of increased image data transfer overheads between the host PC and the DSP cards.

Because of the limitations of the parallel processing hardware and the compromises in the implementation of the proposed parallel image sequence compression algorithm, the overall processing efficiency of parallel image sequence compression process in this study has been adversely affected. Not only do they prevent the proposed parallel image sequence compression algorithm from taking full advantages of parallel processing, but they also cause a substantial increase in parallel processing overheads due to additional image data exchange and processor interactions involved. In addition, at any stage throughout the parallel processing process, there are always at most two active parallel processors. This generally occurs when the host PC does not directly involve in processing operations, but instead acts as a host processor to distribute processing tasks to the DSP cards for parallel processing. It is

for these unfavourable conditions that partially explains why only a moderate average speed-up factor of 2.32 has been obtained in this study.

Although the level of processing improvement achieved in this experiment is not quite sufficient for real-time applications, it is believed that such deficiency could be overcome by utilising more powerful parallel processing hardware, improving programming development tools, and optimising the proposed parallel image sequence compression algorithm. Currently, the experimental results indicate that the average processing time required to compress an image frame is about 23 seconds/frame, whereas the target processing time for real-time image sequence compression at a frame rate of 50 frames/second is in the order of 0.02 second/frame. Therefore, it is quite apparent that the prototype parallel processing system used in this experiment could not meet the processing requirements of real-time video compression applications. In addition, given the magnitude of processing improvement required, it would be necessary that the prototype parallel processing system be radically overhauled in order to ascertain global optimisation throughout the image sequence compression process.

Basically, the shortcoming of the prototype parallel processing hardware in delivering sufficient processing performance in this study can be attributed to the limitations of the host PC architecture, its slow memory transfer speed, and the shortages of on-board and dualport memory on the DSP cards. Specifically, due to the 640-KByte conventional memory restriction on the PC architecture, a third-party driver has been used to gain access to the extended memory on the host PC. While access to the extended memory region is necessary to satisfy the huge memory requirement of the proposed image sequence compression algorithm, its usage also involves performance trade-offs because extended memory access is generally not as efficient as conventional memory access. Therefore, given that large blocks of image data are to be exchanged between the conventional and extended memory regions regularly during image sequence compression operation, the adverse effect of slower extended memory access on the overall processing performance would be quite significant. In addition, PC expansion bus imposes further restrictions on how quickly image data can be transferred from the host PC to the DSP cards for parallel

processing. For a standard PC/ISA bus system, although the bus speed is specified at 8MHz, the actual average data transfer speed is generally much lower because normal bus access may require more clock cycles to complete and the same bus bandwidth has to be shared among various system devices. For a CIF image sequence with a frame refresh rate of 50 frames/sec, the raw image data rate would be around 4.8 MB/sec. This means that for image sequence compression applications with CIF format, a real-time capable parallel processing system must have an average data transfer rate well exceeding 4.8 MB/sec to ensure proper operations. Finally, the shortages of on-board and dualport memory on the DSP cards have significant impacts on the processing performance of the proposed parallel image sequence compression algorithm. This unfavourable aspect has already been discussed earlier in detail.

In addition to parallel processing hardware, programming development tools also play a significant part in enhancing processing efficiency of parallel processing systems. In particular, although high level programming language offers many desirable features such as code portability and the ease of code development, its application may sometimes have to compromise on code size and execution speed because it is generally difficult for high level compilers to produce as efficient code as hand-coded assembly counterparts. This aspect of high level programming language is discussed in more detail in appendix H.

Finally, apart from eliminating implementation trade-offs in the existing parallel image sequence compression algorithm, it is also important that other optimisation be adopted. In particular, eliminating the need to access image sequence data via the host PC hard disk and reducing data transfers between the host PC and the DSP cards would have significantly improved the performance of the proposed parallel image sequence compression algorithm. While accessing image sequence data via the host PC hard disk has been adopted in this experiment to ease implementation difficulties, such an approach is certainly not appropriate for assessing the suitability as well as the actual benefits of parallel processing in real-time image sequence compression applications because of the slow access speed of PC hard disk. In fact, a more common image sequence compression system would have image data acquired and

stored directly in its own primary memory, thus eliminating the need to access image data via the host PC hard disk altogether. Apart from this, reducing unnecessary data transfers between the host PC and the DSP cards would certainly be an effective measure for accelerating the processing speed of the proposed parallel image sequence compression algorithm. For example, during the parallel motion estimation process, the requirement that an entire search window block be down-loaded to the DSP cards for every parallel motion estimation request may not be necessary because search window blocks normally overlap one another. Therefore, if a more efficient scheme could be developed for extracting search window blocks, the DSP cards could then re-use a portion of previously transferred search window data, thereby reducing the amount of image data that need to be transferred to the DSP cards.

## **5.8 Summary**

In this chapter, an overview of DSP chip design has been given and the main features of different parallel DSP platforms has been discussed. Although there has been considerable improvement in the design of advanced DSP chips to enable them to run significantly faster than general-purpose processors, it is unlikely that existing single-chip designs would be able to deliver sufficient processing power for many computationally intensive applications. Therefore, there has been a rising trend in adopting multiple processor designs, hence leading to the development of many parallel DSP systems. Parallel DSP systems can be designed in various forms ranging from the utilisation of an additional DSP coprocessor in a general-purpose uni-processor processing system to the deployment of multiple processors networked into a truly parallel processing system. While parallel processing platforms based on the TMS320C80 processor inherit the on-chip parallel processing capability of the processor itself, others rely on multiple on-board processors or multiple DSP boards to provide parallel processing capability. In particular, a prototype parallel platform consisting of a host PC and 2 PC-32 DSP cards offers a flexible and cost effective parallel processing system and has been adopted to provide basic parallel hardware supports for the study of parallel image sequence compression.

While the host PC is powered by an Intel 486DX33 general-purpose processor, the PC-32 DSP cards each feature a TI TMS320C32 dedicated high performance floating



point DSP engine. The interconnection between the host PC and the two PC-32 cards is primarily via dualport memory to cater for high speed inter-processor communications. As dualport memory is a special form of shared memory, an arbitration scheme has been established to administer all dualport memory access. This arbitration scheme is based on the principle of semaphore and its operation efficiency is further enhanced by a set of hardware semaphore registers on the PC-32 cards. Access to the hardware semaphore registers is always on a first-come-first-serve basis. However, in order to assure fair access to these registers and to avoid possible deadlocks, it is crucial that software running on both the host PC and the PC-32 cards must operate in a cooperative manner. A mailbox messaging scheme has also been adopted to facilitate inter-processor communications between the host PC and the PC-32 DSP cards. Its operation principle relies on the hardware semaphore registers to allow arbitrated access to mailbox data in dualport memory. An additional interconnection between the host PC and the two PC-32 cards is via the host PC I/O bus space. Its provision is primarily for PC-32 hardware control and semaphore register interaction.

The basic processing arrangement for the prototype DSP platform is to have the host PC serve as a host processor and the two PC-32 cards as coprocessors. While the main function of the host processor is to distribute processing tasks to the PC-32 cards for parallel processing and to handle processing tasks which are difficult to be processed efficiently on the PC-32 cards, that of the PC-32 cards is to respond and execute processing requests from the host processor. During parallel processing, data is exchanged between the host PC and PC-32 cards in packed format so as to overcome implementation problems caused by the mismatch in their physical memory structures and the limited size of dualport memory space.

During parallel HOD computation, the relative processing performance in terms of processing time has been recorded for various processing arrangements. A worst case situation has been simulated in which parallel HOD computation was deliberately applied to a pair of completely different image frames instead of consecutive ones. Unlike sequential processing, the application of parallel HOD computation involves additional processing overheads because image data needs to be distributed to

parallel processors to allow parallel processing to take place. During parallel processing operations, not only does the host PC have to undertake its share of processing tasks, but it is also responsible for distributing image data to the DSP cards for parallel processing and coordinating their operations. Being dedicated to processing tasks, the DSP cards only need to respond to processing requests from the host PC and return processed results to the host PC once processing is complete. Image data is transferred to the DSP cards via the common dualport memory region in the form of data packets. Each of these packets contains two scan lines of image data extracted from two test image frames for which the HOD parameter is to be calculated. Access to the common dualport memory is arbitrated by mailbox messaging scheme. Also, dedicated mailbox semaphore directives are used to facilitate the arbitration process. Because of the simplicity of HOD computation, the application of parallel HOD computation only offered a marginal improvement to processing speed. Therefore, a decision has been made not to include parallel HOD computation in subsequent parallel image sequence compression experiments.

Although parallel still-image or intraframe compression has been implemented using similar parallel processing arrangement as in the case of parallel HOD computation, its application offers significant improvement over the sequential processing approach. This is because processing of 8x8 and 4x4 image blocks during intraframe compression can be performed independently from one another in the spatial domain, thus enabling more efficient parallel operation. Experimental results have shown that the greatest improvement came from 4x4 image sub-block processing. Due to relatively small codebook size, a copy of the optimised codebook for 4x4 image sub-block processing was stored locally on each DSP card, hence eliminating altogether the need for dynamic codeword transfers and allowing 4x4 image sub-blocks to be processed without any host PC intervention. Accordingly, the host PC was able to handle data distribution tasks more efficiently while leaving more computationally intensive tasks to the DSP cards.

The application of parallel 8x8 image block processing during intraframe compression experienced was ineffective because it caused the processing time to increase. This problem can be attributed to the limitation of the prototype parallel

processing hardware rather than the deficiency of the concept of parallel processing. The limited size of on-board memory on the DSP cards did not allow the codebook for 8x8 image block processing to be stored locally, hence necessitating costly dynamic codeword transfers during parallel 8x8 image block processing. This means that excessive parallel processing overheads would be involved even though processing arrangement for 8x8 image block processing has been modified to take into account dynamic codeword transfers.

For parallel image sequence compression, experiments have been initially conducted for studying parallel motion compensated coding and parallel residual frame coding algorithms. This is because both motion compensated coding and residual frame coding are regarded as the most crucial and time-consuming operations throughout image sequence compression process. To facilitate this study, several GOPs consisting of only 2 consecutive image frames were extracted from the test image sequence with the first image frame being coded as an I-frame using parallel intraframe coding technique and the other as a P-frame using parallel forward motion predictive coding scheme. Apart from parallel intraframe coding which has been discussed earlier, the study of parallel forward motion predictive coding concentrates on parallel motion estimation for 8x8 and 16x16 image blocks and parallel residual frame processing because they represent the key operations during forward motion predictive coding process.

To improve processing efficiency, a slightly different processing arrangement has been adopted during parallel motion estimation process. Unlike the sequential processing approach, the parallel 16x16 motion estimation process was implemented in such a way that all 16x16 image blocks of an image stripe were processed at the same stage. Such an arrangement would allow the prototype parallel processing platform to operate more efficiently because the host PC only needs to distribute 16x16 image blocks to the DSP cards for parallel motion estimation to take place and then up-load processed results once processing on the DSP cards is complete. These processed results, however, need to be stored temporarily for subsequent reference when a decision needs to be made as to whether 8x8 motion estimation is required. For 8x8 parallel motion estimation, a similar processing arrangement was applied.

However, as quadtree segmentation is applied to individual 16x16 blocks, parallel 8x8 motion estimation will only take place in groups of 4 8x8 image sub-blocks. Experimental results have revealed that the application of parallel 8x8 and 16x16 motion estimation offers marginal to moderate reduction in processing time.

On average, parallel residual frame processing contributes about 15% to the reduction of processing time. Its processing arrangement is very similar to that of parallel intraframe processing. However, because all the codebooks for residual frame processing are relatively small in size, this allows a copy of these codebooks to be stored locally on each DSP card. The result is that the efficiency of parallel residual frame processing has been markedly improved as the host PC can undertake image data distribution tasks and processing coordination role more efficiently, while leaving computationally intensive tasks to the DSP cards.

The overall parallel image sequence compression algorithm was eventually tested against various sample GOPs, each consisting of 4 consecutive image frames obtained from various segments of the test image sequence. Due to the different nature of these sample GOPs and the adaptability of the compression algorithm, the overall processing time required to process these GOPs tends to vary quite substantially. On average, an overall speed-up factor of 2.32 has been achieved with the aggregate contribution from parallel intraframe processing, parallel B-frame processing, parallel P-frame processing and parallel residual frame processing.

The moderate results achieved in this study can be attributed to a number of factors. These include the limitation of the prototype parallel processing platform, a compromise in the development of the overall parallel image sequence compression algorithm, and the inefficiency in certain aspects of the processing arrangement itself. For real-time image sequence compression applications where processing power requirements are many folds of what is achieved in this study, it is necessary that not only should more powerful parallel processing hardware be used, but it is also important that the proposed parallel image sequence compression algorithm be further optimised and any inefficient processing arrangement be removed in order to achieve better parallel processing efficiency.

## **Chapter 6: Conclusions and Recommendations**

The summary provided at the end of the previous chapters serves to highlight the major developments and findings in three key areas, namely, still image compression, image sequence compression and the application of parallel processing to image sequence compression. It is the aim of this chapter to conclude this thesis with an overall perspective on the algorithms themselves and the results obtained in this study, and to provide some recommendations applicable to future research work.

### **6.1 BACVQ algorithm**

#### **6.1.1 Algorithm overview**

A detailed literature survey in Chapter 2 has provided a good foundation for the development of the proposed BACVQ algorithm for still image compression. While the study of transform coding techniques in Section 2.2 has helped identify efficient operators for image feature detection, the investigation of the basic principle of VQ coding and its variants in Section 2.4 has facilitated the algorithmic development of BACVQ. As presented in Section 2.5, BACVQ can be regarded as an adaptive block-based VQ coder developed for still image compression applications. BACVQ adopts both block size and codebook adaptivity in its operation to enhance its coding performance. Block size adaptation has enabled BACVQ to encode low activity and uniform regions using larger block size of 8x8 to improve compression gain and higher activity and complex regions using smaller block size of 4x4 to enhance image reconstruction quality. The adaptive segmentation process of 8x8 and 4x4 image blocks is controlled by an efficient quadtree segmentation scheme. Codebook adaptation via CVQ enables high activity and complex regions to be coded with a very high degree of integrity, thereby substantially improving the perceptual quality of reconstructed images.

As far as implementation is concerned, BACVQ has been developed with further enhancements. As discussed in Sections 2.5.1 through 2.5.3, the adoption of image feature classification in both transform and spatial domains has been proven to be useful in improving image feature classification accuracy during quadtree

segmentation and CVQ processes. In particular, the classification operation in the spatial domain has tried to take advantages of the deficiency of HVS by incorporation dynamic range and contrast sensitivity into its operation. In the transform domain, new and effective measures have been incorporated into the classification operation during CVQ process. An effective partial codebook search algorithm has been developed in Section 2.5.4 to accelerate 8x8 codebook search operation significantly with negligible sub-optimality. More importantly, a new algorithm has been developed in Section 2.5.4 for determining the optimal codebook size for CVQ sub-codebooks. Compared to other approaches described in Section 2.4.6, the application of this algorithm has reduced the computation complexity of codebook size determination process substantially.

### **6.1.2 Coder Evaluation**

Results obtained in this research have confirmed that BACVQ has indeed demonstrated its great potential for still image compression applications in terms of high compression gain, low computation complexity and high perceptual reconstructed picture quality. It has been shown that reconstructed image with PSNR ranging from 28.2 to 35.3dB can be achieved at a coding rate from 0.2784 to 0.461bpp. Although the coding performance of BACVQ is slightly lower than other high performance coding schemes quoted by Lee et al (1994a), this can be attributed mainly to the fact that the largest block size used in BACVQ is only 8x8 as opposed to 16x16 in other schemes.

Because both BACVQ and transform coding are block-based algorithms, they tend to suffer from some degrees of “blocking” effects which occur due to the discontinuity at image block boundaries. However, BACVQ is believed to suffer less from this because the adoption of variable block size coding should make the appearance of blockiness less apparent. The blockiness is in general more noticeable in low activity and uniform regions where image blocks tend to be coded with larger block size. Unlike transform coding, BACVQ does not suffer from any “ringing” artefacts.

Some misclassification of low activity and uniform blocks has been observed in regions with fine details, thus causing noticeable degradation to the quality of

reconstructed images. It is, however, expected that this problem can be rectified if the adaptive segmentation algorithm is further refined. Apart from this, consistent coding results have been achieved among image blocks in high activity and complex regions.

### **6.1.3 Recommendations**

Areas of improvement for still image compression algorithms in general and BACVQ in particular can be identified by addressing the constraints and assumptions involved in the development of BACVQ algorithm. Firstly, the lack of a perceptually accurate and computationally efficient quality metric has placed some limitation on the robustness of BACVQ and has caused some difficulties in evaluating reconstructed image quality in an objective manner. Specifically, the application of MSE distortion measure does not always guarantee the coding integrity of the VQ process. Similarly, the usage of PSNR does not necessarily provide an accurate measure for image quality assessment. For these reasons, improvement on a set of quality metric would be highly beneficial for future study in this area.

The second constraint and assumption in the development of BACVQ involves the selection of a set of coding parameters for image feature classification process. This is because BACVQ relies heavily on the classification process in order to operate efficiently. In particular, as the threshold values for these parameters have been chosen in a trial-and-error manner based on a limit set of training images, further optimisation is believed to be possible. The benefit of this is that the robustness of image feature classification during adaptive block segmentation and CVQ processes would be improved, thereby enhancing the coding performance of BACVQ.

With respect to codebook generation, it is believed that the quality of BACVQ codebooks can be improved by using a larger and more diverse set of training images. In addition, since the algorithm for determining BACVQ sub-codebook sizes has been developed based on the overall average square error distortion measure, this suggests that further improvement is possible. As mentioned earlier, this is because square error distortion measure is not necessarily a perceptually accurate indication of how distortion is perceived by HVS.

Because BACVQ is a block-based coding algorithm, further enhancement to eliminate the “blocking” effects is also highly desirable to improve reconstructed image quality. One possible way to achieve this is to use a special form of weighted distortion measures that would give more weight to distortion at image block boundaries so that the discontinuity at the image block boundaries caused by the vector quantisation process can be reduced. In addition, enhancements to allow the BACVQ algorithm to easily adjust its coding rate are quite beneficial. This property is particularly useful when coding rate adjustment is necessary to meet such conditions as transmission channel and storage medium restrictions.

## **6.2 Adaptive spatial/temporal image sequence compression**

### **6.2.1 Algorithm overview**

Literature survey in the area of image sequence compression in Chapter 3 has laid a good foundation for the development of the proposed adaptive spatial/temporal image sequence compression algorithm. The study of HVS in Section 3.2 has provided good justifications for measures taken to improve the coding performance of the proposed image sequence compression algorithm. In general, visual data which are less sensitive to or even could not be perceived by HVS are maximally compressed with greater distortion to maximise compression gain, whereas those with visual importance shall be coded with higher fidelity to enhance the overall perceptual quality of reconstructed pictures. While the study of scene segmentation in Section 3.3 has facilitated the establishment of the proposed adaptive scene segmentation algorithm, the survey on motion compensated predictive coding and adaptive coding techniques in Sections 3.4 and 3.5 has helped to shape the proposed adaptive image sequence compression algorithm.

As presented in Section 3.7, the proposed adaptive image sequence compression algorithm has been developed with an aim to improve the coding performance of a typical image sequence compression process. It relies on BACVQ during intra-frame coding operation and the proposed variable block size motion compensated predictive coding algorithm during inter-frame coding process.



As discussed in Section 3.7.1, a novel adaptive scene segmentation algorithm based on the proposed MHOD criterion has been developed for segmenting lengthy image sequences into coherent GOPs of manageable lengths. Compared to other scene segmentation schemes discussed in Section 3.3, the proposed adaptive scene segmentation algorithm has demonstrated its ability to detect scene transitions more accurately with high tolerance to spurious image activities, and yet it only involves moderate computation complexity similar to that of the HOD scheme.

While the leading I-frame of a GOP is coded using BACVQ coding algorithm, the remaining frames are dealt with using motion compensated predictive coding techniques. An enhanced adaptive variable block size motion compensation algorithm has been proposed for improving the efficiency of motion compensated predictive coding process. As discussed in Section 3.7.2, motion estimation errors has been adopted as an accurate segmentation criterion for controlling the variable block size motion compensation process since it gives a more direct indication of how effective motion estimation and motion compensation operation is going to be. In addition, provisions for motion compensated predictive coding operation to revert to intraframe coding mode have been provided on individual block basis to deal with situations where unsatisfactory motion compensation operation occur.

An improved motion estimation strategy based on the so-called Modified 2-D search algorithm has been presented in Section 3.7.3.1 to increase motion estimation accuracy for small movements specifically. Its development has been inspired by the fact that motion estimation errors has been observed to be more noticeable in low motion activity regions. Compared to other motion estimation algorithms described in Section 3.4.3, the proposed strategy inherits the good performance of the Modified 2-D algorithm and yet it has the ability to estimate small motion more accurately with slight increase in computation complexity.

An alternative progressive motion estimation technique has also been developed for dealing with long inter-reference frame intervals which may occur as a result the proposed adaptive scene segmentation algorithm. As described in Section 3.7.3.2, the proposed progressive motion estimation algorithm offers a fixed motion search range

regardless of the length of inter-reference frame intervals, better motion vector field coding efficiency, and more importantly the ability to deal with overlapping movement more effectively among moving objects in an image sequence.

### **6.2.2 Coder Evaluation**

The proposed adaptive spatial/temporal image sequence compression algorithm has indeed achieved quite encouraging results. When tested against a group of sample image sequence segments with varying image complexity and motion activity, the proposed algorithm was able to reproduce pictures with acceptable perceptual quality varying from 20 to 40dB at a bit rate between 0.15 and 0.55bpp. These encouraging results have been obtained mainly by adapting its coding process to the nature of original images, thereby enabling spatial/temporal redundancy within image sequences to be exploited more effectively. In fact, such adaptivity is collectively achieved with the aid of adaptive scene segmentation, variable block size motion compensated predictive coding and BACVQ.

The proposed adaptive scene segmentation algorithm has demonstrated very high consistency throughout the temporal segmentation process. As shown in Section 3.7.1, the proposed algorithm has been able to accurately identify all key scene transition points within the test image sequence. In fact, it has been able to operate with a reasonably large margin to protect itself from being affected by spurious image sequence activities.

Good results have also been obtained during motion compensated predictive coding process thanks to the application of variable block size motion compensated prediction, enhanced motion search strategy, and the provision for motion compensated predictive coding process to revert back to intraframe coding. The application of variable block size motion estimation shall offer good motion compensation performance closer to more complicated schemes such as deformable block based motion compensation scheme, and yet require significantly less computation complexity. In addition, the adoption of an alternative approach for progressive motion estimation has facilitated significantly the operation of motion estimation, especially when GOPs with long frame intervals are involved. Any

excessive motion estimation error that occurs during the progressive motion estimation process is self-corrected by the provision for intraframe coding mode invocation. It is believed such a provision is essential to promote the adaptivity of the proposed image sequence compression algorithm although its usage would increase the complexity during the coding process slightly.

In general the performance of intraframe coding process has been observed to be inferior to that of motion compensated predictive coding process when low detailed image sequences with low motion activities are involved. This is because intraframe coding by its virtual nature could not take advantages of the significant level of temporal redundancy. Conversely, motion compensated predictive coding tends to offer less efficient coding performance than intraframe coding when high detailed image sequences with high motion activities are involved because of the normal drop in motion compensated prediction efficiency.

### **6.2.3 Recommendations**

Because sub-optimal block matching has been applied in the proposed motion compensated predictive coding algorithm, some coding inconsistency is generally inevitable. This problem normally occurs when an image sequence experiences complex motion activities and the assumption of uniform motion field is therefore violated. Although variable block size motion compensated prediction has been applied quite successfully in this study to deal with this problem, it is believed that this technique can be applied with even greater success if a larger range of block sizes can be efficiently used during the coding process. Not only does this help to improve the consistency of motion compensation process by allowing smaller block sizes to be used, but it also has a great potential to offer higher compression gain because spatial/temporal redundancy can be better exploited using larger block sizes.

In addition, the consistency of the motion estimation process may occasionally be compromised because of the ineffectiveness of most sub-optimal motion estimation search strategies in handling complex motion fields. The tendency of sub-optimal block-based motion estimation algorithms to converge to local distortion minima is a major factor causing not only errors during the motion estimation process but also

affects the efficiency of motion compensated predictive coding. As such, ideas to incorporate inter-block correlation into the motion estimation process are definitely worth pursuing. In addition, the assumption of  $\pm 15$  pixels/frame made in this study can be adaptively increased to better handle image sequences with very high motion activities. However, this does not necessarily mean that motion estimation resolution has to be maintained uniformly with  $\pm 1$  pixel accuracy throughout the motion search domain to take advantages of the deficiency of HVS.

A lossless coding block could also be added to the final stage of the image sequence compression process to further increase its compression efficiency. This coding stage is usually applied to remove any statistic redundancy remaining in the coded data stream without causing any compatibility issues with preceding coding structures due to its lossless nature. Its omission in this study is to ease implementation issues but should not affect the generality of this study because of its independent nature.

## **6.3 Parallel image sequence compression**

### **6.3.1 Algorithm overview**

The prototype parallel processing platform has been constructed as a small scale MIMD computer which consists of a 486DX33 IBM/PC computer and 2 high performance PC-32 DSP cards connected in a star configuration. The interface between the PC and the PC-32 cards is primarily provided via relocatable dualport memory banks, hence not only offering high speed data transfer capability but also allowing multiple dualport memory bank supports for multiple DSP card operation. Dedicated semaphore hardware is provided on-board the PC-32 cards to facilitate access arbitration and process synchronisation during parallel processing operation.

Basically the proposed parallel image sequence compression algorithm has been developed on the prototype parallel processing platform with the host PC serving as a master processor and the DSP cards as independent parallel processors. Apart from having to handle those image processing tasks which are difficult to execute efficiently on the DSP cards, the host PC is also responsible for distributing image data to the parallel processors via the dualport memory for parallel processing.

During parallel processing operation, a general processing arrangement between the host PC and the DSP cards has been adopted in which the host PC transfers image data to an idle DSP card via the dualport memory region, followed by sending a unique message or command to a corresponding semaphore mailbox to initiate processing on the idle DSP card. This process is repeated until all DSP cards are busy or all image processing tasks have been distributed. When processing is complete, the host PC shall be notified by the parallel processors via its semaphore mailbox to allow it to retrieve processing results and possibly re-initiate another processing cycle.

The proposed parallel image sequence compression algorithm has been developed after a detailed assessment of parallel HOD computation, parallel still image coding and parallel image sequence coding. As discussed in Section 5.7.1, the principle behind the parallel HOD computation algorithm is to allow the host PC to dynamically distribute computation tasks to parallel processors including itself for parallel processing. The partition of image data is performed by the host PC with image scanning lines serving as partition units. The distribution of computation tasks to the DSP cards is generally accomplished by sending scanning lines of image data to the dualport memory region followed by a proper processing request to a corresponding semaphore mailbox. For processing efficiency, the PC32 DSP cards are not required to return processing results immediately. Instead, explicit requests shall be issued to the DSP cards at the end of the parallel HOD computation process to obtain the processing results from them. In spite of this, the parallel HOD computation algorithm still involves too much overhead and therefore has been excluded from the overall parallel image sequence compression algorithm.

As discussed in Section 5.7.2, parallel still image compression has been accomplished mainly by applying parallel 4x4 image block processing. The application of parallel 8x8 image block processing, however, resulted in inferior performance because of hardware limitations and hence was not included in the overall parallel still image compression algorithm. The proposed parallel 4x4 image block processing algorithm was developed with the host PC taking care of the scheduling of 4x4 image block processing tasks to the two DSP cards for parallel

processing and the two DSP cards serving as parallel processors. Because the codebooks needed for 4x4 image block processing are stored locally on both DSP cards, a highly efficient parallel processing operation has been achieved with a general data/command processing arrangement, in which the host PC shall dynamically send 4x4 image data blocks to idle DSP cards via the common dualport memory region followed by a proper processing request to the semaphore mailbox. When processing on a DSP card is complete, the host PC shall be notified via its semaphore mailbox to allow it to upload processing results and possibly to re-initiate another processing cycle for the next 4x4 image block.

The application of parallel processing to image sequence compression has been achieved in the form of parallel intraframe and interframe processing. While parallel intraframe processing can be implemented using the same algorithm as that of parallel still image compression, parallel interframe processing is primarily achieved with the application of parallel motion estimation and parallel residual frame processing. The development of parallel 16x16 motion estimation algorithm in particular required a different approach to be taken to allow multiple 16x16 image blocks within an image stripe to be processed concurrently. Like parallel still image compression, parallel 16x16 motion estimation was implemented based on the general data/command processing arrangement for reliable parallel operation. The proposed parallel 8x8 motion estimation algorithm was also developed in the same way as the parallel 16x16 motion estimation algorithm. However, its parallel operation is restricted to every four 8x8 image sub-blocks at a time following the quadtree segmentation of each 16x16 image block.

Unlike parallel intraframe processing, parallel residual frame processing could be implemented with significant simplification as discussed in Section 5.7.3. During parallel residual frame processing operation, the host PC dynamically distributes 8x8 residual block processing tasks to the DSP cards for parallel processing without having to involve itself directly in the coding process. This simplified processing arrangement is possible because a copy of the codebooks needed for residual block processing is stored locally on the DSP cards and they are capable of processing 8x8 residual blocks without the host PC intervention. Once processing of 8x8 residual

blocks is complete, the host PC will be notified by the DSP cards to upload processing results from them before the whole process is initiated all over again for a new 8x8 residual block.

### **6.3.2 Coder evaluation**

As discussed in Section 5.5, the use of dualport memory for data exchange between the host PC and the DSP cards has reduced the cost of data transfer overheads considerably, thereby enhancing the processing efficiency of the prototype parallel processing platform. This form of interfacing is inherently superior than a common serial interface because its implementation can take advantage of high speed 16 bit data transfers. In addition, its performance is further enhanced by the on-board semaphore hardware on both DSP cards. Its provision indeed offers a very efficient and reliable means for controlling data transfers across the dualport memory. However, the limited size of dualport memory has adversely affected the overall parallel processing efficiency to some degree.

Although care has been taken in the development of the proposed parallel HOD computation algorithm, experimental results have indicated that its application only offers a marginal processing improvement over its sequential counterpart because of the relatively high data transfer overheads during its operation. Therefore, given the computational simplicity nature of HOD computation, such overheads would simply offset almost all the processing benefits of parallel HOD computation.

By contrast the application of the proposed parallel 4x4 image block processing algorithm has offered a significant improvement to the processing speed of intraframe or still image compression process. In a typical example, the intraframe processing time has been reduced from 49 down to 18 seconds, that is, a speed-up ratio of more than 2.7. The general parallel processing arrangement in which the host PC is assigned the task of scheduling 4x4 image processing tasks to the DSP cards for parallel processing has proven to be very effective as it allows the host PC to take advantage of the high processing speed of the DSP cards more effectively. More importantly, the processing efficiency of the parallel 4x4 image processing algorithm during intraframe compression has been ultimately accomplished by allowing a copy

of the codebooks for 4x4 image block processing to be stored locally on both the DSP cards.

Experimental results have shown a consistent reduction in processing time when parallel processing was applied progressively to various stages of interframe coding process although the relative processing gain at each stage may vary from one GOP to another depending on the nature of image sequences being processed. These stages include motion estimation for 8x8 and 16x16 image blocks and residual frame processing. The proposed parallel processing arrangement during 16x16 motion estimation in particular has demonstrated how parallel processing can be sustained on the prototype parallel processing platform in order to enhance its processing efficiency. The application of parallel processing to residual frame processing has proved to be very efficient. On average, it has resulted in a further 15% reduction in the overall processing time of the parallel image sequence compression process. In spite of being based on the same BACVQ coding algorithm, parallel residual frame processing has proved to be much more efficient than parallel intraframe processing because of smaller codebook requirements. Apart from computational advantages, this also allows parallel residual frame processing to be implemented with greatest efficiency as the host PC does not have to be involved in the coding process directly.

As an ultimate assessment for the overall processing performance, the proposed parallel image sequence compression algorithm was tested against a series of representative GOPs consisting of 4 consecutive image frames extracted from the test image sequence. In this assessment, various speed-up factors have been recorded ranging from 1.98 to 2.77 with an average speed-up factor of 2.32. In terms of average processing time, a reduction from around 3'35" to 1'32" has been achieved as a result of the application of the proposed parallel image sequence compression algorithm. The fluctuation is generally regarded as normal and is mainly caused by the different nature among various GOPs and the adaptivity of the proposed image sequence compression process, which in turn determines how much parallel processing gain would be achieved at various stages of image sequence compression process.



Lastly, although the combination of parallel processing algorithm and parallel processing hardware in this study only offers a moderate increase in the overall processing performance, it does not necessarily mean that this is the ultimate improvement that parallel processing could offer. In fact, the moderate improvement in processing performance in this study can be attributed to a number of factors such as the limitations of parallel processing hardware, the compromises involved in the development of the proposed parallel image sequence compression algorithm and some inefficiencies in the existing processing arrangement itself.

### **6.3.3 Recommendations**

Although the level of processing improvement achieved in this study is not sufficient for real-time digital video compression, it is believed that such deficiencies could be overcome by utilising more powerful parallel processing hardware, improving programming development tools to facilitate parallel processing optimisation, and globally optimising the proposed image sequence compression algorithm. Major improvements to parallel processing hardware could be achieved by adopting a more advanced computer system such as a workstation for the master processor. The general IBM/PC architecture has proved to be too restricted for efficient parallel processing operation in this study. The limited size of conventional memory space and the overheads in accessing data in the Extended Memory space both have adversely affected the processing efficiency on the prototype parallel processing platform. In addition, the lack of sufficient on-board and dualport memory on parallel DSP cards should be seriously addressed to avoid unnecessary compromises during parallel processing operation. Even the use of more powerful DSP processing cards such as those based on the TMS320C8x, which is by itself a massive parallel processing system on a single chip, is also desirable for image compression applications. Programming tool improvement should include the use of more efficient compilers and more powerful tools for parallel process performance analysis.

Improvements should also be adopted for the proposed parallel image sequence compression algorithm. These may include eliminating any trade-offs adopted due to parallel processing hardware limitations, avoiding acquiring image sequence data from mass data storage due to its slow access speed and reducing redundant data

exchange during parallel processing operations. In addition, the issue of increasing memory usage efficiency shall be considered in order to take full advantage of parallel processing and reduce system cost.

Finally, being a powerful parallel computation technique, pipeline processing can be adopted in order to exploit data/functional parallelism in most image data processing tasks including image sequence compression. Also, double buffering may offer a more efficient processing arrangement to reduce processor idling time during parallel processing operation.

## References

- [1] Ahlgren, D. R. and et al. (1988). Compression of Digitized Images for Transmission and Storage Applications. Proceedings SPIE, Image Processing, Analysis, Measurement and Quality, vol. 901.
- [2] Almasi, G.S. and Gottlieb, A. (1994). Highly Parallel Computing, 2nd edition. The Benjamin/Cummings Publishing Company Inc.
- [3] Aoki, T. (Jan. 1994). Consumer Electronics: HDTV alignment, Games people play (ed Perry, T). IEEE Spectrum, 30-34.
- [4] AT&T. (Jan./Feb. 1993). Image & Video Coding Standards. AT&T Technical Journal.
- [5] Barnwell, T. P. and et al. (1993). The Georgia Tech Digital Signal Multiprocessor. IEEE Transactions on Signal Processing, 41, no. 7, 2471-2487.
- [6] Bic, L. and Shaw, A.C., (1988). The Logical Design of Operating Systems, 2nd edition. Prentice Hall.
- [7] Blinn, J. F. (July 1993). What's the Deal with the DCT? IEEE Computer Graphics and Applications, 78-83.
- [8] Buhler, Y. and Fortier, M. (1987). Hierarchical Picture Coding Using Quadtree Decomposition. Proceedings SPIE, Advances in Image Processing, 804, 336-343.
- [9] Bursky, Dave. (March 21, 1994). Parallelism Pushes DSP Throughput. Electronic Design, 151-154.
- [10] Chan, M. H. and et al. (1990). Variable Size Block Matching Motion Compensation with Applications to Video Coding. IEE Proceedings, Part I, 137, no. 4, 205-212.
- [11] Chee, Y. K. and et al. (Jan. 1994). An Introduction to Vector Quantisation. Proceedings of the 2nd International Interactive Multimedia Symposium. Perth, Western Australia.
- [12] Chen, C. F. and Pang, K. K. (1992). Hybrid Coders with Motion Compensation. Multi-dimensional Processing of Video Signals, pp. 133-158. Kluwer Academic Publishers.
- [13] Chen, W. and Pratt, W. (1984). Scene Adaptive Coder. IEEE Transactions on Communications, COM-32, 225-232.
- [14] Clarke, R.J. (1985). Transform coding of images, 1st edition. Academic Press.

- [15] Coppisitti, N. and et al. (1993). Low-Complexity Subband Encoding for HDTV images. IEEE Journal on Selected Areas in Communications, 11, no. 1, 77-87.
- [16] Dijkstra, E.W. (1968). Co-operating Sequential Processes (ed Genuys, F). Programming Languages. Academic Press.
- [17] Dubois, E. (1992). Motion-Compensated Filtering of Time-Varying Images. Multi-dimensional Processing of Video Signals, pp. 103-131. Kluwer Academic Publishers.
- [18] Dufour, C. and Nocture, G. (1992). A HDTV Compatible Coding scheme for distribution purposes (ed Yasuda, H. and Chiariglione, L). Signal Processing of HDTV, III. Elsevier Science Publishers.
- [19] Ely, M. (April 1996). Tech and Production: Getting Ready for DVD: Premastering Issues for Digital Video Disk. Advanced Imaging, 26-29.
- [20] Fedele, N.J. and et al. (1988). Real-time Multidirectional Data Compression of Full Motion Video. Proceedings SPIE, Image Processing, Analysis, Measurement and Quality, 901, 82-90.
- [21] Fountain, T.J. (1994). Parallel Computing - Principles and Practice. Cambridge University Press.
- [22] Furukawa, I. and et al. (1992). Hierarchical Coding of Super High Definition Images with Adaptive Block-size Multi-stage VQ (ed Yasuda, H. and Chiariglione, L). Signal Processing of HDTV, III, pp. 361-368. Elsevier Science Publishers.
- [23] Gersho, A. and Gray, R.M. (1992). Vector quantisation and signal compression, 1st edition. Kluwer Academic Publishers.
- [24] Ghanbari, M. (1990). The Cross-Search Algorithm for Motion Estimation. IEEE Transactions on Communications, 38, no. 7, 950-953.
- [25] Gray, R. M. and Davission, L. D. (1976). A Mathematical Theory of Data Compression, pp. 21-25. Data Compression. Hutchinson and Ross Inc..
- [26] Gray, R. M. and et al. (1992). Image Compression and Tree-structured Vector Quantization-Image and Text Compression (ed Storer, J. A). Kluwer Academic Publisher.
- [27] Gray, R. M. (April 1984). Vector Quantization. IEEE ASSP Magazine, 4-27.
- [28] Grolier Multimedia Encyclopedia CDROM. (1995). Grolier Electronic Publishing Inc.

- [29] Guttag, K. M. (June 1994). Multimedia Powerhouse. Byte, 57-64
- [30] Hoang, D. T. and et al. (1994). Explicit Bit Minimization for Motion Compensated Video Coding. Proceedings - Data Compression Conference '94, 175-184. IEEE Computer Society Press.
- [31] Hobbs, L.C. and Theis, D.J. (1970). Survey of Parallel Processor Approaches and Techniques. Parallel Processor Systems, Technologies, and Applications, 3-20. Spartan Books.
- [32] Hsieh, C. H. and et al. (1990). Motion Estimation Algorithm Using Interblock Correlation. Electronic Letters, 26, no. 5, 276-277.
- [33] Huang, C. L. and Hsu, C. Y. (1994). A New Motion Compensation Method for Image Sequence Coding Using Hierarchical Grid Interpolation. IEEE Transactions on Circuits and Systems for Video Technology, 4, no. 1, 42-52.
- [34] Hussain, Z. (1991). Digital Image Processing - Practical Applications of Parallel Processing Techniques. Ellis Horwood Ltd..
- [35] Jain, A. K. (1981). Image Data Compression: A Review. Proceedings of the IEEE, 69, no. 3, 349-389.
- [36] Jain, J. R. and Jain, A. K. (1981). Displacement Measurement and Its Application in Interframe Image Coding. IEEE Transactions on Communications, COMM-29, no. 12, 1799-1808.
- [37] Jeschke, H. and et al. (1992). Multiprocessor Performance for Real-time Processing of Video Coding Applications. IEEE Transactions on Circuits and Systems for Video Technology, 2, no. 2, 221-206.
- [38] Kappagantula, S. and Rao, K. R. (1985). Motion Compensated Interframe Image Prediction. IEEE Transactions on Communications, COMM-33, no. 9, 1011-1015.
- [39] Kim, J. T. and et al. (1993). Subband Coding Using Human Visual Characteristics for Image Signal. IEEE Journal on Selected Areas in Communications, 11, no. 1, 59-64.
- [40] Koga, T. and et al. (1981). Motion Compensated Interframe Coding for Video Conferencing. Proc. Nat. Telecommun. Conf., G5.3.1-5.3.5. New Orleans, LA.
- [41] Kolagotla, R. K. (1993). VLSI Implementation of a Tree-Searched Vector Quantizer. IEEE Transactions on Signal Processing, 41, no. 2, 901-905.
- [42] Kubrick, A. and Ellis, T. (1990). Classified Vector Quantization of Images: Codebook Design Algorithm. IEE Proceedings, Part I, 137, no. 6, 379-386.

- [43] Kumar, V. and et al. (1994). Introduction to Parallel Computing - Design and Analysis of Algorithms. The Benjamin/Cummings Publishing Company, Inc.
- [44] Lee, J. and Dickinson, B. W. (1994). Temporally Adaptive Motion Interpolation Exploiting Temporal Masking in Visual Perception. IEEE Transactions on Image Processing, 3, no. 5, 513-525.
- [45] Lee, M. H. and Crebbin, G. (1994a). Classified Vector Quantisation with Variable Block-Size DCT models. IEE Proceedings, Visual Image Signal Processing, 141, no. 1, 39-48.
- [46] Lee, M. H. & Crebbin, G. (1994b). Image Sequence Coding Using Quadtree-Based Block-Matching Motion Compensation and Classified Vector Quantisation. IEE Proceedings: Vision, Image and Signal Processing, 141, i6, 453-460.
- [47] Li, H. and et al. (1994). Image sequence Coding at Very Low Bit Rates: A Review. IEEE Transactions on Image Processing, 3, no. 5, 589-609.
- [48] Linde, Y., Buzo, A. and Gray, R. M. (1980). An Algorithm for Vector Quantizer Design. IEEE Transactions on Communications, COMM-28, no. 1, 84-95.
- [49] Lo, K. T. and Cham, W. K. (1992). An Efficient Encoding Algorithm for Vector Quantization of Images. Proceedings of EUSIPCO-92, Signal Processing VI: Theories and Applications, vol. III, 1231-1234.
- [50] Malvar, H. S. (1992). Signal Processing with Lapped Transforms. Artech House Inc.
- [51] Moldovan, D.I. (1993). Parallel Processing - From Applications to Systems. Morgan Kaufmann Publishers.
- [52] Musmann, H. G. and et al. (1985). Advances in Picture Coding. Proceedings of the IEEE, 73, no. 4, 523-548.
- [53] Naveen, T. and Woods, J. W. (1994). Motion Compensated Multiresolution Transmission of High Definition Video. IEEE Transactions on Circuits and Systems for Video Technology, 4, no. 1, 29-41.
- [54] Ngan, K. N. and Koh, H. C. (1992). Predictive Classified Vector Quantization. IEEE Transactions on Image Processing, 1, no. 3, 269-280.
- [55] Parker, J. R. and Ingoldsby, T. R. (1990). Design and Analysis of a Multiprocessor for Image Processing. Journal of Parallel and Distributed Computing, 9, 297-303.

- [56] Petajan, E. (Oct. 1992). Digital Video Coding Techniques for US High-Definition TV. IEEE Micro, 13-21.
- [57] Phamdo, N. and et al. (1993). A Unified Approach to Tree-Structured and Multistage Vector Quantisation for Noisy Channels. IEEE Transactions on Information Theory, 39, no. 3, 835-850.
- [58] Plansky, H. (1992). Variable Block-Size Vector Quantization in the Transform Domain. Proceedings of EUSIPCO-92, Signal Processing VI: Theories and Applications, III, 1243-1246. Elsevier Science Publishers.
- [59] Privat, G. and Petajan, E. (Oct. 1992). Processing Hardware for Real-Time Video Coding. IEEE Micro, 9-12.
- [60] Puri, A. and Haskell, B. G. (1992). Digital HDTV coding with motion compensated interpolation. Signal Processing of HDTV, III, 531-537. Elsevier Science Publishers.
- [61] Ramamurthi, B. and Gersho, A. (1986). Classified Vector Quantisation of Images. IEEE Transactions on Communications, COMM-34, no. 11, 1105-1115.
- [62] Rao, K. R. and Yip, P. (1990). Discrete Cosine Transform - Algorithm, Advantages, Applications. Academic Press Inc.
- [63] Rocca, F. and Zanoletti, S. (Oct. 1972). Bandwidth Reduction Via Movement Compensation on a Model of the Random Video Process. IEEE Transactions on Communications, 960-965.
- [64] Sarkar, V. (1989). Partitioning and Scheduling Parallel Programs for Multiprocessors. The MIT Press. PITMAN publishing.
- [65] Seferidis, V. and Ghanbari, M. (1994). Generalised Block-Matching Motion Estimation Using Quadtree Structured Spatial Decomposition. IEE Proceedings - Visual Image Processing, 141, no. 6, 446-452.
- [66] Shen, K. and et al. (1994). An Overview of Parallel Processing Approaches to Image and Video Compression. SPIE Image and Video Compression, 2186, 197-208.
- [67] Shusterman, E. and Feder, M. (1994). Image Compression via Improved Quadtree Decomposition Algorithms. IEEE Transactions on Image Processing, 3, no. 2, 207-215.
- [68] Simar, R. and et al. (August, 1992). Floating-Point Processors Join Forces in Parallel Processing Architectures. IEEE Micro, 60-69.

- [69] Srinivasan, R. and Rao, K. R. (1985). Predictive Coding Based on Efficient Motion Estimation. IEEE Transactions on Communications, COMM-33, no. 8, 888-896.
- [70] Thyagarajan, K.S. and et al. (1987). Encoding of Subband Images Using Adaptive Vector Differential Pulse Code Modulation (DPCM). Proceedings SPIE, Application of Digital Image Processing X, 829, 95-102.
- [71] Uffenbeck, J. (1987). The 8086/8088 family: Design, Programming, and Interfacing. Prentice-Hall International, Inc.
- [72] Vaisey, J. and Gersho, A. (1992). Image Compression with Variable Block Size Segmentation. IEEE Transactions on Signal Processing, 40, no. 8, 2040-2058.
- [73] Vetterli, M. and Metin, K. (1992). Multiresolution Coding Techniques for Digital Television: A review. Multidimensional Processing of Video Signals, 53-79. Kluwer Academic Publishers.
- [74] Webster's Concise Encyclopedia CDROM. (1994). Helicon Publishing Ltd.
- [75] Wen, K. A. and Lu, C. Y. (1993). Hybrid Vector Quantization. Optical Engineering, 32, no. 7, 1496-1502.
- [76] Wright, M. (June, 1996). Special Report: Delivering Digital Video. EDN Asia, 24-42.
- [77] Wu, S. W. and Gersho, A. (1994). Joint Estimation of Forward and Backward Motion Vectors for Interpolative Prediction of Video. IEEE Transactions on Image Processing, 3, no. 5, 684-688.
- [78] Zafar, S. and et al. (1991). Predictive Block-Matching Motion Estimation for TV Coding -- Part I: Inter-Block Prediction. IEEE Transactions on Broadcasting, 37, no. 3, 97-101.
- [79] Zhang, Y. Q. and et al. (1993). A New Approach to Reduce the Blocking Effect of Transform Coding. IEEE Transactions of Communications, 41, no. 2, 299-302.
- [80] Zhang, Y. Q. and Zafar, S. (1991). Predictive Block-Matching Motion Estimation for TV Coding -- Part I: Inter-Block Prediction. IEEE Transactions on Broadcasting, 37, no. 3, 102-105.
- [81] Zou, W. Y. (Feb. 1993). Digital HDTV Compression Techniques for Terrestrial Broadcasting. SMPTE Journal, 127-132.



## **Appendix A: Conference Paper #1**

Block adaptive classified vector quantization by Peter H. S. Truong and Stephen C.Y. Ho

Conference name: IS&T/SPIE Symposium on Electronic Imaging Science and Technology Digital Video Compression: Algorithms and Technologies 1995, San Jose, Calif., Feb. 5-10, 1995

Proceedings details: SPIE Proceedings, v.2419, pp 340-351

**Note: For copyright reasons, Appendix A (pp226-237 of this thesis) has not been reproduced.**

**(Co-ordinator, ADT Project (Retrospective), Curtin University of Technology, 4.12.02)**

## **Appendix B: Conference Paper #2**

Adaptive spatial/temporal coding for image sequence compression by S.C.Y.Ho and P.H.S.Truong.

Conference name: 1995 Conference on Digital Image Computing: Techniques and Applications. Australian Pattern Recognition Society. Brisbane, Dec.6-8, 1995.  
Proceedings details: DICTA-95 Conference Proceedings, pp384-389.

**Note: For copyright reasons, Appendix B (pp238-243 of this thesis) has not been reproduced.**

**(Co-ordinator, ADT Project (Retrospective), Curtin University of Technology, 4.12.02)**

### **Appendix C: Conference Paper #3**

Parallel digital signal processing for image sequence compression by P.H.S.Truong and S.C.Y.Ho.

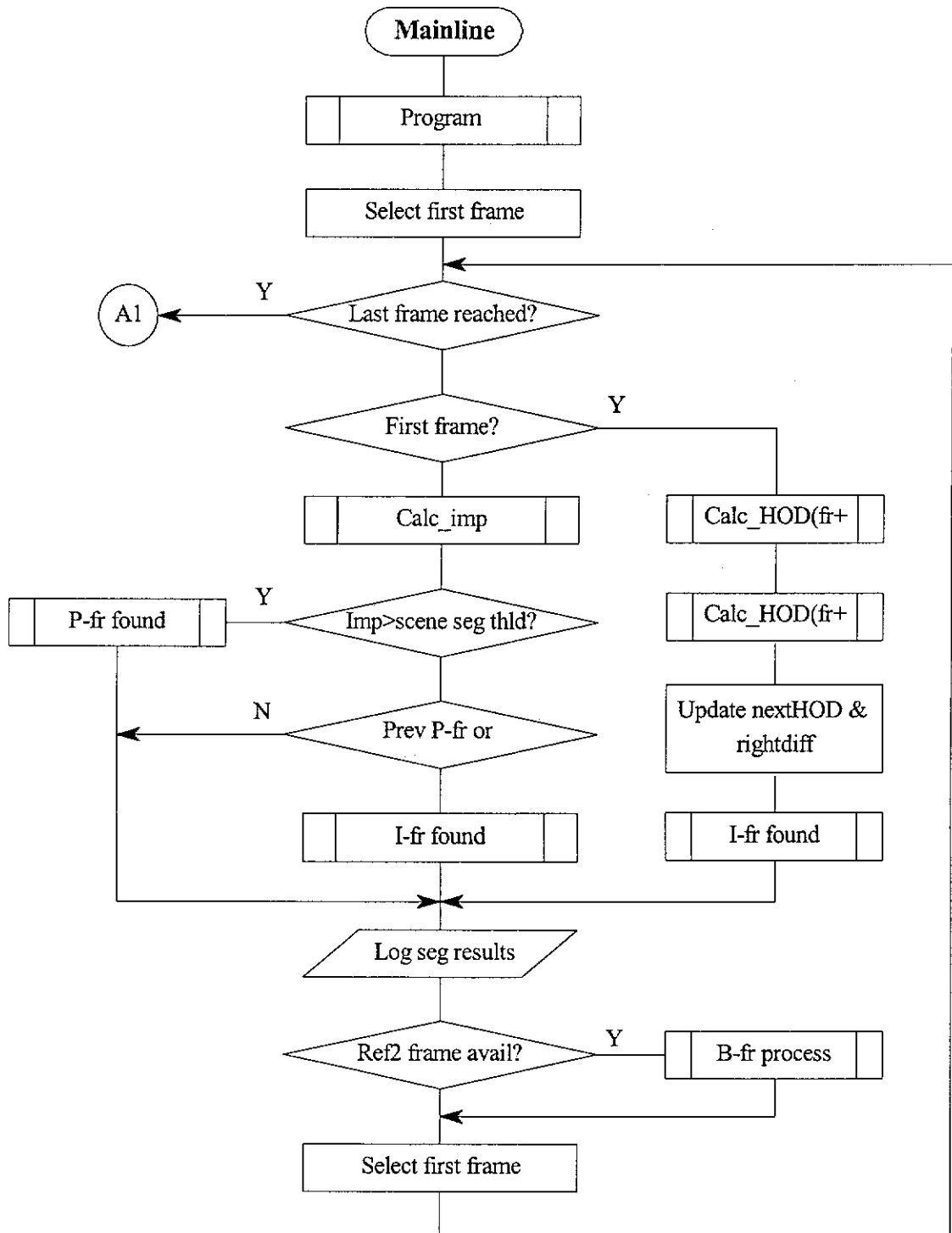
Conference name: 1996 IEEE Region Ten International Conference. Digital Signal Processing Applications. Perth, Nov. 27-29, 1996.

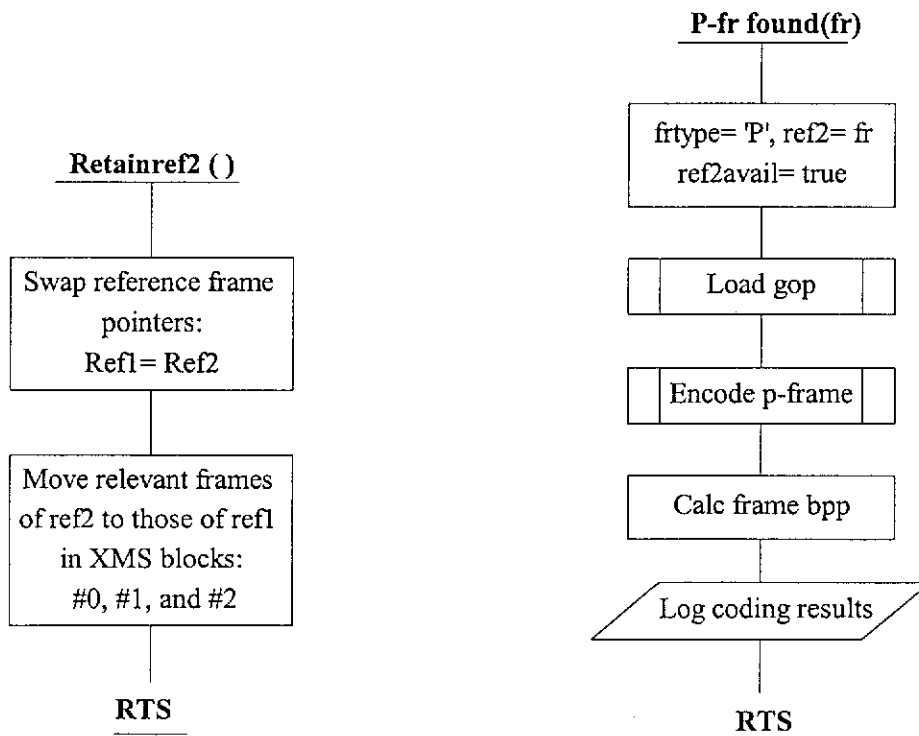
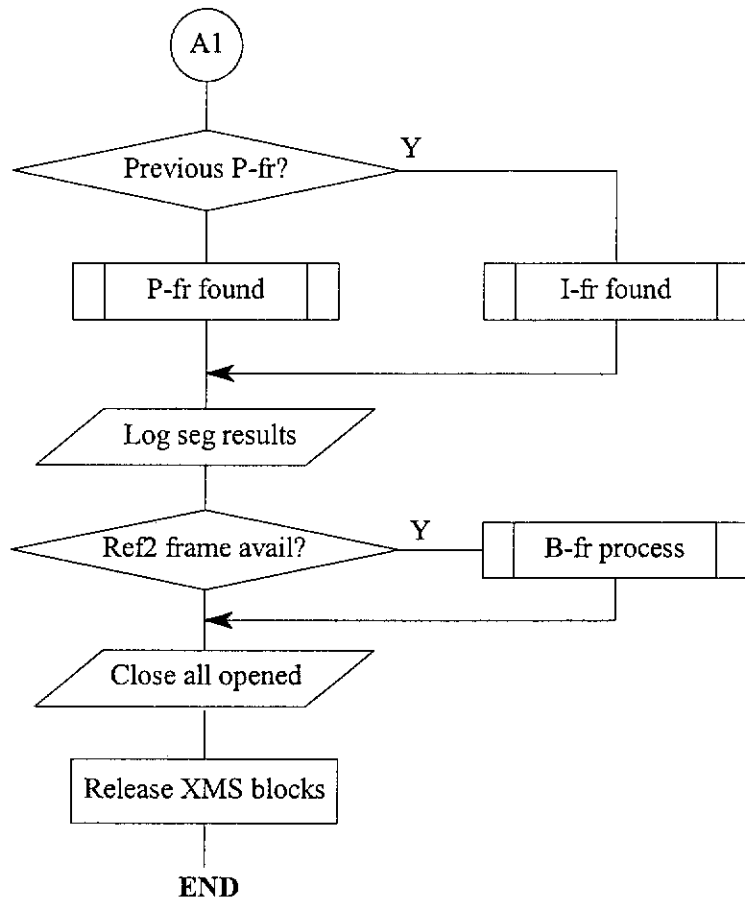
Proceedings details:1996 IEEE Region 10 Conference Proceedings, v. 2, pp716-721.

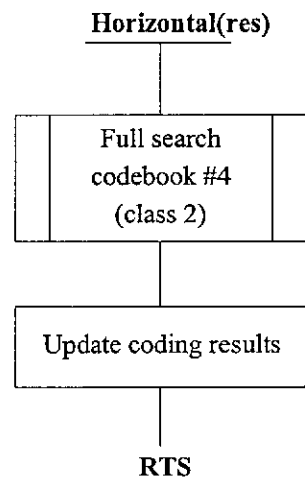
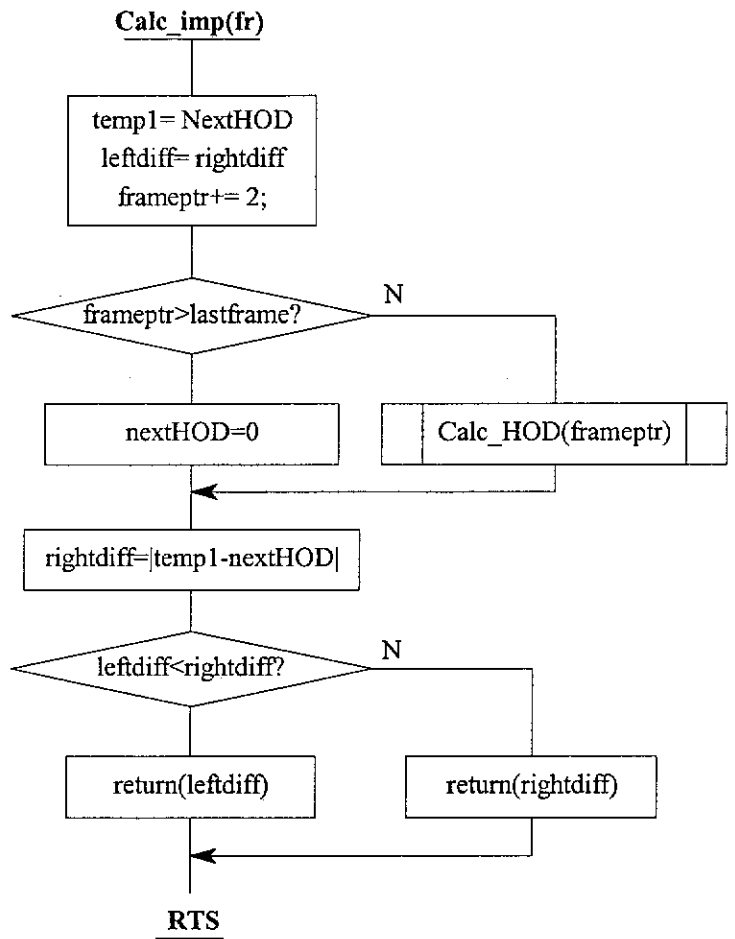
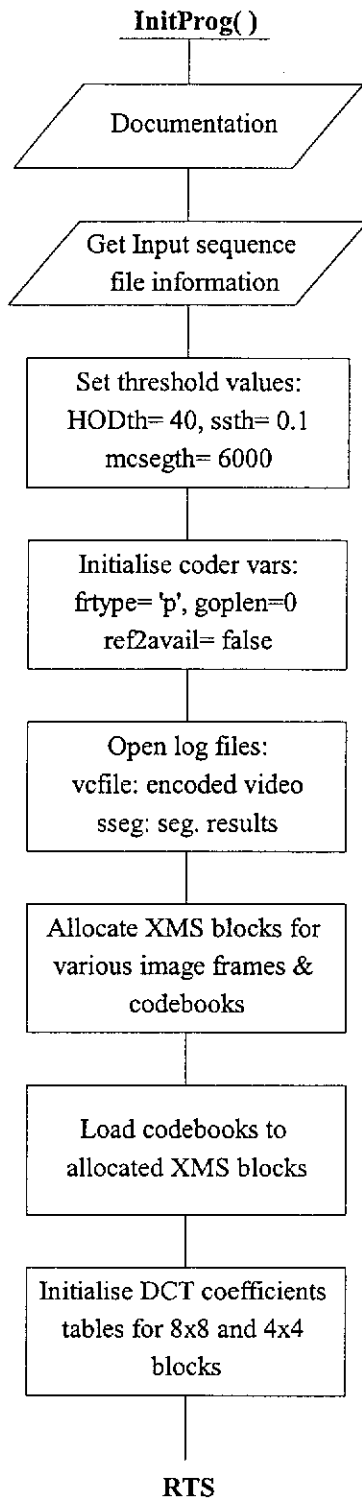
**Note: For copyright reasons, Appendix C (pp244-249 of this thesis) has not been reproduced.**

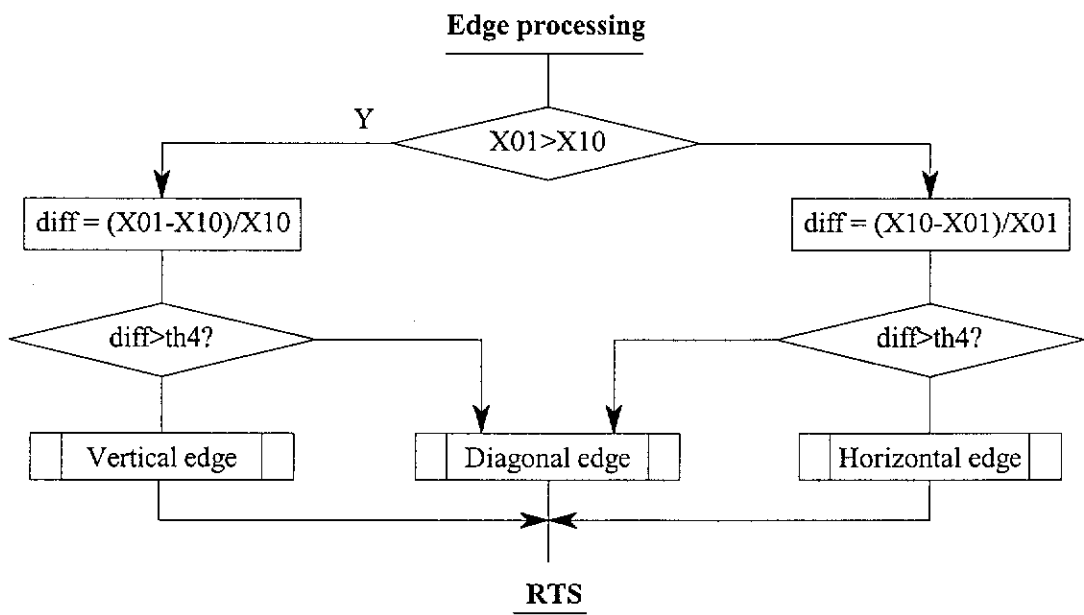
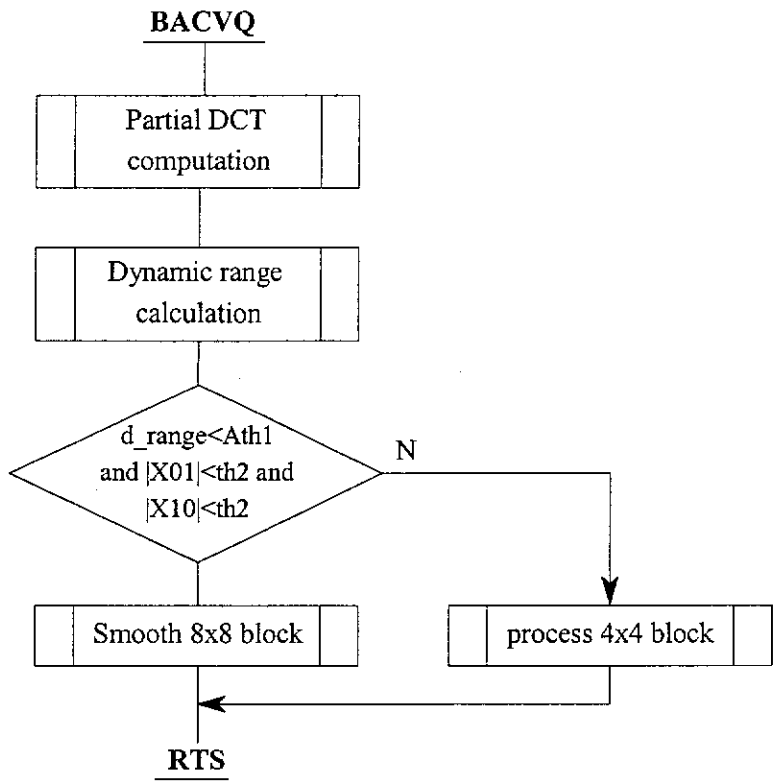
**(Co-ordinator, ADT Project (Retrospective), Curtin University of Technology, 4.12.02)**

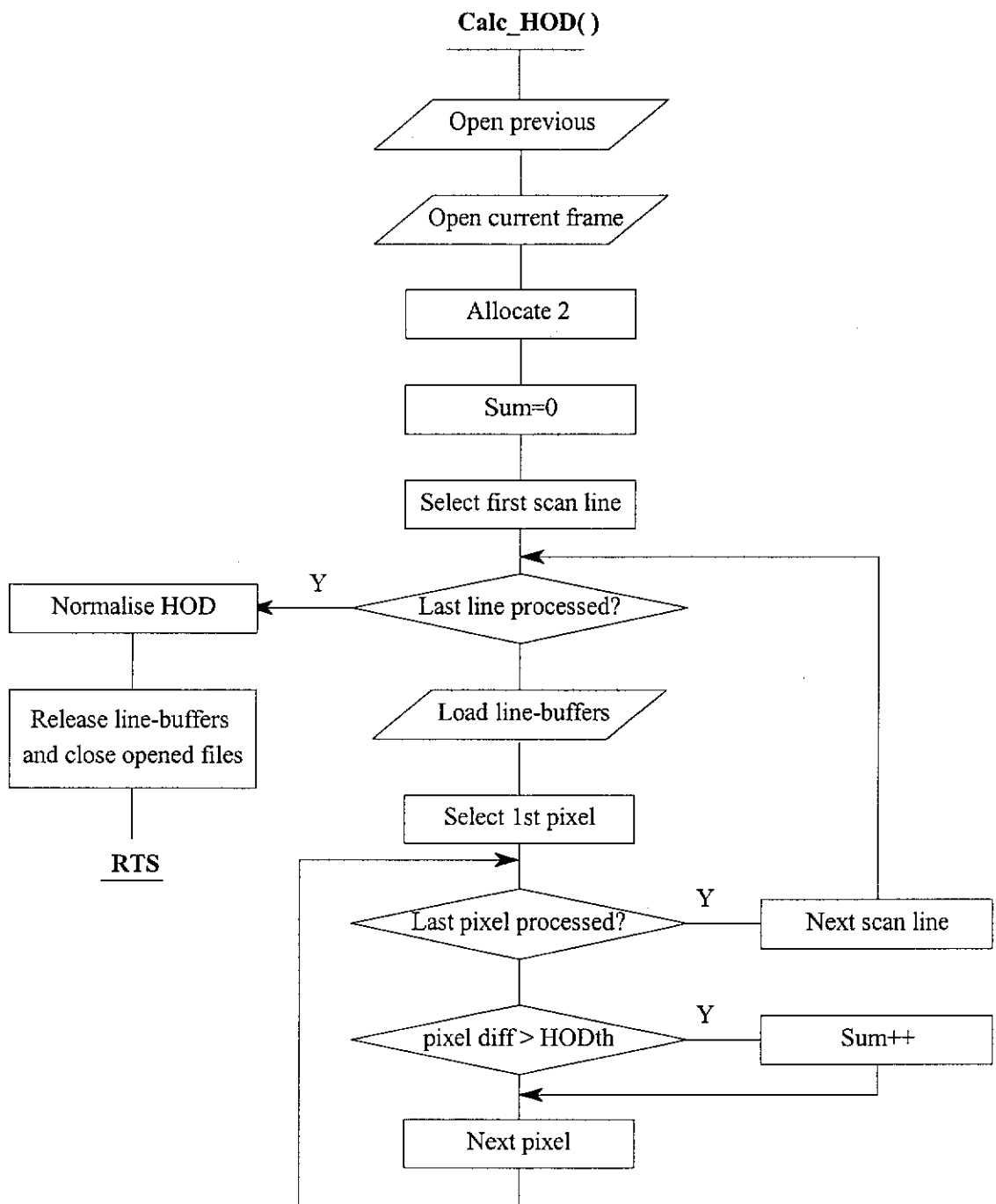
## Appendix D: Still Image Compression Flowchart



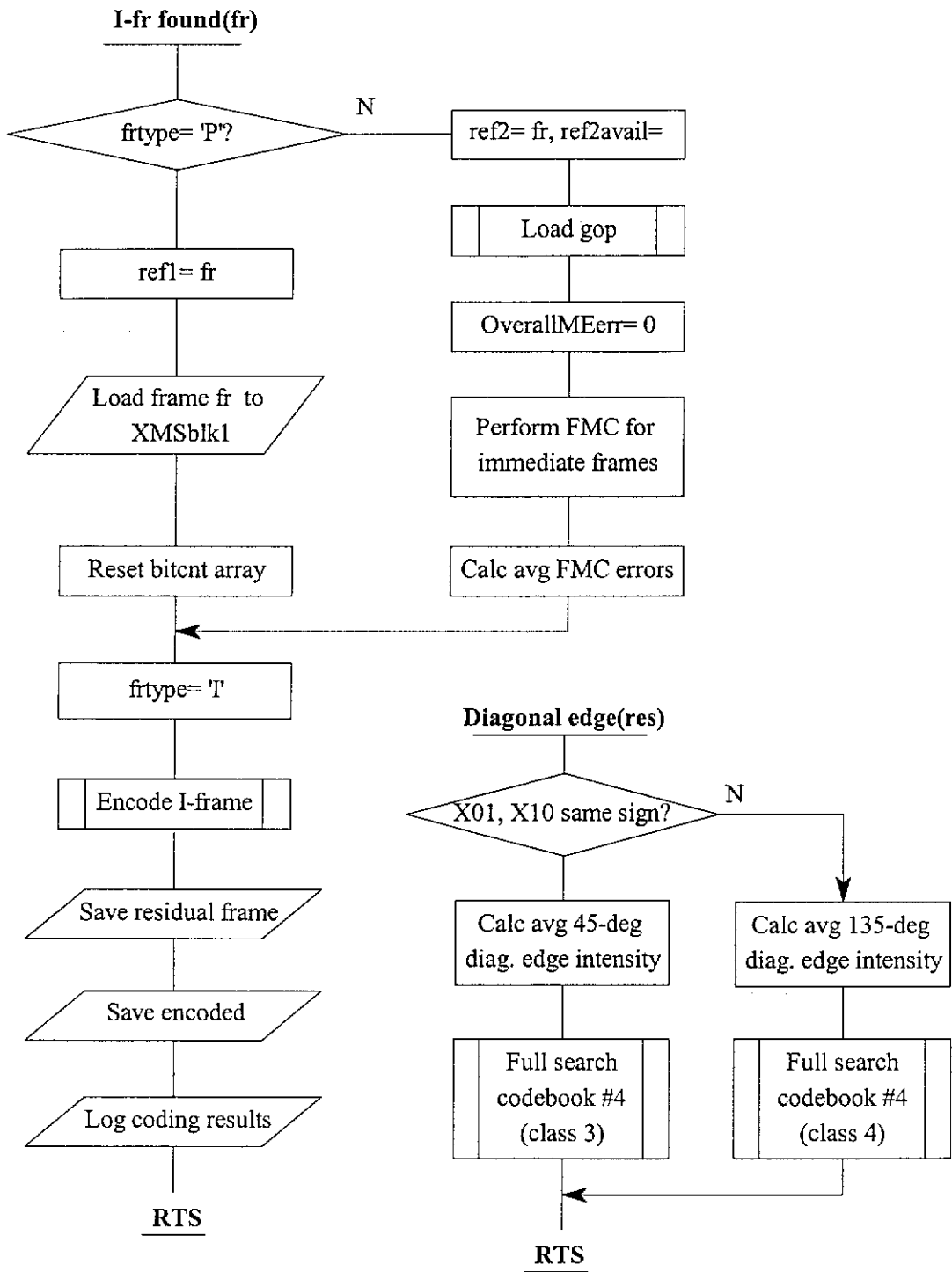


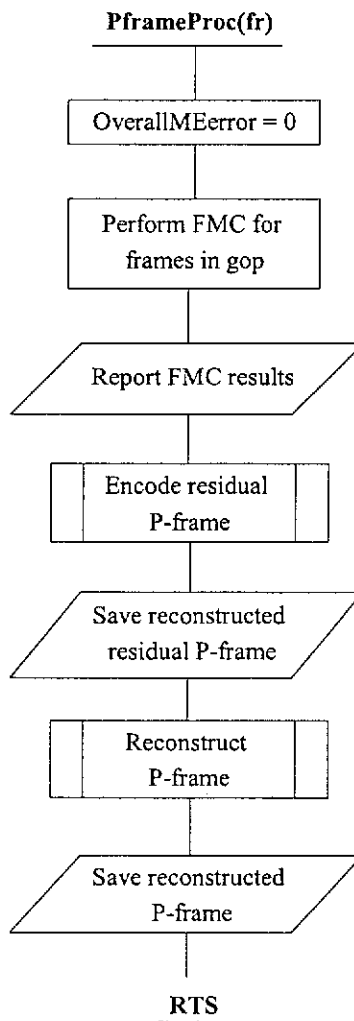
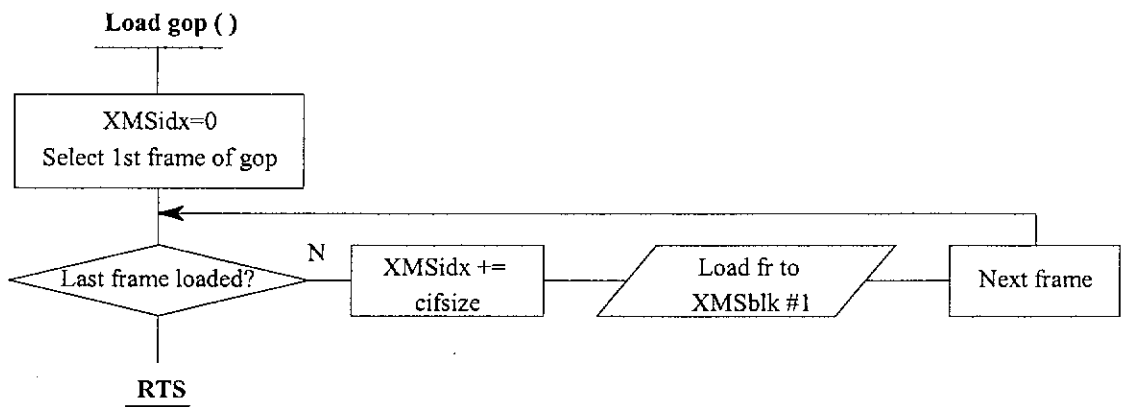


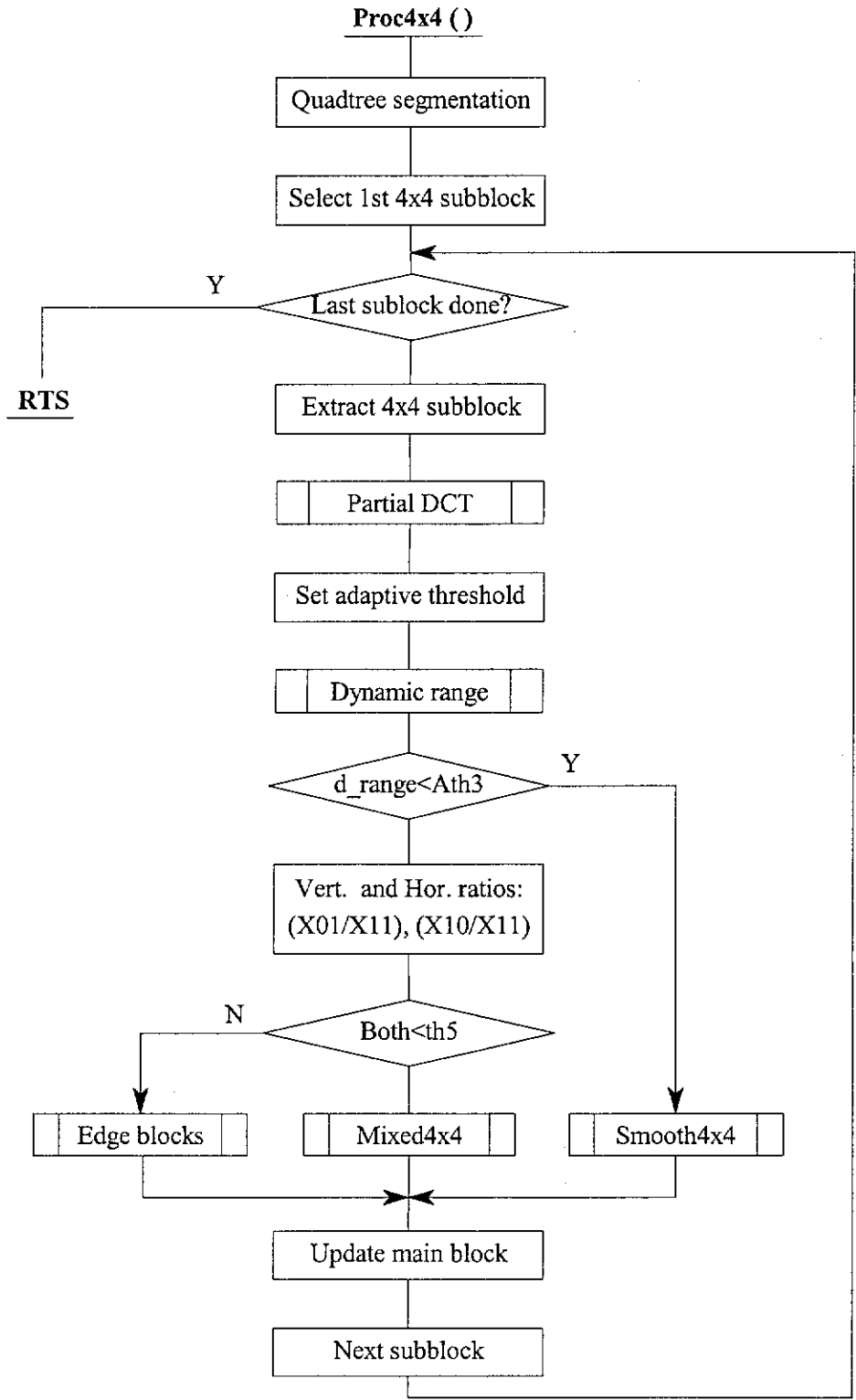


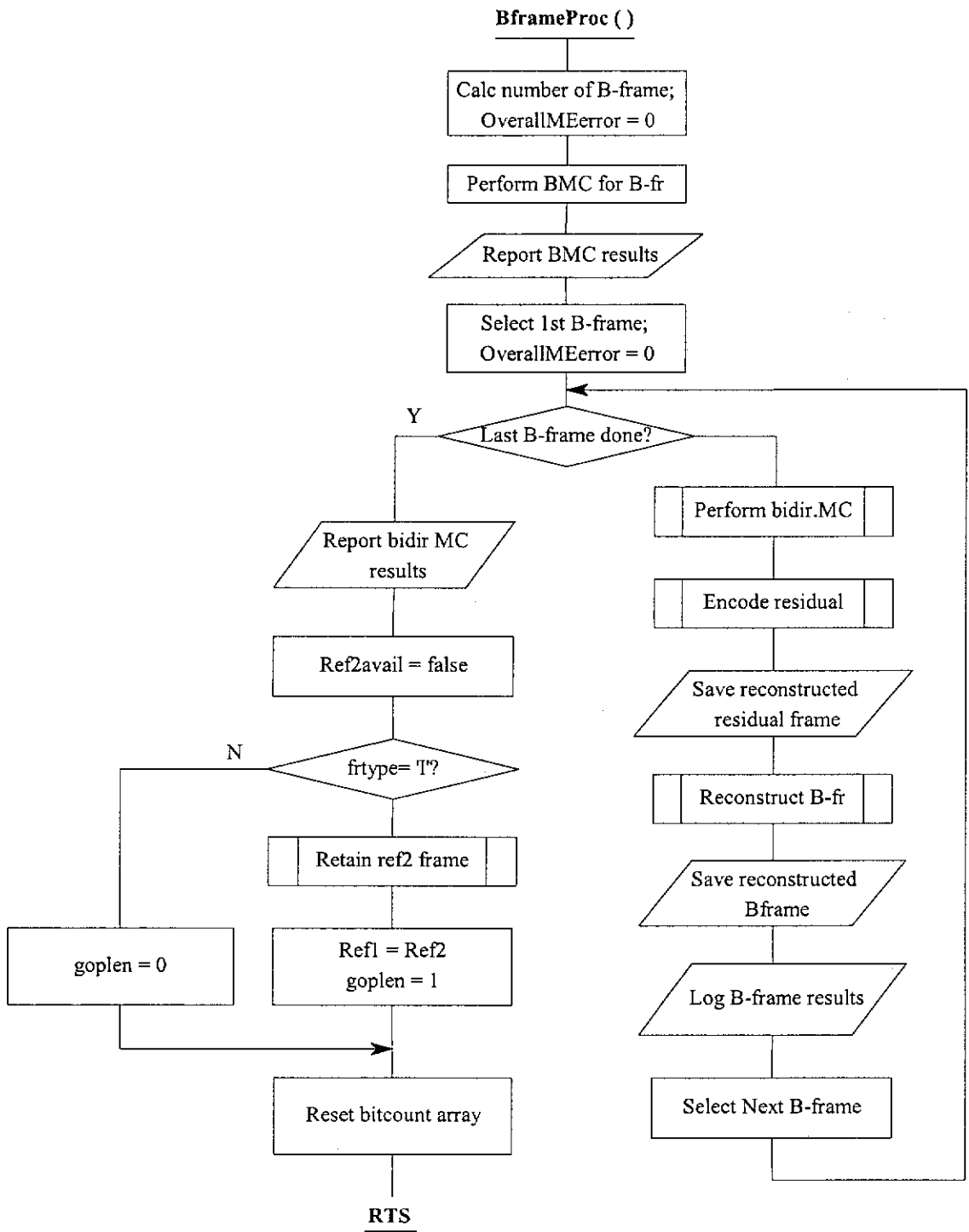


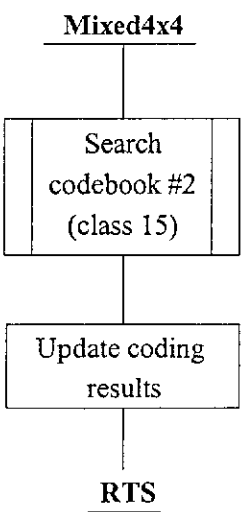
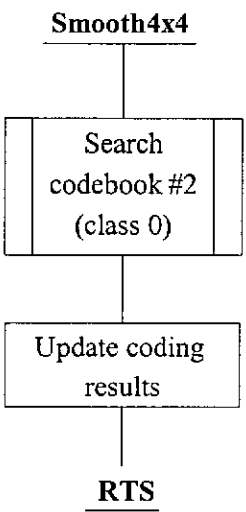
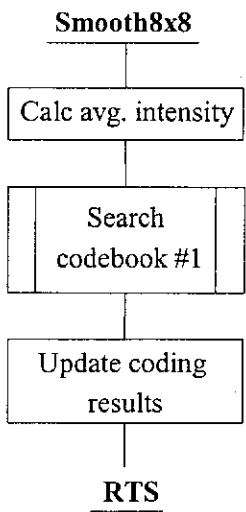
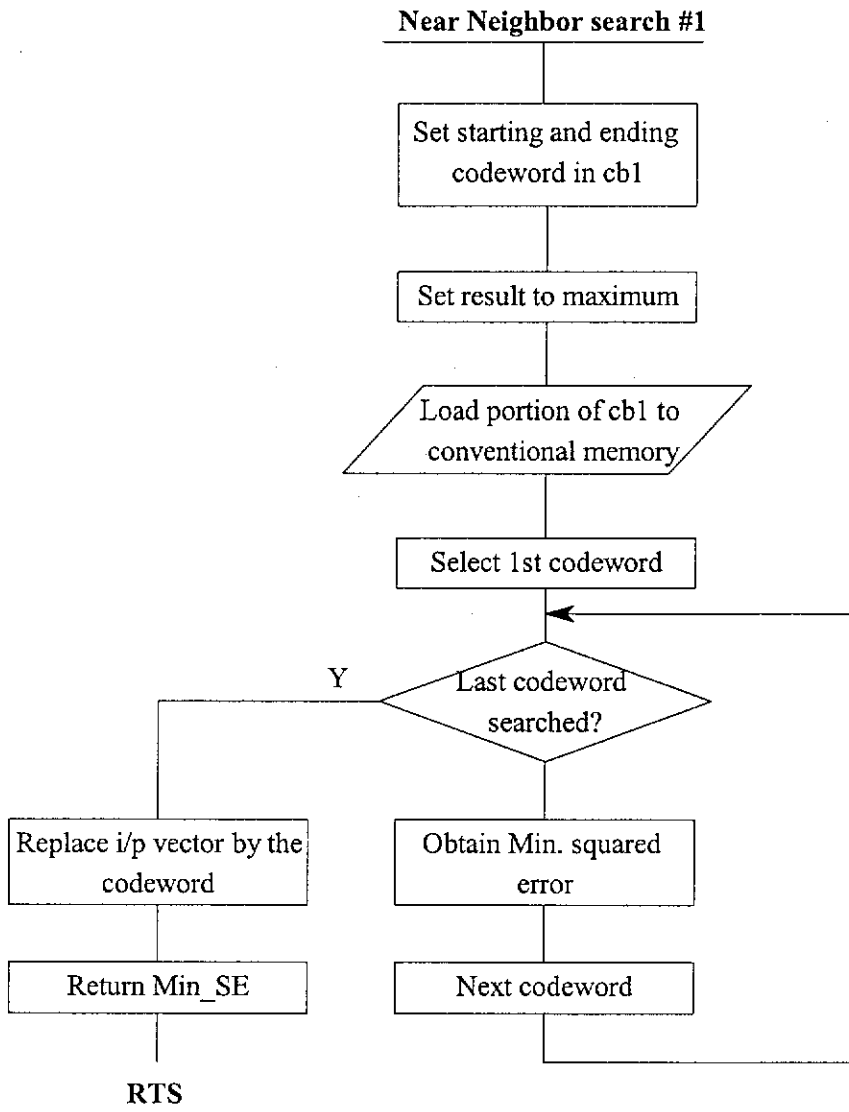


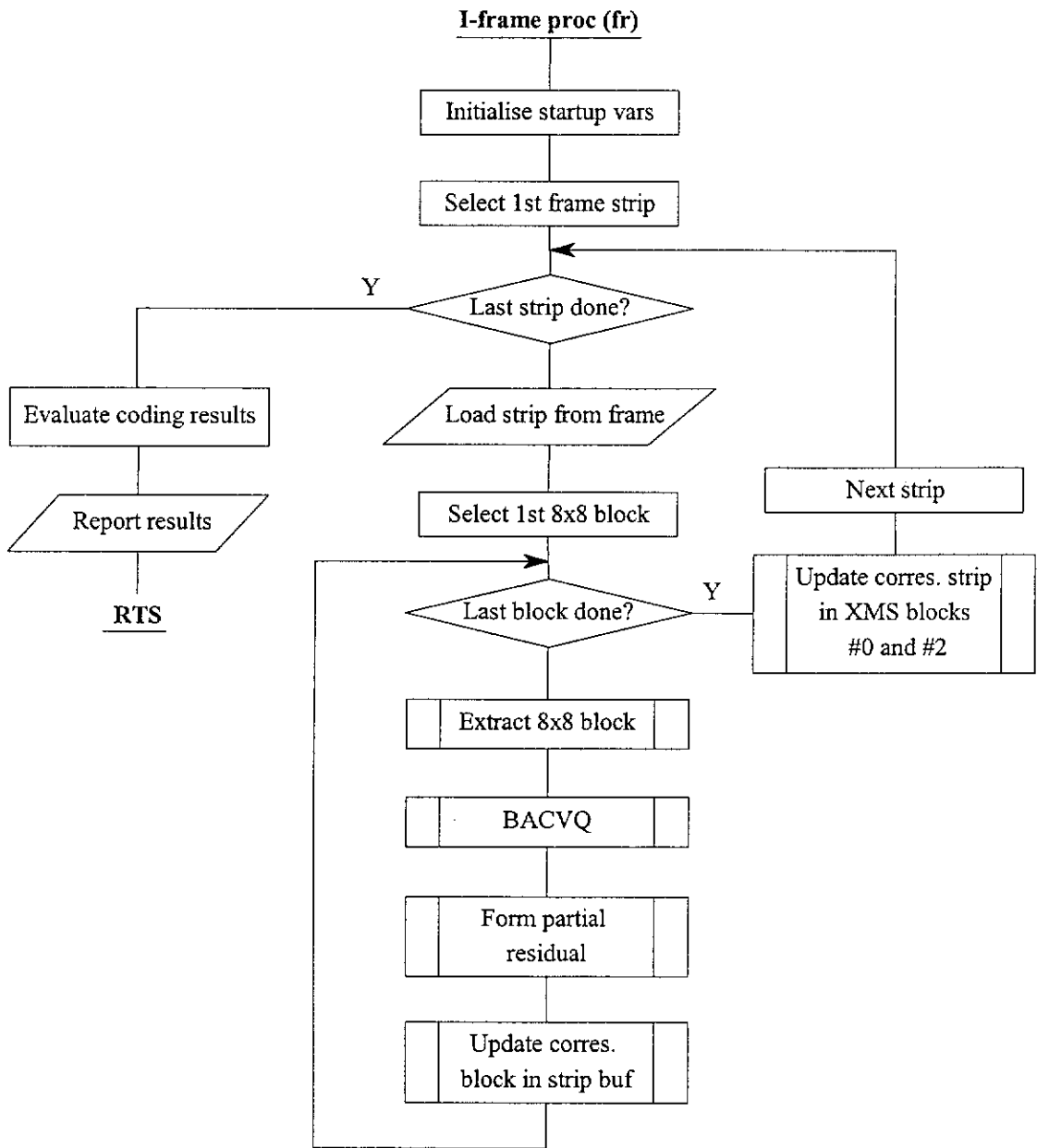


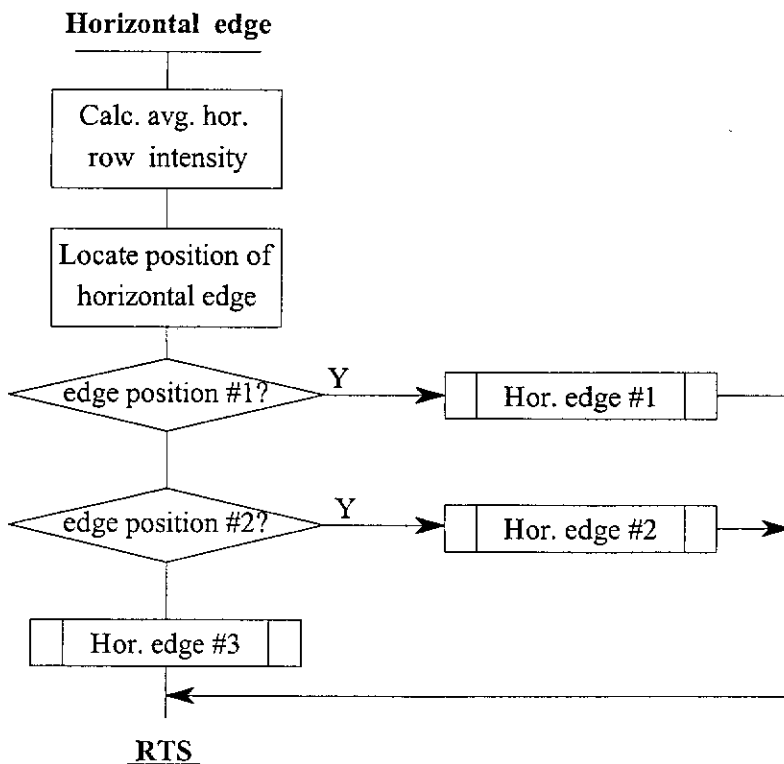
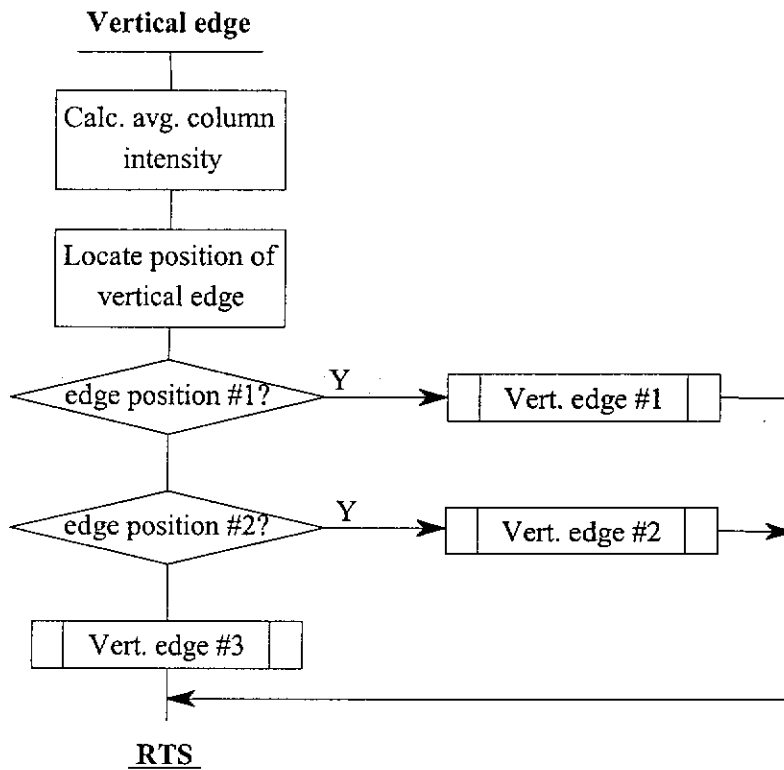


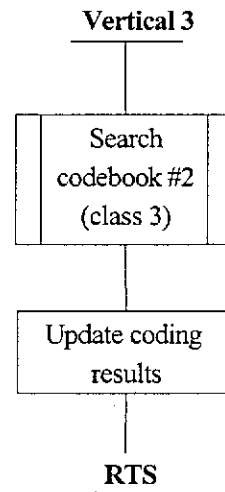
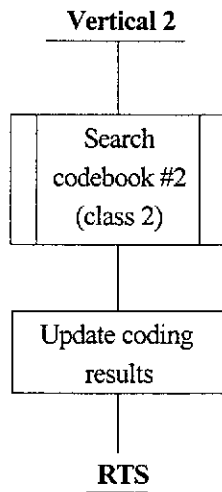
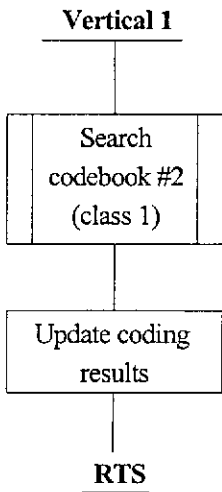
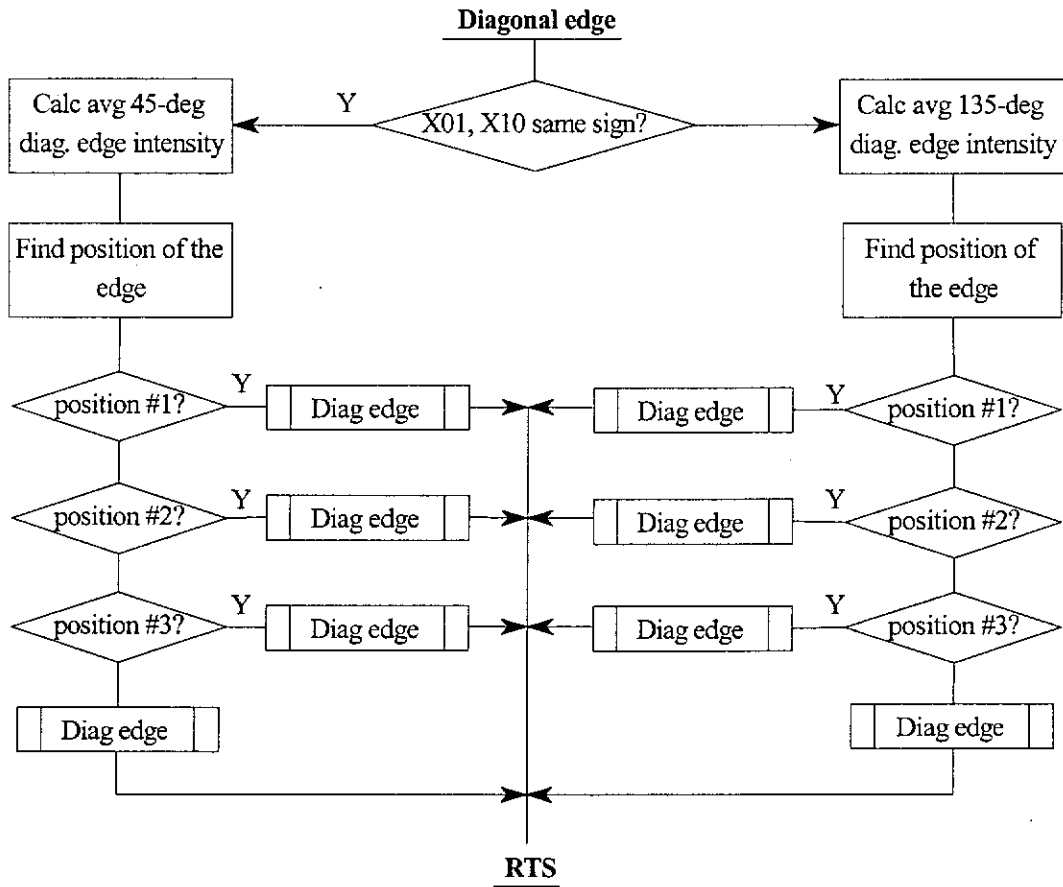




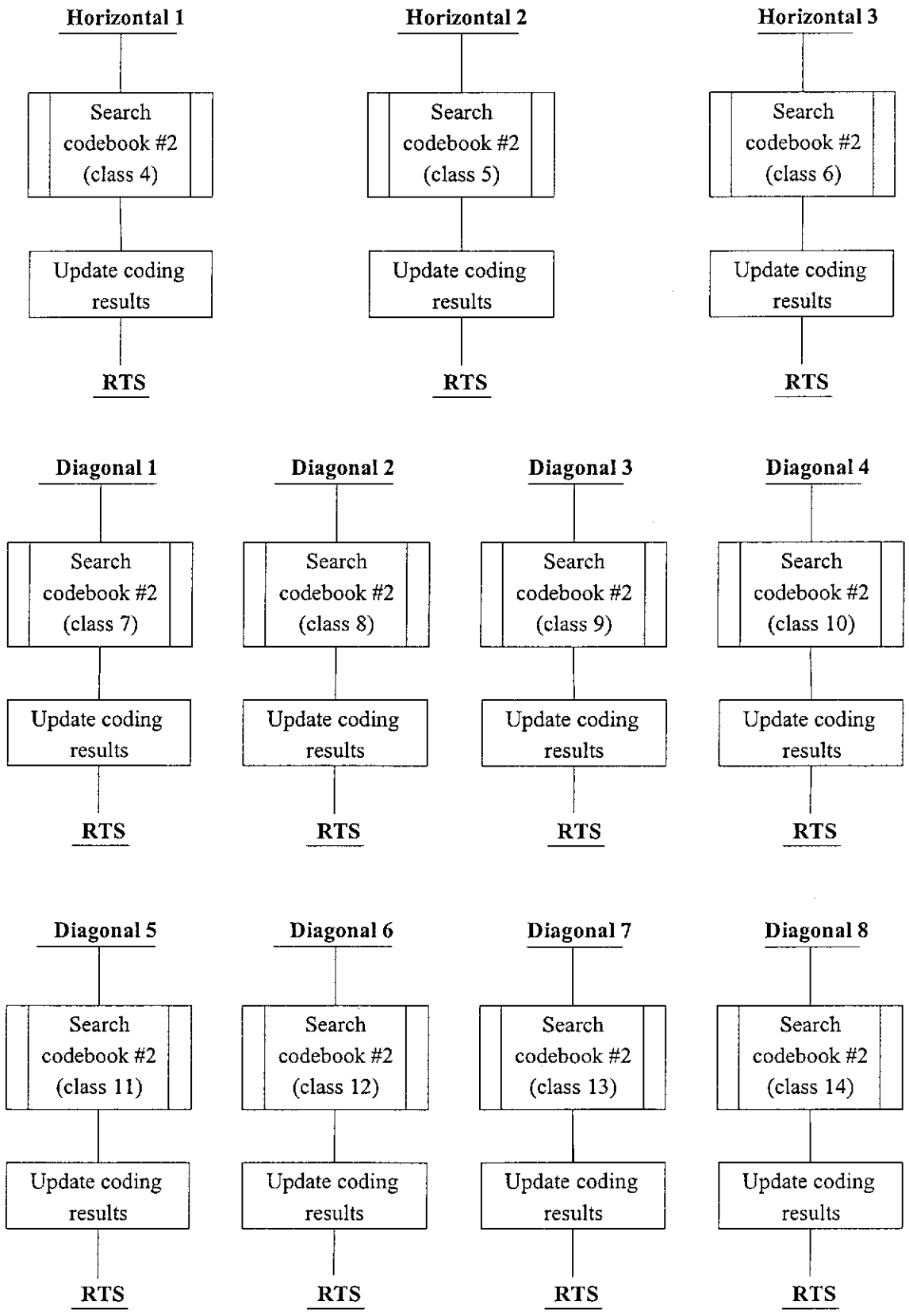


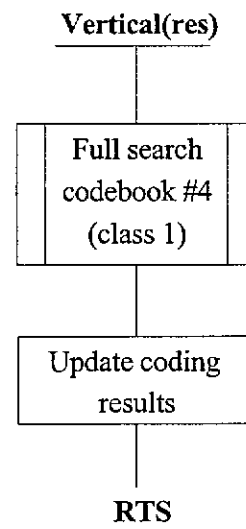
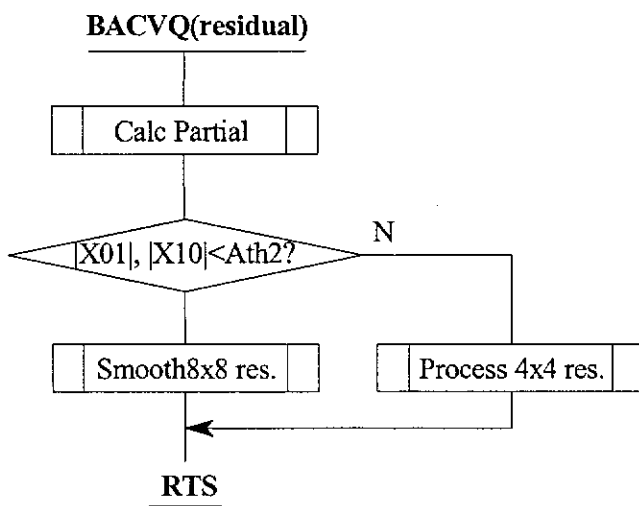
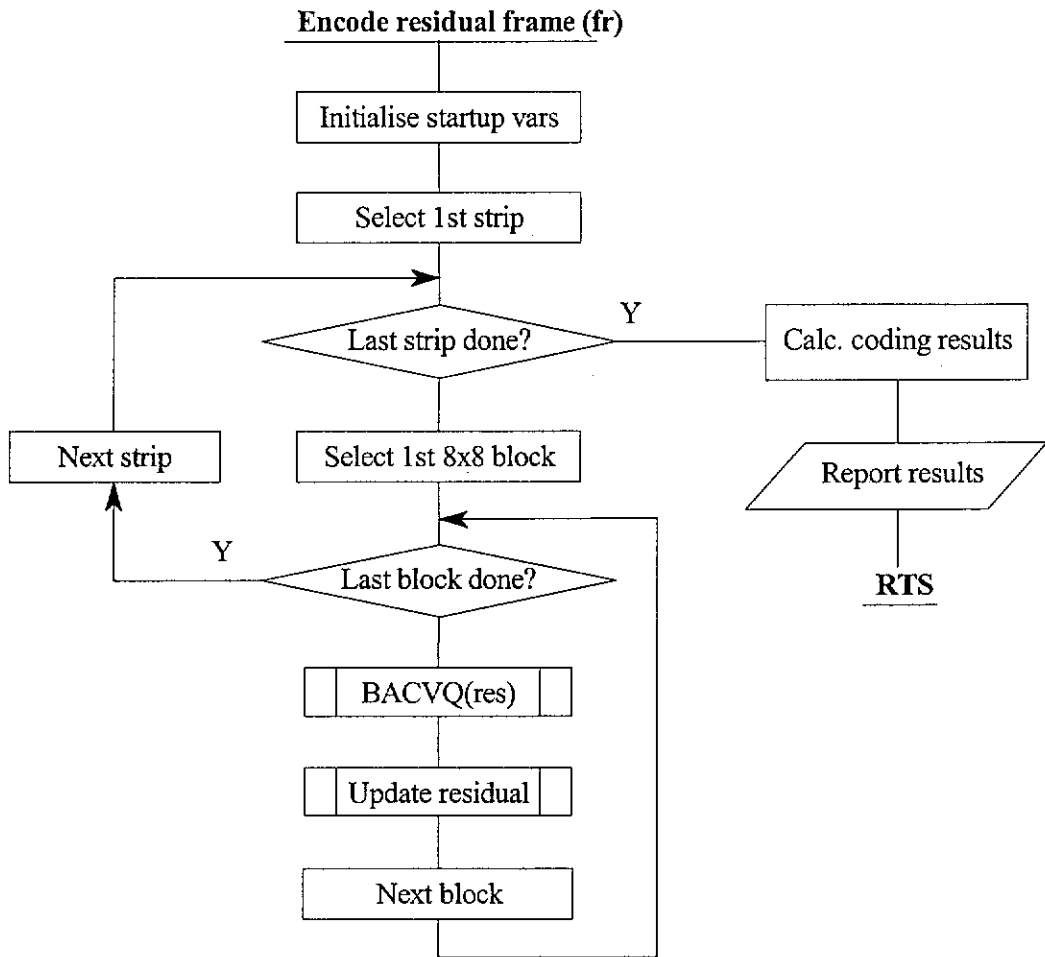


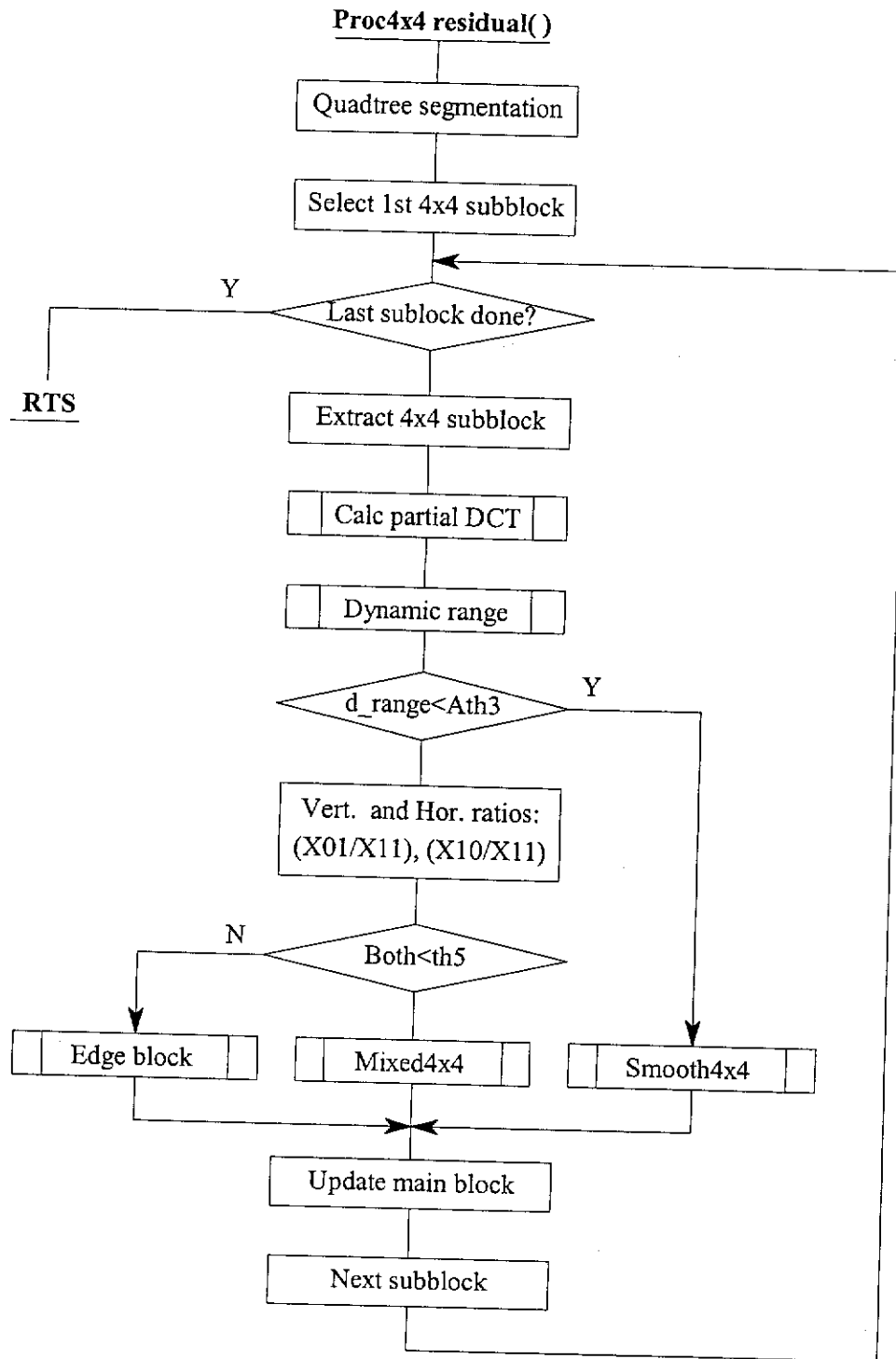


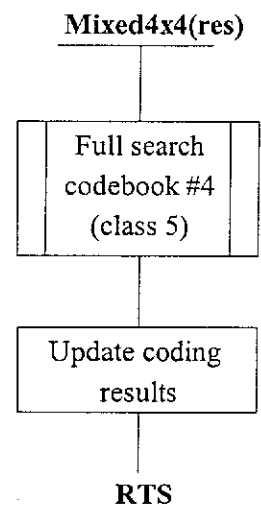
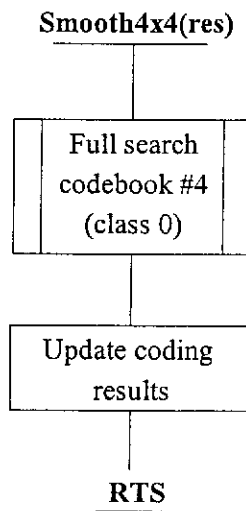
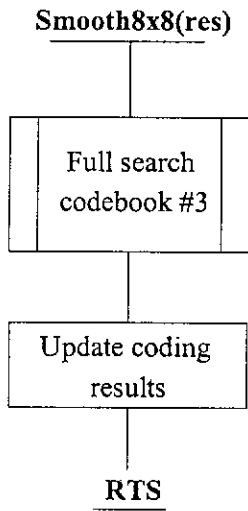
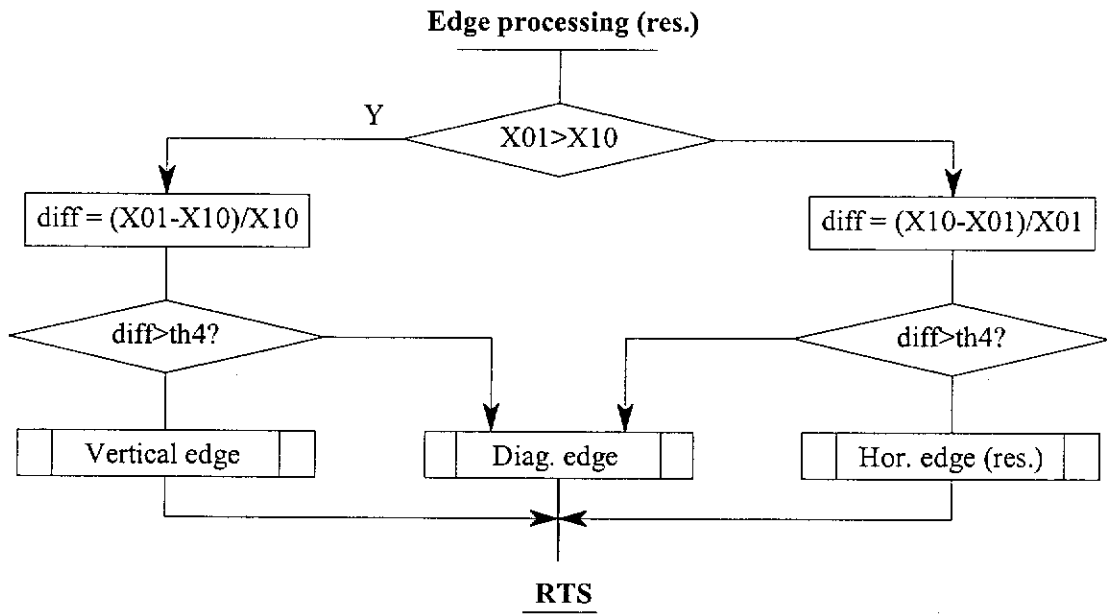




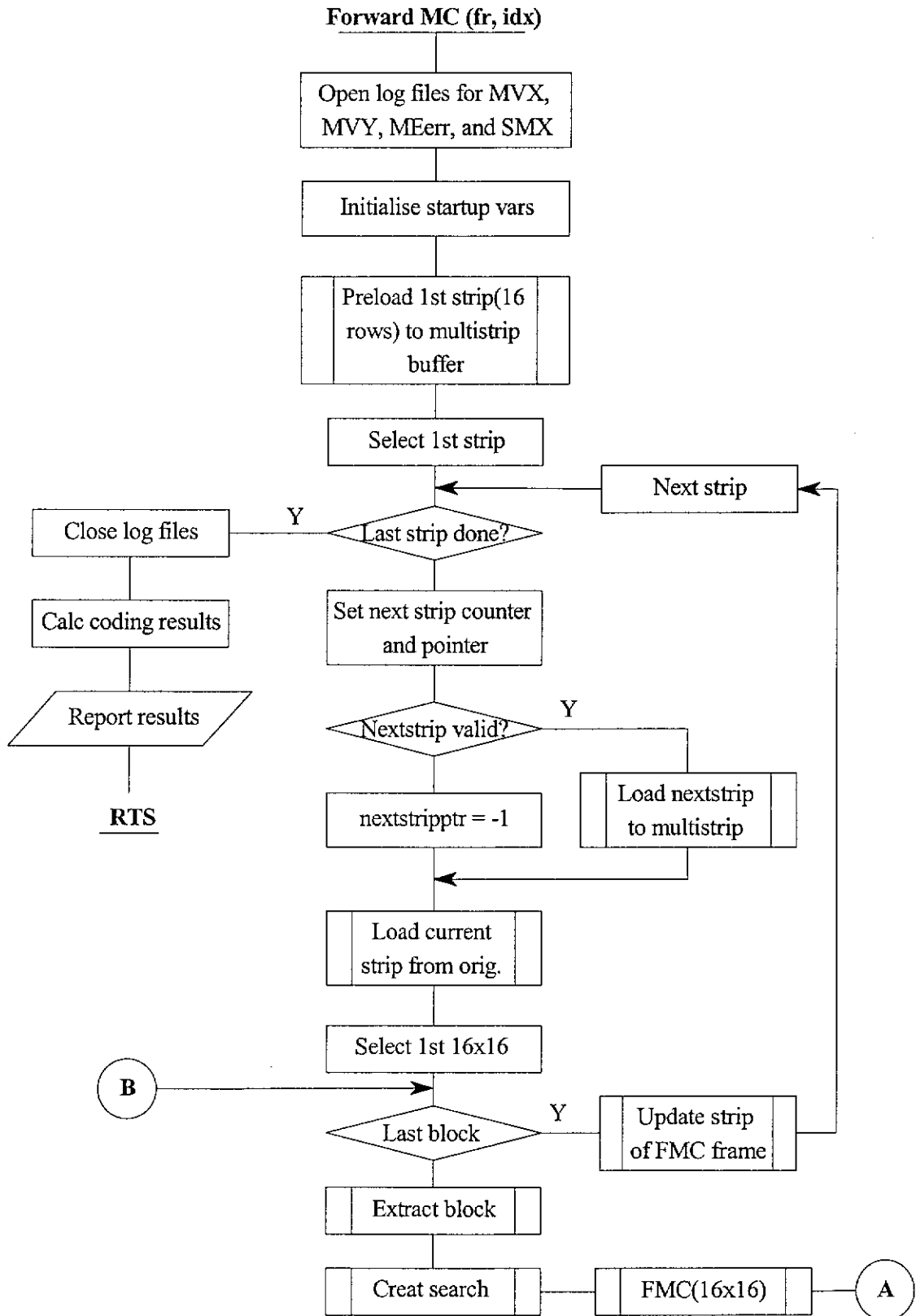


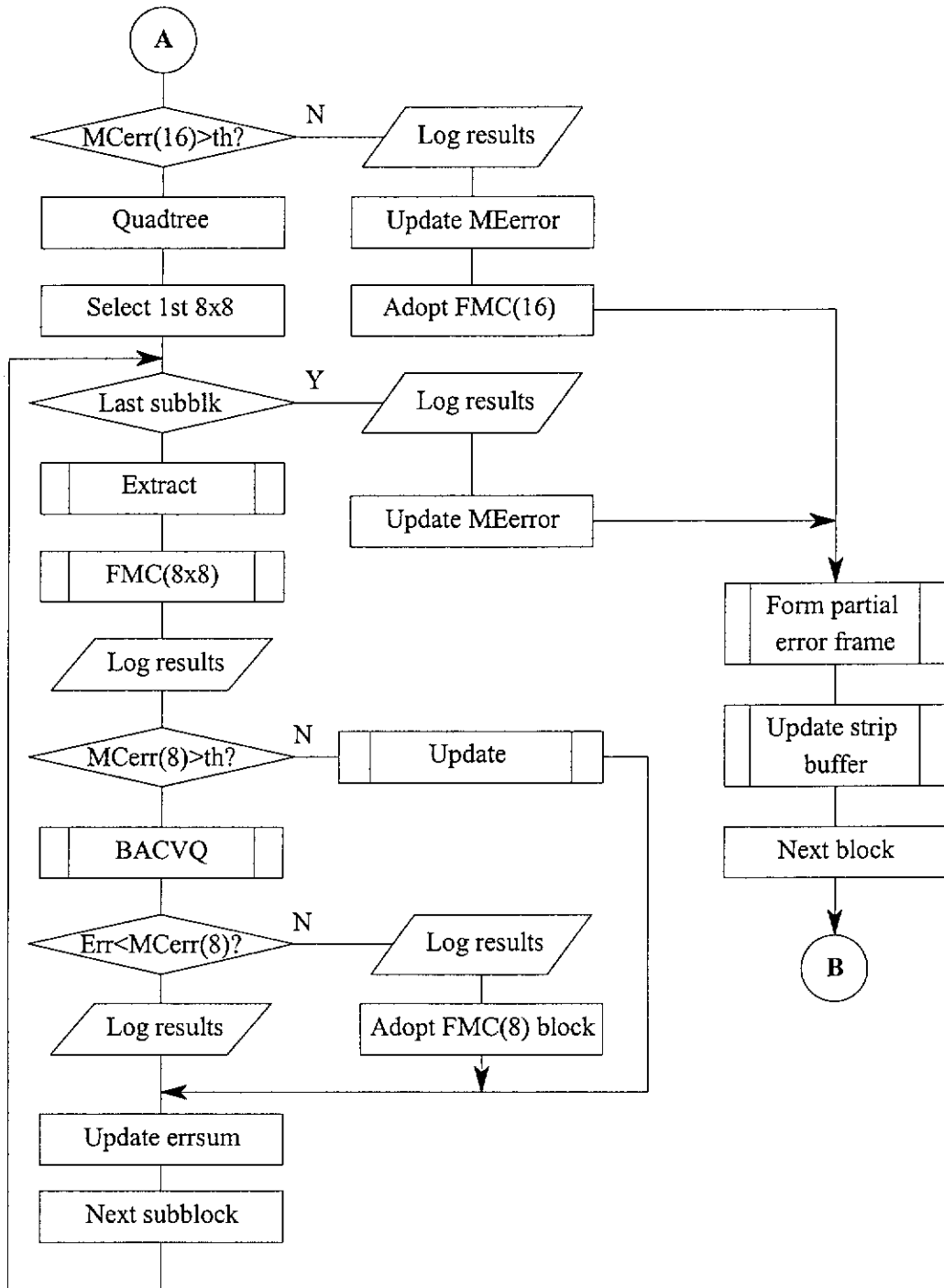


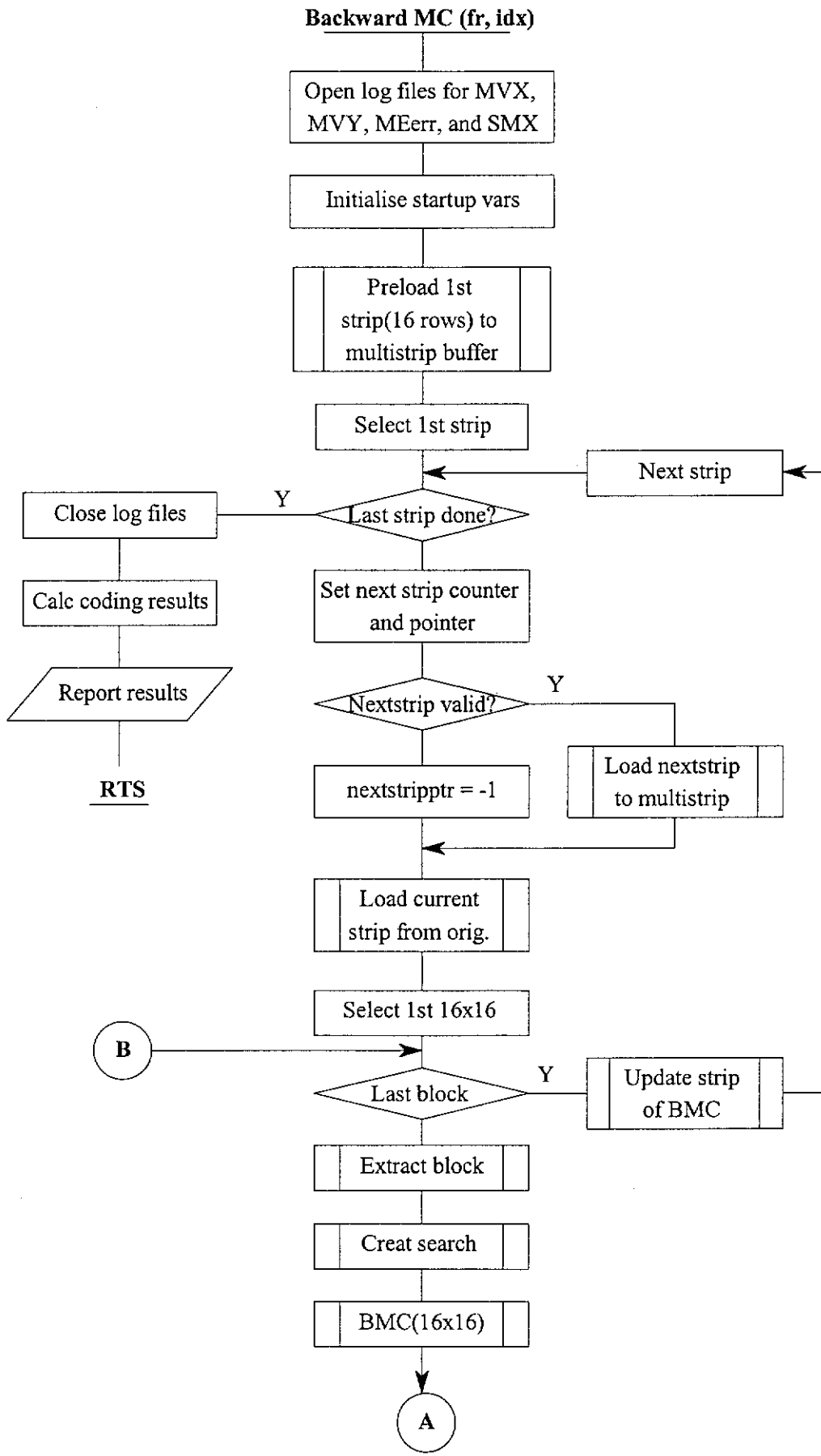


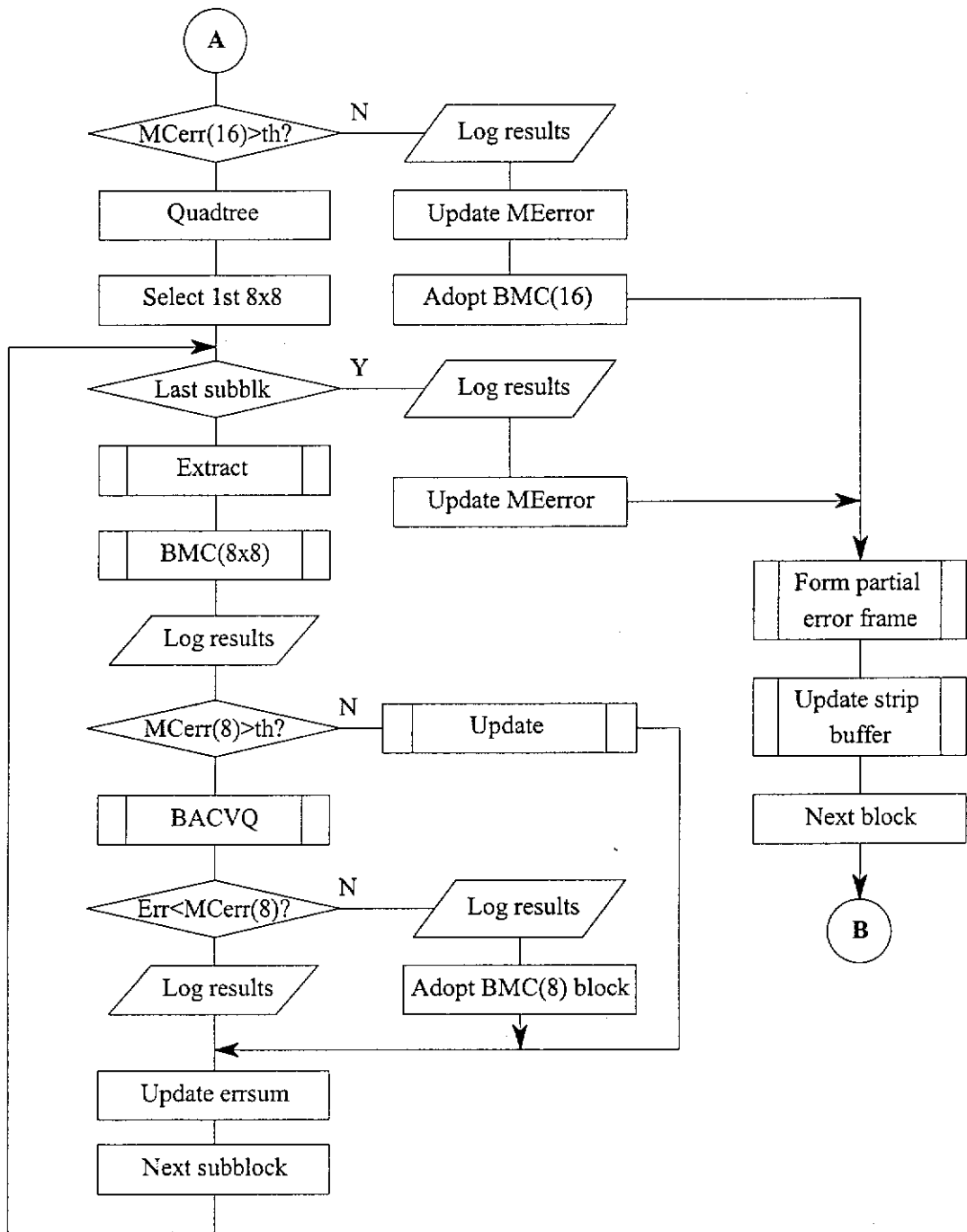


## Appendix E: Image Sequence Compression Flowchart

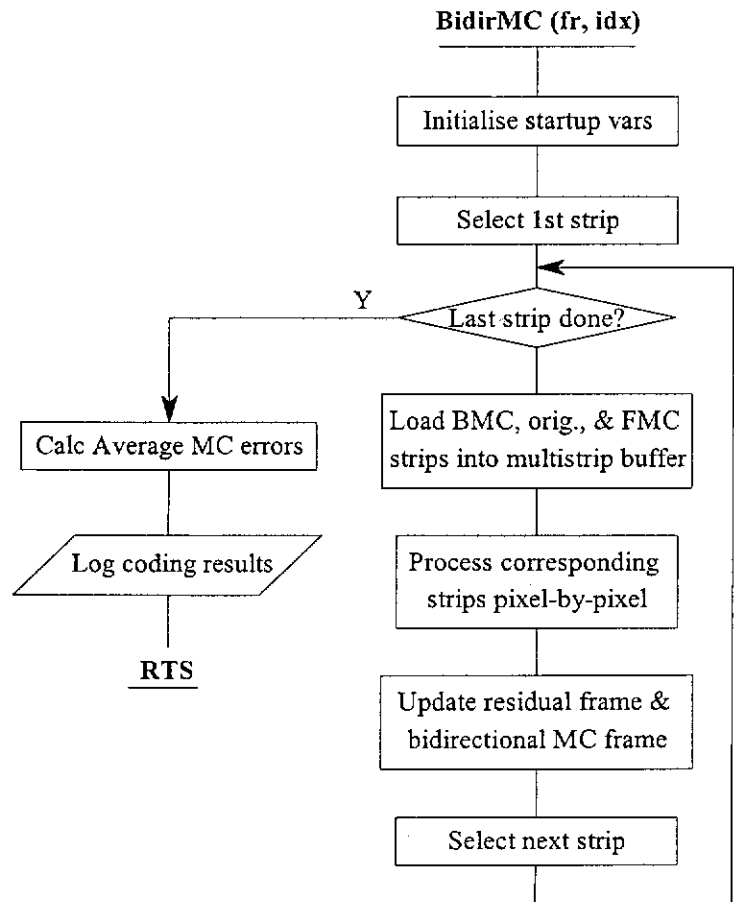
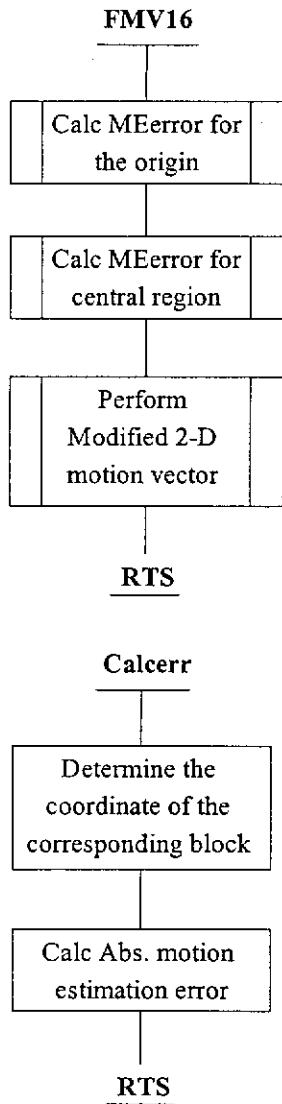
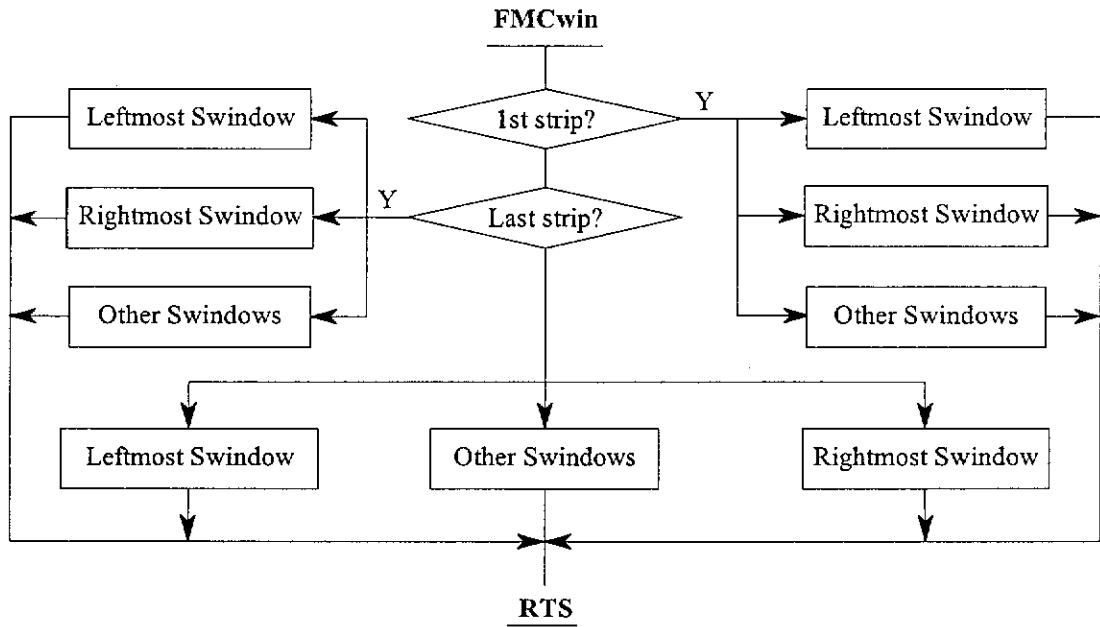






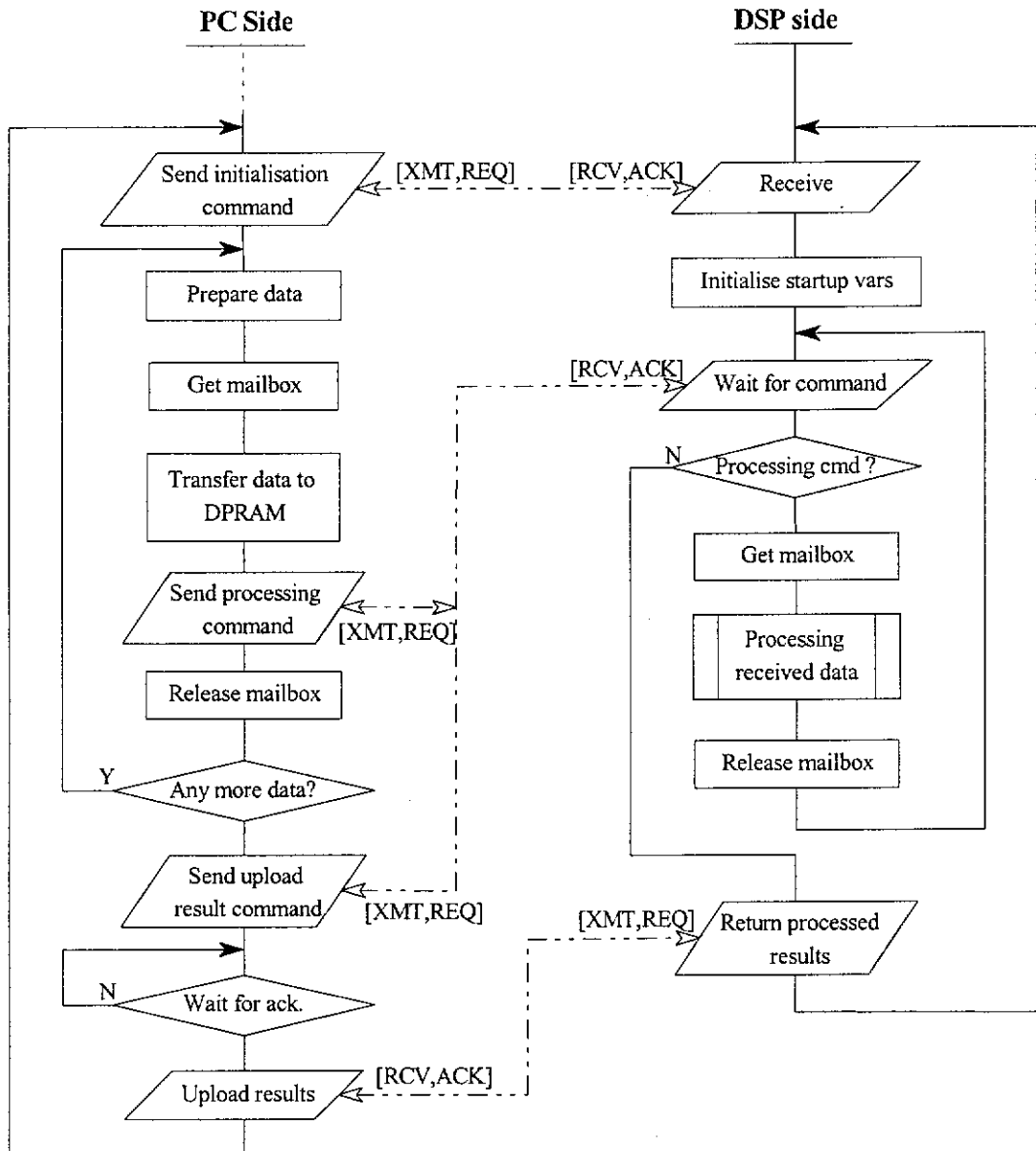




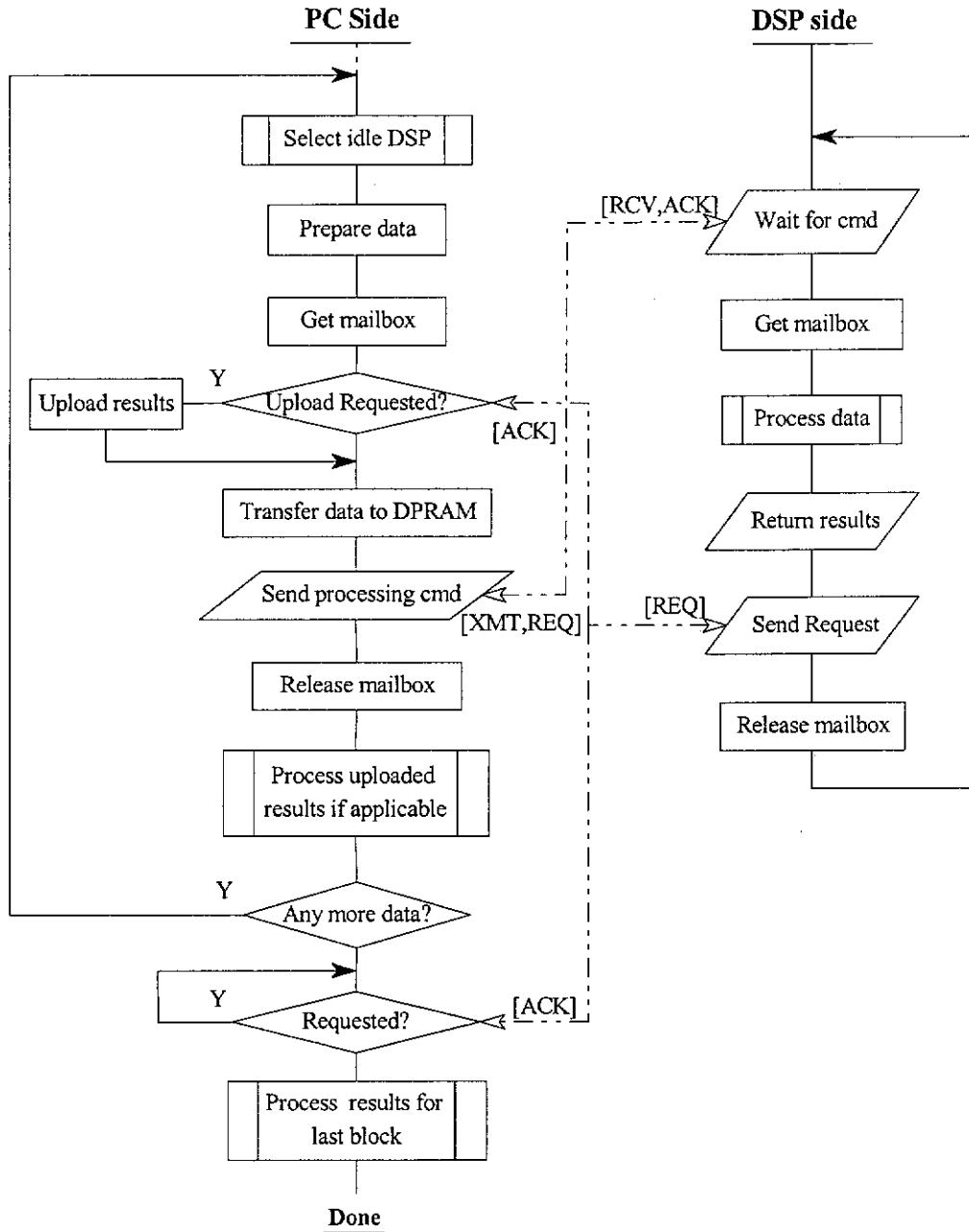




**2. Data-command processing arrangement #1 (This scheme was adopted for parallel HOD computation and parallel NNS for 8x8 image block processing)**



**3. Data-command processing arrangement #2 (This scheme was adopted for parallel 4x4 image block processing and parallel motion estimation)**



## Appendix G: Boot loading operations

Boot loading or bootstrapping is an important process for initialising a target PC32 card following a power up reset. Its operation offers a convenient mechanism for down-loading application programs to the PC32 for execution. In the prototype parallel DSP platform, the host PC is responsible for down-loading the application program corresponding to the proposed image sequence compression algorithm to both the PC32 cards using the boot loading mechanism. Once the boot loading process is complete, the down-loaded programs will start executing, thus allowing the PC32 to starting serving processing requests from the host PC.

Although the TMS320C32 processor on the PC32 DSP cards can support multiple bootstrapping modes, depending on the logic states of its interrupt input pins coming out of reset, the PC32 can only support boot loading from a single external memory. For flexibility, the PC32 was designed to boot directly from its dualport memory so that no on-board boot ROM is required.

During the process of bootstrapping the PC32 cards, the following steps are typically executed by the host PC:

1. Hold the PC32 in the reset state by asserting bit 0 of the I/O port register at address (base + 0).
2. Enable the PC32 access to the dualport memory by clearing bit 1 of the same register.
3. Write the boot loader records in the supplied TALKER.BIN file to the dualport memory (eg: D000:0000 in the host PC memory space).
4. Release the PC32 from reset by de-asserting bit 0 of the I/O port register (base +0).
5. As the C32 comes out from reset, the boot loader records in the dualport memory are processed so that code and data of TALKER.BIN are transferred to the corresponding locations in the on-board SRAM region starting from \$90000 in the PC32 memory space.

6. As soon as the last boot loader record is processed, the program counter of the PC32 will be loaded with the address contained in the first boot loader record and start executing from there. At this point, the PC32 is fully under the control of the TALKER program. This major function of this small program is to provide basic access to all of the internal and external memory resources of the PC32.
7. Once the TALKER program establishes communication with the host PC via the monitor mailbox, application programs can be down-loaded to the PC32 for execution. In the II development environment, this down-loading process is handled by the supplied `load()` library routine. Also, when the down-loading process is complete, the supplied `start_app()` library routine will have to be executed to get the down-loaded program to run.
8. Once the application program is running, it may establish further host/target communications and interact with the host for data storage and I/O functionality. Alternatively, it may also run independently without further host intervention.

## Appendix H: Influence of high level languages on processing speed

Although high level languages can be used for program development, their code generation may not always be optimum in terms of code size and execution speed. An example of this is demonstrated below where an identical test program written in C is separately compiled using Borland C and TI's optimised C compilers for the PC and the PC32 respectively; and their execution time is then compared. Basically, the test program was formulated to simulate a situation in which the processor has to perform a full search VQ operation for an 8x8 input vector with a codebook size of 50000 codewords. The pseudo code for this operation is as follows:

```
for i=1 to 50000
  Sum_of_Square_Error = 0
  for j=1 to 64
    Sum_of_Square_Error += |Input_vector(j) - Codeword(j)|2
  next j
  Total_Sum_of_Square_Error += Sum_of_Square_Error
next i
```

While the test program running on the PC serves as a reference benchmark, that running on the PC32 is subject to various optimisation stages to show how it can be further optimised for smaller code size and faster execution speed. These optimisation stages are referred to as category A, B, C and D in the table below. The results recorded in this table clearly show that code execution time has been constantly improved from 7.24 seconds down to 1.8 seconds as the code size was optimised further and further. Nevertheless, even though the optimisation in category B were to be removed due to its irrelevance to the code optimisation process, the combined improvement from category C and D would have reduced the processing time by around 3 seconds; thus resulting in an overall execution time of around 4.2 seconds on the PC32.

	Execution time (s)
Reference program in C running on PC	4.40
PC32 execution time for category A	7.24
PC32 execution time for category B	4.84
PC32 execution time for category C	4.52
PC32 execution time for category D	1.80

Category A: Using a similar program to the one running on the PC without any optimisation.

Category B: As in category A except that the absolute operation on the difference term is removed.

Category C: As in category B but with the following modification to its assembly output:

```

SUBI *AR4,*AR5,R5
ADDI 1,AR4          ==>   SUBI *AR4++,*AR5++, R5
ADDI 1,AR5

```

Category D: As in category C but replacing 32-bit software multiplication by 24-bit integer multiply instruction.

Table A.1: Execution time of full search VQ operation for an 8x8 input vector with a codebook of 50000 code words.

From this example, it can be seen that while the application of high level languages has facilitated the process of program development considerably, its implementation may involve some compromise in terms of code size and execution speed. Therefore, for applications where execution speed is of great concern, hand-coded assembly programs or modified assembly programs based on the assembly outputs from C compilers as illustrated in this example could be used for better results. The major advantage of the latter approach is that shorter program development time can generally be achieved while retaining most of the desirable features of high level languages.



## Appendix I: Samples of Parallel Processing Results

This program implements a video coding scheme with an adaptive scene segmentation algorithm based on impulse HOD to obtain variable length of GOP structure, variable block-sized motion compensation and block-adaptive classified vector quantisation.

```
* Default filename of i/p frame sequence = cif###
* Default filename for encoded frames = enc\enc_###
* Default filename for residual frames = res\res_###
* Default video coding structure data file = vc3cifb.dat
* Default output segmentation data file = ssegvc3.dat
* Default motion vector search range = -15..15
```

=====

Run debug mode [n] ?

Please specify image sequence directory [x:\]:

Please enter the starting frame number in the sequence:

Please enter the ending frame number in the sequence:

```
* XMS available: 11366400 bytes
* Largest block: 11366400 bytes
* BACVQ codebooks loaded successfully...
* Program initialisation completed successfully.
```

DSP card #0 has responded to host PC.

DSP card #1 has responded to host PC.

Processing I-frame #000:

```
*****
* Smooth 8x8 blocks = 1200
* Smooth 4x4 blocks = 264
* High texture blocks = 59
* Total edge blocks = 1213
* Blks with vert edges (v1..v3) = 157, 187, 126
* Blks with hor edges (h1..h3) = 153, 181, 175
* Blks with 45-deg edges (d1..d4) = 43, 12, 20, 35
* Blks with 135-deg edges (d5..d8) = 40, 25, 19, 40
* MSE/pel of 8x8 smooth blocks = 39.51
* MSE/pel of 4x4 smooth blocks = 14.84
* MSE/pel of edge blocks = 139.22
* MSE/pel of mixed blocks = 266.86
* Overall MSE/pel and PSNR = 59.69, 30.37dB
* Avg bit rate & compression ratio = 0.3395bpp, 23.56
```

Forward motion vector estimation...

```
* Frame #001: => FMC error= 5.5374/pel, f16,f8,i8= 396, 0, 0( 0)
* Frame #002: => FMC error= 5.2480/pel, f16,f8,i8= 396, 0, 0( 0)
* Frame #003: => FMC error= 5.6158/pel, f16,f8,i8= 396, 0, 0( 0)
=> Avg FMC error= 5.467060/frame
```

Encode residual frame #003: \*\*\*\*\*

```
* Smooth 8x8 blocks = 1584
* Smooth 4x4 blocks = 0
* High texture blocks = 0
* Total edge blocks = 0
* Blks with vert edges (v1..v3) = 0
* Blks with hor edges (h1..h3) = 0
* Blks with 45-deg edges (d1..d4) = 0
* Blks with 135-deg edges (d5..d8) = 0
* MSE/pel of 8x8 smooth blocks = 16.00
* Overall MSE/pel and PSNR = 16.00,36.09dB
* Avg bit rate & compression ratio = 0.1250bpp,64.00
=> Actual absolute MC error= 5.4764/pel
```

Processing B-frame...

Backward motion vector estimation...

```
* Frame #002: => BMC error= 5.2006/pel; f16,f8,i8=396, 0, 0( 0)
* Frame #001: => BMC error= 5.4712/pel; f16,f8,i8=396, 0, 0( 0)
=> Avg BMC error= 5.3359/frame
```

Bidirectional motion compensation...

```
* Frame #001:
=> BiMC error= 5.3042/pel
Encode residual frame #001: *****
* Smooth 8x8 blocks = 1584
```

```

* Smooth 4x4 blocks           = 0
* High texture blocks        = 0
* Total edge blocks          = 0
* Blks with vert edges (v1..v3) = 0
* Blks with hor edges (h1..h3) = 0
* Blks with 45-deg edges (d1..d4) = 0
* Blks with 135-deg edges (d5..d8) = 0
* MSE/pel of 8x8 smooth blocks = 14.89
* Overall MSE/pel and PSNR = 14.89,36.40dB
* Avg bit rate & compression ratio = 0.1250bpp,64.00
=> Actual absolute MC error= 5.3075/pel

* Frame #002:
=> BiMC error= 5.0323/pel
Encode residual frame #002: *****
* Smooth 8x8 blocks           = 1584
* Smooth 4x4 blocks          = 0
* High texture blocks        = 0
* Total edge blocks          = 0
* Blks with vert edges (v1..v3) = 0
* Blks with hor edges (h1..h3) = 0
* Blks with 45-deg edges (d1..d4) = 0
* Blks with 135-deg edges (d5..d8) = 0
* MSE/pel of 8x8 smooth blocks = 14.28
* Overall MSE/pel and PSNR = 14.28,36.58dB
* Avg bit rate & compression ratio = 0.1250bpp,64.00
=> Actual absolute MC error= 5.0719/pel

=> Avg biMC error of B-frame= 5.1897/frame

* Program finished.
* Execution starting date and time = Thu Mar 14 18:50:21 1996
* Execution ending date and time   = Thu Mar 14 18:51:46 1996

```