Image Processing and DSP Technology Applied to Remote Activity Monitoring

A thesis is presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Technology at Massey University.

by

Nick B. Body

2000

Massey University
COLLEGE OF SCIENCES

Institute of Information Sciences & Technology
Private Bag 11 222,
Palmerston North,
New Zealand
Telephone: 64 6 350 5799
Facsimile: 64 6 350 5723

Integrated Research and Teaching in the Fields of:

Statistics & Applied Statistics

Computer Science & Information Systems

Information & Electronic Engineering

TO WHOM IT MAY CONCERN

This is to state that the research carried out for my Doctoral thesis entitled "Image Processing and DSP Technology Applied to Remote Activity" in the Institute of Information Sciences and Technology, Massey University, Turitea, New Zealand is all my own work.

This is also to certify that the thesis material has not been used for any other degree.

Candidate: Nick B Body

Candidate Signature: 

Date: 6/11/00

Image Processing and DSP Technology Applied to Remote Activity Monitoring

Nick B. Body

Institute of Information Sciences and Technology

Massey University

NEW ZEALAND

Submitted:

6th November, 2000

Abstract

This thesis describes the development of image processing algorithms to achieve industrial remote activity monitoring. The design of such algorithms is constrained by the requirement that they must be implemented in real-time on a cost effective digital signal processor (DSP). A review of the literature revealed that two viable alternatives were available. These are JPEG, a well established lossy block-based image compression algorithm and methods using wavelets. A suitable family of wavelets was identified as the biorthogonal-7.9, the coefficients indicating the number of taps in the quadrature mirror filters at the heart of the transform. Data compression is then achieved through the construction of zerotrees followed by sequential baseline coding. The two candidate algorithms were then systematically compared for recovered image quality and implementation cost at high compression ratios in the range of 16:1 to 64:1. On this basis the wavelet approach was selected and its implementation on a DSP studied. The architectural features of the Motorola 56303 DSP are presented and analysed. It is shown that the various components required for the wavelet based algorithm can be efficiently mapped onto the DSP architecture. Motion detection and image watermarking algorithms were designed and co-operatively implemented with the compression algorithm. A new method of watermarking highly compressed images was developed and this algorithm has been named the Image Authentication Watermark. A new way of representing optimal Huffman code tables has been developed to enable Huffman entropy coding to perform competitively with the more complex arithmetic coding. A product of this research is a smart digital camera that has been integrated into an automated video surveillance system now in industrial production.

Keywords:

image processing, compression, watermarking, digital camera, digital signal processor.

Contents

Abstract ..... **iv**

Preface ..... **xi**

Acknowledgements ..... **xiii**

List of Figures ..... **xv**

Publications ..... **xxi**

1. Introduction **1**

1.1 Scope of research 2

1.2 Thesis overview 3

1.3 Contents by chapter 4

1.4 References 6

2. Low Bit-rate Image Coding **7**

2.1 Image data compression techniques 7

2.2 Lossless and lossy image compression 8

2.3 Predictive coding 11

2.3.1 One- and two-dimensional differential modulation 12

2.3.2 Interframe techniques 13

2.3.3 Vector quantisation 13

2.4 Transform coding 15

2.4.1 Two- and three-dimensional image transforms 17

2.4.2 Block based and global based 18

2.4.3 Coefficient coding strategies 18

2.4.3.1 Zonal coding 19

2.4.3.2 Threshold coding 19

2.4.3.3 Run-length coding 20

2.5 Adaptive methods 22

2.5.1 Adaptive predictive coding 22

2.5.1.1 Predictor rule 23

2.5.1.2 Quantiser function 23

2.5.2 Adaptive transform coding 23
2.5.3 Adaptive coding summary 24
2.6 Wavelet coding 26
2.6.1 Image analysis techniques 26
2.6.1.1 Fourier analysis 26
2.6.1.2 Short-Time Fourier analysis 27
2.6.1.3 Wavelet analysis 28
2.6.2 Wavelet transform of one-dimensional signals 30
2.6.3 Signal extension of one-dimensional signals 40
2.6.4 Wavelet transform of two-dimensional images 45
2.6.5 Survey of some existing wavelet based image codecs 53
2.6.5.1 Wavelet filter choice 53
2.6.5.2 Wavelet decomposition choice 54
2.6.5.3 Wavelet coefficient coding choice 55
2.7 Miscellaneous coding techniques 60
2.8 Discussion 61
2.9 References 61

3. Efficient Image Compression Algorithms Implemented on DSP Architecture 65
3.1 Data flow description of the image compression algorithm 66
3.2 Specialised architectural features of the DSP 74
3.2.1 MAC and advanced Harvard architecture 74
3.2.2 Indirect addressing modes 76
3.2.3 Data marshalling and pipelining 77
3.2.4 Barrel shifter 79
3.3 Specific algorithm implementations 79
3.3.1 QMF kernel implementation 80
3.3.2 Tree building and coefficient coding implementation 83
3.3.2.1 Zerotree construction 83
3.3.2.2 Zerotree coefficient coding 89
3.3.2.3 Summary 91
3.3.3 Huffman coding implementation 92
3.3.3.1 Generation of bit patterns to construct Huffman codes 93

3.3.3.2 Delta coding of Huffman code tables 96
3.3.3.3 Efficient concatenation of variable length codes 96
3.3.3.4 Summary 97
3.4 Efficient use of memory 99
3.4.1 Stage 1: Wavelet transformation 100
3.4.2 Stage 2: Coefficient coding 105
3.4.3 Stage 3: Huffman coding 107
3.5 Compression algorithm performance 109
3.5.1 Profile of DSP compression algorithm 112
3.5.2 Profile of Pentium decompression algorithm 114
3.5.3 Comparison of algorithm performance 115
3.6 Comparison with a JPEG like image compression algorithm 118
3.6.1 Algorithm structure and memory usage 118
3.6.2 Execution speed 119
3.6.3 Image quality 121
3.7 Discussion 124
3.8 References 126

4. Motion Detection and Image Watermarking 129
4.1 Motion detection 130
4.1.1 Requirement for motion detection 132
4.1.2 Review of motion detection techniques 132
4.1.2.1 Root Mean Square Error measurement 132
4.1.2.2 Optical flow analysis 132
4.1.2.3 Object segmentation and tracking 134
4.1.2.4 Motion detection for surveillance 134
4.1.3 Wavelet decomposition and motion measurement 134
4.1.4 Efficient DSP implementation of the motion metric 142
4.1.4.1 Implementing the SSE metric 142
4.1.4.2 Arithmetic overflow considerations 145
4.1.4.3 System considerations 145
4.2 Image watermarking 147
4.2.1 Reasons for image watermarking 148

4.2.1.1 Image authentication 148
4.2.1.2 Image ownership 150
4.2.2 Review of image watermarking techniques 151
4.2.2.1 Original-referencing verses self-referencing watermarking 151
4.2.2.2 Pixel based watermarking techniques 155
4.2.2.3 Transform based watermarking techniques 156
4.2.3 Watermarking in the wavelet domain 157
4.2.3.1 Proposed co-operative scheme 157
4.2.4 Performance of co-operative scheme 163
4.2.5 Efficient DSP implementation of the image watermark 169
4.2.5.1 Watermark bit rotation 171
4.2.5.2 Modulo-8 checksum 172
4.2.5.3 Insertion sort 173
4.2.5.4 Coefficient perturbation 174
4.3 Discussion 176
4.4 References 177

5. The Complete Remote Activity Monitoring System and its Performance 179
5.1 System evolution 180
5.1.1 Prototype system 180
5.1.1.1 Hardware architecture 181
5.1.1.2 Software architecture 183
5.1.1.3 Bottlenecks and latency 188
5.1.2 Production system 191
5.1.2.1 Hardware architecture 192
5.1.2.2 Software architecture 195
5.1.2.3 Bottlenecks and latency 197
5.2 Compression algorithm performance 202
5.2.1 Target SWT compression ratios and quantiser settings 203
5.2.2 Reference images 204
5.2.3 Results of compression performance test 205
5.2.4 Discussion of compression performance test 212
5.3 Watermark algorithm performance 214

5.3.1 Results of watermark performance test 214
5.3.2 Discussion of watermark performance test 217
5.4 Discussion 219
5.5 References 220

6. Discussion and Conclusions 221
6.1 Thesis discussion 221
6.1.1 Low bit-rate image coding 222
6.1.2 Efficient implementation on a DSP 223
6.1.3 Motion detection and image watermarking 225
6.1.4 Performance of complete system 226
6.2 Thesis conclusions 228
6.3 Future work 230
6.4 References 231

Bibliography 233

Appendix .....

Preface

It was in 1983 that I first became interested in computers and electronics, I was eleven years old. My school had recently acquired a Sinclair ZX-81 (with a 16 kbyte RAM expansion pack). Not many of my peers seemed interested in the chunky graphics that could be produced by this home computer, but I was hooked. After some bargaining with my parents, I was soon to be the owner of a series of home computers, each one boasting more RAM and better (even coloured) graphics.

It was not until the final year of my undergraduate degree in 1993 that I did any serious image processing or digital signal processor (DSP) work. Dr Wyatt Page, later one of my supervisors, challenged the class to develop an image compression algorithm in Matlab using many of the techniques described in this thesis (less wavelets). I remember enjoying this programming challenge immensely. I suspect that is why I ended up spending over three years of my working career back at Massey University trying for the extra bonus marks in that assignment! I also enjoyed the DSP related projects I worked on in my final year. The challenge of writing complex algorithms in raw assembler language appealed to my sense of detail especially when there were many demanding memory and processing constraints.

I worked as a software engineer for two and a half years and then jumped at the chance to return to university to work on a project that would involve both DSPs and image processing. The company I worked for gave me three years of "study leave" and sponsored the project (and me). As I write this last part of this thesis, I am back at work with the same company and I find it gratifying to know that the smart digital camera is in commercial production. I guess it has the potential to make the owner of the company rich (or richer) and that alone should keep me and my peers employed for some time to come.

Nick Body, 9th October 2000

Acknowledgements

I wish to thank the following people:

- My supervisor Professor Bob Hodgson who is Head of the Institute of Information Sciences and Technology at Massey University. Although a busy man, he always made time for our often lengthy fortnightly supervision meetings. Your feedback on my writing is slowly sinking into my writing style. Such feedback certainly has been required to polish many of my raw thoughts from the jumbled sentences that first appeared on the page.
- My second supervisors Dr Wyatt Page and Dr Jamil Khan both offered support and encouragement as well as a wealth of technical knowledge about image processing and DSPs. Wyatt proved to be a great person to bounce ideas off and to help guide me in designing the image processing algorithms.
- Dr Don Bailey was my colleague who occupied the office next to me for the three years I worked on this research. Don was the head of the Image Analysis Unit at Massey and what he doesn't know about image processing is probably not worth knowing about! Your constant "availability" to answer my questions or to listen to my ramblings was much appreciated. Our joint research into "stuff that hadn't yet been done with Huffman coding" proved to be most amusing!
- My employer (Cardax International) and FRST (Foundation for Research, Science and Technology) for financing the project to make it look appealing to my need to be paid well whilst returning to be a student. The funding of the trip to Chicago to present a conference paper at ICIP-98 was also provided by the project and was the experience of a lifetime.
- My colleagues at Cardax International for supporting the development of the commercial smart digital camera. Felix Collins and Charles Oram integrated my image processing code into the commercial system. Tony Smith designed and constructed the test-rig hardware for the prototype camera. Grant Allen provided the used of his flat bed scanner to assist with some of the illustrations in this thesis.

- My wife, Melicia, for being ever so patient as the time to write up my thesis seemed to drag on forever! I suspect you will see more of me now and I will no longer talk about DSPs in my sleep…
- My parents for assisting with the proofreading of the final copy of the thesis. Not a task I would normally wish on anyone, but they were kind of handy!
- And finally I would like to thank God, not just a person but a complete *tuple* of [Father, Son, Holy Spirit]. Many times when debugging the complexity of the assembly language code, the quiet reassuring voice of God would point me in the right direction. Your assistance in what must seem like a trivial exercise is much appreciated.

List of Figures

Figure 1.1 – Topology comparison between current and future AVS systems 3
Figure 2.1 – Quantiser function, many-to-one relationship 10
Figure 2.2 – Block diagram of a typical hybrid image coder 10
Figure 2.3 – Differential pulse code modulation codec 11
Figure 2.4 – Pixels (A,B,C,D) used in two-dimensional prediction 13
Figure 2.5 – Vector quantisation of image blocks 14
Figure 2.6 – One-dimensional transform coding 15
Figure 2.7 – Zonal coding of quantised transform coefficients 19
Figure 2.8 – Threshold coding of quantised transform coefficients 20
Figure 2.9 – Run-length coding of quantised transform coefficients 21
Figure 2.10 – Zig-zag and Hilbert curve scanning order for an 8 x 8 block 21
Figure 2.11 – Fourier analysis of a one-dimensional signal 27
Figure 2.12 – Short-Time Fourier analysis of a one-dimensional signal 28
Figure 2.13 – Time/frequency analysis grids for various techniques 29
Figure 2.14 – Sampled signal S(n) 31
Figure 2.15 – Daubechies-4 (db4) wavelet filters 32
Figure 2.16 – Structure of a single level wavelet decomposition 33
Figure 2.17 – First level decomposition of S(n) with db4 wavelet 33
Figure 2.18 – Four level decomposition of S(n) with db4 wavelet and the resulting leaf node octave bins 34
Figure 2.19 – Coefficient signal components of the four level decomposition of S(n) 35
Figure 2.20 – Reconstruction of successive approximations of S(n) 36
Figure 2.21 – Structure of a single level of wavelet reconstruction 37
Figure 2.22 – Creation of A_4 from cA_4, upsampling followed by low pass filtering 38
Figure 2.23 – Creation of D_4 from cD_4, upsampling followed by high pass filtering 39
Figure 2.24 – Summation of A_4 and D_4 to create cA_3 39
Figure 2.25 – Implied periodicity of the DFT and DCT 40

Figure 2.26 – Signal extension of a finite length signal 42
Figure 2.27 – Pyramidal decomposition structure of the expansive WT with an 8 element filter on a 512 element input signal 42
Figure 2.28 – Various padding strategies for expansive wavelet transform 43
Figure 2.29 – Non-expansive wavelet transform signal extension 44
Figure 2.30 – Intensity plot of first 256 pixels in first scanline of the Lena image 45
Figure 2.31 – Biorthogonal-7.9 wavelet filters 46
Figure 2.32 – Block diagram of the two-dimensional wavelet decomposition 48
Figure 2.33 – Various tree forms of three level wavelet decompositions 49
Figure 2.34 – Set of one-dimensional horizontal wavelet decompositions 50
Figure 2.35 – Set of one-dimensional vertical wavelet decompositions 51
Figure 2.36 – Four level wavelet transform of the Lena image 52
Figure 2.37 – Four level wavelet packet transform of the Lena image 52
Figure 2.38 – Simplified WSQ encoder 55
Figure 2.39 – Quad-tree structure from wavelet decomposition 56
Figure 2.40 – Scanning method used in the DWT 58
Figure 3.1 – Wavelet compression signal processing functional diagram 67
Figure 3.2 – Single level one-dimensional wavelet decomposition (QMF followed by 2:1 decimation) signal processing flow graph 68
Figure 3.3 – Single level two-dimensional wavelet decomposition signal processing flow graph 69
Figure 3.4 – Four level two-dimensional wavelet decomposition signal processing flow graph 70
Figure 3.5 – Zerotree building, quantisation, scanning and run-length coding of detail coefficients signal processing flow graph 71
Figure 3.6 – DPCM coding of approximation coefficients signal processing flow graph 72
Figure 3.7 – Huffman coding signal processing flow graph 73
Figure 3.8 – Subset of data Arithmetic Logic Unit (ALU) block diagram 74
Figure 3.9 – Subset of Address Generation Unit (AGU) block diagram 76
Figure 3.10 – Inner loop operation of the wavelet filter bank for filters of size FILTLEN = 9 81
Figure 3.11 – Reading from four pools of data to construct a zerotree 84

Figure 3.12 – Offsets associated with reading a sequence of blocks 85
Figure 3.13 – Timeline of double buffered DMA and DSP core processing 86
Figure 3.14 – Timeline including double buffering of the output stream 88
Figure 3.15 – Timeline where DSP core processing exceeds DMA controller usage 88
Figure 3.16 – Scan pattern for run-length coding of a vertical edge zerotree 90
Figure 3.17 – Huffman codes generated from symbol bit lengths 94
Figure 3.18 – Procedure using bit-reversed addressing to build Huffman codes 95
Figure 3.19 – Inner loop operation of Huffman code concatenation using the "`insert`" op-code 98
Figure 3.20 – Memory pools available for image compression algorithm 99
Figure 3.21 – Internal memory usage for wavelet transform 101
Figure 3.22 – External memory usage for first level of wavelet transform 102
Figure 3.23 – Non-critical nature of high frequency information in a natural image 103
Figure 3.24 – External memory usage for first level of truncated wavelet transform 104
Figure 3.25 – External memory usage after four levels of the initially truncated wavelet transform 105
Figure 3.26 – Internal memory usage for coefficient coding 106
Figure 3.27 – External memory usage for coefficient coding 106
Figure 3.28 – Internal memory usage for Huffman coding 107
Figure 3.29 – External memory usage for Huffman coding 108
Figure 3.30 – Time spent in each stage of the wavelet based algorithm 113
Figure 3.31 – Proportion of time spent in each stage of the algorithm 116
Figure 3.32 – Comparison between DCT and wavelet based algorithms 118
Figure 3.33 – Time spent in each stage of the DCT based algorithm 120
Figure 3.34 – 160 x 160 pixel subsection from 512 x 512 pixel Lena image 122
Figure 3.35 – 60 x 60 pixel subsection from 512 x 512 pixel Lena image 122
Figure 4.1 – 512 x 288 pixel raw images from 140 frame sequence 135
Figure 4.2 – 512 x 288 pixel raw difference images from 140 frame sequence 136
Figure 4.3 – 32 x 18 pixel approximation images from 140 frame sequence 137
Figure 4.4 – Full set of 32 x 18 pixel approximation images 138

Figure 4.5 – Full set of 32 x 18 pixel difference images 139
Figure 4.6 – Two regions of interest where motion is to be measured 140
Figure 4.7 – Root Mean Square Error over 140 frames for region 1 141
Figure 4.8 – Root Mean Square Error over 140 frames for region 2 141
Figure 4.9 – Signal processing flow graph showing SSE calculation for a single region 143
Figure 4.10 – Computation of SSE for a single region 144
Figure 4.11 – Visible image watermark 147
Figure 4.12 – Digital camera sends compressed image to computer for storage 148
Figure 4.13 a)-c) – Original-referencing watermark example 152
Figure 4.14 a)-e) – Self-referencing image watermark example 153
Figure 4.15 a)-c) – Original- and self-referencing image watermark examples 154
Figure 4.16 – Domains the image data is transformed into during compression 158
Figure 4.17 – Layout of watermark bit embedding locations 160
Figure 4.18 – Four largest coefficient magnitudes from a set of four zerotrees 162
Figure 4.19 a)-f) – Watermark and tamper detection with 14:1 compressed image 164
Figure 4.20 a)-c) – Increased watermark visibility with 43:1 compressed image 166
Figure 4.21 a)-d) – Original and high compression Goldhill and Barbara images 168
Figure 4.22 – Signal processing flow graph showing where IAW is added during image coding 169
Figure 4.23 – Signal processing flow graph showing processing required to add IAW 170
Figure 4.24 – Rotation of 96-bit watermark data 172
Figure 4.25 – Calculating modulo-8 checksum of four zerotrees and the watermark bit 172
Figure 4.26 – Insertion sort to find location of four largest coefficient magnitudes 174
Figure 4.27 – Application of the perturbation rule to correct the modulo-8 checksum 175
Figure 5.1 – Block diagram of prototype hardware for smart digital camera 181
Figure 5.2 – Photo of prototype hardware for smart digital camera 183
Figure 5.3 – Software architecture for prototype smart digital camera 184

Figure 5.4 – Visual Basic "Camera Viewer" user interface for prototype smart digital camera 185
Figure 5.5 – Visual Basic "Video Viewer" user interface for prototype smart digital camera 186
Figure 5.6 – Visual Basic display of statistics for prototype smart digital camera 187
Figure 5.7 – Motorola development environment for debugging the DSP 187
Figure 5.8 – Process timing diagram for prototype smart digital camera 190
Figure 5.9 – Photo of production hardware for smart digital camera 191
Figure 5.10 – Block diagram of production hardware for smart digital camera 192
Figure 5.11 – Photo of production hardware main circuit board for smart digital camera 193
Figure 5.12 – Photos of production hardware for smart digital camera 194
Figure 5.13 – Software architecture for production system 195
Figure 5.14 – C++ "Command Centre FT" user interface for production system 196
Figure 5.15 – Change in frame rate when an event triggers an archive sequence 198
Figure 5.16 – Process timing diagram for production smart digital camera 200
Figure 5.17 – Quantiser multiplier values used for the 10 SWT quality settings 204
Figure 5.18 – SWT algorithm performance against its peers on Lena image 206
Figure 5.19 – High, medium and low compression on Lena image 207
Figure 5.20 – SWT algorithm performance against its peers on Barbara image 208
Figure 5.21 – High, medium and low compression on Barbara image 209
Figure 5.22 – SWT algorithm performance against its peers on Goldhill image 210
Figure 5.23 – High, medium and low compression on Goldhill image 211
Figure 5.24 – Increase in RMSE error after IAW added to Lena image 216
Figure 5.25 – Visibility of the IAW in Lena image at high compression ratios 216

Publications

Papers 4, 5, 6 & 7 were published in internationally refereed conference proceedings.

1. **Body, N. B.,** *"Digital Camera for Low Bit-Rate Transmission of Slow Frame-Rate Video,"* Proceedings of the Third NZ Conference of Postgraduate Students in Engineering and Technology, University of Canterbury, Christchurch, New Zealand, pp.467-468, 1996. ISBN 0-473-03889-7.

2. **Body, N. B.; Page, W. H., and Hodgson, R. M.,** *"Application of the Wavelet Packet Transform to Very Low Bit Rate Coding of Images,"* The NZ Communication Research Workshop 1997, Wellington Town Hall, Wellington, New Zealand, 1997.

3. **Body, N. B.; Page, W. H.; Khan, J. Y., and Hodgson, R. M.,** *"Efficient Mapping of Image Compression Algorithms on a Modern Digital Signal Processor,"* Proceedings of the 4th Annual New Zealand Engineering and Technology Postgraduate Students Conference, University of Waikato, New Zealand, pp.59-64, 1997. ISBN 0-473-04578-8.

4. **Body, N. B.; Page, W. H.; Khan, J. Y., and Hodgson, R. M.,** *"Efficient Digital Signal Processor Implementation of a Wavelet Transform Based Image Compression Algorithm,"* Proceedings of Digital Image and Vision Computing: Techniques and Applications (DICTA/IVCNZ-97), Albany Campus, Massey University, Auckland, New Zealand, pp.71-76, 10-12 December, 1997. ISBN 0-473-04947-3.

5. **Body, N. B.; Page, W. H.; Khan, J. Y.; Hodgson, R. M., and Collins, F. A. H.,** *"Efficient Wavelet Image Coding on a Digital Signal Processor Based Digital Camera,"* Proceedings of International Conference on Signal Processing Applications & Technology (ICSPAT-98), Toronto, Canada, pp.782-786, 13-16 September, 1998.

6. **Body, N. B. and Bailey, D. G.,** *"Efficient Representation and Decoding of Static Huffman Code Tables in a Very Low Bit Rate Environment,"* Proceedings of IEEE International Conference on Image Processing (ICIP-98), Chicago, USA, pp.90-94, vol. 3, 4-7 October, 1998. ISBN 0-8186-8821-1.

7. **Body, N. B.; Page, W. H., and Hodgson, R. M.,** *"An Image Authentication Watermark for Wavelet Compressed Digital Images,"* Proceedings of Image and Vision Computing '99 New Zealand (IVCNZ-99), Canterbury University, Christchurch, New Zealand, August 30-31, 1999. ISBN 0-478-09333-0.

Glossary of Acronyms

ADC – Analogue to Digital Converter

AGU – Address Generation Unit

ALU – Arithmetic Logic Unit

AVS – Automated Video Surveillance

CCD – Charge Coupled Device

CCTV – Closed Circuit Television

CODEC – Coder-Decoder

CWT – Continuous Wavelet Transform

DCT – Discrete Cosine Transform

DFG – Data Flow Graph

DFT – Discrete Fourier Transform

DLL – Dynamic Link Library

DMA – Direct Memory Access

DPCM – Differential Pulse Code Modulation

DSP – Digital Signal Processor

DWT – Discrete Wavelet Transform

EZW – Embedded Zerotree Wavelet

FIR – Finite Impulse Response

GPP – General Purpose Processor

HVS – Human Visual System

IAW – Image Authentication Watermark

IIR – Infinite Impulse Response

JND – Just Noticeable Distortion

JPEG – Joint Photographic Expert Group

JTAG – Joint Test Action Group

KL – Karhunen-Loève

LAN – Local Area Network

LSB – Least Significant Bit

MMX – Multimedia Extensions

MOS – Mean Observer Score

MPEG – Motion Pictures Expert Group

MSE – Mean Square Error

PIR – Passive Infra-Red

PSNR – Peak-to-peak Signal to Noise Ratio

QMF – Quadrature Mirror Filter

RMSE – Root Mean Square Error

SAD – Sum of Absolute Differences

SBC – Sequential Baseline Coding

SCI – Synchronous Communications Interface

SIMD – Single Instruction, Multiple Data

SPIHT – Set Partitioning In Hierarchical Trees

SPL – Signal Processing Library

SRAM – Static Random Access Memory

SSE – Sum of Squared Error

STFT – Short-Time Fourier Transform

SWT – Symmetric Wavelet Transform

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

WSQ – Wavelet/Scalar Quantisation

1

Introduction

In days gone past men, women and children used to toil in coal mines to extract resources to the benefit of the privileged few. The working conditions were terrible and the workers suffered physically from their environment both due to the lack of sunlight and as a result of breathing coal dust for long periods. Looking back at such working conditions appals us today, yet at that time the work was seen as vital. Moving to the late twentieth century and the new millennium where no responsible government would condone such physical drudgery, we find many new kinds of drudgery in the workplace. One such example is the important job of protecting assets from thievery and destruction.

As companies have become more sophisticated and competition has become tougher, an organisation has a large amount of capital invested in its assets, comprised of both physical and intellectual property. The security personnel contracted by a company are responsible for protecting the company's assets. One tool that is available to security personnel is video surveillance. This can be used to centralise the role of the security personnel so that larger areas can be monitored with less manpower. The use of closed-circuit TV (CCTV) cameras in surveillance applications is growing more rapidly than almost any other sector of the security industry [1.1]. Unfortunately, for each new set of cameras installed comes the responsibility to employ someone to monitor the images from the cameras for up to 24 hours a day, every day. This activity entails a lot of mental drudgery; most of the time each camera is not witnessing any abnormal activity. The need for the operator to remain alert in what is essentially a very boring job is a major cause of the ineffectiveness of video

surveillance. The cost of maintaining video surveillance is mainly related to the ongoing operating costs and not the cost of the installation of the cameras.

It is likely that in 20 years time we will look back with disdain at much of the mental drudgery that occupied the workplace around the end of the twentieth century. As with coal mining, technology will evolve to relieve the physical burden from humans. Technologies will both eliminate mental drudgery from the workplace and make it more secure.

1.1 Scope of research

The research reported in this thesis represents early steps towards the goal of achieving sentient buildings. Such buildings would have an awareness of their local and internal environment and would be able to react to changes. Central to this goal is the equipping of buildings with sensors and actuators to allow sophisticated control systems control the changing environment.

Systems such as air conditioning systems use sensors to measure air temperature and humidity and use this information to control air quality. With building security, electronic systems are commonly used to control access throughout the building via identity cards with magnetic stripes or proximity cards. Video surveillance is valuable for validating alarm conditions and capturing evidential images of intruders. Low cost automated video surveillance has yet to be achieved, some of the work reported in this thesis represents a step towards this goal.

This thesis describes the development of a low cost system in which the surveillance function is carried out in the camera so that only information about scenes that contain prescribed movements need to be transmitted to the central surveillance control room. The system is completely digital, all transmitted events can be compressed, encrypted and archived to hard disk for later viewing. Most current commercial systems today rely on the existing installations of CCTV cameras and use expensive Automated Video Surveillance (AVS) systems to detect activity from analogue camera video

signals. The topology of these AVS systems for say an installation of 100 CCTV cameras requires the ability to process 100 analogue signals in real-time to isolate significant activity. This places a processing bottleneck in the central surveillance control room. Research into combining the surveillance function with image capture within the camera allows the processing to be distributed over the network of cameras and also reduces the net cost of the hardware. The cabling network that connects the cameras to the central control room is also simplified as a Local Area Network (LAN) can be used to connect the cameras rather than dedicated wide bandwidth analogue cable runs between each camera and the control room [1.2]. These fundamental differences in topology are illustrated in Figure 1.1.

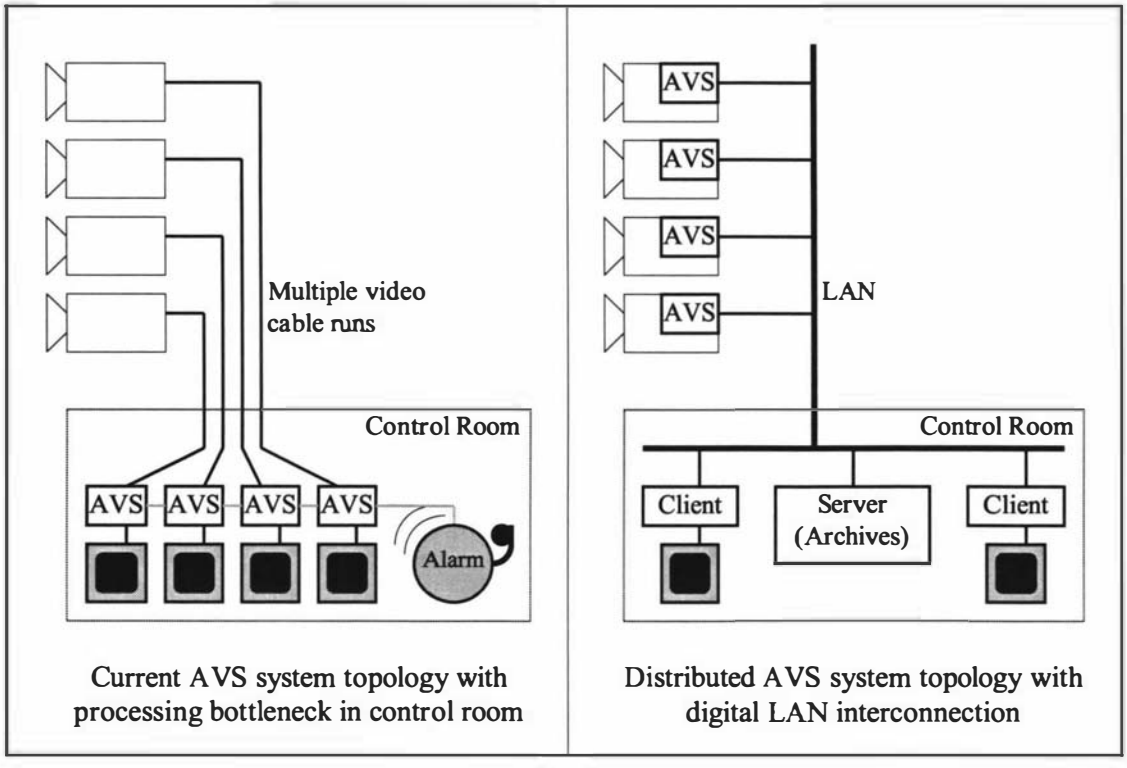


Figure 1.1 – Topology comparison between current and future AVS systems

1.2 Thesis overview

The theme of this thesis is that Digital Signal Processor (DSP) technology can be applied to image compression and image processing with considerable advantage.

This thesis describes research into the development of suitable algorithms and systems to achieve an automated surveillance function at the camera.

Such a system should have the following properties:

- Robust detection of significant motion.
- Rejection of false alarms caused by noise.
- Ability to communicate scene content information in a compressed format.
- Be easy to use.
- Run on inexpensive hardware.

The key achievements claimed for this work are:

- Real-time image compression at the camera.
- Efficient windowed motion detection within the wavelet domain to provide a robust indication of significant motion in the presence of noise.
- Co-operative Image Authentication Watermark (IAW) developed to protect the compressed image from tampering.
- Realisation of the above algorithms in a co-operative and highly efficient manner on an inexpensive DSP. Such a DSP is primarily designed for one-dimensional signal processing applications such as audio processing.

1.3 Contents by chapter

The review of literature and background material is found at the beginning of each major section in each chapter.

Chapter 2 contains details of the image compression elements of the research. Various algorithms are examined in light of the high compression and low computational cost required for this application. Compression ratios between 20:1 and 30:1 are targeted to enable an average of two compressed images to be transmitted from each camera every second. The compression algorithm must be able to be

executed on an 80 MHz DSP in real-time. This imposes a constraint on the computational complexity of the algorithm. The Symmetric Wavelet Transform (SWT) combined with Sequential Baseline Coding (SBC) of zerotrees is chosen as the best algorithm to be used. A suitable complete compression algorithm is then described in detail. The level of detail presented in this chapter may seem excessive but such algorithmic detail is essential when an efficient DSP based implementation is to be considered in the following chapter.

Chapter 3 examines the computational efficiency of the SWT based image compression algorithm for execution on a DSP architecture. DSPs are well suited to image compression and are practical as they are a low cost and highly integrated form of processor. The computational complexity is compared to the well-known JPEG image compression algorithm. The performance of implementing the SWT based decompression algorithm on a Pentium processor is also compared to the DSP implementation.

Chapter 4 discusses research into robust motion detection in the wavelet domain. There are inherent computational efficiencies in performing motion detection in the wavelet domain as the amount of data to be processed can be greatly reduced. Part of the transform data is low pass filtered. This reduces noise and enables more robust measures of significant motion. The chapter then examines techniques to ensure that images archived by the system can be authenticated to deter and detect tampering. Steganography is defined as the art of hiding messages inside images and the techniques involved are also related to digital image watermarking. A watermark can be visible or invisible and is used to provide a seal of authenticity to the image it is protecting. An Image Authentication Watermark (IAW) is developed that is compatible with the highly compressed nature of the image files. The technique enables the authentication information to become an integral part of the image and thus offers additional protection to the integrity of the image file. This is important as digital images are very easy to alter and steps need to be taken to prevent any tampering of the image files. The ability to use the wavelet transform to serve both the functions of image compression, motion detection and watermarking enables a more efficient system to be realised.

Chapter 5 discusses the complete system and its performance. Initially a prototype camera was constructed and this is described in terms of its hardware and software architecture. The prototype has recently been followed by a production smart digital camera that is being sold commercially as an integrated part of a well-established building access control and alarm management solution. The prototype and production systems are compared and the performance of each component along the image processing pipeline is measured. The objective quality of the SWT based compression algorithm is compared with other algorithms at various compression ratios. Finally the visibility of the IAW watermark is measured at various compression ratios.

Chapter 6 summaries the key points from the previous chapters and describes the important areas of research covered by this thesis. Conclusions are drawn and further areas of enhancement and research are listed.

References are contained at the end of each chapter and a comprehensive bibliography is included after chapter 6. The appendix contains three selected papers that were published during the course of this study. At the end of the appendix is a reprint of the "Cardax FT System Catalogue" brochure produced by Cardax International, the industrial partner to this research. This product brochure overviews the way the smart digital camera integrates with the complete Cardax FT access control and alarm management system.

1.4 References

[1.1] **McLeod, A.,** *"The impact and effectiveness of low-cost automated video surveillance systems,"* IEEE International Carnahan Conference on Security Technology, Lexington, Kentucky, USA, pp.204-211, 1996.

[1.2] **Meyer, M.; Hotter, M., and Ohmacht, T.,** *"A New System for Video-Based Detection of Moving Objects and its Integration into Digital Networks,"* IEEE International Carnahan Conference on Security Technology, Lexington, Kentucky, USA, pp.105-110, 1996.

2

Low Bit-rate Image Coding

Low bit-rate image coding is a specific type of image data compression technique. The compression ratio is high in low bit-rate image coding; at such high compression ratios image quality is a compromise trade-off against the bit-rate. This chapter will first look at image data compression techniques in general and then focus on techniques that give high compression ratios with minimal loss in image quality.

2.1 Image data compression techniques

Image data compression techniques involve reducing the number of bits required to represent an image. A digitised image is represented as a two-dimensional array of pixels. Each pixel is assigned a value to describe its intensity. Typical grey scale images use 256 shades of grey which corresponds to 8 bits of data per pixel. In most natural images individual pixels are not statistically independent of neighbouring pixels, this lack of pixel independence indicates that there is redundancy in such images. Image data compression techniques exploit this redundancy to allow a new image to be reconstructed in such a way that it ideally is visually the same as the source image, yet requires less information to be encoded for storage or transmission.

2.2 Lossless and lossy image compression

In lossless compression the following constraint is imposed on the image data compression technique used: any image redundancy encoding (data compression) performed on the source image must be completely reversible when reconstructing the image. This implies that the reconstructed image is **exactly** the same as the source image. In lossy data compression this constraint of exactness is relaxed to that of producing a visually **similar** image. The human visual system (HVS) can be modelled by parameters which describe how sensitive the system is to changes or errors in an image. In some image regions pixel values can be significantly changed without the change being noticeable by a human observer. The term "region" can be defined as an area of pixels within an image that have come from a similar statistical distribution. Regions with a low standard deviation are "uniform" and even small errors in such regions may be quite noticeable. Regions with a high standard deviation can conceal larger errors. The term just noticeable difference (JND) [2.1] is often used to describe the amount of change allowable in a lossy image compression system before a human observer detects the change. Lossy image compression systems often have a quality parameter that can be user specified. This parameter then limits the amount of compression that can be achieved. The highest compression ratio is obtained when the quality parameter is set to a low value, as there is a direct trade-off between compression ratio and image quality.

An indication of the visual degradation associated with a lossy image coding technique is given by the peak-to-peak signal to noise ratio (PSNR) difference between the original image $u(p,q)$ and the reconstructed image $\dot{u}(p,q)$ where the image size is given by $P \times Q$. The measure is dependent on N, the number of bits used to represent each pixel. PSNR measurement is performed in decibels (dB) and is given in equation (2.1).

$$\text{PSNR} = 10\log_{10}\left(\frac{(2^N-1)^2}{\sum_p\sum_q [u(p,q)-\dot{u}(p,q)]^2/(P\times Q)}\right) \tag{2.1}$$

Perfect reconstruction would result in a PSNR of ∞ dB but often visually lossless reconstruction can be obtained with a PSNR as low as 30 dB. This simple-to-calculate measure cannot be considered a general quality measure for images because it is very sensitive to image content. PSNR does not model the sensitivity of the human visual system to change and so the usefulness of the metric rapidly deteriorates with values less than 30 dB usually corresponding to severe distortion in the image.

A better form of image quality assessment is obtained through the Mean Observer Score (MOS) measure [2.2,3]. A five-point scale is used to measure the amount of visual impairment the compressed image suffers from when compared to the original image. The scale is defined as follows:

1. Very annoying
2. Annoying
3. Slightly annoying
4. Perceptible, but not annoying
5. Imperceptible

The MOS measure is derived from a large number of human observers who score a wide range of images at various compression ratios. Although this metric gives superior results in predicting the quality of compressed images, it is not machine computable as it is subjective and relies on human observation. Most of the literature only quotes PSNR values for image degradation as it is much simpler to calculate.

Current state of the art low bit-rate image coding techniques are invariably lossy due to the high compression ratio required. A quantiser is a function that is almost always present in a lossy compression system. The function maps a large domain of values onto a more restrictive set of range symbols. The function cannot be reversed exactly as there is a many-to-one relationship in the quantisation process; see Figure 2.1.

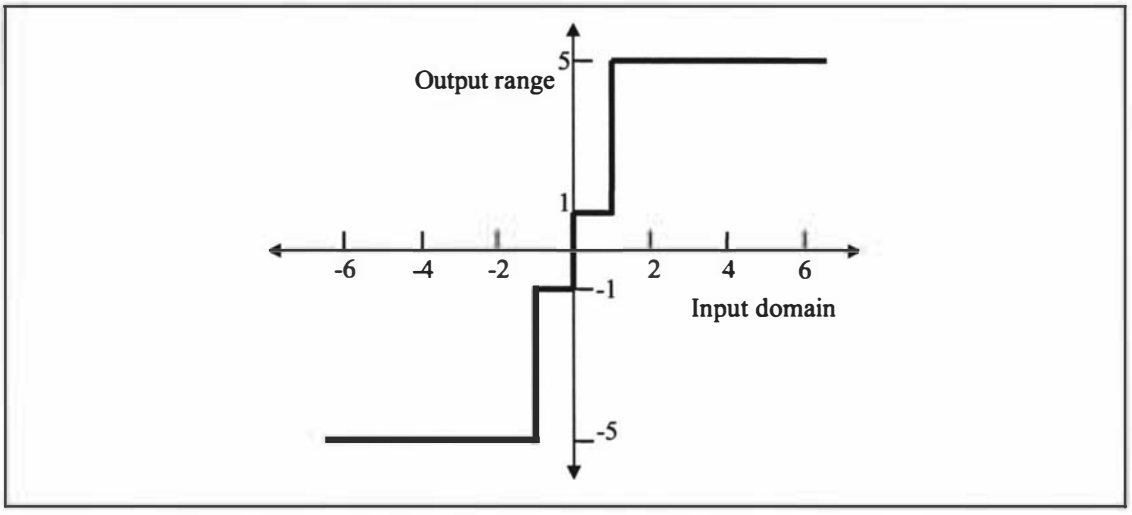


Figure 2.1 – Quantiser function, many-to-one relationship

There are two broad categories of image coding algorithms. Predictive coding and transform based coding. Additional compression can be achieved by constructing hybrid coders which use both techniques. The use of entropy coding techniques such as Huffman [2.4] or arithmetic [2.5] coding can be used to further reduce the required bit-rate after the image is coded by removing additional redundancy. A block diagram of a typical hybrid image coder utilising both predictive and transform coding is shown in Figure 2.2. The unprocessed image is a digitised image which has already been quantised in both space and amplitude. It is assumed that the losses associated with both the digitisation and visualisation processes, when the compressed image is displayed, are visually negligible.

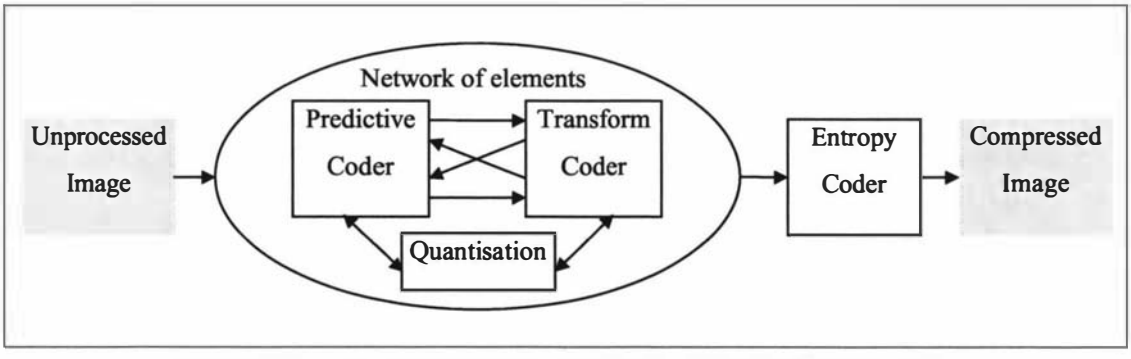


Figure 2.2 – Block diagram of a typical hybrid image coder

2.3 Predictive coding

Predictive techniques aim to remove mutual redundancy between successive pixels. If $u(1,\ldots,n)$ represents a sequence of pixels already coded and $\dot{u}(1,\ldots,n-1)$ represents a sequence of pixels already decoded, then the decoding process will calculate the next pixel $\dot{u}(n)$ based on the previous $n-1$ decoded pixels.

$$\bar{\dot{u}}(n) = \psi(\dot{u}(n-1), \dot{u}(n-2), \ldots) \tag{2.2}$$

Where $\psi()$ denotes the prediction rule. It is sufficient to code only the prediction error:

$$e(n) = u(n) - \bar{\dot{u}}(n) \tag{2.3}$$

If $\dot{e}(n)$ is the quantised value of $e(n)$, then the reproduced value of $u(n)$ is taken as:

$$\dot{u}(n) = \bar{\dot{u}}(n) + \dot{e}(n) \tag{2.4}$$

This coding process is called differential pulse code modulation (DPCM) [2.1] and continues recursively in this manner. Figure 2.3 shows a DPCM coder-decoder (codec). The coder has to calculate the reproduced sequence $\dot{u}(n)$ in order to model the behaviour of the decoder. This feedback stabilises the codec and prevents DC drift and accumulation of error in the reconstructed signal $\dot{u}(n)$.

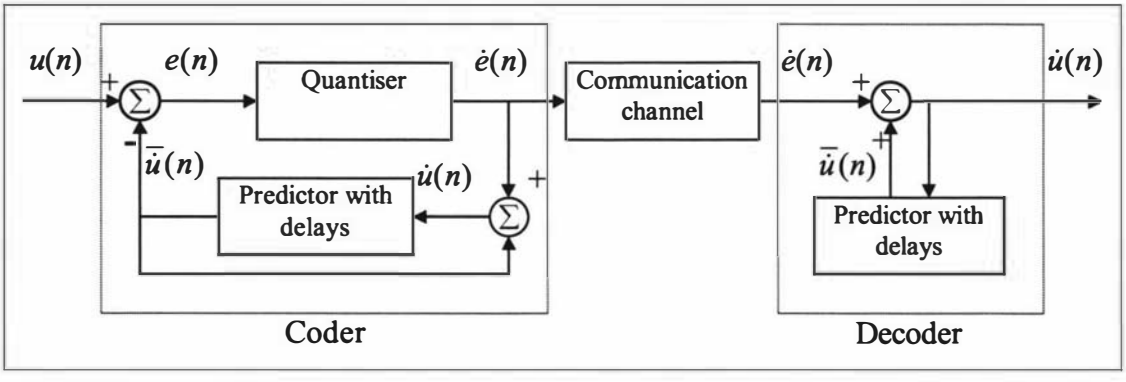


Figure 2.3 – Differential pulse code modulation codec

The advantage of using DPCM to encode a signal $u(n)$ is that the differential $e(n)$ signal has a lower mean square value than $u(n)$. This implies that the quantiser requires fewer levels to encode $e(n)$ with little or no loss of quality observed in practice. The use of fewer quantiser levels requires less bits to be used for each pixel and results in image compression occurring. For example, if only 8 levels, corresponding to 3 bits, are used to quantise the prediction error signal, then a compression ratio of 8:3 or 2.67:1 is achieved with generally little observed loss in picture quality [2.1].

2.3.1 One- and two-dimensional differential modulation

DPCM can be applied to multidimensional signals including images. In the simplest case, the two-dimensional image can be represented as a one-dimensional signal equivalent to scanning the image line by line. The prediction rule is often based on a function of the previous few pixels on the scanline. This prediction can be erroneous at the start of a new scanline if it is based on the last few pixels of the previous scanline thus exceptions to the prediction rule are often made at the start of each scanline. Equations (2.5) - (2.8) describe a one-dimensional DPCM system where the prediction is based on the p previous pixels.

Predictor:
$$\bar{\dot{u}}(n) = \sum_{k=1}^{p} a(k)\dot{u}(n-k) \tag{2.5}$$

Quantiser input:
$$e(n) = u(n) - \bar{\dot{u}}(n) \tag{2.6}$$

Quantiser output:
$$\dot{e}(n) = f(e(n)) \tag{2.7}$$

Reproduced output:
$$\begin{aligned} \dot{u}(n) &= \bar{\dot{u}}(n) + \dot{e}(n) \\ &= \sum_{k=1}^{p} a(k)\dot{u}(n-k) + \dot{e}(n) \end{aligned} \tag{2.8}$$

The one-dimensional case can be extended to two dimensions by defining the predictor over a two-dimensional area $\hat{W}_1$:

Predictor: $$\bar{u}(m,n) = \sum_{(k,l)\in\hat{W}_1} \sum a(k,l)\dot{u}(m-k,n-l) \qquad (2.9)$$

For row-by-row scanned images the four pixels shown in Figure 2.4 give a prediction for the current pixel marked with an "×". For the prediction to be causal, only pixels that have been previously scanned can be used for the prediction. For most images prediction of the current pixel does not improve appreciably by adding additional pixels to the prediction.

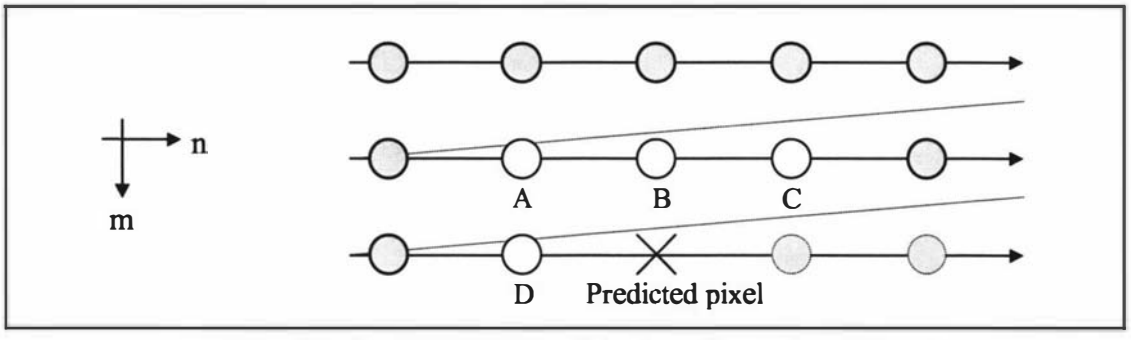


Figure 2.4 – Pixels (A,B,C,D) used in two-dimensional prediction

2.3.2 Interframe techniques

Predictive coding can be extended to three dimensions when coding a sequence of images. Each pixel is predicted from its immediate spatial neighbours and its value in the previous frame or group of frames. In most image sequences there are significant regions in the image that do not change from frame to frame, thus significant redundancy can be removed by basing predictions on the value of a pixel in the previous frame [2.1].

2.3.3 Vector quantisation

Vector quantisation divides the image into many small blocks of pixels and classifies each block using a codebook of templates. The address of the closest matching

template is transmitted to the decoder along with the block's mean and standard deviation to enable the decoder to scale the template to achieve a close match to the original block in the source image. The blocks are usually limited to 4 x 4 pixels to reduce the complexity of the template matching process as well as to reduce the size of the codebook required at both the encoder and decoder. Figure 2.5 shows a block diagram of a vector quantiser.

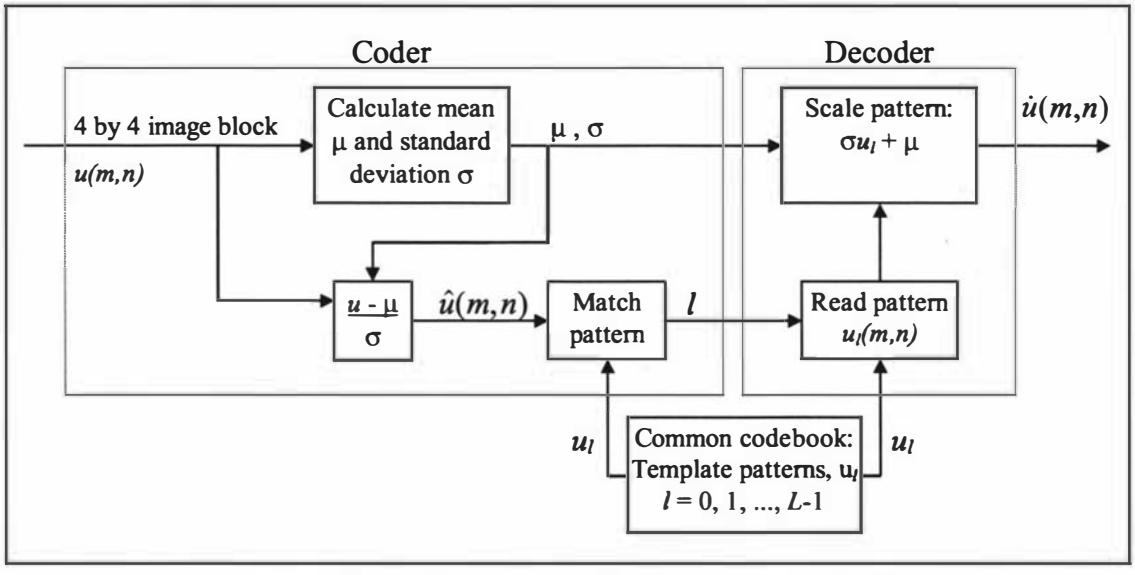


Figure 2.5 – Vector quantisation of image blocks

Vector quantisation achieves compression by using a variable number of bits to describe the address of the template in the codebook. Commonly occurring templates will have addresses that are assigned short codes whereas less common templates will have longer assigned address codes. This is a form of entropy encoding of the templates. If an image block does not closely match any entry in the codebook, then just the mean value of the block is transmitted and the block is reconstructed as a block of constant value. To generate the codebook, the vector quantisation system is trained with images that are typical of the set of images to be compressed. Thus a codebook for x-ray images would probably be quite different from a codebook for natural images. The need for specific training to achieve good compression prevents vector quantisation from being a universal technique for compressing all classes of images equally well.

2.4 Transform coding

Transform coding is an alternative to predictive coding. Practical implementations usually divide the image into 8 x 8 pixel blocks that are then coded. Some methods process the image as one large block of pixels. The block of data is unitarily transformed so that a large fraction of its total energy is packed in relatively few transform coefficients. A transform is called unitary if its transform matrix is unitary. This implies that the inverse of the matrix is equal to its conjugate transpose [2.1]. Important properties of unitary transforms are listed below [2.1]:

1. *Energy and Entropy Conservation.* A unitary transform preserves the signal energy and information content.
2. *Energy Compaction.* Most unitary transforms have a tendency to pack a large portion of the average energy of the image into relatively few components of the transform coefficients. As signal energy is conserved, this means that many of the transform coefficients will contain very little energy.
3. *Decorrelation.* For an input image that has a high degree of correlation between neighbouring pixels, the transform coefficients tend to be uncorrelated.

Once the image is transformed, each coefficient is quantised independently. To reconstruct the image, the quantised data is decoded and then an inverse transform is applied to the transform coefficients. This process is shown in Figure 2.6 for a one-dimensional transform coding.

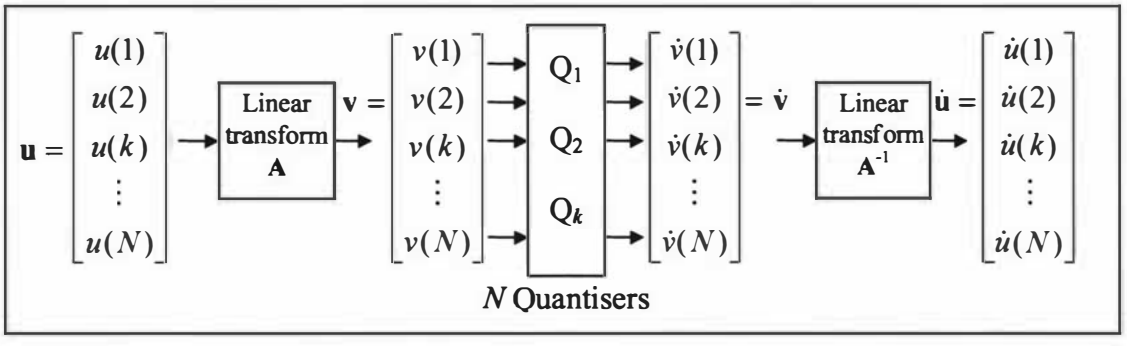


Figure 2.6 – One-dimensional transform coding

Mathematically, the one-dimensional transform coder is described in equations (2.10) - (2.12).

Forward transform: $\mathbf{v} = \mathbf{A}\mathbf{u}$ (2.10)

Quantisation: $\dot{\mathbf{v}} = Q(\mathbf{v})$ (2.11)

Inverse transform: $\dot{\mathbf{u}} = \mathbf{A}^{-1}\dot{\mathbf{v}}$ (2.12)

The transform matrix **A**, of size N x N, and the quantisation functions $Q_{1...N}$ need to be selected to minimise the mean square distortion of the reproduced data for a given number of total bits. The transform matrix **A** is also chosen according to its energy compaction ability. The Karhunen-Loève (KL) transform is optimal in the above sense [2.1] but is difficult to implement so sinusoidal transforms are usually used instead. The best fast unitary transform substitute for the KL transform is dependant on the type of images to be compressed. Table 2.1 shows a list of transforms and the situations where they work best [2.1].

Transform	Comments
Cosine	Performs best for highly correlated data typically found in natural images
Sine	For fast KL or recursive block coders
Discrete Fourier	Requires working in complex variables - this is useful for CT (computerised tomography) and MRI (magnetic resonance imaging) medical image data
Hadamard	Simple implementation for small block sizes such as 4 x 4 pixels
Haar	Emphasises high spatial frequencies
Slant	Best non-sinusoidal fast transform

Table 2.1 – KL transform substitutes

Data compression is achieved because of the energy compaction property of the transform used; many of the transform coefficients in **v** will be close to zero and only a few coefficients will have significant values. The quantisers discard small coefficients below a threshold whilst coding the significant coefficients with minimal

loss. This results in a bitstream in which most of the bits are used to code the value and location of the more significant coefficients.

2.4.1 Two- and three-dimensional image transforms

Virtually all transform based image coders process image data in two or three dimensions depending on whether a single image or sequence of images are to be compressed. Table 2.2 gives the typical compression ratios for images compressed as one-, two- and three-dimensional sets of data [2.1]. The compression ratios listed are to achieve PSNRs in the 30-36 dB range which correspond to near visually lossless coding. Adaptive methods are mentioned in the table and these will be expanded upon in section 2.5.

Method	Typical compression ratios for images
One-dimensional	2-4
Two-dimensional	4-8
Two-dimensional adaptive	8-16
Three-dimensional	8-16
Three-dimensional adaptive	16-32

Table 2.2 – Overall performance of transform coding methods

The one-dimensional transform coding can be easily generalised to two dimensions by mapping a given N x M image $u(m,n)$ to a one-dimensional NM x 1 vector **u**. The KL transform of **u** would require a matrix **A** of size NM x NM which is prohibitively large for images sizes of 512 x 512 pixels. In practice, this transform is replaced by a separable fast transform such as the Cosine, Sine, Fourier or Hadamard. These transforms are capable of packing a considerable amount of energy in a small number of coefficients.

2.4.2 Block based and global based

To make transform coding practical, a given image is divided into small blocks and each block is transform coded independently. If each block is of size p x q pixels then an N x M image is divided into NM/pq blocks. This reduces the main storage requirement for implementing the transform by a factor of NM/pq. The computational load is reduced by a factor of $\log_2 NM/\log_2 pq$ for a typical fast transform requiring $\alpha N \log_2 N$ operations to transform an N x 1 vector [2.1]. For an image of 512 x 512 pixels divided in blocks of 8 x 8 pixels, the storage requirement for the transform is reduced by a factor of 4096 and the computational load is reduced by a factor of 3. Although the operation count is not greatly reduced, the complexity of the hardware for implementing small-size transforms is reduced significantly. Smaller block sizes do reduce the compression ratios achievable for a constant amount of distortion. Block sizes between 16 x 16 and 32 x 32 pixels provide significant improvements to the compression achievable with an 8 x 8 pixel block yet for historical reasons, the popular JPEG image compression algorithm standard uses an 8 x 8 pixel block size. The JPEG algorithm was standardised by the Joint Photographic Expert Group in 1991 at a time where the available hardware favoured the computational complexity of an 8 x 8 pixel transform.

2.4.3 Coefficient coding strategies

Once the image has been transformed from a set of pixels into a set of transform coefficients, several strategies are available to efficiently code the quantised coefficients. The transform coefficients prior to quantisation typically have a larger dynamic range than the original range used to describe the pixel intensity values. The quantisation functions ensure that the quantised data is well constrained to a small number of integer step indices that can be further coded. The quantisation process also causes many close-to-zero transform coefficients to be rounded to zero. The following sections describe strategies that take advantage of the fact that many of the quantised coefficients are rounded to zero.

2.4.3.1 Zonal coding

Zonal coding uses a zonal mask to predetermine a subset of the coefficients that will be transmitted to the decoder. In block based coding, an array of 8 x 8 pixels is transformed and quantised into an array of 8 x 8 quantised coefficients. Most of the energy is packed into the DC coefficient in the top left corner element of the array. The rest of the energy is mainly found in the low frequency AC coefficients which surround the DC coefficient in the top left corner of the array. By creating a zonal mask that only transmits the coefficients in the top left corner of the array, most of the energy can be transmitted in just a few of the total number of coefficients but with some loss in image quality due to the truncation of high frequency content. Figure 2.7 shows the steps involved in zonal coding the quantised transform coefficients for a 4 x 4 block of pixels.

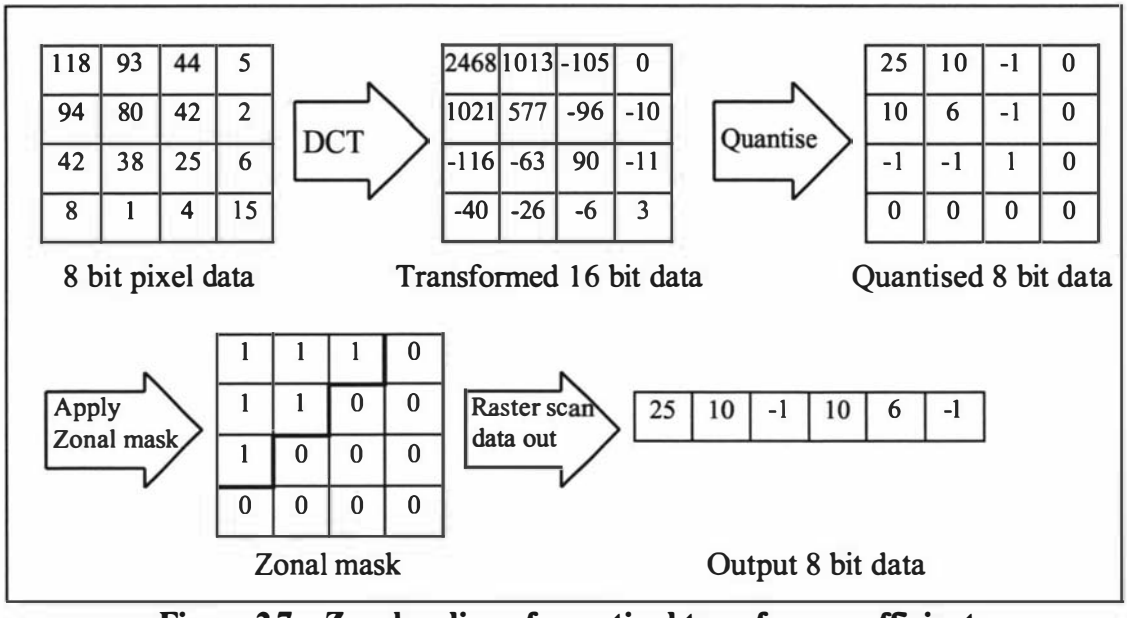


Figure 2.7 – Zonal coding of quantised transform coefficients

2.4.3.2 Threshold coding

Threshold coding is similar to zonal coding except that the zonal mask may vary from block to block. The mask is defined by choosing a suitable threshold value η so that only coefficients that have an absolute value greater than η are transmitted. This

method is more adaptive than zonal coding and will produce less distortion for the same number of transmitted coefficients. The disadvantage is that the threshold mask needs to be transmitted for each block as an overhead. This overhead is shown in Figure 2.8 as a set of addresses for each of the transmitted data points.

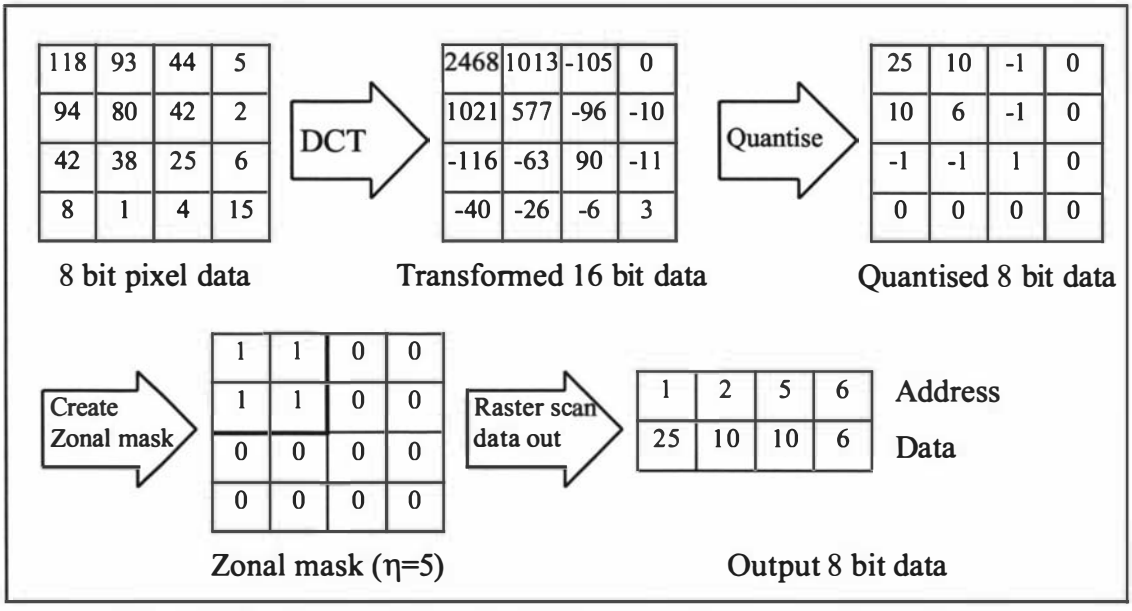


Figure 2.8 – Threshold coding of quantised transform coefficients

2.4.3.3 Run-length coding

Run-length coding when applied to the threshold coding allows the data values and effective addresses of the data values to be transmitted in an interleaved fashion [2.1]. A scanning order needs to be defined that will favour long runs of zero symbols and a zig-zag scanning order is often used for this purpose. An escape symbol is used to signify the presence of a run of two or more zero symbols (**RUN**) and the escape symbol is followed by a count. An additional escape symbol called "zero-to-end" (or **Z2E**) may be used to signify that there are no more non-zero symbols left in the block. Figure 2.9 shows how run-length coding can be applied to a threshold coded block of data to efficiently encode the locations of the non-zero data.

An alternative to the zig-zag scanning order is to scan along the Hilbert curve [2.6]. This scanning order is not used in JPEG but it does consider the closeness of pixels in

many directions as opposed to the diagonal direction only considered in zig-zag scanning. A Hilbert curve can be constructed iteratively from a 2 x 2 block due to its fractal nature and it is shown for an 8 x 8 block alongside the zig-zag scan in Figure 2.10.

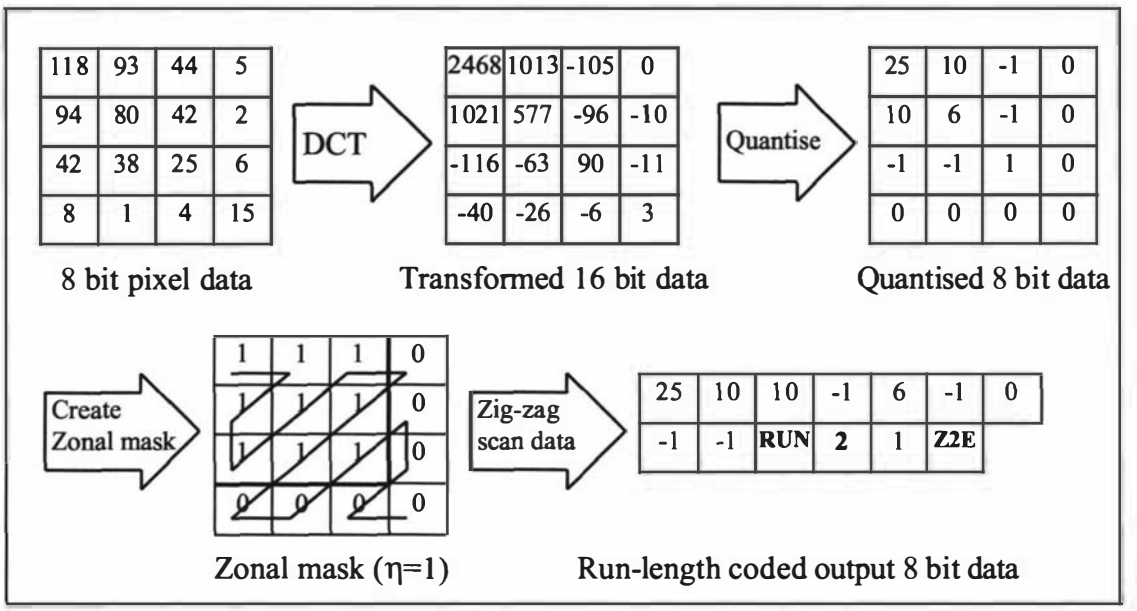


Figure 2.9 – Run-length coding of quantised transform coefficients

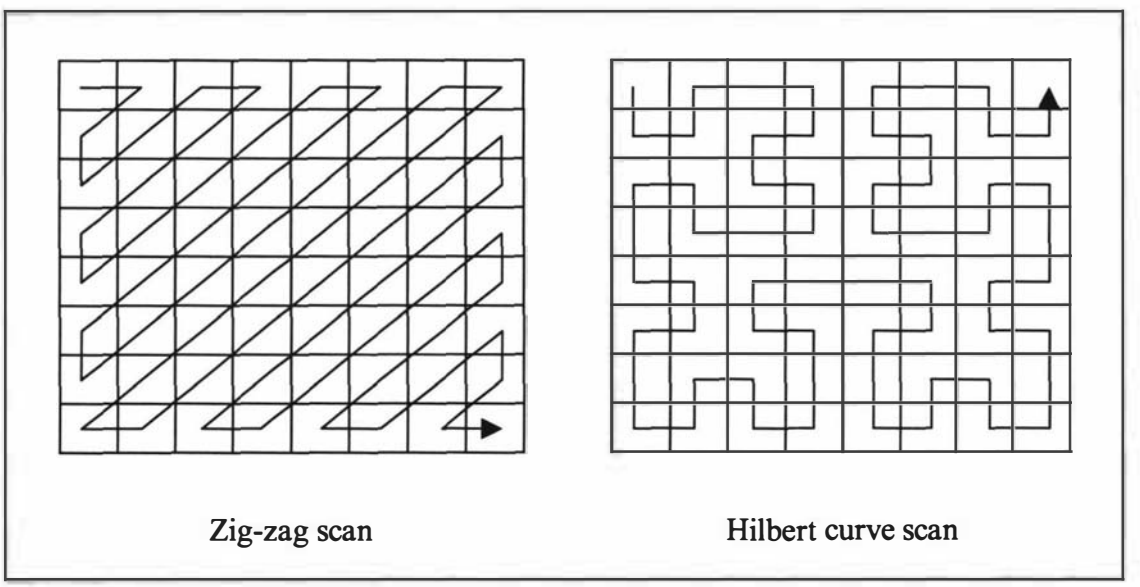


Figure 2.10 – Zig-zag and Hilbert curve scanning order for an 8 x 8 block

2.5 Adaptive methods

The image coding methods covered in the previous sections can be described as non-adaptive algorithms which rely on set processes and tables to control the compression process. This has the consequence that independent of image content, the compression algorithm will go through the same steps. However, the actual compression ratio achieved by an image compression scheme is sensitive to image content. An image that contains a lot of high frequency information, which may for example be in the form of highly textured regions within the image, is considered to be visually complex. Visual complexity can be estimated from the width of the autocorrelation function of an image [2.7]. Predictive coding of such an image will produce large prediction errors and will require more bits at the entropy encoder. Correspondingly in a transform based algorithm there will be more of the total energy spread throughout the high frequency coefficients. This change in energy distribution will cause poor run-length coding due to the scattered nature of the coefficients and more bits will be required to code each block. In both cases a visually complex image will produce more data to be coded than a simple image without a lot of texture or detail.

Adaptive algorithms are usually designed as extensions to otherwise non-adaptive algorithms. The main goal in adaptive coding is to improve the visual quality of an image at the same or lower bit-rate when compared to the non-adaptive coding. This is accomplished by adjusting the algorithm as the image is being coded to take advantage of the changing image content. Measurement of local image spatial activity or frequency content can be used to control the adaptation so that the current optimal set of parameters is selected.

2.5.1 Adaptive predictive coding

In a simple predictive coder the quantiser function and the predictor rule are both fixed. An adaptive predictive coder can adapt to image content by adjusting the

quantiser and predictor to account for change in image content. The adjustment can be signalled to the decoder without overhead if only the previously coded pixels are used to determine the mode of operation. The main improvement that can be made with predictive coders is the subjective quality of the image especially at edges [2.1].

2.5.1.1 Predictor rule

By sensing to the orientation of an edge, a more suitable predictor rule can be used to predict the pixels surrounding the edge. There will be a computational cost associated with choosing the appropriate predictor rule so usually only simple measurements are made to establish the rule.

2.5.1.2 Quantiser function

Over the image the variance of the prediction error fluctuates with change in spatial activity. One simple adaptation is to use adaptive gain control to normalise the variance of the prediction error prior to quantisation. A more complex technique called adaptive classification can be used to segment the image regions into activity classes. This allows flat regions in the image to be quantised finely and coarser quantisation to be used around edges and textured regions. This adaptation takes advantage of the human visual system's decrease in sensitivity to noise with increased activity [2.1]. The classification scheme may require a small overhead to be transmitted to the decoder to represent to classification of each image block.

2.5.2 Adaptive transform coding

The three main features of a transform based coding that can be adapted are the transform itself, the quantiser function and the allocation of bits over the set of blocks (assuming a block based coder), in the image. The basis vectors for the image transform can be selected from a set of predefined transforms to better suit the

statistical properties of the block being transformed. The quantiser function can be adapted to image content in the same way as discussed above for a predictive coder. The bit allocation for each image block can also be varied in response to spatial activity to give a better bit utilisation over the entire ensemble of image blocks.

As the transform component of the image compression algorithm is often the most computationally intensive part, sometimes the accuracy and hence speed of the transform may be varied by selecting between alternative algorithms. In the paper by Girod [2.8], the observation is made that most of the quantised transform coefficients are zero for the majority of the blocks in a DCT based image coder. The adaptive implementation proposed by him is to classify a block prior to transform coding with a simple activity measure and then use one of a set of DCT algorithms that vary in accuracy and speed. This form of adaptation is used to achieve an overall increase in algorithm speed rather than an increase in subjective quality for a given bit-rate.

2.5.3 Adaptive coding summary

The advantages of adaptive algorithms are:

1. Adaptive algorithms exploit the non-stationary nature of images and image streams by flexibly adapting to each image presented to the algorithm.
2. Bit allocation is varied across the image so that more bits are spent encoding areas of the image that are visually significant.
3. High level human vision system heuristics can be used to create a visually pleasing image at low bit-rates [2.9].

Disadvantages of adaptive algorithms are:

1. Increase in algorithm complexity. Adaptive techniques add a greater computational cost as measurement and classification operations are required to control the adaptation.

2. Additional information may be needed to signal the decoder that a parameter has been adaptively modified. The decoder does not have access to the source data in the raw image which may have prompted a mode change in the encoder. The adapted parameter will need to be transmitted as an overhead to the decoder which may nullify the benefits of adaptation where bit-rate conservation is important.

In summary, most of the common accepted still image compression standards such as JPEG do not contain adaptive techniques. When coding sequences of images it is common to use a simple form of global adaptation in order to create a near constant bit-rate. If the complexity of the scene changes over a sequence of frames then a global compression quality parameter may be altered over time to attempt to keep the bit-rate constant. It is important to maintain a constant bit-rate when transmitting sequences of images in real-time applications otherwise frames may need to be dropped from the sequence when the available fixed bandwidth is exceeded.

Fourier Transform (STFT) and it represents a compromise between time based and frequency based analysis of the signal [2.11]. The size of the time window limits the precision of the analysis obtainable. If the window covers the entire signal the analysis reverts back to the Fourier transform of the signal where there is no time information available. At the other extreme, a very narrow time window reveals no information about the frequency content of the signal. In many applications a sensible compromise can be reached where adequate precision can be allocated to both time and frequency. The disadvantage of STFT is that for a particular window size, the grid of frequency and time precision is fixed over all frequencies. A time-frequency rectangular grid is shown in Figure 2.12 where each window consists of 40 samples. In this example the signal being analysed is 400 samples long producing 10 separate windows in time over the duration of the signal. Only the amplitude of the complex valued Fourier transform is graphed in Figure 2.12 and it is represented by the intensity of each rectangular grid location.

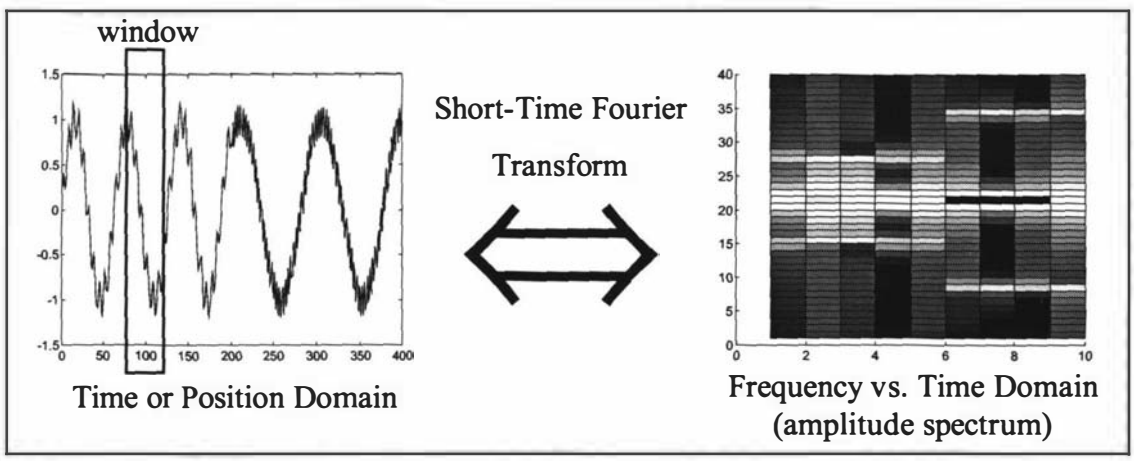


Figure 2.12 – Short-Time Fourier analysis of a one-dimensional signal

2.6.1.3 Wavelet analysis

Wavelet analysis can be performed by passing a signal though a set of filters called a filter bank [2.12]. The analysis filter bank often has two filters, low-pass and high-pass. Each filtering operation separates the signal into two frequency bands called subbands. Wavelet analysis allows a different window size to be used with each

frequency subband being analysed. This variation in window size provides a superior time/frequency trade-off to be made when compared with STFT analysis. The uncertainty principle [2.13] only allows the time and frequency content of a signal to be measured to a certain precision. As the precision of the measured signal frequency is increased (by using longer time intervals), the precision of time related trends in the signal decreases. Figure 2.13 shows the different analysis grids used to trade-off frequency and time precision for the various techniques considered. The time domain analysis shown in **a)** is related to Shannon's sampling theorem and offers no information about frequency content. The frequency domain analysis shown in **b)** uses the Fourier transform of the entire signal and offers no information on how the signal changes over time. The STFT shown in **c)** and wavelet analysis technique shown in **d)** illustrate a tiling effect where the STFT tiles are uniform in shape and the wavelet tiles vary in shape. The uncertainty principle forces a trade-off to be made between the precision of measured time and frequency in a signal. As the precision of frequency measurement increases (by using a longer time window), less precision is available for measuring time related trends in the signal. The wavelet windowing technique uses long time regions when analysing for low frequency content and short time regions when analysing for high frequency content. The term "scale" is used instead of "frequency" when dealing with the wavelet transform. Each step in the scale represents a doubling or halving of the band of frequencies represented [2.11].

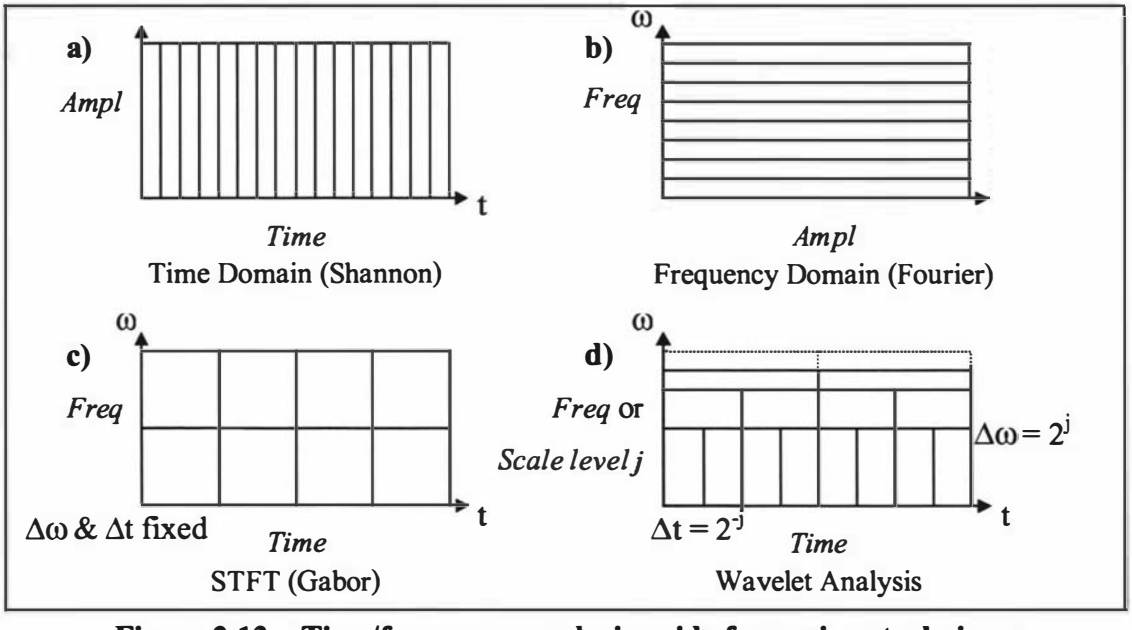


Figure 2.13 – Time/frequency analysis grids for various techniques

The wavelet analysis considered above is the Discrete Wavelet Transform (DWT). The Continuous Wavelet Transform (CWT) is not considered in this thesis as it produces a highly over-determined set of coefficients by analysing the input signal at every possible scale and every possible position. The CWT is useful in analysing signals to extract specific information from a signal but is less useful in the field of signal data compression. If the choice of scales and positions are based on powers of two, often called *dyadic* scales and positions, the CWT reduces to the non-redundant DWT [2.11]. The relationship between the CWT and the DWT is analogous to the relationship between the Fourier transform and the discrete Fourier transform [2.13]. Figure 2.13 **d)** shows how the time and frequency bandwidths of each of the tiles are related by powers of two based on a scale parameter j.

For the purposes of transforming the signal to another domain to enable energy compaction and data compression, the discrete version of the WT is most suitable. The DWT produces a non-redundant set of output coefficients that can then be encoded. The DWT achieves good energy compaction due to the compact support property of most of the wavelet filter banks [2.11].

2.6.2 Wavelet transform of one-dimensional signals

The following informal discussion will show how the wavelet transform is performed and what the wavelet analysis of a one-dimensional signal can reveal. The one-dimensional signal to be analysed is denoted by s(t). The signal is sampled 500 times over its duration to produce S(n), $n \in 0 \ldots 499$. The signal S(n) is shown in Figure 2.14, at this sampling density the signal appears to be continuous.

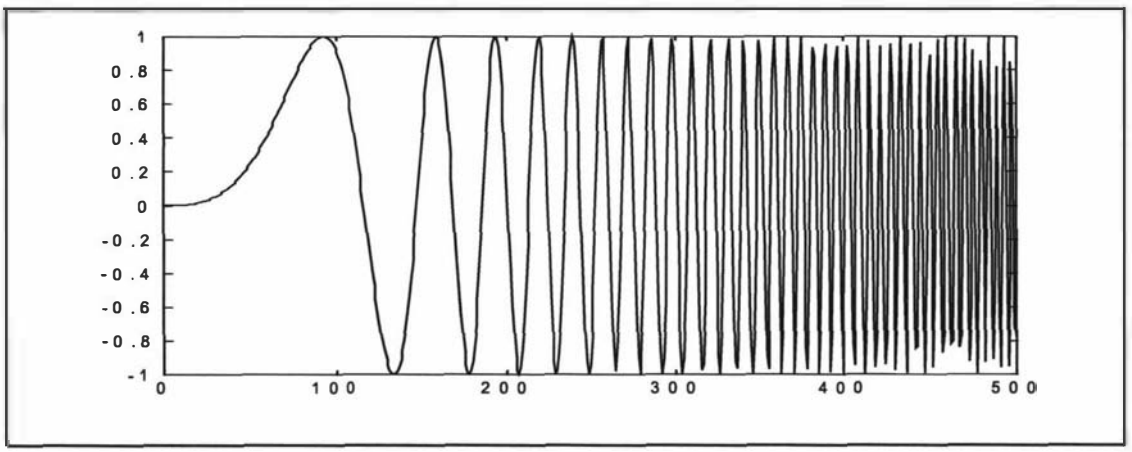


Figure 2.14 – Sampled signal S(n)

A single level wavelet decomposition of the signal S(n) involves filtering the signal using a quadrature mirror filter (QMF) which consists of a low-pass scaling function $\Phi(n)$ and a high-pass wavelet function $\Psi(n)$. The Daubechies family of wavelets is commonly used for signal analysis due to its compact support [2.11] and the Daubechies-4 (db4) wavelet is shown in Figure 2.15. The Daubechies family of wavelets is one of many wavelet families that have been designed by researchers. The decomposition low-pass and high-pass filters each have 8 coefficients for the db4 wavelet.

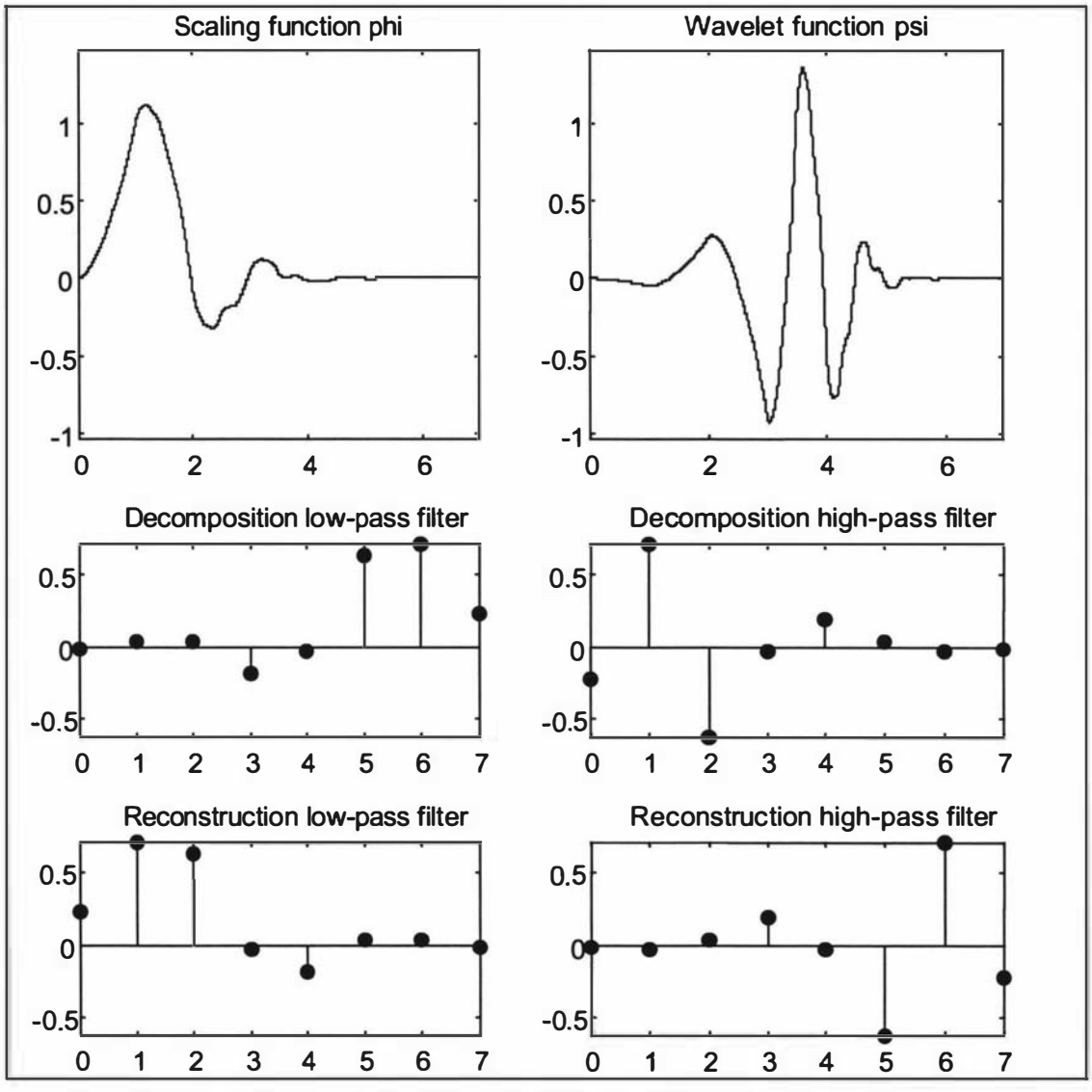


Figure 2.15 – Daubechies-4 (db4) wavelet filters

These filter coefficients are convolved with the 500 sample points in S(n) to separate the low frequency components of the signal into $cA_1(n)$ (approximation coefficients at level 1) and the high frequency components into $cD_1(n)$ (detail coefficients at level 1). The two output signals $cA_1(n)$ and $cD_1(n)$ can be downsampled by a factor of two as they each represent only half of the original bandwidth of the signal S(n). Downsampling is accomplished by removing every second sample. The process of decomposition is shown in Figure 2.16. Figure 2.17 shows a composite plot of S(n) with the downsampled signals $cA_1(n)$ and $cD_1(n)$ after a single level decomposition.

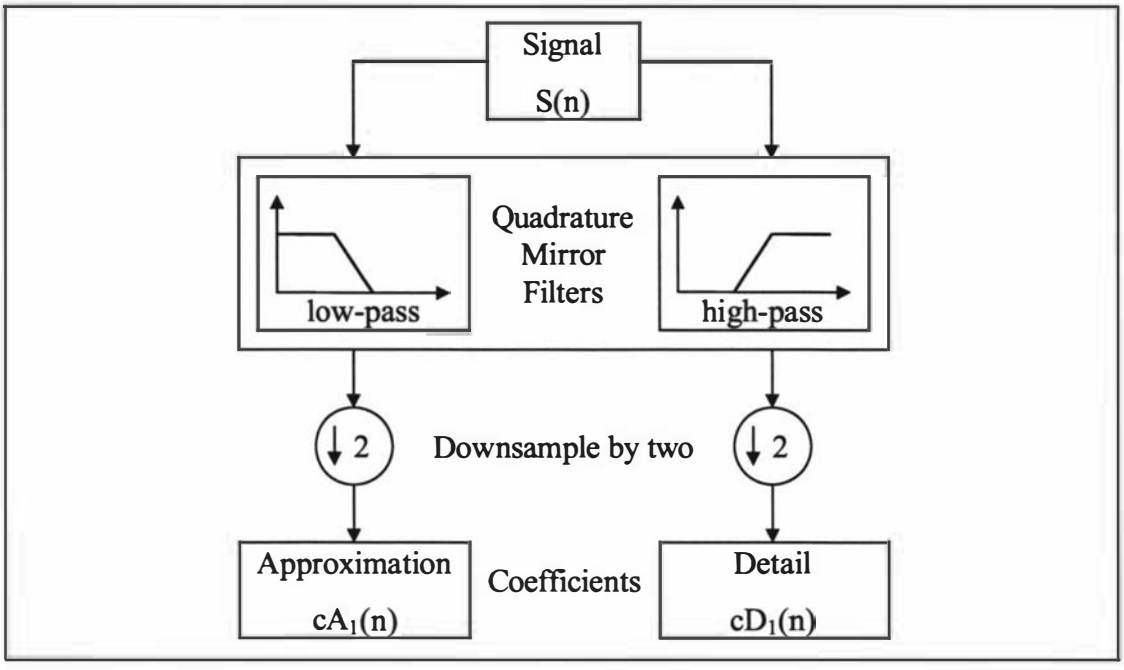


Figure 2.16 – Structure of a single level wavelet decomposition

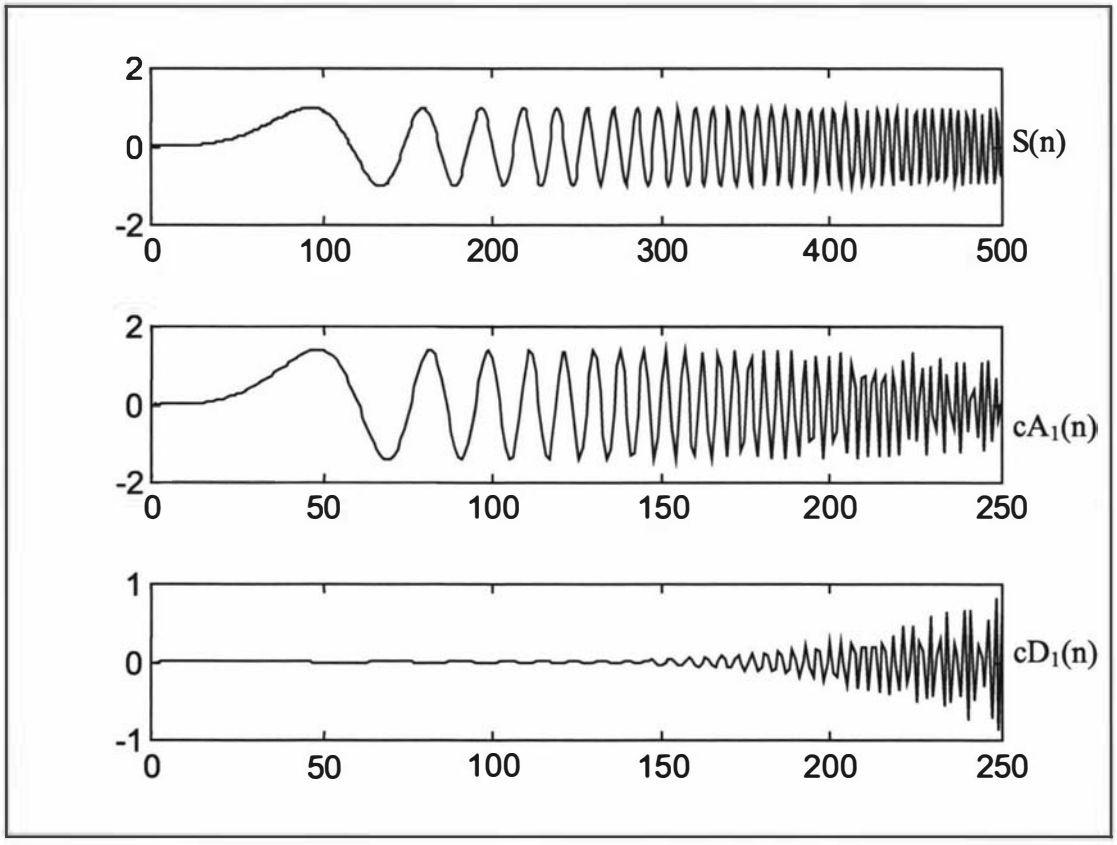


Figure 2.17 – First level decomposition of S(n) with db4 wavelet

The results of the first level of decomposition of signal S(n) reveals that most of the energy in the signal is restricted to the lower half of the sampled bandwidth which is

contained in $cA_1(n)$. There is only significant high frequency content in the last 20% of the duration of the signal which is contained in $cD_1(n)$. With a single level of decomposition, the signal frequency spectrum has only been analysed into two frequency subbands, 0 Hz to half the Nyquist frequency in $cA_1(n)$ and half the Nyquist frequency to the Nyquist frequency in $cD_1(n)$. While the lack of frequency discrimination may appear to be a disadvantage, there are 250 estimates in time across the duration of S(n) of the energy contained in each of the two subbands. This example shows that in the first level of the wavelet decomposition, a low resolution in frequency has been traded-off against a high resolution in time.

Wavelet analysis is an *iterative* technique and the standard wavelet decomposition algorithm decomposes the previous level 1 $cA_1(n)$ signal into a new pair of level 2 $cA_2(n)$ and $cD_2(n)$ signals. This process is continued until the desired number of levels of analysis has been reached by decomposing one or both of the output signals at each level. The ideal form of decomposition and number of levels is signal dependant. Most real world natural images are typically low-pass in nature and the dyadic DWT at four to five levels is often used. Each level of the transform covers an octave of signal bandwidth and only the approximation signal is decomposed at each level.

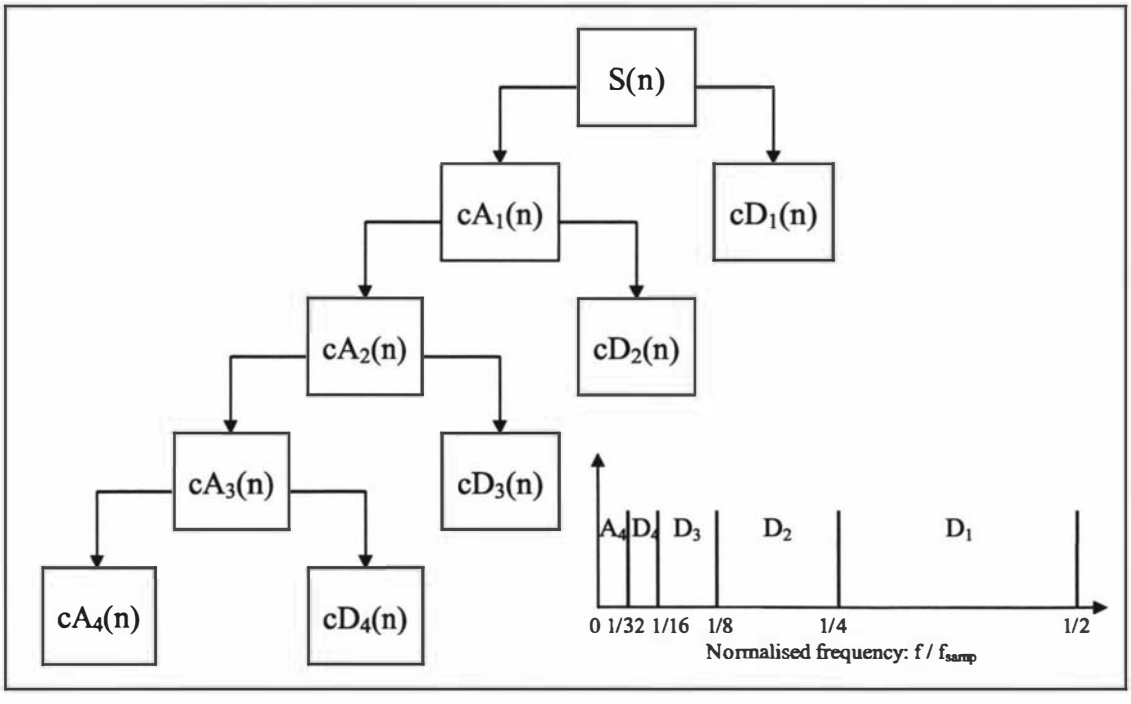


Figure 2.18 – Four level decomposition of S(n) with db4 wavelet and the resulting leaf node octave bins

Figure 2.18 shows the hierarchy of coefficient signals produced in a 4 level decomposition of the signal S(n). The regions marked on the frequency scale in Figure 2.18 show what spectrum portion of the signal is contained in each of the coefficient signals. Figure 2.19 shows the individual coefficient signal components of signal S(n) decomposed to four levels rather than just the single level decomposition shown in Figure 2.17.

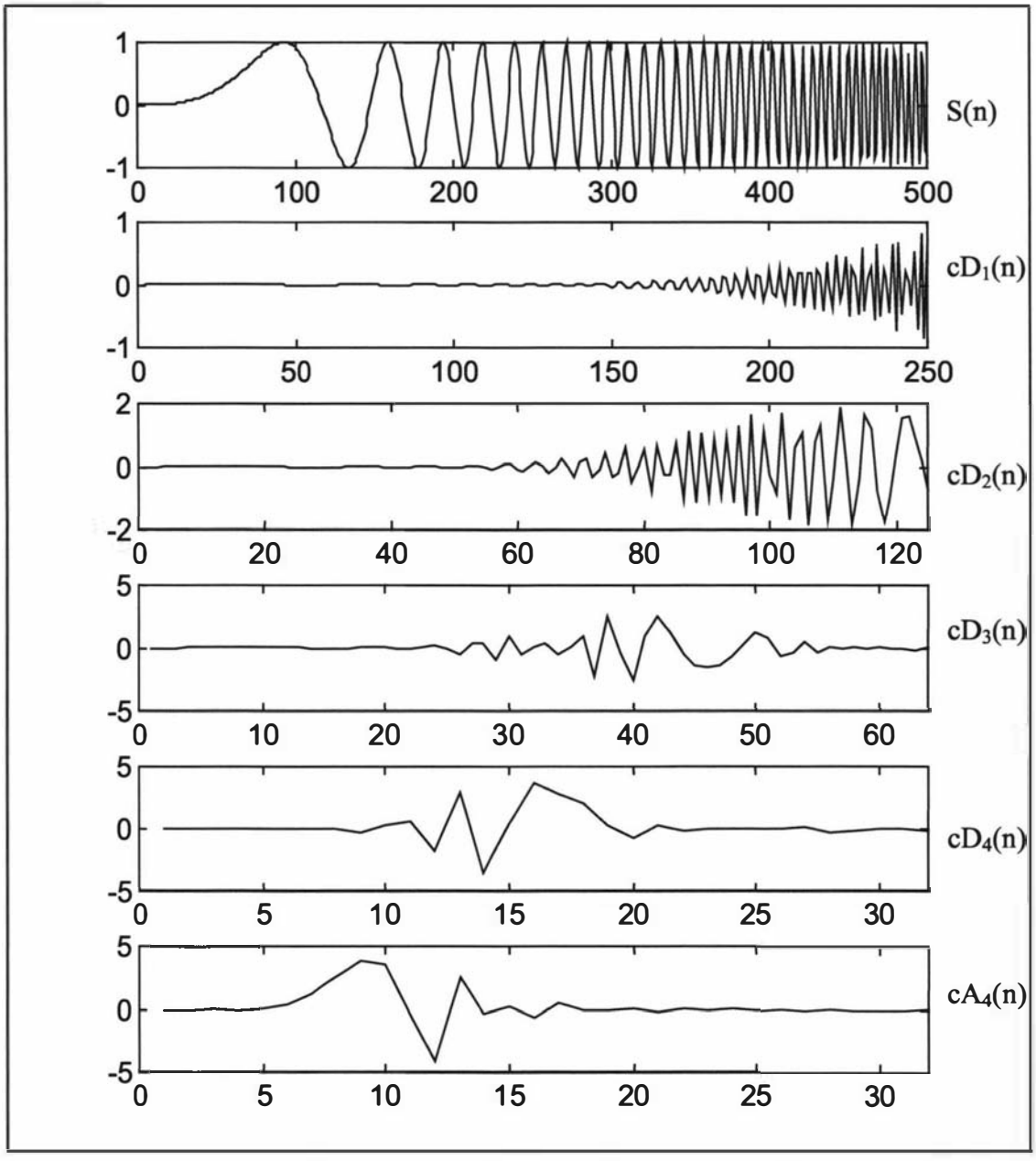


Figure 2.19 – Coefficient signal components of the four level decomposition of S(n)

The analysis of S(n) at four decomposition levels shows at a high time resolution the energy for the high frequency range contained in $cD_1(n)$. As the frequency range being analysed is reduced by factors of two corresponding to $cD_2(n)$ to $cD_4(n)$, the time resolution also is reduced by a factor of two for each level. The horizontal axis in each of the graphs in Figure 2.19 shows the number of points measured in time decreasing with each level. Coefficient signals $cD_1(n)$ to $cD_4(n)$ each reveal the energy distribution as a function of time for the signal S(n) in each subband. The final coefficient signal $cA_4(n)$ represents the DC and lowest analysed octave of the signal. It is known as the *approximation signal* as it approximates the overall shape of the original signal at a lower time resolution.

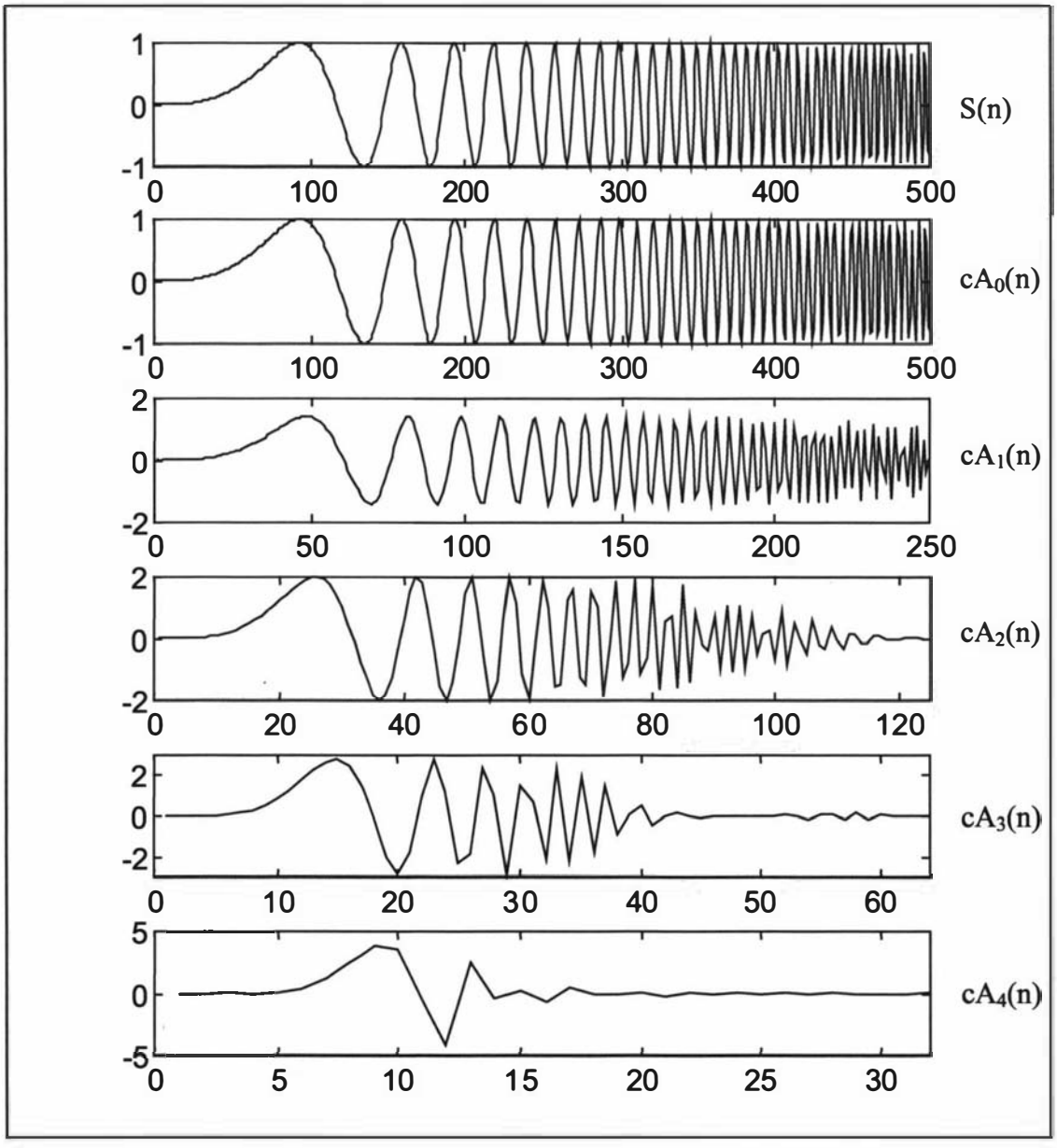


Figure 2.20 – Reconstruction of successive approximations of S(n)

An important property of the wavelet decomposition "tree" shown in Figure 2.18 is that the coefficients within the signals represented by $cD_1(n)$, $cD_2(n)$, $cD_3(n)$, $cD_4(n)$ and $cA_4(n)$ are all that is required to perfectly reconstruct the original signal S(n) [2.11]. These five signals are at the leaf nodes in the decomposition tree. The additional approximation coefficient signals $cA_1(n)$, $cA_2(n)$ and $cA_3(n)$ are located at the internal nodes of the tree. They can be recreated from the leaf node signals. The coefficient signals $cA_4(n)$, $cA_3(n)$, $cA_2(n)$ and $cA_1(n)$ are successive approximations of the original signal S(n) as shown in Figure 2.20. When the coefficient signals $cA_1(n)$ and $cD_1(n)$ are combined in the reconstruction process, the coefficient signal $cA_0(n)$ is created which is identical to the original signal S(n).

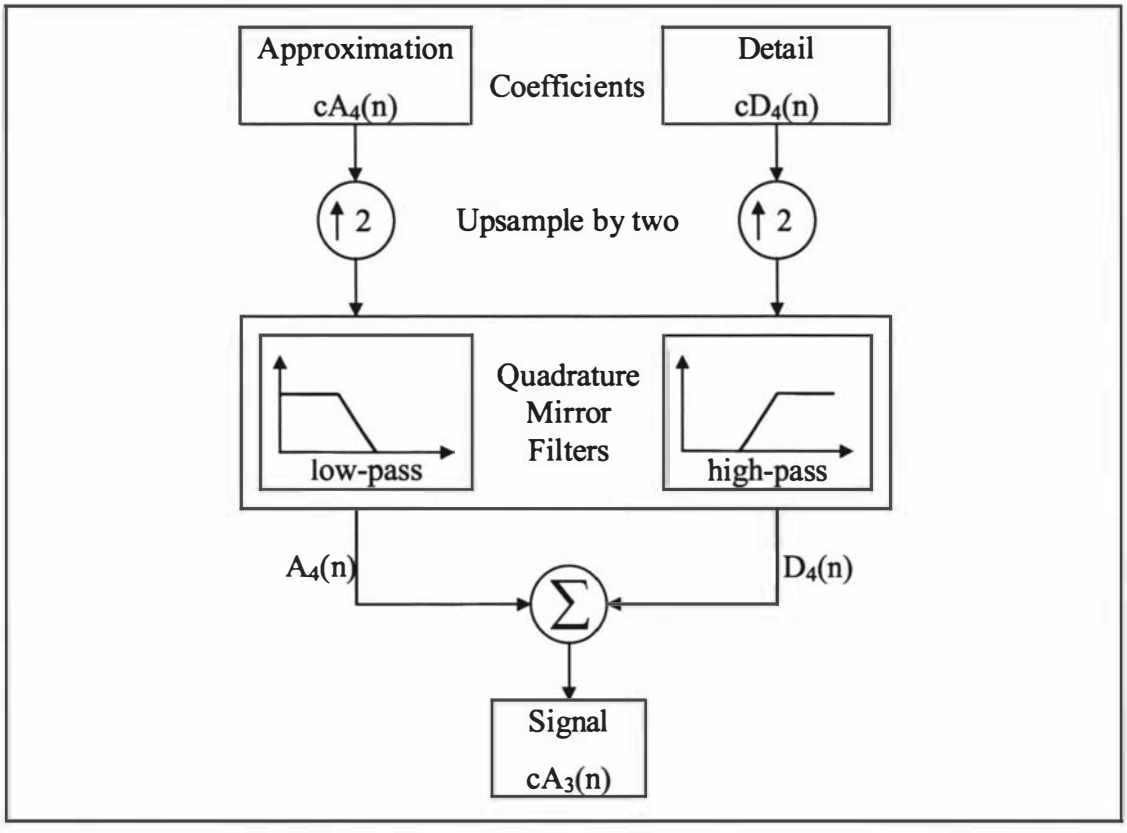


Figure 2.21 – Structure of a single level of wavelet reconstruction

Both the decomposition and reconstruction processes are iterative. Figure 2.21 shows the first level of reconstruction where coefficient signals $cA_4(n)$ and $cD_4(n)$ are combined to create the coefficient signal $cA_3(n)$ which is then used as an input for the subsequent reconstruction level. The two input coefficient signals are upsampled by a

factor of two by inserting zeros between each sample point. Figure 2.22 details this process for the $cA_4(n)$ coefficient signal and Figure 2.23 details this process for the $cD_4(n)$ coefficient signal. After upsampling, the signals are filtered using the appropriate reconstruction filter for the wavelet family being used. Finally the two upsampled and filtered signals, $A_4(n)$ and $D_4(n)$ are summed together to form a new approximation coefficient signal $cA_3(n)$ as shown in Figure 2.24.

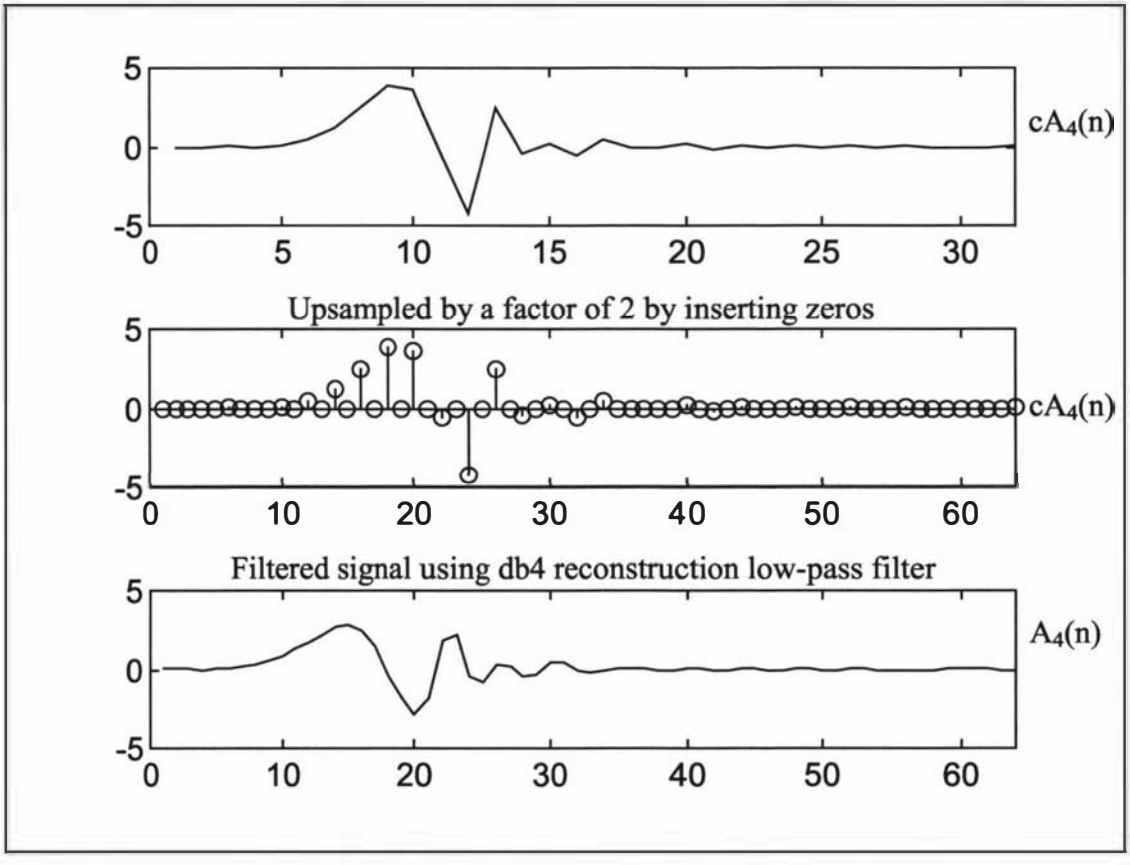


Figure 2.22 – Creation of A_4 from cA_4, upsampling followed by low pass filtering

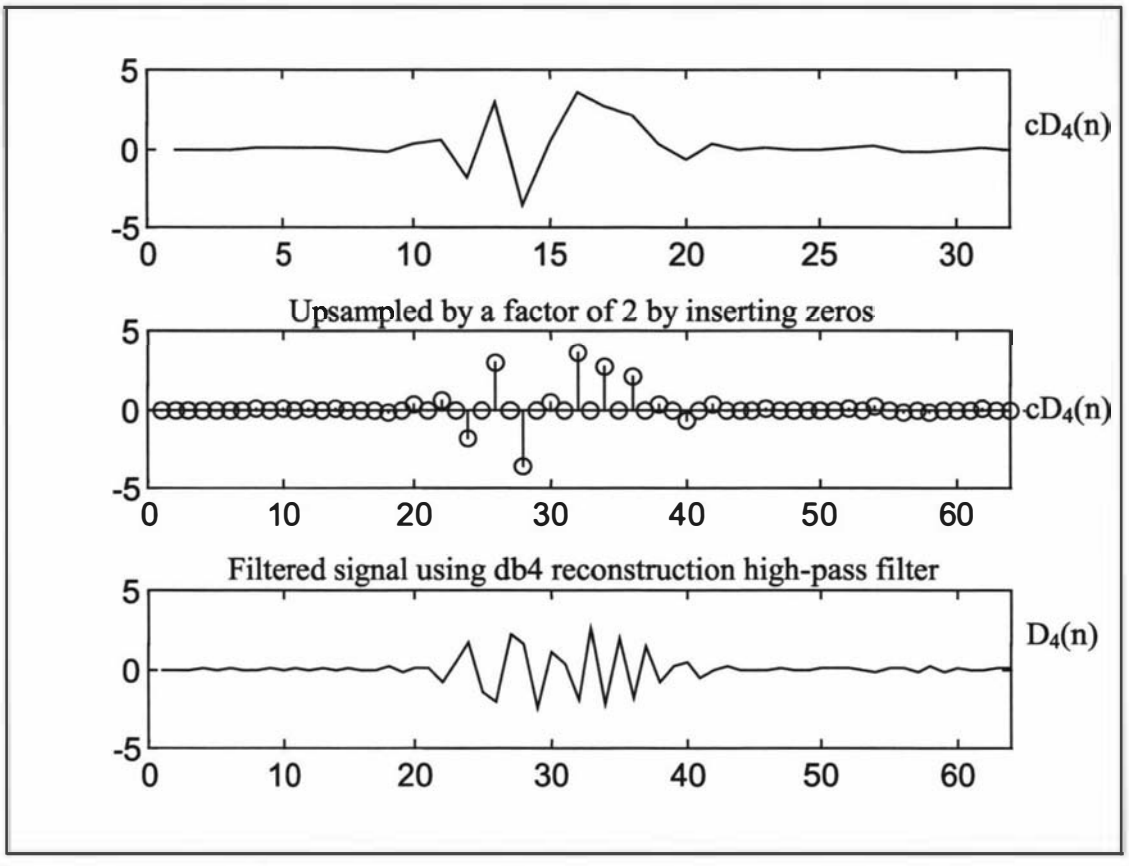


Figure 2.23 – Creation of D_4 from cD_4, upsampling followed by high pass filtering

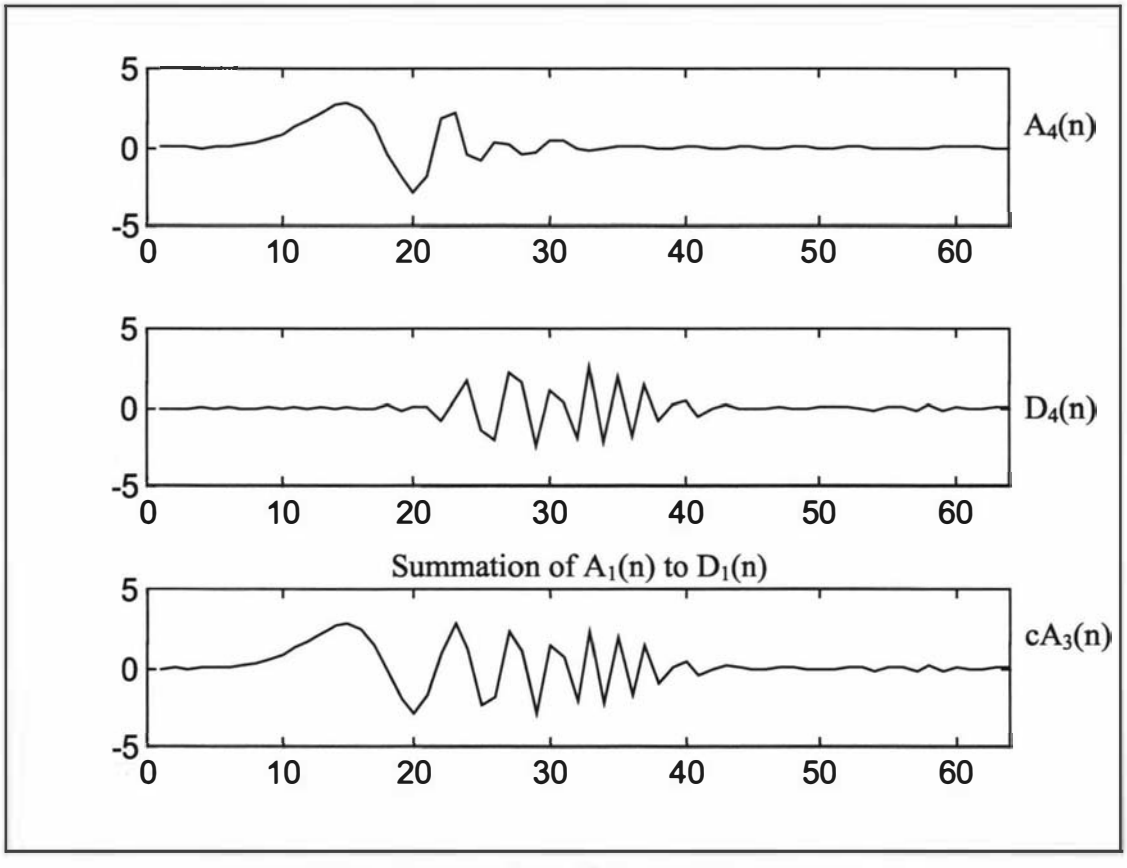


Figure 2.24 – Summation of A_4 and D_4 to create cA_3

2.6.3 Signal extension of one-dimensional signals

Signal extension is an important topic when considering the transform of a discrete finite length signal from one domain to another. With the Discrete Fourier Transform (DFT) and the DCT, the signal extension is not immediately obvious as no additional data points need to be appended to the signal boundaries prior to transformation. Figure 2.25 reveals the implied signal extension present in both of these transformations given that the input signal is of length N samples [2.7]. If the extended signal contains discontinuities at the boundary of the signal then border distortion arises when the signal is compressed as the high frequencies are usually removed in signal compression. Figure 2.25 shows that the DCT is superior to the DFT in that the mirror symmetry implied by the extension removes the discontinuity at the edges of the signal. The signal extension associated with the wavelet transform needs to be explicitly dealt with as it is not implied by the transform.

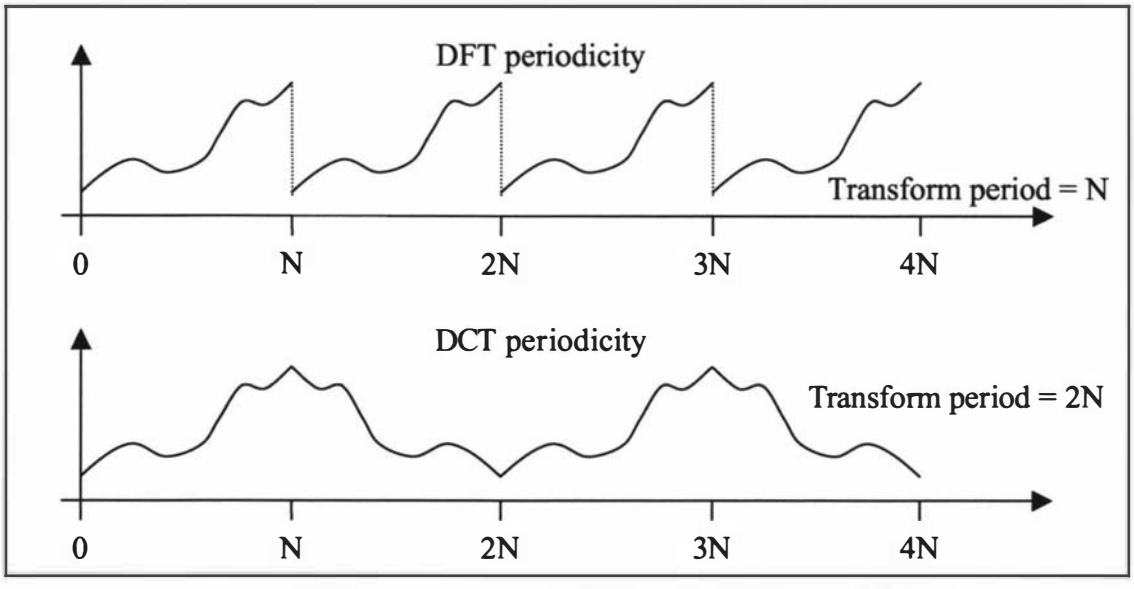


Figure 2.25 – Implied periodicity of the DFT and DCT

The wavelet transform of a one-dimensional signal is accomplished by convolving the signal with two banks of wavelet filter coefficients followed by a downsampling process. Edge distortion arises when a finite length signal is convolved with the wavelet filter coefficients. In the preceding wavelet transform examples, no attention has been directed to how the endpoints of the finite length signals are to be treated. It

is important to consider this level of detail, as there are a number of different techniques for extending the signal at its endpoints. The choice of wavelet filter and style of wavelet transform also depends on the extension technique chosen.

To avoid edge distortion the signal needs to be extended by the filter length minus one samples at each end [2.12,14]. Noting that the Daubechies-4 wavelet has a filter length of eight coefficients, this extension equates to a seven sample extension at each end of the signal. If the original signal was 512 samples in length then the extended signal is now 526 samples long. Figure 2.26 illustrates how there are 519 locations along the extended signal to perform the convolution operation with the wavelet filter coefficients. Each of the two resulting convolved signals, one from each filter bank, contain 519 values. The filter outputs are then decimated by a factor of two. Each decimated signal has the *floor of (519 / 2)* = 259 values, slightly over half the original length of the signal. Here the *floor of* x is defined as the rounded down integer value of x. This extension process is shown in Figure 2.26 for the low pass filter bank. The first level decomposition of the input signal of 512 samples contains a total of 518 output coefficients. The output is comprised of 259 low pass decimated coefficients and 259 high pass decimated coefficients.

Ideally, an energy compaction transform such as the wavelet transform would produce the same number of coefficients in the output space as there are in the original signal. This is a property of the Fourier and Discrete Cosine transforms. The signal extension method described when applied to the asymmetric db4 wavelet, results in an expansive wavelet transform. The expansion gradually worsens as either, the filters get longer, or the number of levels of the decomposition increases. Figure 2.27 illustrates the pyramidal structure for just one half of the filter bank outputs associated with an expansive one-dimensional transform of an input signal of 512 elements. When both halves are considered there are a total of 538 coefficients produced in a four level transform using an eight coefficient wavelet filter.

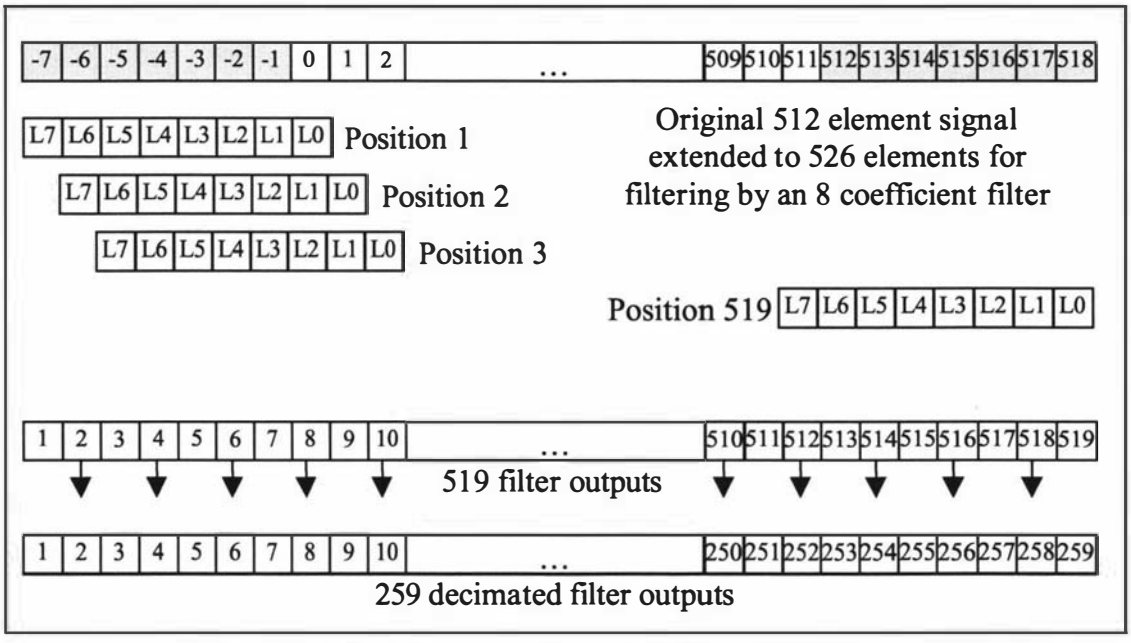


Figure 2.26 – Signal extension of a finite length signal

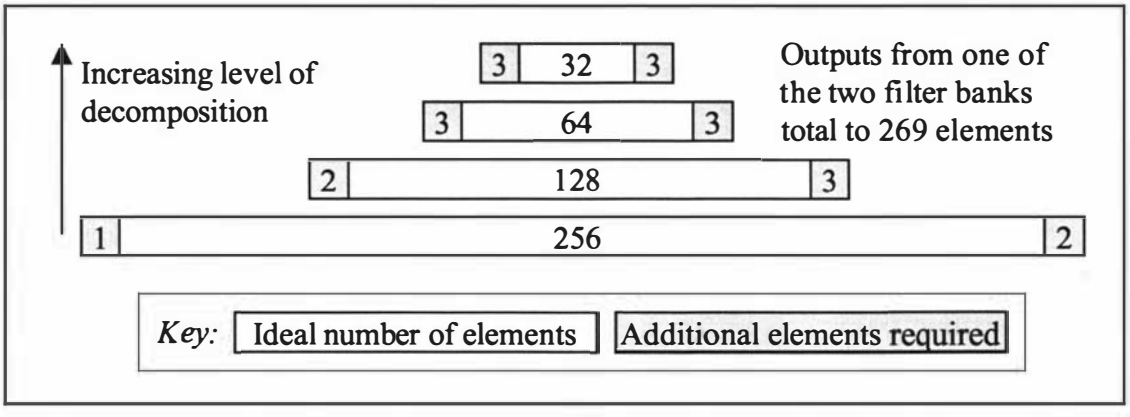


Figure 2.27 – Pyramidal decomposition structure of the expansive WT with an 8 element filter on a 512 element input signal

Several different techniques can be used with the expansive wavelet transform to extend the signal prior to filtering. In applying the zero-padding technique it is assumed that the signal is zero outside its range. As a simple padding method discontinuities are created at the edges of the signal. When coding a signal to maximise its potential for compression it is important to not introduce discontinuities that will then need additional bits to encode them. The symmetrisation [2.11] technique, also called symmetric extension, replicates a mirrored view of the boundary values of the signal and only creates a reduced discontinuity. The discontinuity is caused by a jump in the first derivative of the signal due to it being

reflected at the boundary [2.12]. A simple smooth padding can be used to preserve the first derivative at the boundary by using it to extrapolate the signal. These padding methods are illustrated in Figure 2.28.

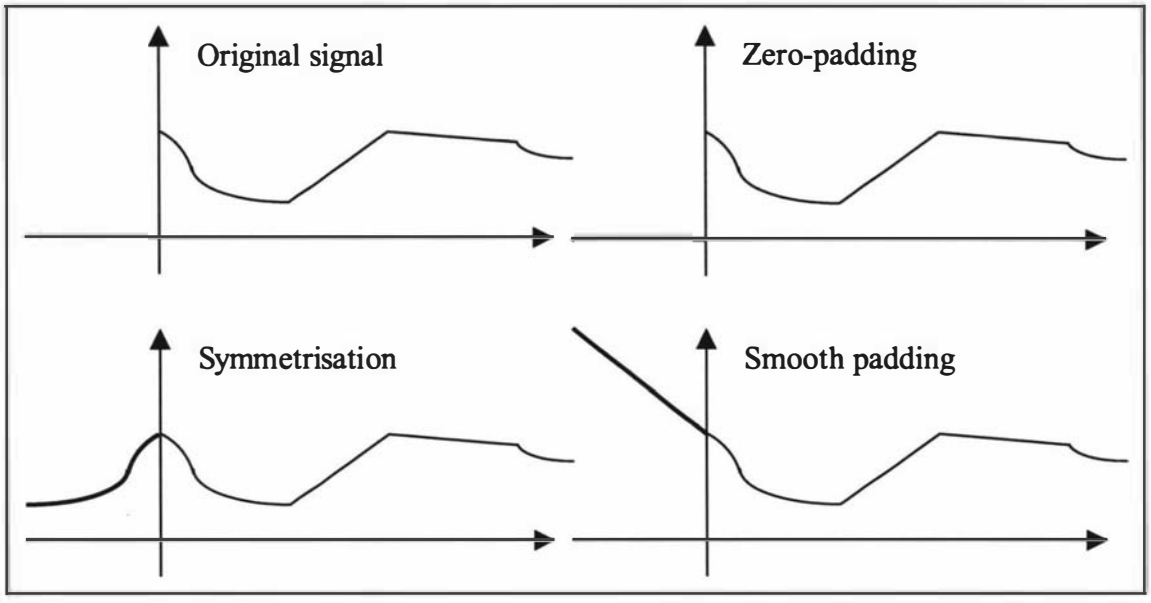


Figure 2.28 – Various padding strategies for expansive wavelet transform

There are two non-expansive wavelet transforms. In a compression algorithm a transform which is non-expansive is preferred as fewer coefficients are produced to be coded. The use of the periodised wavelet transform assumes that the signal is periodic which is not the general case for most real world signals. Periodised wavelet transforms will often create discontinuities at the boundary of the signal. The symmetric wavelet transform uses symmetrisation to extend the signal in the same way as it can be used in the expansive wavelet transform. The critical difference is that if a symmetry constraint is placed on the choice of wavelet filter bank then the symmetric wavelet transform becomes non-expansive. This result is due to the symmetry of both the signal and the filter bank coefficients at the signal edges. The two non-expansive wavelet transforms are shown in Figure 2.28.

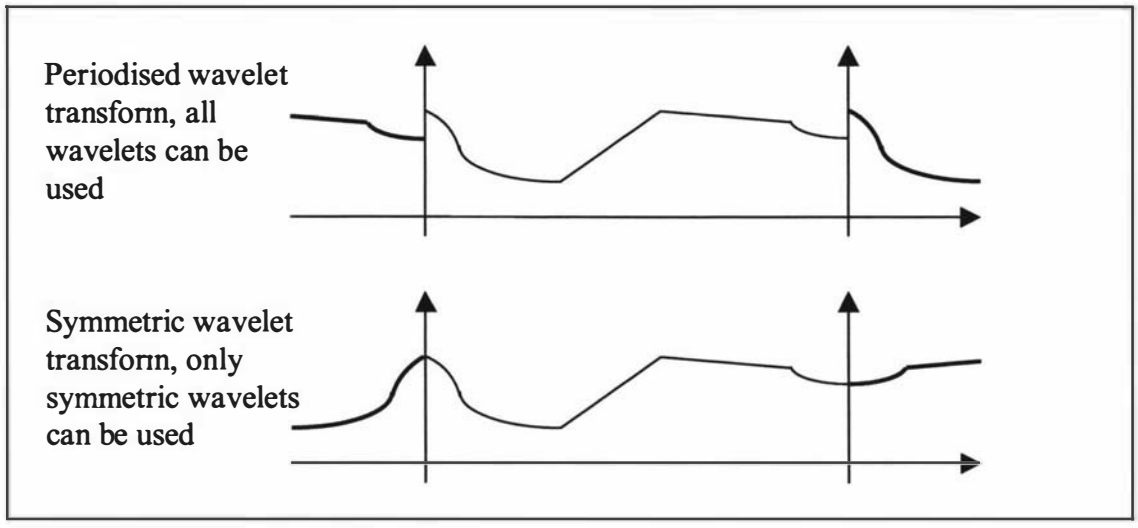


Figure 2.29 – Non-expansive wavelet transform signal extension

In reviewing the above signal expansion techniques and considering that the signals to be transformed are images, the following observations can be made:

1. Smooth padding appears to be the best method for reducing signal edge distortion but the signals must be smooth to favour this technique. When considering a scanline from a real world image, there will be a noise component superimposed on what is normally a smooth signal. An intensity plot of the first 256 pixels in the first scanline of the Lena image is shown in Figure 2.30 to demonstrate this point. As smooth padding uses the first derivative of the signal, it is sensitive to noise. In practice the extension often appeared discontinuous when tested on real images. Using smooth padding to extend an amplitude limited signal, such as an 8-bit digitised video scanline, may cause saturation as the signal is extrapolated. Saturation of the signal again leads to further artificial discontinuities in the extended signal.
2. The use of symmetric extension will overcome the above limitations of smooth padding when the signal is corrupted with noise. It is therefore a more robust extension technique for use with images.
3. A non-expansive signal transform is advantageous for data compression. Of the two non-expansive signal extensions implied by the non-expansive transforms, the symmetric wavelet transform appears to be the best for images as image scanline signals are not periodic in nature.

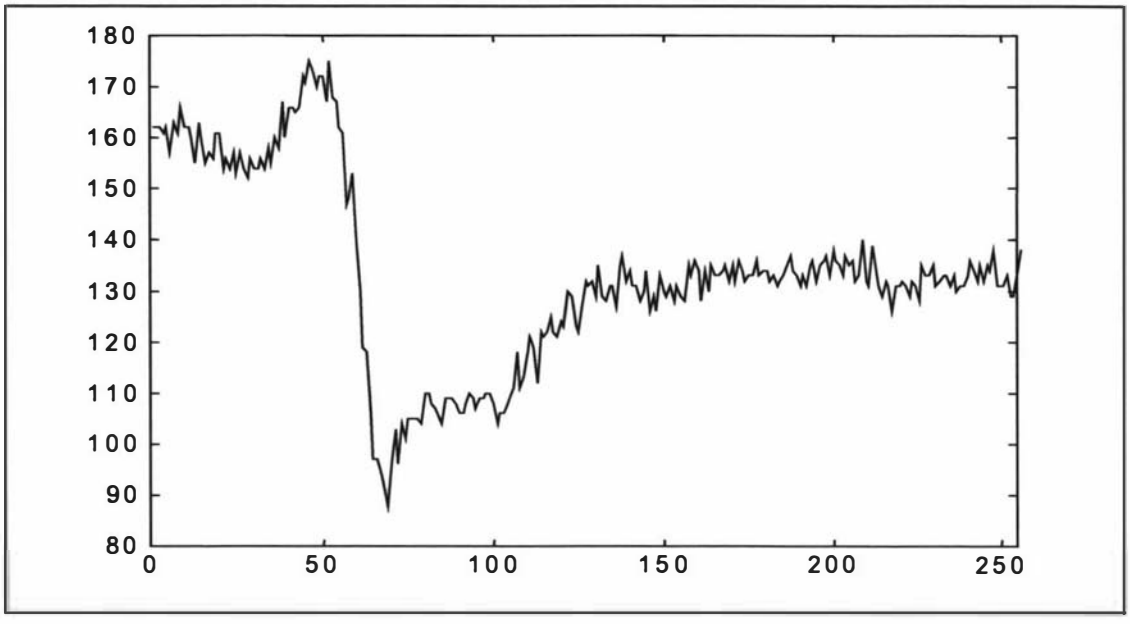


Figure 2.30 – Intensity plot of first 256 pixels in first scanline of the Lena image

The above observations show that the best signal extension technique for image compression using wavelets is symmetrization. If a symmetric wavelet filter bank is used for the wavelet transform, then the symmetric wavelet transform can be used for the decomposition. This results in a non-expansive transform that has minimal distortion about the edges of the image. The Daubechies-4 wavelet filter that has been considered in section 2.6.2 is not symmetric and is not suitable for use with the symmetric wavelet transform. The selection of a more suitable wavelet filter will be explained in the following section as the progression from one-dimensional signals to two-dimensional images is considered.

2.6.4 Wavelet transform of two-dimensional images

The wavelet transform of one-dimensional signals can be extended to two-dimensional signals with ease if the wavelet filter bank is separable. Most of the common wavelet filters families such as Daubechies, symlets, Coiflets and the biorthogonal wavelets are separable [2.11]. Of particular interest are wavelet filters that are separable and symmetric so that the non-expansive symmetric wavelet transform can be realised. The only symmetric wavelet filter family in the above list

is the biorthogonal wavelet family. A specific member of this family in the biorthogonal-7.9 wavelet filter bank and this is shown in Figure 2.31. The FBI fingerprint image compression standard uses the symmetric wavelet transform together with the biorthogonal-7.9 wavelet filter bank to achieve good results [2.15].

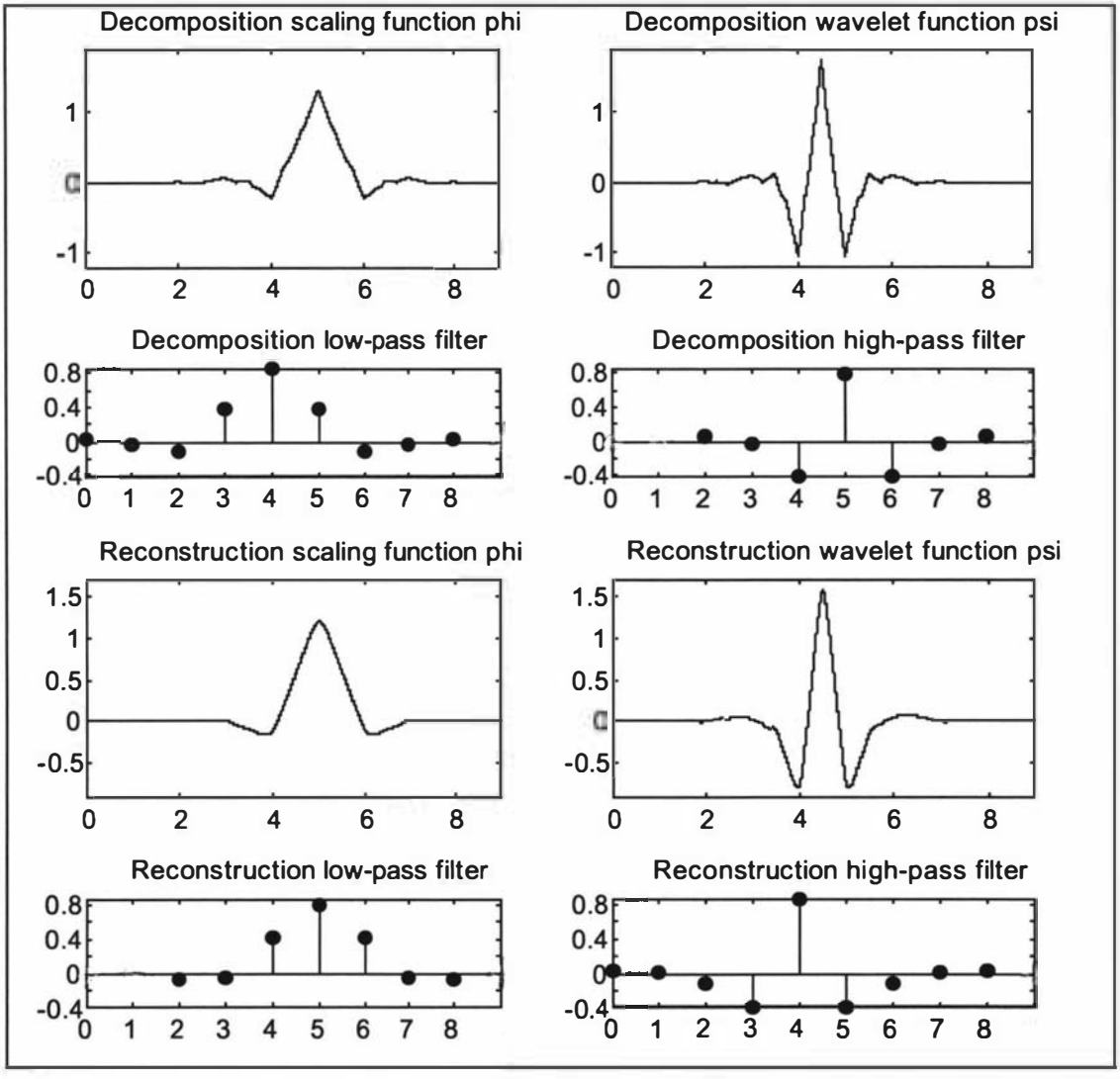


Figure 2.31 – Biorthogonal-7.9 wavelet filters

Biorthogonal wavelets are fundamentally different from orthogonal wavelet families such as the Daubechies family. An important property of the biorthogonal wavelet family is linear phase which is needed for image reconstruction. Linear phase implies that the plot of phase verses frequency for the filter frequency response function is a straight line [2.12]. Separate wavelets are used for decomposition and reconstruction to achieve this property. An advantage of using separate wavelet filters for analysis

and synthesis is that different properties can be emphasised in each filter [2.11]. For analysis the properties of oscillation and zero moments can be concentrated. For synthesis the property of regularity can used to create a smoother reconstructed image.

Another result of the linear phase biorthogonal filters is that the image contents do not shift between the subbands. With the orthogonal Daubechies-4 filter, features in the subbands do not quite line up as the filters do not have linear phase [2.12]. In the paper by da Silva and Ghanbari [2.16], it is proven that orthogonal filters cannot have linear phase. The precise alignment of coefficients across the subbands allows for higher coding gains if structures are built upon the relationships between coefficients in different subbands.

Figure 2.32 shows in block form the two main stages of how the separable two-dimensional wavelet transform is implemented. The first stage is accomplished by initially performing one-dimensional wavelet filtering of the image (say) horizontally. This produces a set of horizontal low pass approximation coefficients and a set of horizontal high pass detail coefficients which can both be downsampled by a factor of two. The downsampling is done because each of the filter outputs contain one half of the original bandwidth of the input signal. This is shown graphically on the Lena image in Figure 2.34. The aspect of the two output images in Figure 2.34 appears squashed due to the downsampling that has occurred. The two sets of coefficients produced are not the final product of the two-dimensional separable wavelet transform. Each set of coefficients need to be wavelet filtered a second time in the vertical direction to produce a total of four downsampled sets of coefficients as shown in Figure 2.35.

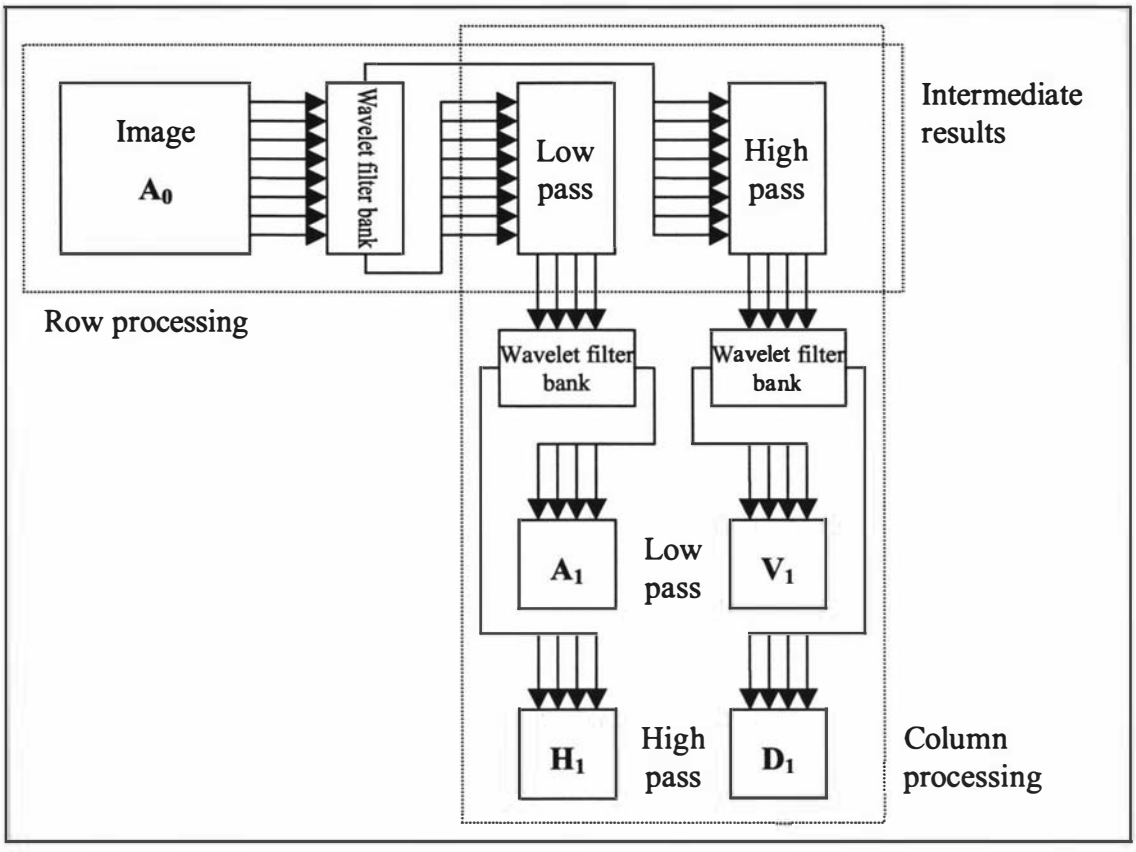


Figure 2.32 – Block diagram of the two-dimensional wavelet decomposition

At this point a single level of the two-dimensional wavelet transform has been performed. The A_1 set of approximation coefficients represent a low pass filtered and downsampled version of the original image which can be further decomposed in subsequent levels of the wavelet transform. The H_1, V_1 and D_1 sets of detail coefficients represent various combinations of low and high pass filters in the horizontal and vertical directions. Essentially they contain edge information in one of **three** orientations. Low pass horizontal filtering followed by high pass vertical filtering produces a set of horizontal edges shown in the H_1 set of detail coefficients. High pass horizontal filtering followed by low pass vertical filtering produces a set of vertical edges shown in the V_1 set of detail coefficients. Finally, high pass horizontal and vertical filtering produces a set of diagonal edges shown in the D_1 set of detail coefficients.

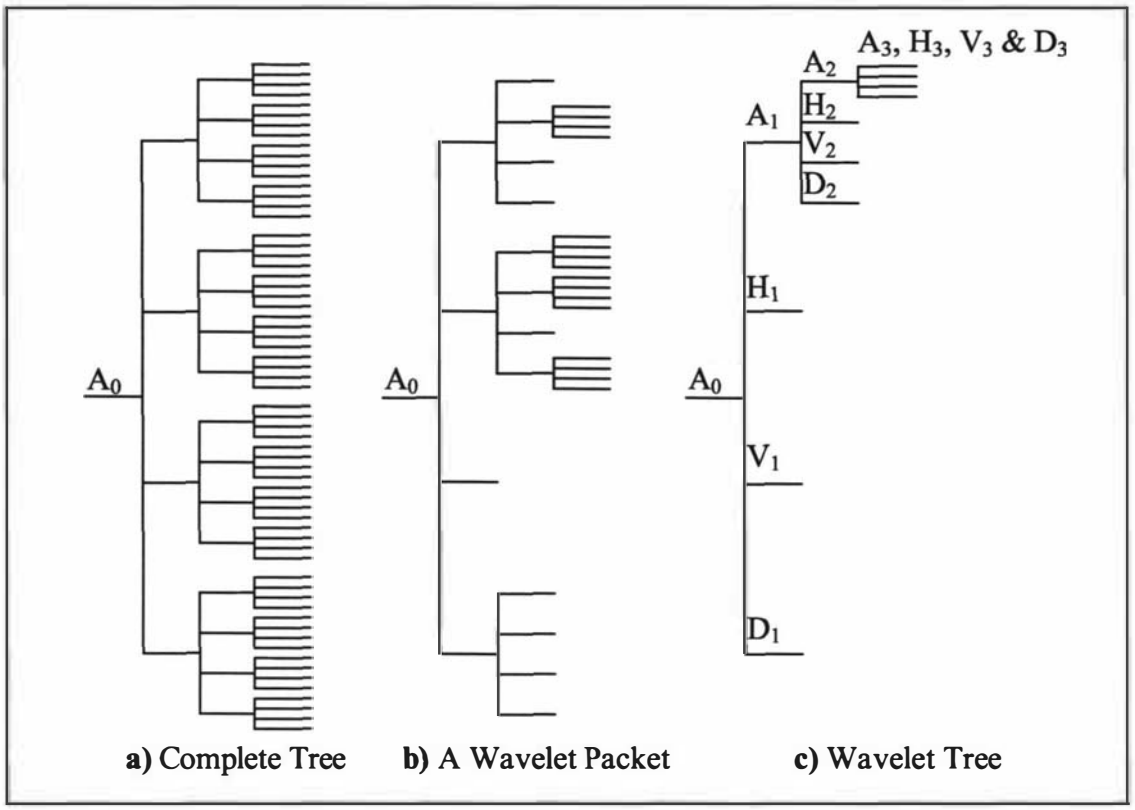


Figure 2.33 – Various tree forms of three level wavelet decompositions

The wavelet decomposition can be repeated on the A_1 set of low pass coefficients to further decompose the image into more subbands. Figure 2.36 shows the Lena image decomposed to four levels. The figure was produced in the Matlab software package in conjunction with the Wavelet Toolbox. At this point the original 512 x 512 pixel image has been reduced to a 32 x 32 pixel A_4 image by low pass filtering and most of the energy compaction has already occurred. There is little gain to be had by decomposing to any further levels. This form of decomposition is known as the wavelet decomposition tree in [2.11] or the logarithmic wavelet tree in [2.12] of an image. The tree structure for this form of decomposition of image A_0 to three levels is shown in Figure 2.33 **c)** along with two other decomposition trees. The complete tree decomposition [2.12] is produced when all four coefficient sets (A_n, H_n, V_n & D_n) at each level are further decomposed at subsequent levels (Figure 2.33 **a)**). The more general wavelet packet decompostion tree [2.11,12] uses an application specific metric [2.12] to determine whether or not a node should be decomposed further at the next level (Figure 2.33 **b)**). Figure 2.37 shows the four level wavelet packet transform of the Lena image using the complete tree decomposition.

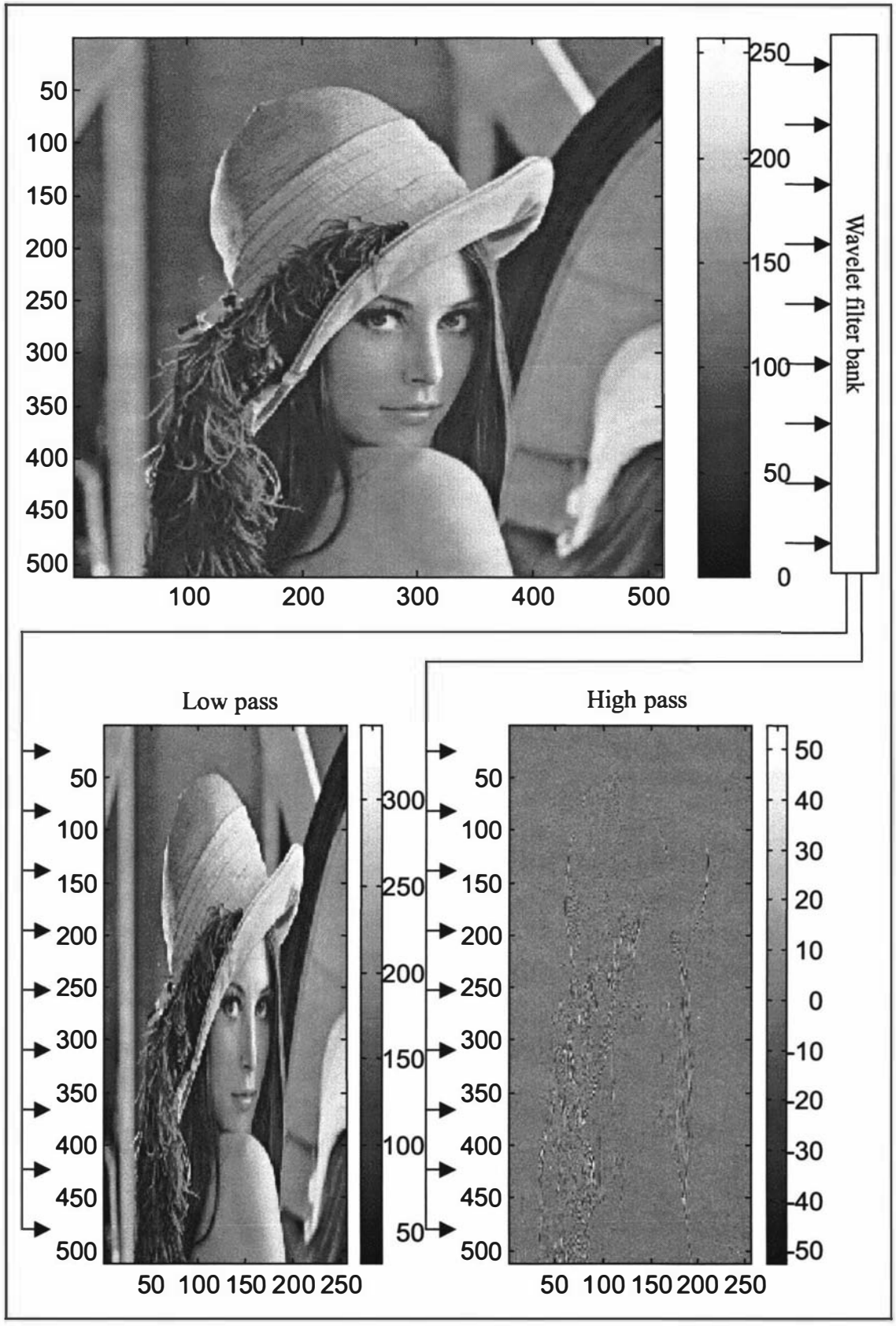


Figure 2.34 – Set of one-dimensional horizontal wavelet decompositions

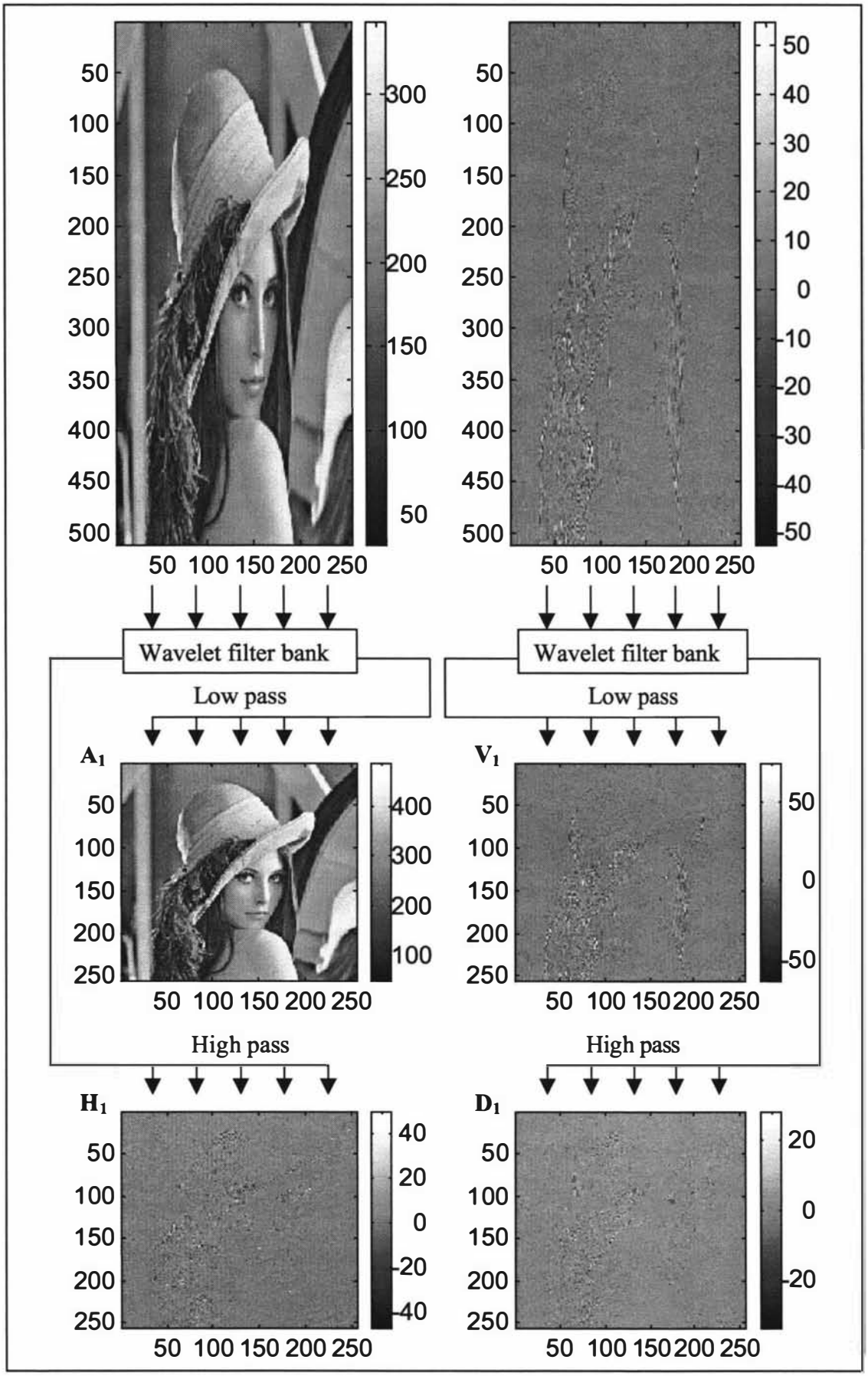


Figure 2.35 – Set of one-dimensional vertical wavelet decompositions

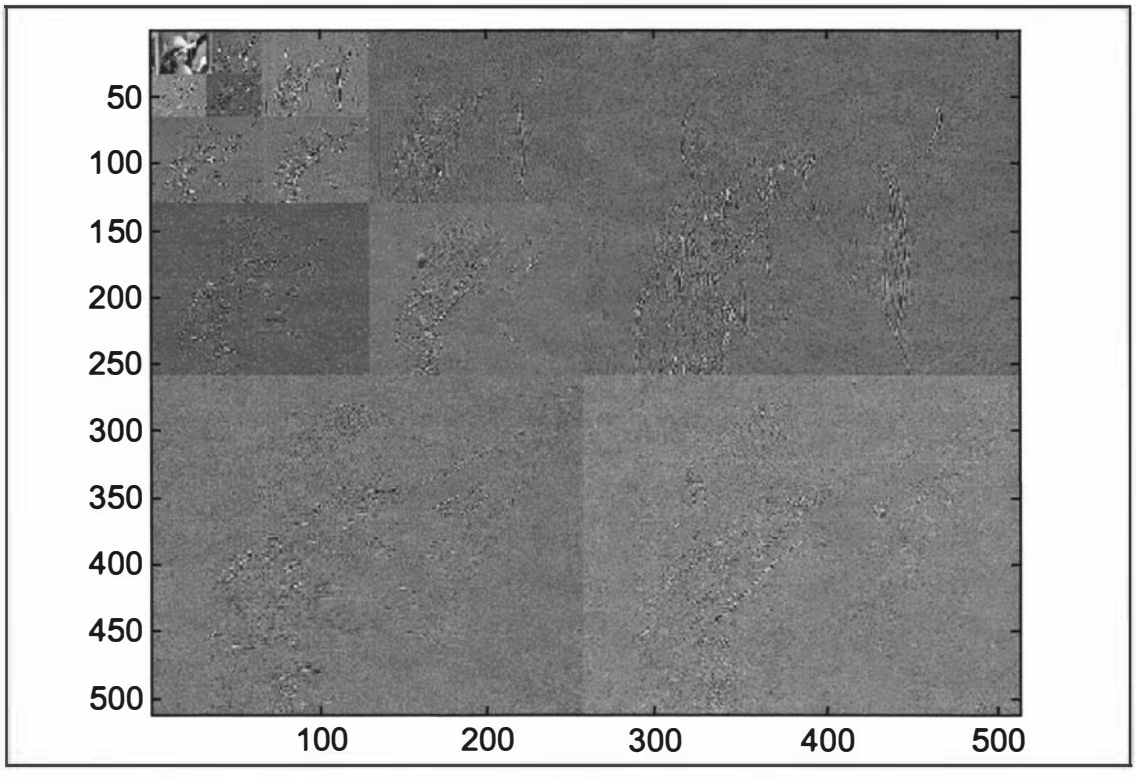


Figure 2.36 – Four level wavelet transform of the Lena image

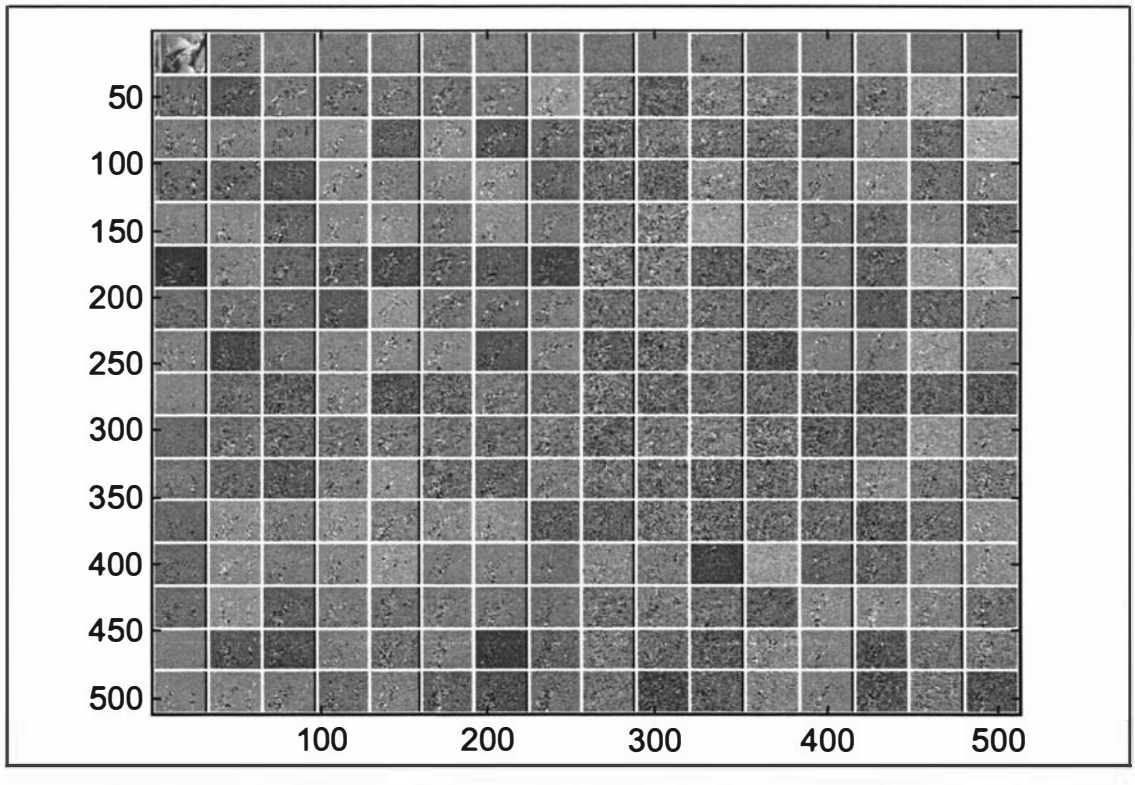


Figure 2.37 – Four level wavelet packet transform of the Lena image

2.6.5 Survey of some existing wavelet based image codecs

The following three sub-sections look at the specific forms of some existing wavelet based compression/decompression systems (codecs). The first sub-section reviews the popularity of the biorthogonal-7.9 wavelet filter bank in several recent image codecs. Next the form of wavelet decomposition is examined in light of computational load and suitability for use in compressing a wide range of images that may be typical in a surveillance application. Finally some methods of coding the resultant wavelet transformed image are given.

There are many combinations of the above parameters that appear in the literature, but the central theme of this thesis is to achieve high compression of images with a relatively low computational load. As the system is to be implemented on an embedded DSP system, memory is also a significant constraint on algorithmic complexity. At the end of each sub-section a choice is made as to a suitable form for an embedded wavelet based image compression system.

2.6.5.1 Wavelet filter choice

The biorthogonal-7.9 wavelet filter appears in the FBI fingerprint image codec [2.17] and is reported to provide superior results to the asymmetric Daubechies-4 wavelet filter [2.12]. The regions of support are similar between these two wavelet filter banks but there are two main reasons why the biorthogonal filter bank is superior. As the filter bank is symmetric it is possible to apply a symmetric extension to the image at the boundaries. This in turn allows the Symmetric Wavelet Transform (SWT) to be used which generates less coefficients to be subsequently encoded. The second advantage gained by the linear phase biorthogonal filters is that the image contents do not shift between the subbands.

Other papers utilising the biorthogonal-7.9 wavelet filter include Barreto and Mendonça [2.18] who compare its compression properties favourably with the longer biorthogonal-10.14. Also Antononi *et al* [2.19] uses biorthogonal wavelets for

decomposing the image in their image codec. The most significant recent confirmation that the biorthogonal-7.9 wavelet filter is appropriate for low bit-rate image coding is its use in the JPEG2000 image compression standard [2.20].

2.6.5.2 Wavelet decomposition choice

The discrete wavelet transform is most appropriate for the task of image compression as it produces a minimal set of coefficients that can then be encoded. A fast transform exists to calculate the discrete wavelet transform for compactly supported wavelets. The algorithm was produced by Mallat in 1988 and is the same algorithm that is used in the signal processing community to perform a two channel subband coder [2.12]. The algorithm uses quadrature mirror filters (QMF) followed by downsampling by a factor of two as shown previously in Figure 2.16.

Another parameter to the wavelet decomposition is the number of levels deep the decomposition should be iterated through as well as the style of decomposition. Wavelet packet decomposition potentially decomposes every output data packet at each level into four new data packets, each with a quarter of the parent data. Computationally this means that each level of the decomposition requires approximately the same amount of effort and any computational resource restrictions will limit the practical number of levels achievable. The number of packets produced at the end of N iterations is 4^N which shows that the number of packets to be subsequently encoded increases rapidly with N, see Figure 2.33 **a)** for an example of $N=3$.

The simpler and less general form of decomposition that is often used in reported wavelet based image codecs only decomposes the approximation packet at each level of iteration. The number of packets produced at the end of N iterations is $3N + 1$ which shows that the number of packets to be subsequently encoded increases with *order(N)*, see Figure 2.33 **c)** for an example of $N=3$. The amount of computation required also quarters with each iteration thus there is less constraint imposed by computational resource when choosing the number of levels to decompose.

The above constraints led to the choice of a wavelet decomposition to 4 levels producing 13 varying in size packets of data to be encoded per image. At this point the input image, if of dimensions 512 x 512 pixels, has been decomposed into four octaves of frequency subbands. The approximation image is now only 32 x 32 pixels in size and most of the energy compaction has been achieved at this point.

2.6.5.3 Wavelet coefficient coding choice

Once the image is decomposed into a set of subbands, coding strategies as discussed in section 2.4.3 are required to achieve data compression.

The FBI fingerprint image compression standard [2.15] uses Wavelet/Scalar Quantisation (WSQ) to encode the transformed coefficients. After wavelet packet transforming the image into 64 subbands, each band is uniformly scalar quantised. The quantised symbols are run-length coded in a simple raster scan fashion and optimal Huffman entropy coding is applied to the symbol stream. A similar coding scheme is used in the Analog Devices ADV601 wavelet based video codec integrated circuit [2.21,22]. This single-chip, video-codec uses a truncated wavelet decomposition similar to WSQ but omits the computation of V_1 and D_1. The basic building blocks for the WSQ coder are shown in Figure 2.38 [2.17].

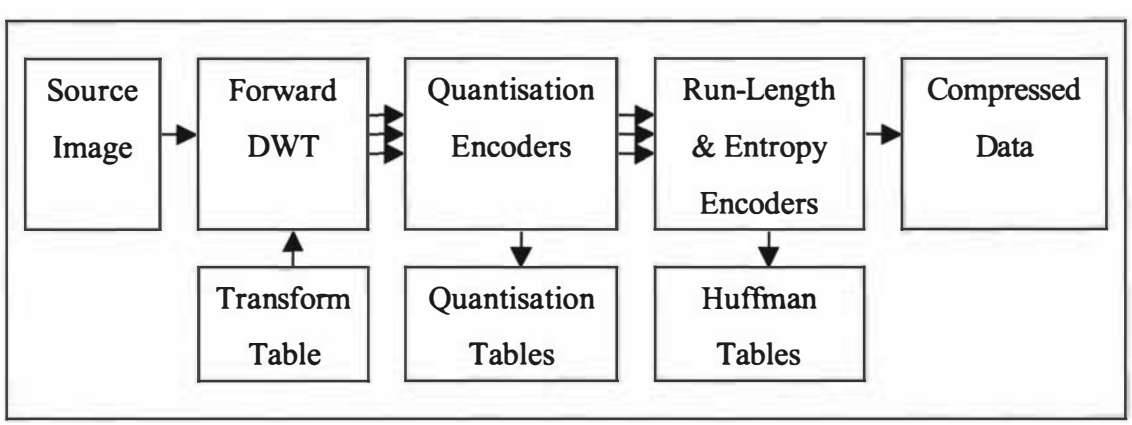


Figure 2.38 – Simplified WSQ encoder

An alternative to raster scanning of the coefficients is the construction of tree structures. The coefficients in each of the three filter orientation spaces, H_n, V_n & D_n, can be viewed as separate pyramidal structures, see Figure 2.39. A coefficient in level 4 has four spatially related coefficients in the level below it, which in turn have four descendants. This structure is known as a *quad-tree*; two early references to its use in coding wavelet coefficients are found in papers by Lewis and Knowles [2.23,24] where it is called a *four branch tree.* The main advantage in using quad-trees to represent spatially related coefficients is that the tree allows inactivity in the lower branches to be predicted. If a quantised coefficient near the top of the tree is zero then it is very likely that the four descendent coefficients, representing the octave of frequencies above the parent, are also zero [2.12].

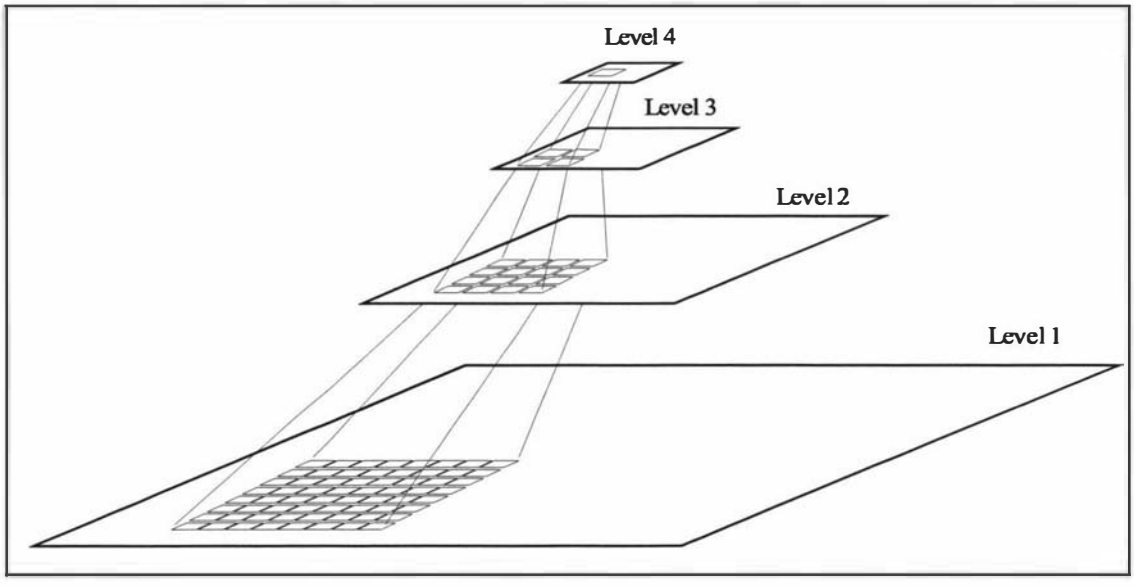


Figure 2.39 – Quad-tree structure from wavelet decomposition

In 1993 Shapiro further refined the use of quad-trees by defining a *zerotree* structure [2.25] which was primarily used to improve the compression of the *significance map* of wavelet coefficients. A significance map conveys the location of significant wavelet coefficients within the transformed image. As the quantised wavelet coefficient spaces are sparse, it is important to be able to encode efficiently the location of significant coefficients. The zerotree structure efficiently encodes the significance map by compactly representing the sparse regions within the tree structure. Shapiro also used an embedded code to represent the compressed data

stream. An embedded code can be truncated at any point and an image can be successfully decoded from the truncated data stream. Shapiro's codec is called an Embedded Zerotree Wavelet (EZW) coder and is useful for progressive transmission of image data as a crude representation of the image can be decoded from only a small percentage of the full bitstream. As more bits are received, the image can be reconstructed at greater levels of quality through the use of successive-approximation quantisation. For a fixed bandwidth environment the embedded code is useful for maximising the bandwidth available without ever exceeding it. Shapiro's paper is also significant in that it broke the trend of ever increasing complexity to achieve better compression by being a fast technique that was competitive with the most complex techniques at the time [2.26].

Vector quantisation and progressive transmission are used in the coding scheme proposed by Antonini [2.19]. One disadvantage of vector quantisation is that the codebook used in the scheme is produced from a training set of images. This has the disadvantage that coding performance is adversely affected if the nature of the coded images was significantly different from the training set. The progressive transmission scheme ensures that all of the information to reconstruct the lowest resolution of the wavelet decomposition arrives first so that a low resolution image can be constructed after only a small percentage of bits are received. This technique is different from an embedded code in that reconstruction is only possible in large steps after certain numbers of symbols have been received. The embedded code in EZW allows for an improvement in image quality as each symbol is received.

In 1996, Said and Pearlman [2.26], reported on a significant enhancement to the EZW algorithm called Set Partitioning In Hierarchical Trees (SPIHT). The way the subsets of coefficients are partitioned and significance information is conveyed are fundamentally different in SPIHT. The algorithm is computationally efficient and produces an embedded bitstream. More recently there have been minor enhancements to the EZW coder made by altering the scanning order [2.18] whilst still maintaining the embedded property of the bitstream. The EZW coder has also been used as the basis for image sequence compression where the embedded property has been abandoned in favour of achieving slightly better compression [2.27,28].

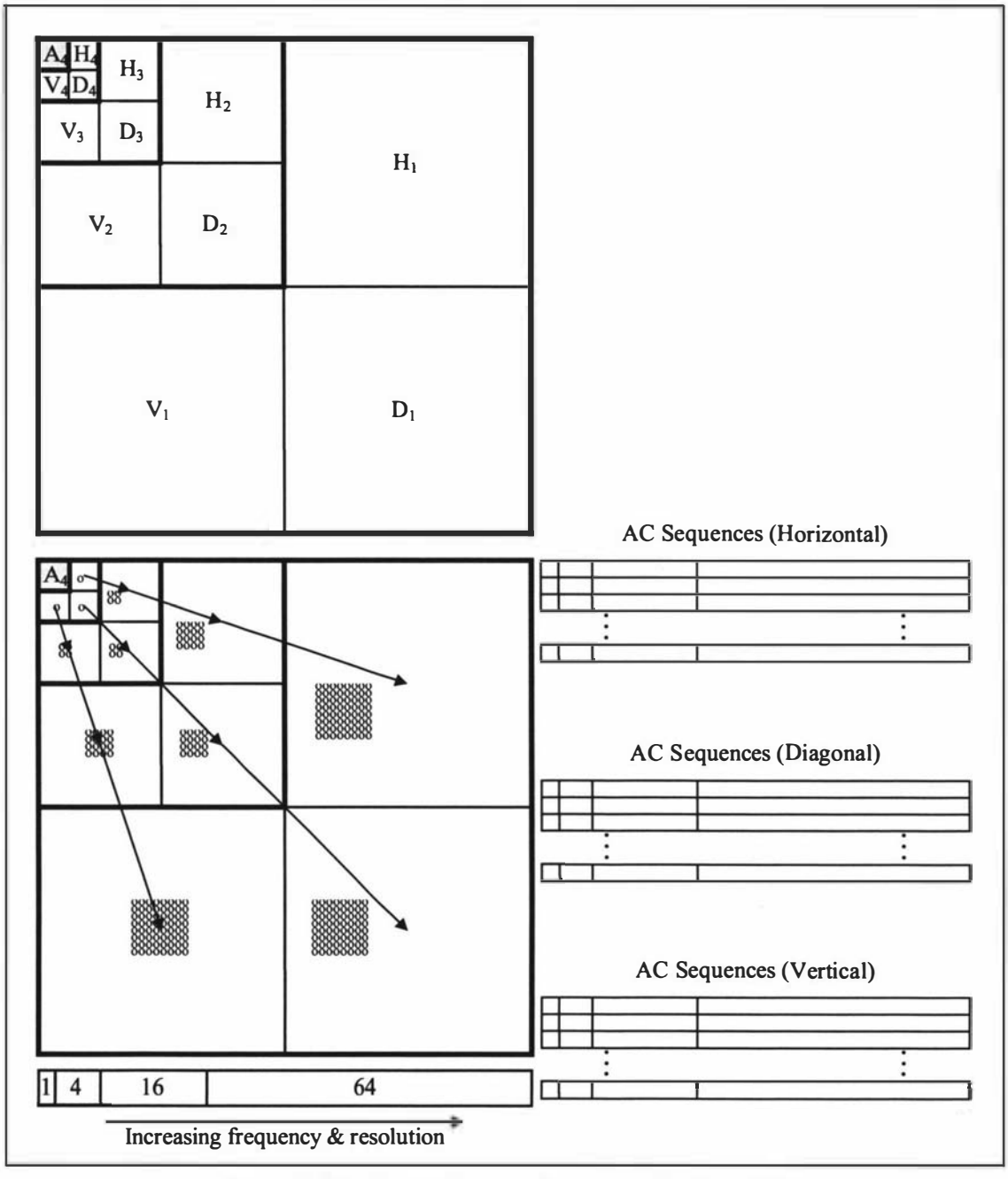


Figure 2.40 – Scanning method used in the DWT

In Strang and Nguyen's book [2.12], the zerotree structure from Shapiro's coder is used to create a wavelet based coder that is very similar to the Sequential Baseline Coder (SBC) used in the JPEG image compression standard. Each of the zerotrees is raster scanned to produce an AC sequence. The AC sequence for a four level decomposition is shown in Figure 2.40, each tree produces 85 ordered coefficients. This is similar to the 64 ordered coefficients that are produced in JPEG after each 8 x 8 pixel block is transformed using the DCT and zig-zag scanned. The ordered

coefficients are then run-length coded after quantisation as the scanning methods employed favour long runs of zeros. Finally, entropy encoding is performed on the resulting symbol set. This method is simple and performs well due to the sparse nature of both DCT and wavelet transformed images. The amount of storage required to encode each tree is minimal as the large set of trees are coded sequentially without any further interaction. The EZW and SPIHT methods require many more passes of the wavelet coefficients to encode successive-approximations of the data and are more complex in terms of amount of computation and storage required.

In an embedded DSP application of wavelet coefficient coding, it is important to choose a method that is computationally simple and does not require a large amount of storage as the working memory on a DSP chip is usually limited to a few thousand words. Embedded code techniques require many iterations over the wavelet coefficients to create the output bitstream. Although the properties of progressive transmission and strict bandwidth adherence are useful, the memory bandwidth and capacity of the DSP are limiting factors for the speed of encoding.

Vector quantisation of the wavelet coefficients require expensive searches in the code book for best matches which will not be optimal for a wide range of images. Again memory bandwidth limitations make such a scheme ill suited for DSP implementation in a real-time system.

The two techniques most promising for efficient implementation on a DSP are the FBI WSQ and Strang's Sequential Baseline Coding of wavelet zerotrees. Both of these techniques are relatively straightforward to implement from simple building blocks. The computation time of each is readily predictable for all image types due to the linear nature of the algorithms. Other than a buffer for the transformed wavelet coefficients, the amount of temporary storage required at any one time is constrained to only a few hundred words.

The SBC of wavelet zerotrees is likely to achieve higher compression ratios than WSQ due to the zerotree structure exploiting the spatial similarities between the different subband levels as well as the efficiencies gained from run-length coding

within a subband. The ability to process the large set of wavelet coefficients in small packets of 85 elements at a time representing a single zerotree is well suited to a pipelined DSP implementation.

2.7 Miscellaneous coding techniques

There are a number of image coding techniques that do not fit into the categories of predictive or transform coding. These techniques include Fractal coding [2.29-31] and Vector Quantisation [2.32-35]. The above set of coding techniques are generally considered to be first-generation techniques [2.36] as there is no real modelling of the image content in terms of a scene consisting of a set of objects.

Second-generation techniques include model based techniques such as Delaunay Triangulation [2.37] and image region segmentation for coding [2.36,38-41]. These techniques are mostly experimental and have not yet enjoyed the widespread practical application that many of the first-generation techniques have.

All of the techniques mentioned in this section are asymmetric in terms of the amount computation required to compress and then decompress an image. They all require the use of extensive search and match computations when compressing an image, thus the computational complexity of the compression process is several orders of magnitude greater than any of the transform or predictive techniques covered. The compression performance of these techniques is often not very different from that obtained with more simple algorithms making the current first-generation techniques such as the DCT and WT much more practical.

2.8 Discussion

This chapter has given a detailed overview of the most popular first-generation techniques for image coding. An emphasis has been on computationally simple techniques to ensure that the chosen technique will be suitable for real-time implementation on a modest DSP. Second-generation techniques are still experimental and are generally constrained to only work well with certain classes of image. An image codec for a surveillance application needs to be able to perform well over a variety of situations with predictable performance characteristics. Techniques based on the discrete Symmetric Wavelet Transform (SWT) have recently been accepted by the image compression community as being suitable for compressing a wide variety of images. As the SWT is based on filtering operations, it is very suitable for real-time DSP implementation. The next chapter of this thesis will examine closely the implementation of a SWT based image codec and compare this with the popular JPEG image compression standard with respect to image quality and implementation complexity.

2.9 References

[2.1] **Jain, A. K.,** *Fundamentals of Digital Image Processing,* 1989. Prentice-Hall.

[2.2] **Jayant, N.,** *"Signal compression based on models of human perception,"* Proceedings of the IEEE, vol. 81, October, pp.1383-1422, 1993.

[2.3] **CCIR,** *"Method for the Subjective Assessment of the Quality of Television Pictures,"* Rec. 500-1, 1978.

[2.4] **Nelson, M. and Gailly, J.,** *The Data Compression Book,* Ed. 2, 1996. M&T Books, ISBN 1558514341.

[2.5] **Witten, I. H.; Radford, M. N., and Cleary, J. G.,** *"Arithmetic Coding for Data Compression,"* Communications of the ACM, vol. 30, no. 6, pp.520-540, 1987.

[2.6] **Simon, B.,** *How Lossy Compression Shrinks Files*; PC Magazine, pp.371-382, July 1993.

[2.7] **Weeks, A. R.,** *Fundamentals of Electronic Image Processing,* Ed. 1, 1996. IEEE Press.

[2.8] **Girod, B. and Stuhlmuller, K. W.,** *"A Content-Dependant Fast DCT for Low Bit-Rate Video Coding,"* ICIP, Chicago, USA, p.WA3.01, 1998.

[2.9] **El-Sakka, M. R. and Kamel, M. S.,** *"Adaptive Image Compression Based on Segmentation and Block Classification,"* ICIP, Chicago, USA, p.TP3.01, 1998.

[2.10] **Ramchandran, K.,** *"Wavelets, subband coding, and best bases,"* Proceedings of the IEEE, vol. 84, April, pp.541-560, 1996.

[2.11] **Misiti, M.; Misiti, Y.; Oppenheim, G., and Poggi, J. M.,** *Wavelet Toolbox User's Guide,* 1996. The MathWorks, Inc.

[2.12] **Strang, G. and Nguyen, T.,** *Wavelets and Filter Banks,* 1996. Wellesley-Cambridge Press.

[2.13] **Bracewell, R. N.,** *The Fourier Transform and its Applications,* 1986. McGraw-Hill.

[2.14] **Bates, R. H. T. and McDonnell, M. J.,** *Image Restoration and Reconstruction,* Ed. 1, 1986. Oxford.

[2.15] **Brislawn, C.,** *"WSQ Gray-scale Fingerprint Image Compression Specification,"* 1993.

[2.16] **da Silva, E. A. B. and Ghanbari, M.,** *"On the Performance of Linear Phase Wavelet Transforms in Low Bit-Rate Image Coding.,"* IEEE Transactions on Image Processing, vol. 5, no. 5, pp.689-704, 1996.

[2.17] **Brislawn, C.,** *How Wavelet/Scalar Quantisation Works* [Web Page]. 1996. Available at: http://www.c3.lanl.gov/~brislawn/FBI/OverCompress/HowItWorks/howitworks.html.

[2.18] **Barreto, C. S. and Mendonça, G. V.,** *"Enhanced Zerotree Wavelet Transform Image Coding Exploiting Similarities Inside Subbands,"* Proceedings 1996 IEEE International Conference on Image Processing, Lausanne, Switzerland, pp.549-551, 1996.

[2.19] **Antonini, M.; Barlaud, M.; Mathieu, P., and Daubechies, I.,** *"Image Coding Using Wavelet Transform,"* IEEE Transactions on Image Processing, vol. 1, no. 2, pp.205-220, 1992.

[2.20] **JPEG,** *JPEG2000 Committee Drafts* [Web Page]. 2000. Available at: http://www.jpeg.org/CDs15444.htm.

[2.21] **Analog-Devices,** *Low Cost Multiformat Video Codec - ADV601 Datasheet* [Web Page]. 1997. Available at: http://www.analog.com/pdf/2011_0.pdf.

[2.22] **McGoldrick, P.,** *Video-Compression Chip is the First to use Wavelets*; Electronic Design, pp.150-152, August 1996.

[2.23] **Lewis, A. S. and Knowles, G.,** *"Video Compression Using 3D Wavelet Transforms,"* Electronics Letters, vol. 26, no. 6, pp.396-398, 1990.

[2.24] **Lewis, A. S. and Knowles, G.,** *"Image Compression Using the 2-D Wavelet Transform,"* IEEE Transactions on Image Processing, vol. 1, no. 2, pp.244-250, 1992.

[2.25] **Shapiro, J. M.,** *"Embedded Image Coding Using Zerotrees of Wavelet Coefficients,"* IEEE Transactions on Signal Processing, vol. 41, no. 12, pp.3445-3462, 1993.

[2.26] **Said, A. and Pearlman, W. A.,** *"A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees,"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp.243-249, 1996.

[2.27] **Martucci, S. A. and Sodagar, I.,** *"Zerotree Entropy Coding of Wavelet Coefficients for Very Low Bit Rate Video,"* Proceedings 1996 IEEE International Conference on Image Processing, Lausanne, Switzerland, pp.533-536, 1996.

[2.28] **Martucci, S. A.; Sodagar, I.; Chiang, T., and Zhang, Y. Q.,** *"A Zerotree Wavelet Video Coder,"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1, pp.109-118, 1997.

[2.29] **Barthel, K. U.,** *"A new image coding technique unifying Fractal and transform coding,"* ICIP-94, pp.112-116, 1994.

[2.30] **Kim, J. S.,** *"A fast feature-based block matching algorithm using integral projections,"* IEEE Journal on Selected Areas of Communications, vol. 10, June, pp.968-971, 1992.

[2.31] **Wright, A.,** *Squeezing into the picture*; Electronics World + Wireless World, pp.663-666, August 1992.

[2.32] **Ghafourian, M. A.,** *"Comparison between several adaptive search vector quantisation schemes and JPEG standard for image compression,"* IEEE Transactions on Communications, vol. 43, February-April, pp.1308-1312, 1995.

[2.33] **Nguyen, H.,** *"Concentric-shell partition vector quantisation with application to image coding,"* IEEE Transactions on Communications, vol. 42, February-April, pp.1911-1918, 1994.

[2.34] **Lookabaugh, T.,** *"Variable rate vector quantisation for speech, image, and video compression,"* IEEE Transactions on Communications, vol. 41, January, pp.186-199, 1993.

[2.35] **Qiu, G.,** *"Image coding based on visual vector quantisation,"* Image Processing and its Applications, pp.301-305, 1995.

[2.36] **Kunt, M.,** *Object based image and image sequence coding,* 1990. Elsevier Science.

[2.37] **Zhaoyang, L.,** *Application of Delaunay triangulation in image coding,* 1990. World Scientific.

[2.38] **Moura, J. M. F.,** *"Video over wireless,"* IEEE Personal Communications, February, pp.44-54, 1996.

[2.39] **Oehler, K.,** *"Digital video compression,"* TI Technical Journal, March-April, pp.27-38, 1996.

[2.40] **Pearson, D. E.,** *"Developments in model-based video coding,"* Proceedings of the IEEE, vol. 83, June, pp.892-905, 1995.

[2.41] **Salembier, P.,** *"Region-based video coding using mathematical morphology,"* Proceedings of the IEEE, vol. 83, June, pp.843-857, 1995.

3

Efficient Image Compression Algorithms Implemented on DSP Architecture

The image compression algorithm selected in chapter 2 of this thesis was the Symmetric Wavelet Transform (SWT) followed by *zerotree* construction and Sequential Baseline Coding (SBC). The main reasons that the algorithm was chosen was its computational simplicity and the significant compression ratio improvement available over the well-known JPEG technique. Both of these factors are important when the image compression algorithm is to be implemented on a real-time Digital Signal Processor (DSP) embedded system. Embedded systems typically do not have a lot of memory and the fast access memory required for a real-time system usually costs as much as the processor itself. For a cost sensitive application, the amount of memory and the speed of the processor needs to be minimised.

The reference image size chosen for the examples in chapter 2 was 512 x 512 pixels. In this chapter where a specific hardware implementation of the compression algorithm is considered, the image size is 512 x 288 pixels to match the resolution of the Sharp LZ2326J CCD (Charge Coupled Device) image sensor on board the digital camera. The chosen DSP is the Motorola 56303 processor clocked at 80MHz. The processor contains a 24-bit data bus, and requires 15ns access time SRAM memory

chips to achieve single wait state access to the external memory. The above constraints create the following bandwidth limitations:

- Instructions per second: 80 million
- External bandwidth in bytes per second: 120 million (40 million 24-bit words)

It is easy to calculate such bandwidth limitations, yet they are of no practical use when deciding whether or not there is sufficient processing power to implement a specific algorithm. A thorough understanding of the DSP architecture is critical when creating an efficient implementation of an algorithm. Section 3.1 of this chapter will describe the image compression algorithm in terms of its data flow. This form of description clearly identifies the primitive operations required and the flow of data streams between the primitive operations. Section 3.2 will describe the features of the DSP architecture that allow the primitive operations to be implemented efficiently. The rest of the chapter will examine the specific implementation issues to ensure an efficient implementation of the chosen image compression algorithm.

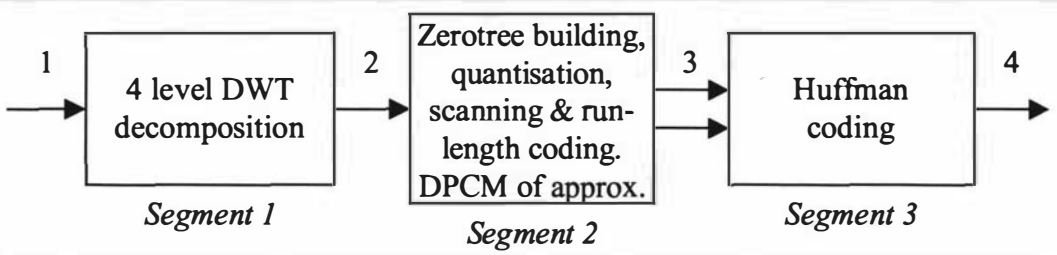
3.1 Data flow description of the image compression algorithm

Figure 3.1 shows the three main segments of the wavelet based image compression algorithm and a description of the interface between each segment. The 8-bit per pixel raw image is initially decomposed by a four level DWT. This produces an equal number of wavelet coefficients as there are pixels in the original image. The wavelet coefficients have a greater dynamic range than 8-bits due to the energy compaction property of the DWT[1]. As the DSP addresses data locations as 24-bit quantities, each wavelet coefficient is stored in a 24-bit word. A significant reordering of the wavelet coefficients is then required to construct a set of zerotrees. The quickest way to achieve the reordering is to reserve enough storage to contain the entire transformed image resulting from the DWT before starting the zerotree construction in segment 2.

[1] Also the convolution of the filter taps with the pixel data naturally increases the dynamic range of the output data due to the accumulation of products.

Each zerotree is quantised, scanned and run-length coded as it is constructed and the resulting symbol stream is stored in memory prior to processing in segment 3. Only the detail coefficients from levels one to four are used in the construction of the zerotrees. This leaves the level four approximation image to be coded separately. Delta Pulse Code Modulation (DPCM) is a suitable technique to encode the quantised approximation image, as described in section 2.3. The approximation image is a 256^{th} of the size of the original image as it has been downsampled by a factor of $2^4 = 16$ in each dimension.

The stored symbol stream resulting from the detail and approximation coefficient processes in segment 2 is next entropy encoded in segment 3. Three 8-bit symbols can be packed into a 24-bit word in order to maximise the use of the 24-bit data locations. The optimal Huffman table is used to entropy encode the symbol stream, this table is calculated from the frequency distribution of the used symbols. The frequency distribution is produced in segment 2 by dynamically updating the symbol probability table as each symbol is assigned to the stored symbol stream. Segment 3 begins with the calculation of the optimal Huffman table based on the symbol probability table. The Huffman table is then used as a lookup table as each symbol is read in from storage. The table is used to form a variable length bit pattern which is concatenated to form an output Huffman bitstream stored in 24-bit words.



Interface	Description	Sampling Rate	Wordlength Bits
1	Image pixels	F	8
2	Wavelet coefficients	F	24
3	Coded symbols	<<F	8
	Symbol probability table	fixed	24
4	Huffman bitstream	<<F	24

Figure 3.1 – Wavelet compression signal processing functional diagram

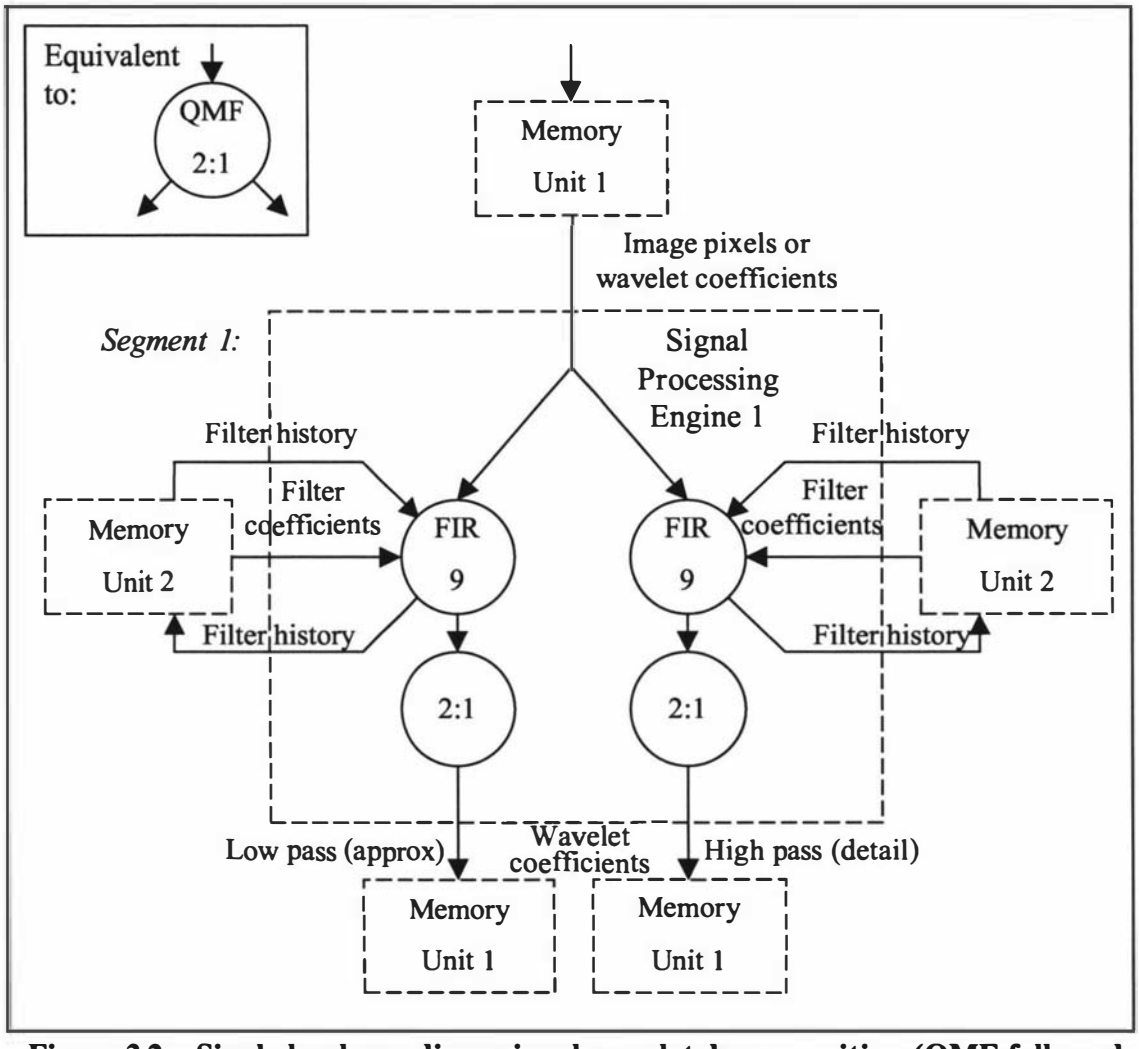


Figure 3.2 – Single level one-dimensional wavelet decomposition (QMF followed by 2:1 decimation) signal processing flow graph

Figure 3.2 through Figure 3.7 represent a hierarchical Data Flow Graph (DFG) of the image compression algorithm described above. The circles are *nodes* that contain primitive signal processing operations. The rectangular boxes are *memory units* where "Memory Unit 1" represents mass off-chip storage and "Memory Unit 2" represents more limited on-chip storage with fast access. "Memory Unit 3", which appears only in Figure 3.3 and Figure 3.4, represents off-chip scratch-pad memory for the large temporary structures required when computing the two-dimensional wavelet decomposition. The connecting lines are *queues* of data which represent the flow of data between the various nodes and memory units.

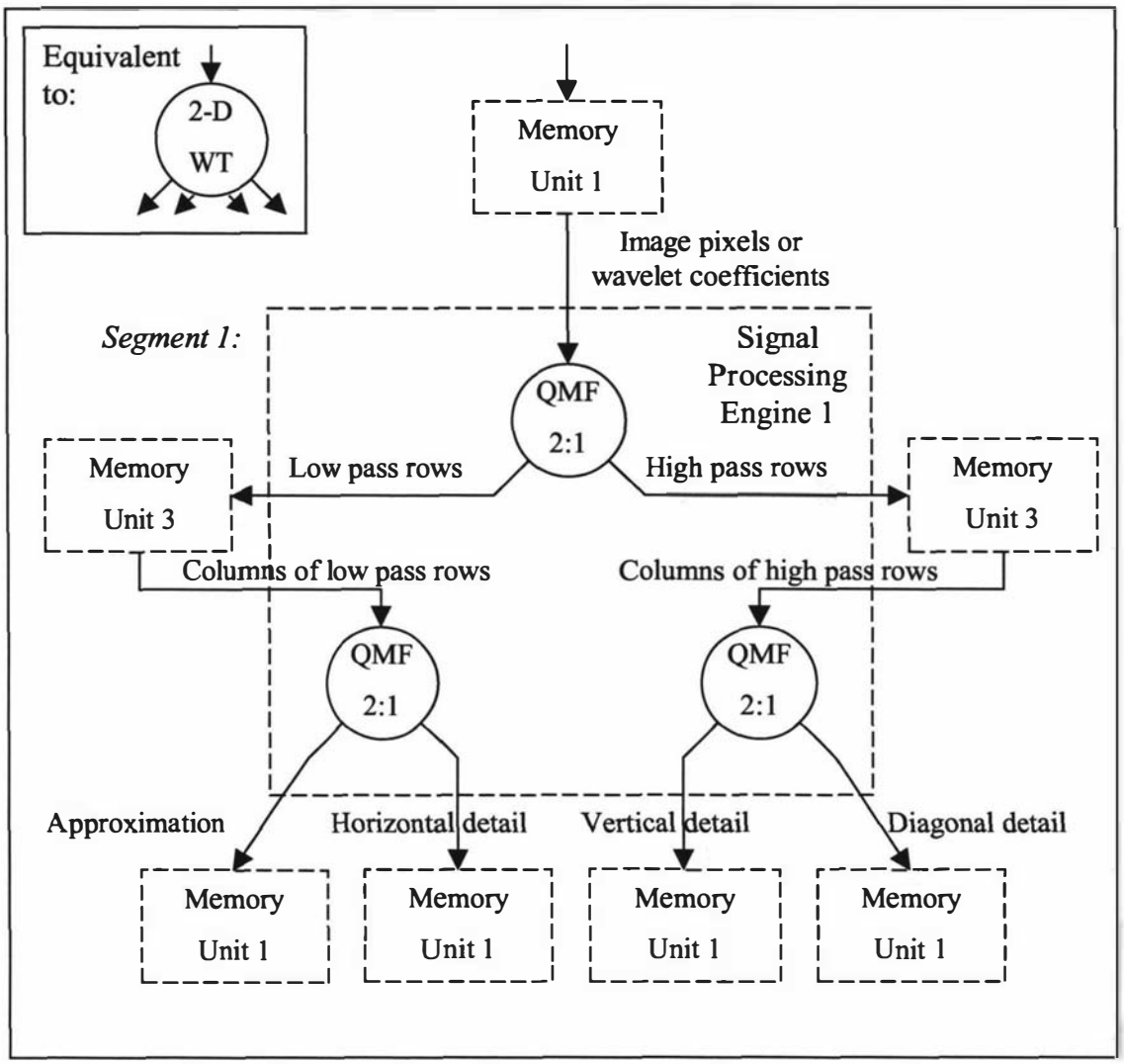


Figure 3.3 – Single level two-dimensional wavelet decomposition signal processing flow graph

Figure 3.2 defines and describes a Quadrature Mirror Filter (QMF) with 2:1 decimation which is used as a primitive node operation in Figure 3.3. Figure 3.3 defines a single level two-dimensional wavelet decomposition which is used as a primitive node operation in Figure 3.4.

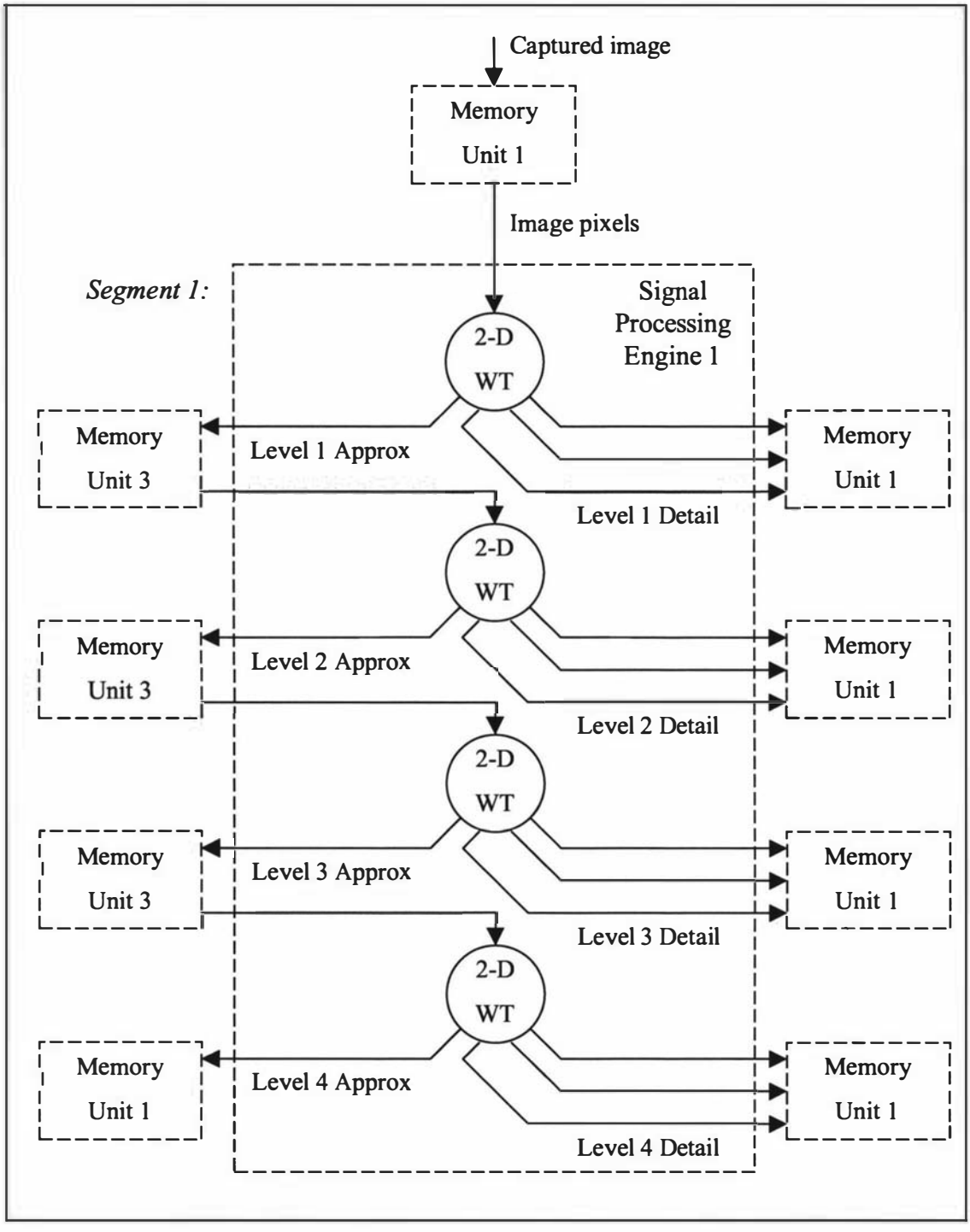


Figure 3.4 – Four level two-dimensional wavelet decomposition signal processing flow graph

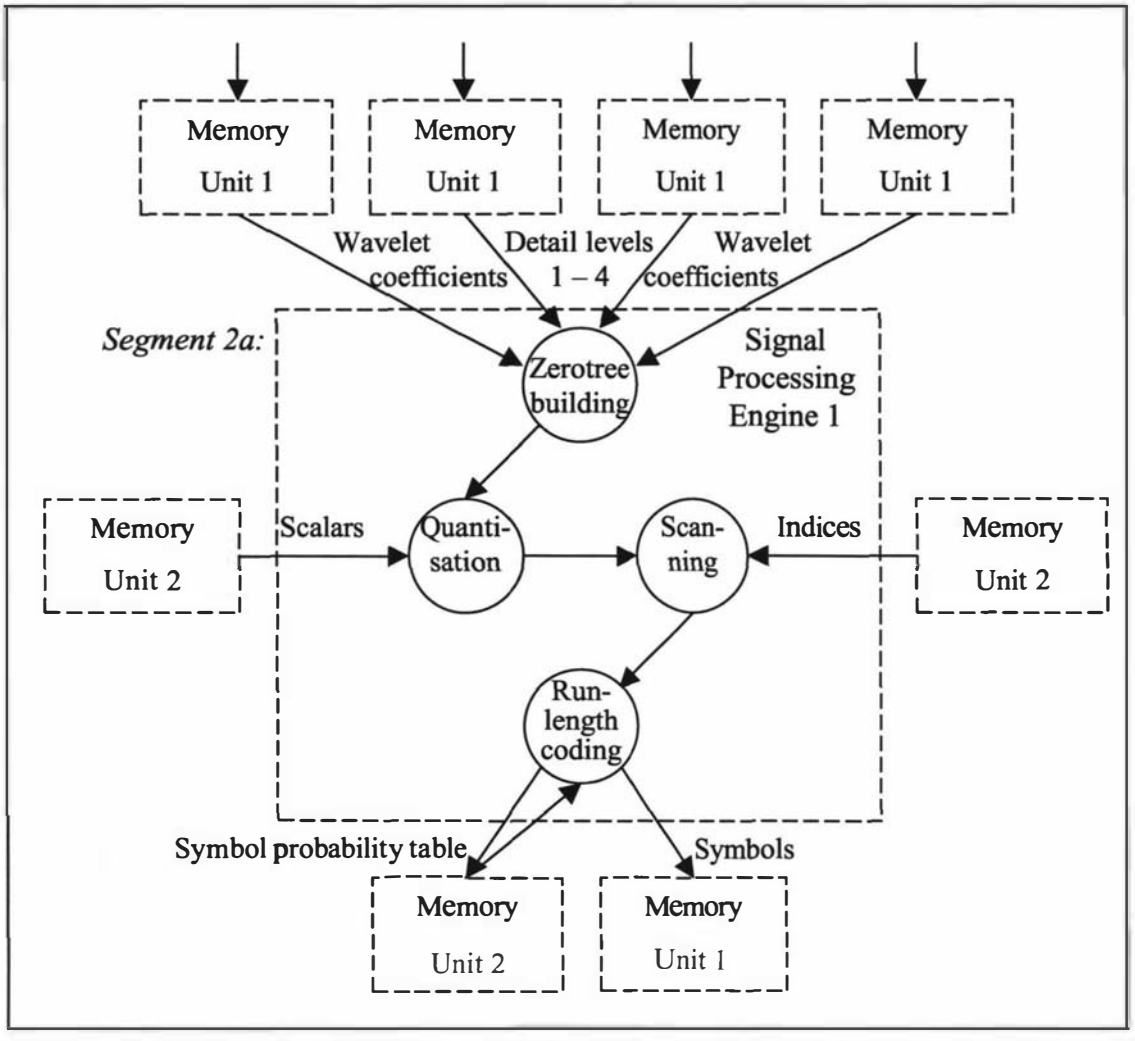


Figure 3.5 – Zerotree building, quantisation, scanning and run-length coding of detail coefficients signal processing flow graph

Segment 1 of the algorithm, the four level two-dimensional DWT, is shown in Figure 3.4. Segment 2 of the algorithm is divided into two parts, the processing of the detail coefficients in Figure 3.5 and the processing of the approximation coefficients in Figure 3.6. The entropy encoding of the symbol stream is shown in Figure 3.7 and represents segment 3 of the algorithm.

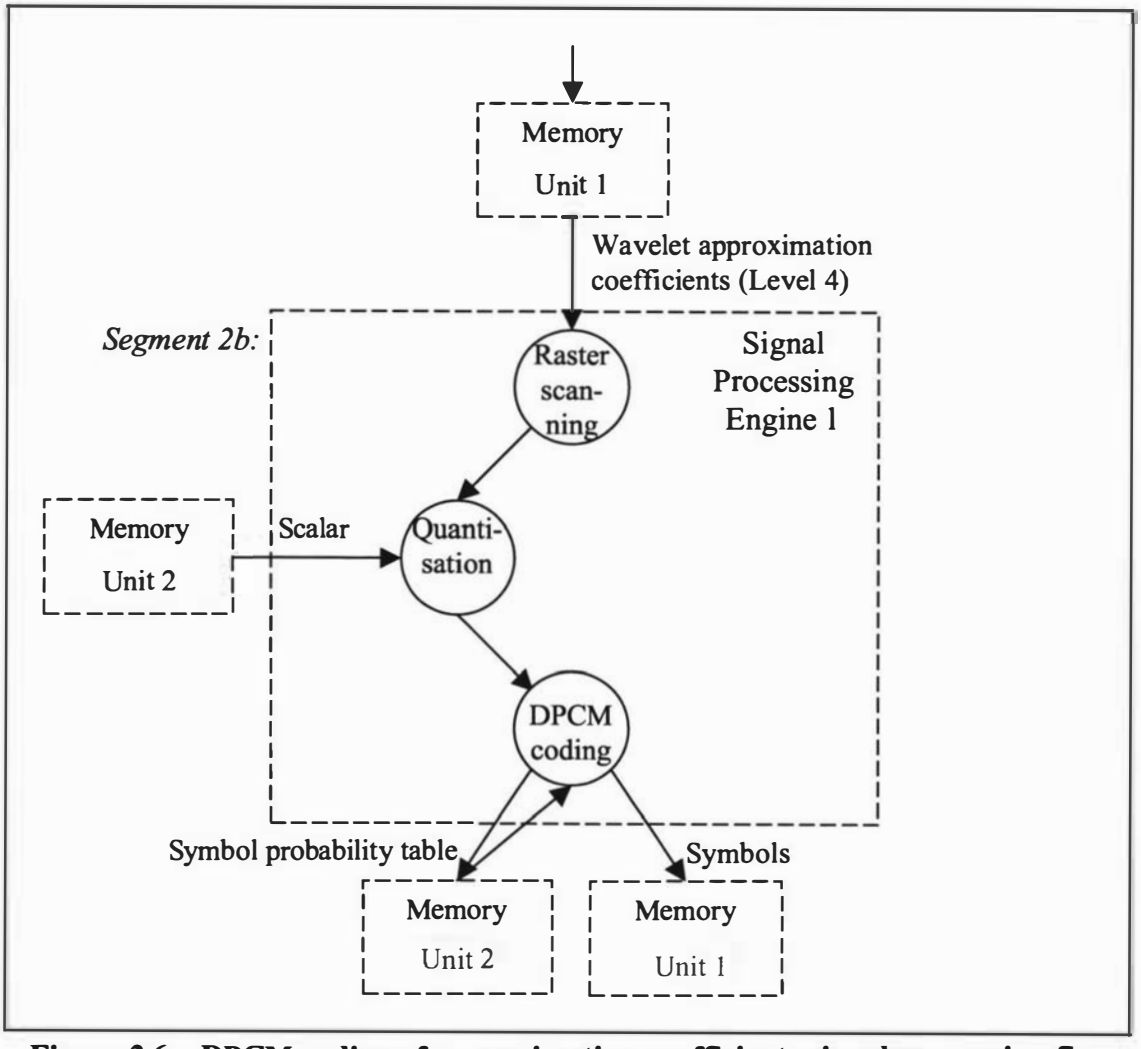


Figure 3.6 – DPCM coding of approximation coefficients signal processing flow graph

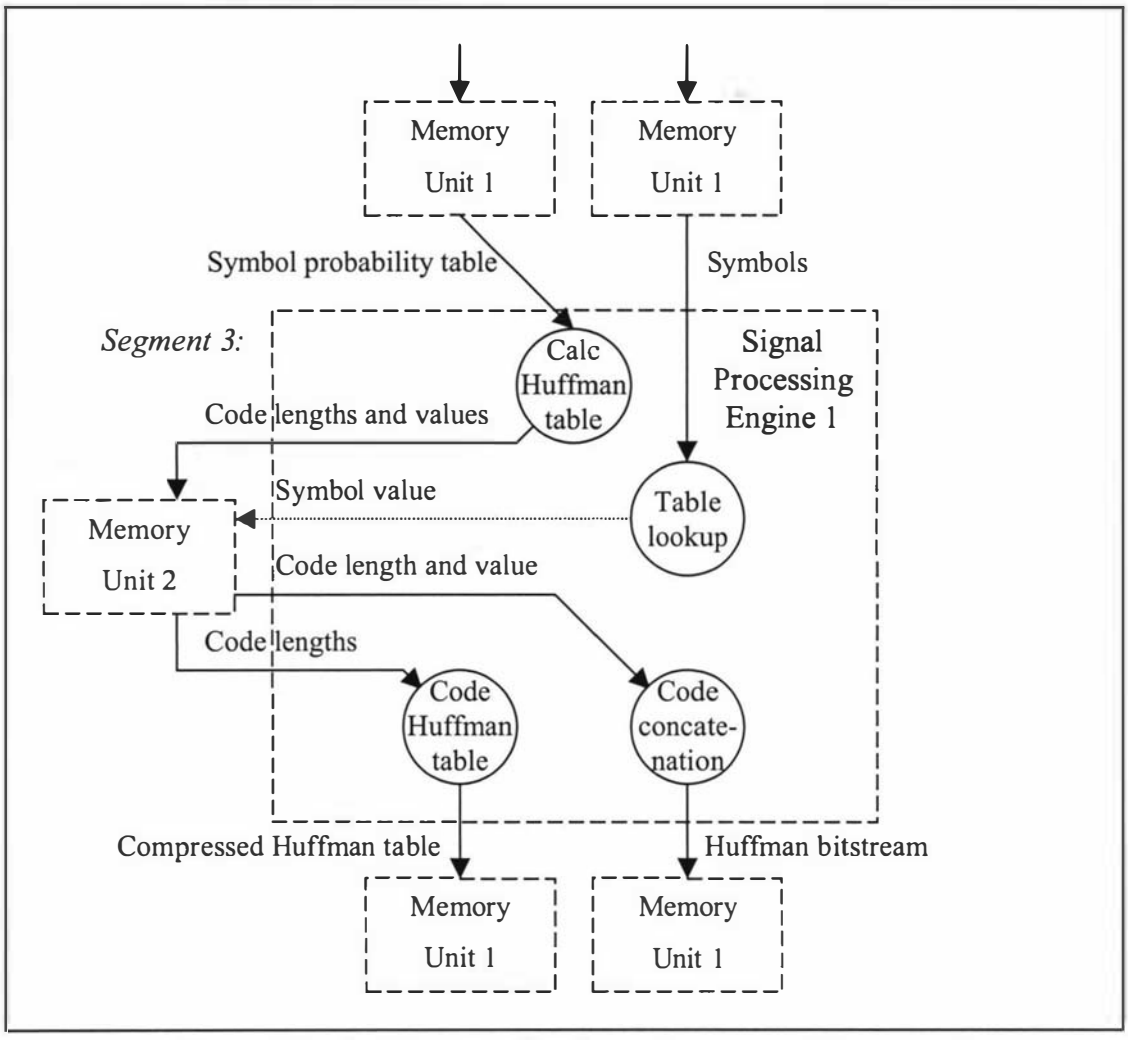


Figure 3.7 – Huffman coding signal processing flow graph

3.2 Specialised architectural features of the DSP

This section will examine a number of specialised architectural features of the Motorola 56303 DSP that can be used to efficiently implement the primitive operations detailed in the DFGs.

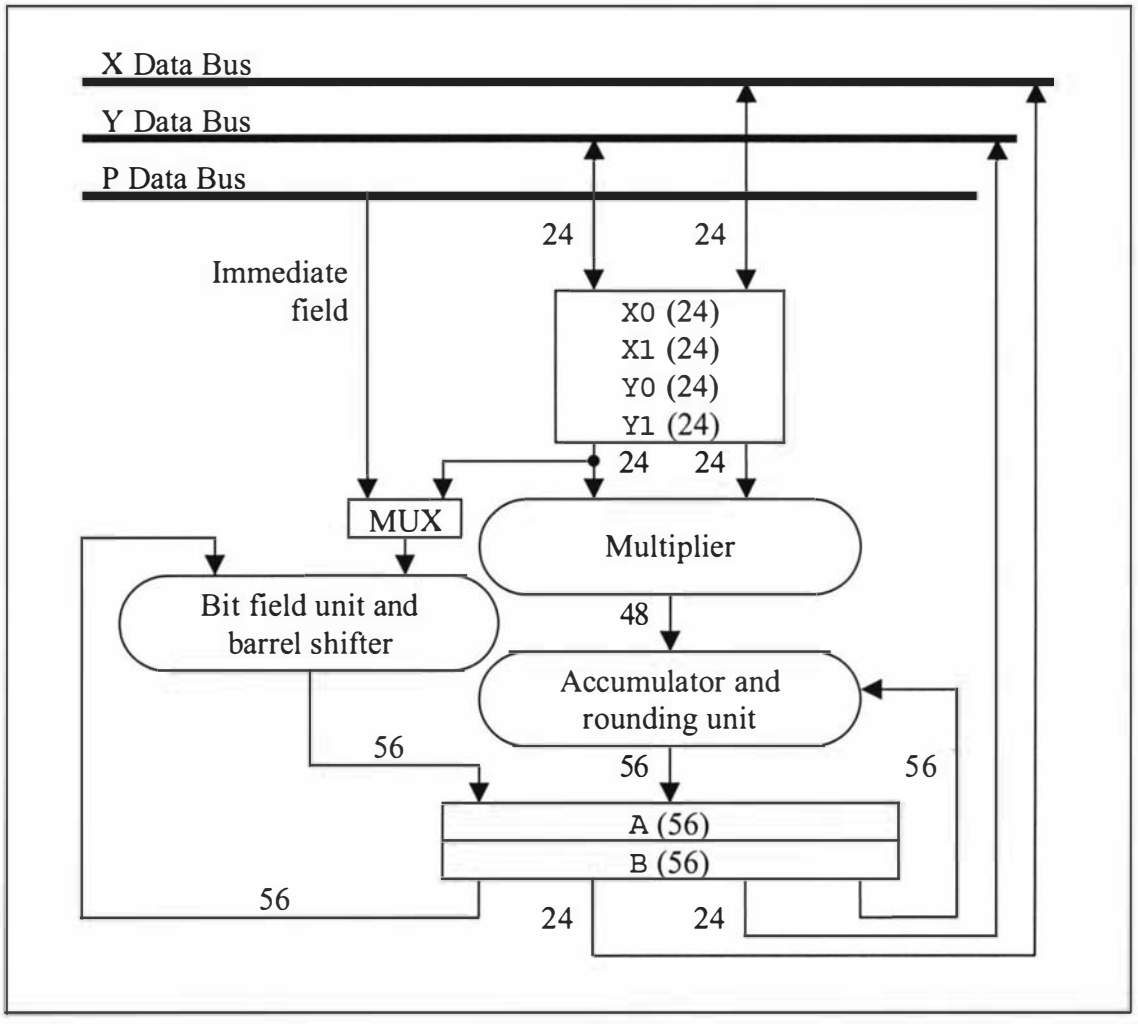


Figure 3.8 – Subset of data Arithmetic Logic Unit (ALU) block diagram

3.2.1 MAC and advanced Harvard architecture

The Multiply and Accumulate (`mac`) operation is at the heart of most digital signal processing operations. Much of the DSP architecture is designed to optimise the data

throughput to support an efficient `mac` operation. Figure 3.8 shows a subset of the data Arithmetic Logic Unit (ALU) of the Motorola 56303 DSP. An important feature of the ALU is the inclusion of two 56-bit accumulators (`A` and `B`) attached to the Multiplication and Accumulation function chain. Either accumulator can be used to accumulate a set of products produced by the multiplier unit. The operands for the multiplication are generally loaded into a pair of source registers chosen from a total set of four 24-bit source registers, `X0`, `X1`, `Y0` and `Y1`. The 48-bit product is accumulated in one of the 56-bit accumulators. The additions of 8 upper-bits in each of the accumulators provide a generous amount of guard-bit protection to prevent overflow from the accumulation of multiple 48-bit products.

All of the data ALU operations are performed in two machine cycles in pipeline fashion so that a new instruction can be initiated in every machine cycle. This yields an effective execution rate of one instruction per machine cycle. In order to take advantage of the rapid computational ability of the DSP, the data path to and from the data ALU must be sufficient. There needs to be enough bandwidth to the data ALU to allow for a new set of operands to be read into the ALU as well as a result written out from the ALU in a single machine cycle. The advanced Harvard architecture of the Motorola DSP allows three distinct memory spaces to exist on the DSP. Internally there are three separate address and data buses for the Program (`P:`) memory, X Data (`X:`) memory and Y Data (`Y:`) memory. External to the DSP chip these three sets of buses are multiplexed into a single bus to reduce the external pin count. The advantage of having three separate buses internal to the DSP is that three reads or writes to memory can occur in a single machine cycle. One of the reads will be to the `P:` memory space to fetch the next instruction to be executed. This allows for any combination of two read and write operations to take place concurrently in the `X:` and `Y:` memory spaces.

The `mac` instruction coupled with the advanced Harvard architecture of the DSP is an essential architectural feature for rapid calculation of the convolution function. In signal processing applications, convolution is used to implement the correlation function, digital filters such as the Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters as well as the QMF filter required for wavelet decomposition.

The QMF calculation will be covered in more detail in section 3.3.1. Quantisation is a necessary part of a lossy compression algorithm, see section 2.2. Uniform quantisation can be performed inexpensively with a single cycle fractional multiply with hardware rounding. This operation is shown in section 3.3.2.

3.2.2 Indirect addressing modes

Indirect addressing is used in many stages of the algorithm to access data structures such as image data, wavelet filter taps and scanning pattern data. The Motorola DSP provides a dedicated Address Generation Unit (AGU) to compute address register updates to a set eight address registers (R0-R7). The AGU can operate in parallel with the data ALU to ensure that each instruction can still be executed in a single machine cycle. The AGU itself is split into two further address Arithmetic Logic Units (ALUs). Each of the address ALUs operate on a subset of the eight address registers to allow two address register updates to occur each machine cycle. This structure is shown in Figure 3.9.

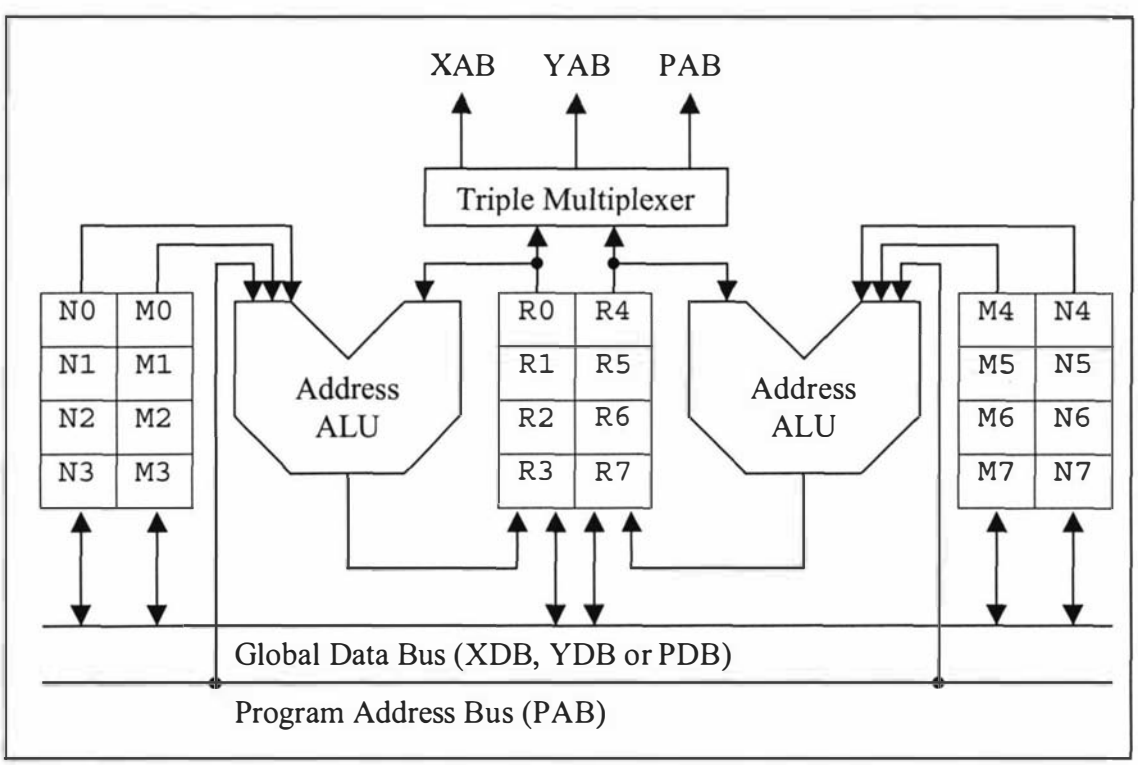


Figure 3.9 – Subset of Address Generation Unit (AGU) block diagram

The two address ALUs are identical and can each perform the following updates on an address register:

1. A simple increment or decrement can be performed.
2. The value of the respective offset register (`N0-N7`) can be added to or subtracted from the address register.
3. Any addition or subtraction can be constrained to fall within a circular buffer by using the modulo value stored in the respective modifier register (`M0-M7`).
4. Any addition or subtraction can be performed with bit reversed carry if required.

Indirect addressing with modulo and offset address modifications is particularly useful in creating circular buffers and scanning data structures in a non-linear ordering. Sections 3.3.1 and 3.3.2 will illustrate the use of these data addressing methods in detail.

3.2.3 Data marshalling and pipelining

A critical difference between one-dimensional audio and two-dimensional image processing is the amount of data required to represent the signal over time. In audio processing, a single sample is all that is required to represent the audio signal amplitude level at a given time. With image processing at a frame size of 512 x 288 pixels, 144 kbytes of 8-bit samples are required to represent a single image at a given time. The amount of fast access internal SRAM present on the Motorola DSP is only 4 kwords where each word is a 24-bit quantity. This small amount of SRAM is adequate for most one-dimensional signal processing applications that use a short window of consecutive samples. The internal SRAM is not sufficient to hold even a small subset of a two-dimensional image so the image must be stored in external SRAM in its entirety.

A significant penalty occurs when accessing image data from external SRAM. At least a single cycle wait state is inserted by the DSP when accessing the SRAM. In addition because the `X:` and `Y:` data spaces are multiplexed on the external bus, there

can only be a single access in progress at any point in time. With data stored in the internal `X:` and `Y:` SRAM memory spaces, there is no wait state penalty and two accesses can occur in each machine cycle.

The above limitation in memory capacity and access speed requires that the image be processed as a sequence of smaller image regions. A single level of the two-dimensional wavelet decomposition is a result of many executions of the QMF primitive as shown in Figure 3.3. The QMF operation is performed on a single row or column of data at a time, in the case of a 512 x 288 pixel image this will be no more than 512 elements. Each time the level of decomposition increases, the size of the rows and columns decrease by a factor of two. By ensuring that the correct row or column of data to be processed is present in the internal SRAM, the QMF operation can be implemented efficiently by taking full advantage of the advanced Harvard architecture.

The term "data marshalling" has been used to describe the process of ensuring that only the necessary data for the current primitive operation is present in the limited resource of the internal SRAM. The usage of the term is consistent with the meaning applied by Microsoft when describing a method to ensure the correct data structures are in place when performing remote procedure calls. As each primitive operation has one or more corresponding output data streams, there needs to be sufficient free space to store these output data streams in internal SRAM as they are being produced. If the central data ALU was required to shift the needed data to and from external SRAM there would be no advantage in processing the data internally due to the overhead required in transferring the data. The Motorola DSP has a third execution unit, in addition to the data ALU and the Address Generation Unit, called a Direct Memory Access (DMA) controller. The DMA controller contains six channels, which can each be programmed to transfer data to and from the external SRAM without any further intervention of the data ALU.

Effective use of the DMA controller can ensure that the primitive operations running on the DSP core are constantly supplied with the necessary input data in the internal SRAM for processing. At the same time another channel in the DMA controller can

be used to transfer the output data streams to external SRAM to free up the internal SRAM resource. In the ideal situation this architecture would allow the DSP core to run as efficiently as possible as all of the processing would be based in internal SRAM. The DMA controller would be responsible for pipelining the large amount of external image data through the DSP core and then pipeline the results from the core back to the external SRAM. The use of DMA transfer to internal SRAM with concurrent DSP core processing in SRAM has the potential for conflict. If both units try to access memory locations within the same 128 word page then the accesses cannot be concurrent and the DMA controller or DSP core will suffer a wait state. By careful design of memory usage it is usually possible to ensure that the buffers being used by the DMA controller are within a separate page from the buffers being used by the DSP core.

The techniques of data marshalling and pipelining are used in all of the stages of the image compression algorithm. Sections 3.3 and 3.4 explain the efficiencies obtained through the use of these techniques.

3.2.4 Barrel shifter

The data ALU block diagram shown in Figure 3.8 contains a bit field unit and barrel shifter. The contents of either accumulator can be shifted up to 56 bits left or right in a single clock cycle. Multibit masks can be generated automatically to allow the insertion of a multibit pattern into a particular offset in either of the 56-bit accumulators. These specialised hardware features and the large 56-bit accumulators allow the concatenation of Huffman codes to be implemented efficiently.

3.3 Specific algorithm implementations

This section examines three specific primitive operations that were detailed in the DFGs. The efficient implementation of these primitive operations will ensure that the overall algorithm is efficiently executed on the DSP. Each of the primitives are

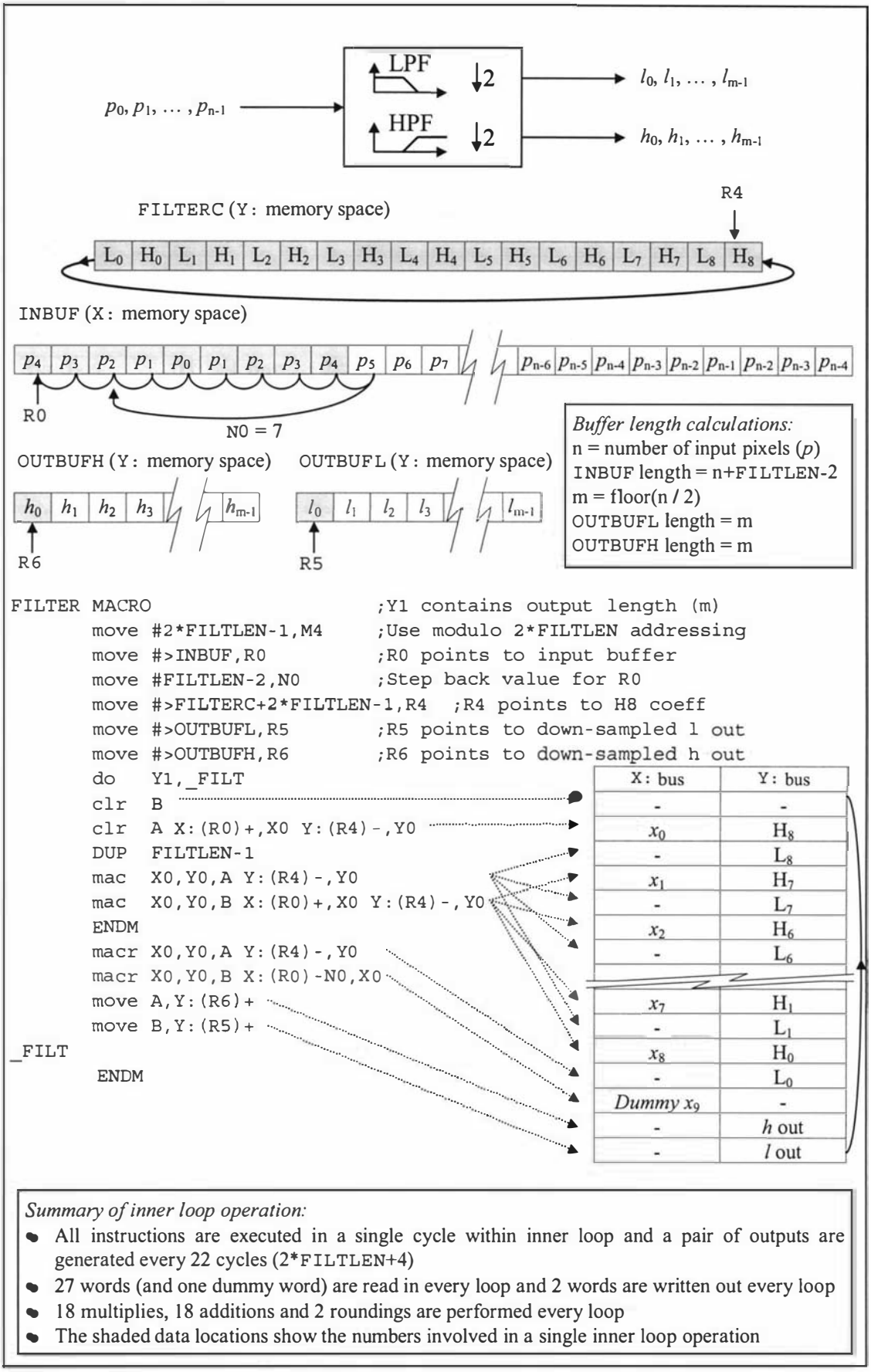
executed many thousands of times over the image which means that any small improvement that can be made to the implementation of each primitive will have a significant effect on the final performance of the algorithm.

3.3.1 QMF kernel implementation

The kernel algorithm of the DWT is the QMF operation which operates on a single row or column of data and produces two output streams each at half the input data rate. The DFG in Figure 3.2 shows that the QMF is constructed from two separate FIR filters that share a common input data stream. Each FIR filter operation requires access to the previous nine input data points *p(n-i)* and a store of nine filter coefficients *L(i)* or *H(i)*. The vector dot product of the nine input data points and the nine filter coefficients for each FIR filter produces a pair of output results as shown in equation (3.1).

$$\begin{aligned} l(n) &= \sum_{i=0}^{8} L(i)p(n-i) \\ h(n) &= \sum_{i=0}^{8} H(i)p(n-i) \end{aligned} \qquad \text{for even } n \qquad (3.1)$$

Due to the downsampling by a factor of two after filtering, only even values of *l(n)* and *h(n)* are calculated. Figure 3.8 shows a subset of the data Arithmetic Logic Unit (ALU) of the Motorola 56303 DSP. An important feature of the ALU is the inclusion of two 56-bit accumulators (`A` and `B`) attached to the Multiplication and Accumulation function chain. This allows both the low *and* high pass filter operations to be calculated in a single loop over the nine input data points *p(n-i)*. Accumulator `A` maintains the sum of products associated with the high pass filter and Accumulator `B` is used for the low pass filter. Figure 3.10 shows an assembly language macro that calculates the two filtered and downsampled output vectors *l(n)* and *h(n)* from the input vector *p(n)*. The internal 4 kwords of data SRAM is divided into two address spaces, `X:` and `Y:`. Every cycle, the core can access a word from both address spaces without any penalty. This allows the next filter coefficient and data point to be read into the ALU in the same cycle as the previous data is being multiplied and accumulated.




```
FILTER MACRO                    ;Y1 contains output length (m)
       move #2*FILTLEN-1,M4     ;Use modulo 2*FILTLEN addressing
       move #>INBUF,R0          ;R0 points to input buffer
       move #FILTLEN-2,N0       ;Step back value for R0
       move #>FILTERC+2*FILTLEN-1,R4  ;R4 points to H8 coeff
       move #>OUTBUFL,R5        ;R5 points to down-sampled l out
       move #>OUTBUFH,R6        ;R6 points to down-sampled h out
       do   Y1,_FILT
       clr  B
       clr  A X:(R0)+,X0 Y:(R4)-,Y0
       DUP  FILTLEN-1
       mac  X0,Y0,A Y:(R4)-,Y0
       mac  X0,Y0,B X:(R0)+,X0 Y:(R4)-,Y0
       ENDM
       macr X0,Y0,A Y:(R4)-,Y0
       macr X0,Y0,B X:(R0)-N0,X0
       move A,Y:(R6)+
       move B,Y:(R5)+
_FILT
       ENDM
```

X: bus	Y: bus
-	-
x_0	H_8
-	L_8
x_1	H_7
-	L_7
x_2	H_6
-	L_6
x_7	H_1
-	L_1
x_8	H_0
-	L_0
Dummy x_9	-
-	*h* out
-	*l* out

Summary of inner loop operation:

- All instructions are executed in a single cycle within inner loop and a pair of outputs are generated every 22 cycles (2*FILTLEN+4)
- 27 words (and one dummy word) are read in every loop and 2 words are written out every loop
- 18 multiplies, 18 additions and 2 roundings are performed every loop
- The shaded data locations show the numbers involved in a single inner loop operation

Figure 3.10 – Inner loop operation of the wavelet filter bank for filters of size FILTLEN = 9

Note that the R0 data pointer is incremented a total of 9 times to calculate the first filter bank pair of outputs and then decremented by 7. This operation leaves R0 pointing to the correct starting point for the next iteration of the filter bank so that only every second output pair is actually calculated. The fact that the DSP architecture supports a decrement of N0 = 7 in a single cycle whilst multiply, accumulate and rounding operations are concurrently occurring allows this structure to be implemented *very* efficiently.

The low and high pass filter coefficients (L_0, L_1, ... , L_8 and H_0, H_1, ... , H_8), are stored in an interleaved fashion in Y: memory space so that the input data can be accessed simultaneously in the X: memory space. The DSP is configured to decrement the R4 register that points to the next filter coefficient within the bounds of a circular buffer through the use of the M4 modulo register. This feature of the DSP eliminates the need to explicitly reset the filter coefficient pointer at the end of each loop.

The presence of data on the X: and Y: buses are shown on Figure 3.10 for the wavelet filter bank calculations. It can be seen that for every second multiply and accumulate instruction, there are two parallel memory accesses which read in the next piece of input data and the next high pass filter coefficient. The other multiply and accumulate instructions only need to read in the next low pass filter coefficient as a single data stream is being filtered by two sets of filter coefficients in this algorithm.

The entire inner loop is encompassed by a hardware "do" loop which has no overhead other than a 5 cycle setup time for the first iteration. The assembly code given in Figure 3.10 has been optimised for the wavelet filter bank algorithm and produces an output for both the high and low pass filters once every 22 machine cycles. The Motorola 5630x DSP family manual [3.1] contains a set of optimised benchmark signal processing kernels. The wavelet filter bank kernel assembler code is based on the optimal "Real * Complex Convolution" kernel provided by Motorola. If the filter length were to be extended then the cycle count would have the relationship to filter length as described by equation (3.2).

$$Cycles = (2 \times Filtlen) + 4 \tag{3.2}$$

At the end of the multiply and accumulate set of operations there is a fixed overhead of 4 cycles needed to accommodate the clearing of the two accumulators and the storing of the two filter output values.

3.3.2 Tree building and coefficient coding implementation

There are two classes of coefficients to be encoded into a stream of symbols prior to the entropy encoding stage, the detail and the approximation coefficients. The majority of the coefficients are the detail coefficients corresponding to the high frequency information in the image. The DFG in Figure 3.5 shows the data flow associated with the zerotree building, quantisation, scanning and run-length coding processes for the detail coefficients. The coding process for the smaller set of approximation coefficients shown in Figure 3.6 is simple by comparison. One-dimensional DPCM coding of the quantised approximation coefficients is easily implemented on the DSP without any need for advanced pipelining or data marshalling.

3.3.2.1 Zerotree construction

Figure 3.5 implies that a single zerotree is constructed from four distinct pools of detail coefficients, each pool relating to a separate level of detail. To construct the zerotree only a small number of coefficients (1, 4, 16 or 64) are required to be read in at a time from each pool of data. A single 85 element zerotree requires an 8 x 8 region from level 1 of the decomposition, followed by a 4 x 4 region from level 2, a 2 x 2 region from level 3 and a single element for the head of the tree from level 4. Figure 3.11 shows the layout in external memory of the different sized levels of decomposition and the relative sizes of the coefficient regions to construct the top-left zerotree of a 64^2 pixel image.

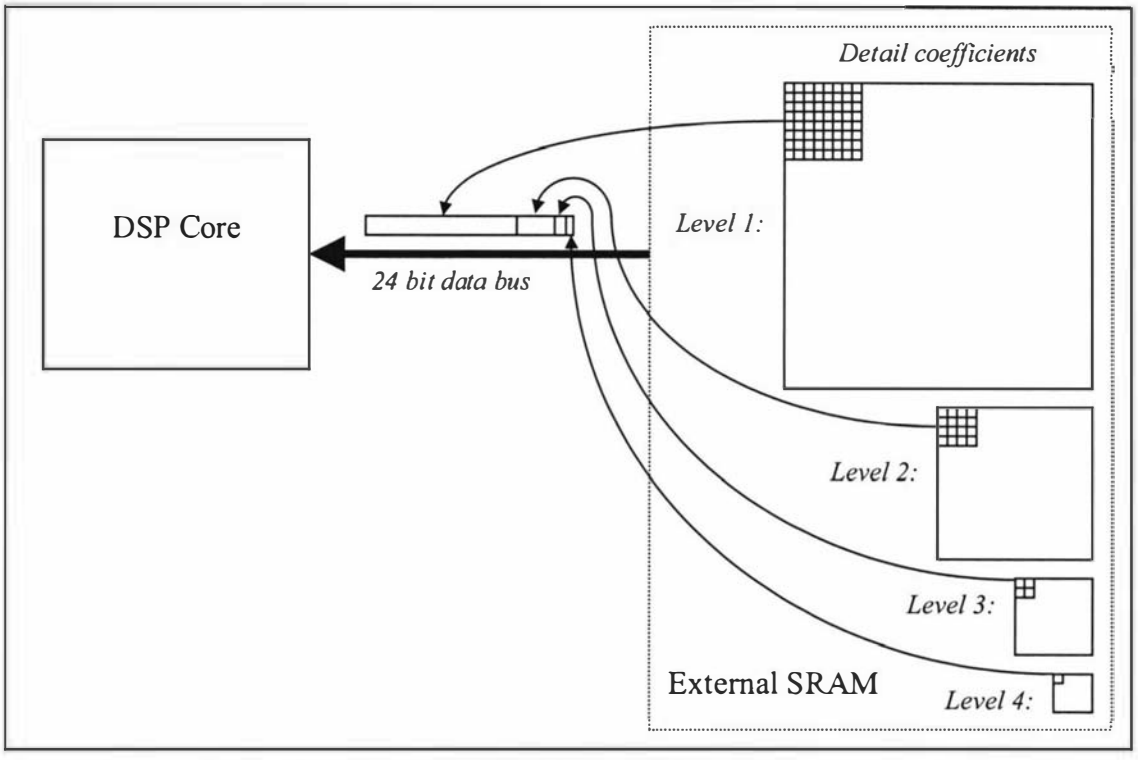


Figure 3.11 – Reading from four pools of data to construct a zerotree

The two-dimensional regions to be read into the DSP core are located within larger two-dimensional structures, each representing the entire image at a particular scale and orientation. The addressing techniques required to read in the required data in the correct order is non-trivial for a single tree. This is further complicated by the fact that the trees are required to be streamed into the DSP core in sequence to cover the complete image. Figure 3.12 shows the offsets associated with reading a series of 8 x 8 element blocks from a larger 32 x 32 element structure in memory. The straight-line arrows represent the operations required to read a single block from memory. Eight elements are read from the start address of the block then an offset of 24 is added to point to the beginning of the second line of the block. This process is repeated eight times.

At the completion of a single block read, one of the two curved arrow offsets are added to point to the next block to be read in. In this example there are 4 x 4 blocks within the complete structure. An offset of -224 is added to the end of each block to locate the next block within a row. After a complete row of blocks has been read in, no offset is required to point to the start of the next row of blocks.

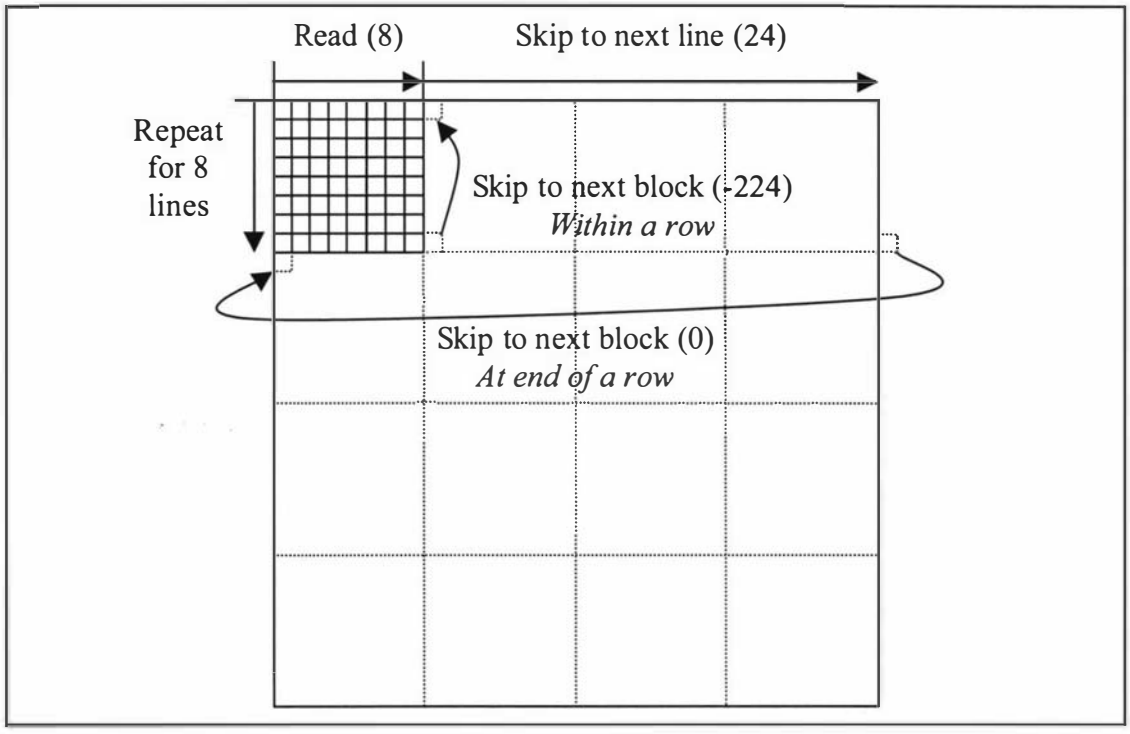


Figure 3.12 – Offsets associated with reading a sequence of blocks

In a pipelined system, where a block at a time is processed by the DSP core, it is necessary to transfer a new block into the internal SRAM without using the arithmetic or addressing units of the DSP. The DMA controller is programmed at the start of a new block to read in an 8 x 8 element structure from external SRAM with an offset of 24 added to the end of each row. This programming is accomplished by initialising a set of registers associated with a particular DMA channel. The available DMA registers are:

- Source start address [DSRx]
- Destination start address [DDRx]
- Size of block (height and width) [DCOx]
- Offset at end of each row [DORx]
- Control register [DCRx]

Once the control register is written, the DMA controller initiates the memory transfer while the DSP core processes at full speed the previous block of data read into a separate buffer. After the core finishes processing the current block, a check is made

to ensure the DMA channel has completed the transfer. The core then reprograms the DMA channel to read the next block of data into the core by simply updating the Start and Destination address registers and re-enabling the Control register. A timeline for the double buffered DMA and DSP core processing is shown in Figure 3.13.

The upper section in Figure 3.13 shows for any instant in time which block of data occupies each of the two SRAM buffers on the DSP core. A block of data occupies a buffer for as long as it takes to read in the block using DMA and the time it takes for the DSP core to process it.

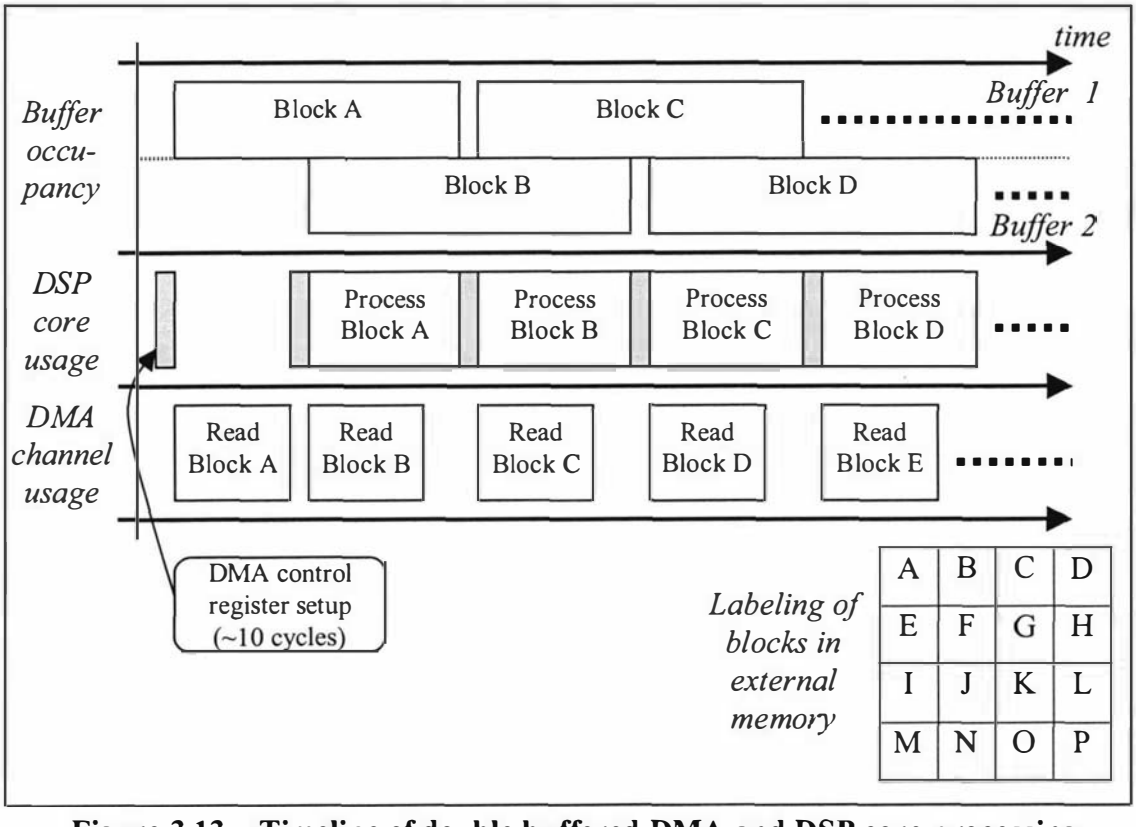


Figure 3.13 – Timeline of double buffered DMA and DSP core processing

The next two timelines in Figure 3.13 show the usage of the DSP core and the DMA controller. The shaded region represents the small overhead required for the DSP core to setup the DMA control registers between processing blocks of data. At an 80 MHz clock speed the 10 cycle setup time lasts only 125ns, this is insignificant when compared to the thousands of clock cycles taken to process the data within a buffer. The final timeline shows the case where the DSP core is the bottleneck in the system

since there is no stalling of the DSP core. This is the preferred situation as maximum processing throughput is achieved when the core is the bottleneck in the system.

By using a double buffer and the DMA controller, it is possible to increase the effective throughput of the DSP core. If the DMA controller or a double buffer was not used then the DSP core would spend a considerable amount of time copying data to and from external memory. There are expenses associated with using double buffering and DMA which need to be justified by the gain in processing throughput. Double buffering requires twice as much internal SRAM to be used to hold a block of input data. Given that the internal SRAM is of a fixed size, the largest amount of data that can constitute a block is limited to half the internal SRAM capacity. A small expense with using the DMA controller is the overhead associated with setting up the control registers to read in each block of data.

For algorithms that do not process the input data *in-place*, the internal SRAM resource must also have room to hold the output data structure. An in-place algorithm uses the input buffer to also hold the output data eliminating the need for an additional output buffer. In a double buffered system, *both* the input and output buffers must be doubled in number. This further limits the size of the blocks of data that can be processed at a time. None of the algorithms in the image compression system are implemented as in-place algorithms so separate input and output buffers are required for each stage of the algorithm.

Figure 3.11 shows that to construct the zerotree, two-dimensional blocks from four separate regions in memory are required to be read into the core. Additionally, the run-length coded zerotree needs to be written back out to external SRAM to free up an output buffer. This data marshalling operation can be performed automatically by using five of the six DMA channels of the DMA controller. Each channel can be triggered to start as soon as the preceding channel has completed a transfer. This protocol allows the DSP core to program all five channels in advance without having to manually trigger each channel to commence its transfer. Channel 1 is used for writing out the previously run-length coded zerotree and channels 2-5 are used to read in the four memory regions needed to construct the next zerotree.

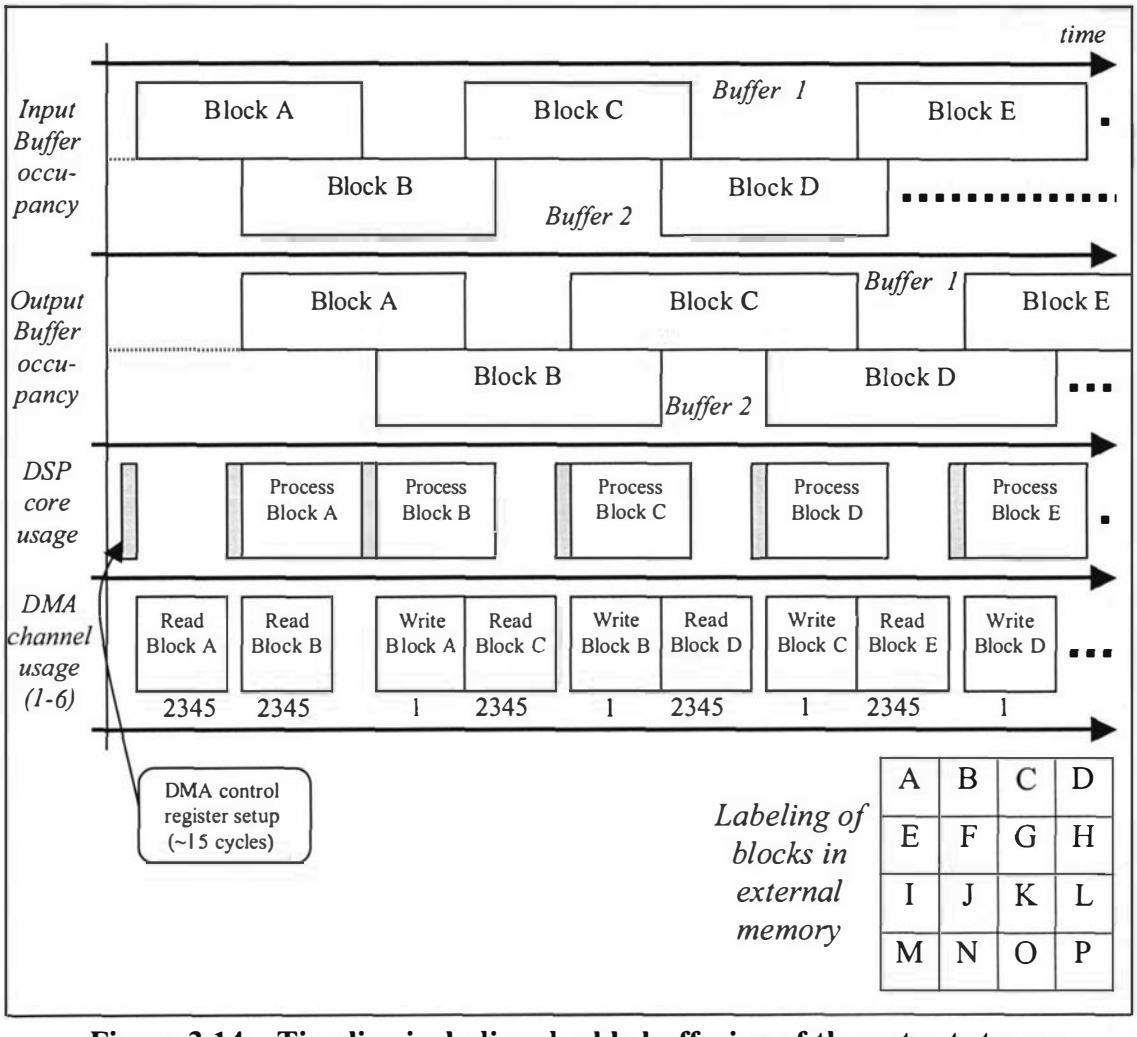


Figure 3.14 – Timeline including double buffering of the output stream

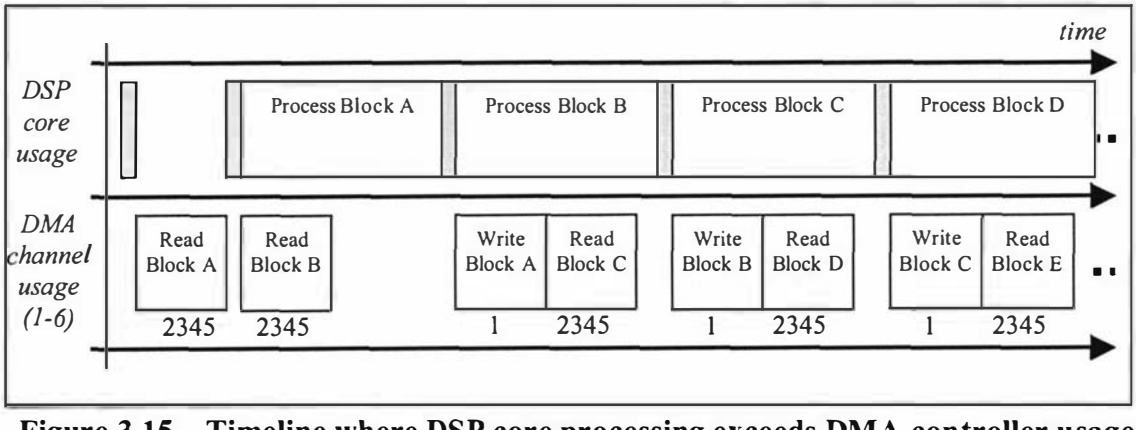


Figure 3.15 – Timeline where DSP core processing exceeds DMA controller usage

An expanded timeline is shown in Figure 3.14 for five DMA channels and the double buffering of both the input and output streams. In this hypothetical example, the combined time required to write out an old output buffer and read new data into an

input buffer exceeds the time required to process the data. In consequence the DSP core is stalled until the DMA engine has finished its transfers. The bottleneck here is the DMA controller bandwidth to external memory. Figure 3.15 shows the alternative and preferred case where the DSP core limits the throughput of the system. If a considerable number of clock cycles of processing are required for each input data point, then the DMA transfers will be completed before the core processing. This will cause a stall in the DMA controller as it waits for new instructions from the DSP core. This second form of stalling is preferred over the first as the DSP core is at its maximum processing throughput and is not limited by inefficiencies arising from the external memory interface.

3.3.2.2 Zerotree coefficient coding

Once the 85 coefficients that make up a zerotree have been read into the internal SRAM input buffer, quantisation, scanning and run-length coding are performed to achieve lossy data compression. Each of the four levels in the zerotree is quantised by a different scale factor to allow for the varying quantisation step size for each level of the decomposition[2]. To maximise system throughput, the algorithm for zerotree coefficient coding is split into two passes within the internal SRAM buffer. The initial pass over the coefficients performs the variable quantisation based on the level of the decomposition. This process has been streamlined to make optimal use of the dual accumulators and to operate to the limit of the instruction sequencing restrictions imposed by the pipelined nature of the processor. When an accumulator is the destination for an arithmetic operation such as a multiplication, then the next instruction cannot be a write from the same accumulator. If a write operation is attempted then the instruction pipeline will stall for a cycle effectively turning a single cycle instruction into a two cycle instruction. The following list of six operations when performed by interleaving the multiplication operations between the two accumulators, have the result that the pipeline does not stall and each coefficient can be quantised in three machine cycles. The six steps to quantise two coefficients are:

1. Read coefficient *n* into ALU.

[2] Sample quantisers are shown in Table 5.1 on page 203.

2. Multiply coefficient *n-1* into Accumulator A.
3. Write quantised coefficient *n-2* from Accumulator B.
4. Read coefficient *n+1* into ALU.
5. Multiply coefficient *n* into Accumulator B.
6. Write quantised coefficient *n-1* from Accumulator A.

If only a single accumulator was used, then four cycles would be required to execute the three instructions used to quantise a single coefficient:

1. Read coefficient *n* into ALU.
2. Multiply coefficient *n* into Accumulator A.
3. *Pipeline stall.*
4. Write quantised coefficient *n* from Accumulator A.

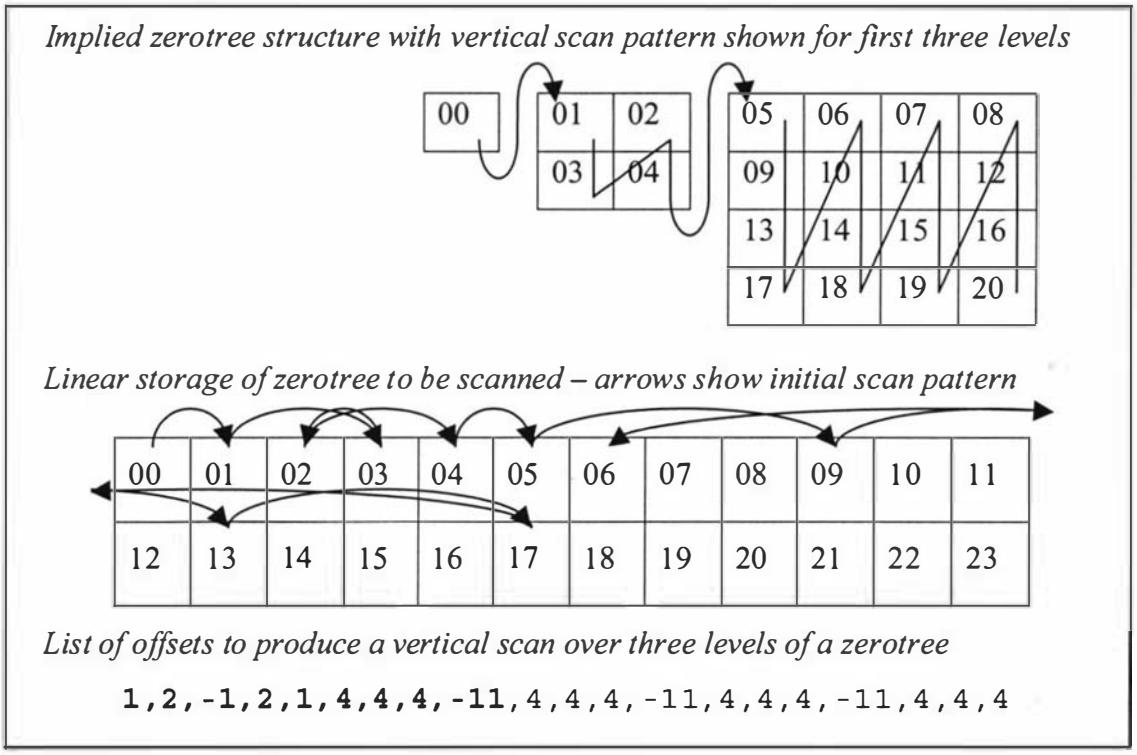


Figure 3.16 – Scan pattern for run-length coding of a vertical edge zerotree

The second pass of the coefficient coding algorithm performs the reordering of the coefficients into a scan pattern favourable to the production of long runs of zero symbols. As the symbols are scanned into the new order, they are run-length coded to

a compact stream of symbols that can be efficiently stored into the output buffer. Depending on whether the zerotree contains horizontal, vertical or diagonal edge information, one of three scan patterns is used. The indirect addressing mode with a variable post increment is available in the processor to produce a complex scanning pattern with minimal overhead. One address register points to a table of offsets to be added to a secondary address register. This secondary address register is then used to index the next coefficient to be read in. The development of the list of offsets to be stored for a vertical scan of a three level zerotree is shown in Figure 3.16.

3.3.2.3 Summary

The performance of the algorithm has been improved by making use of the features of the DSP architecture. This has been specifically accomplished by:

- The use of double buffering techniques: This allows the use of the internal SRAM to buffer the input and output data, to ensure *zero* wait state access to the data by the DSP core. Double buffering ensures that the DSP core and the DMA controller do not access the same data at the same time.
- The DMA controller: Enables the DSP core to execute at full speed while the DMA controller concurrently accesses the slower external memory.
- The use of instruction interleaving: Due to the pipelined nature of the DSP core architecture, certain instruction sequences can cause the pipeline to stall. By reordering these instructions and making use of the additional accumulator, pipeline stalls can be eliminated resulting in a higher data processing throughput.

3.3.3 Huffman coding implementation

Most modern image compression methods use a lossy coder consisting of transformation and quantisation stages to give good compression. The "back-end" of the coder uses some form of lossless entropy coding performed on the resultant symbol stream to minimise the total output file size. The lossless coding stage typically compresses the symbol stream by about 2:1. The "back-end" entropy coding used in many image coding schemes is often overlooked as most research is based around the "front-end" lossy stages of image compression. It is usually assumed that established entropy encoding methods are optimal and that the more sophisticated the scheme, the better the resultant compression will be. This section challenges these assumptions by showing that under certain conditions the simplest of schemes can be superior in both execution *and* compression performance.

Optimal Huffman coding [3.2] is the lossless entropy encoding stage chosen for the image compression algorithm that allows the entropy of the symbol stream to be modelled. At compression ratios of 30:1 for monochrome 512 x 288 pixel images, the resulting entropy encoded symbol stream is about 4.8 kbytes. Optimal Huffman coding requires that a Huffman coding table be sent with each image. Research has been performed to minimise the overhead of the Huffman coding table relative to the modest size of the compressed image bitstream [3.3]. This research shows that optimal Huffman coding with a highly compressed coding table produces smaller files than the more advanced adaptive arithmetic coding scheme. Adaptive arithmetic coding is at least an order of magnitude more computationally complex, but when compressing short symbol streams it is less efficient. Very low bit-rate images typically consist of symbol streams that are only several thousand symbols in length and the overhead due to adaptation is larger than that required to represent a compact Huffman code table.

Computational simplicity along with good use of the DSP architecture are central to the development of the Huffman coding implementation. Section 3.3.3.1 discusses the method of generating the bit patterns for each symbol so that only the bit length

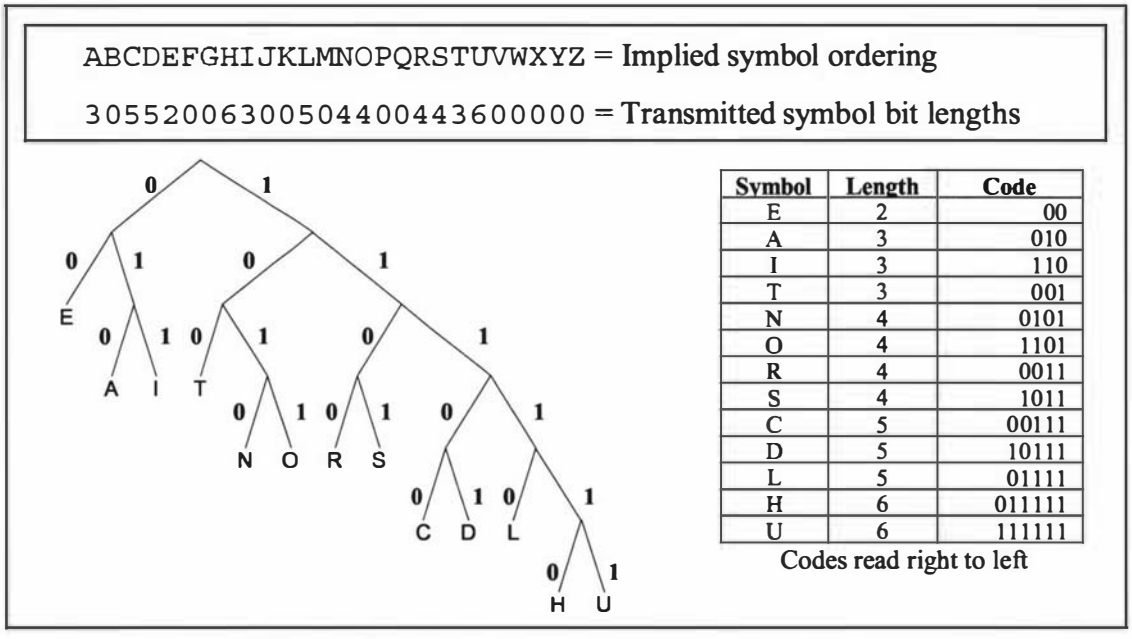
needs to be transmitted for each symbol. Section 3.3.3.2 describes a new method of coding the Huffman tables that reduces the table overhead for most images to less than 300 bits. Finally section 3.3.3.3 looks at the DSP architectural features that allow for the efficient concatenation of variable length codes to produce the Huffman coded bitstream.

3.3.3.1 Generation of bit patterns to construct Huffman codes

The conventional way of assigning bit patterns to each symbol in a Huffman code tree is to label each branch in the tree with a '0' or a '1' depending on the direction of the branch. The bit patterns can then be read by traversing the tree to each leaf node symbol. This method of assigning bit patterns assumes that the complete table structure can be transmitted to the receiver so that an identical tree can be created. The usual way this is done is by transmitting the frequency count for each symbol so that the decoder can build the same tree as the encoder. Sending frequency information requires at least 8 bits to be used for each symbol to produce a good model of the system entropy distribution [3.4]. A typical lossy image coding scheme may have the need for 256 unique symbols to encode an image, this form of representation would require 2,048 bits of table overhead which is a 5% overhead for an 4.8 kbyte file.

A more efficient way of transmitting Huffman code tables is to send only the bit length of each symbol in the tree. These quantities can be constrained to be encoded in 4 bits per symbol, reducing the overhead by a factor of two. The disadvantage of only sending the symbol bit lengths is that many different trees can be constructed from this information. The structure of the tree conveys the assignment of the '0' and '1' bits that make up the bit pattern for each symbol. This assignment needs to be performed in an identical manner in the encoder and decoder. Figure 3.17 shows that by using the implicit "alphabetical" ordering of the symbols, a unique tree can be generated at both the encoder and decoder from just the transmitted symbol bit lengths. The structure of the tree is determined by the number of symbols to be inserted into the tree of a particular length. In Figure 3.17 there is a single 2-bit

symbol, followed by three 3-bit symbols, four 4-bit symbols, three 5-bit symbols and two 6-bit symbols. Within each level of the tree, the symbols are placed in the leaf nodes in alphabetical order ensuring that the bit pattern is assigned correctly at both the encoder and decoder.



Symbol	Length	Code
E	2	00
A	3	010
I	3	110
T	3	001
N	4	0101
O	4	1101
R	4	0011
S	4	1011
C	5	00111
D	5	10111
L	5	01111
H	6	011111
U	6	111111

Figure 3.17 – Huffman codes generated from symbol bit lengths

The first stage of the Huffman coding algorithm will compute the correct lengths for each of the symbols based on their relative probability of occurrence. The second stage of the algorithm is responsible for assigning the bit pattern to each symbol. An address register in the DSP core is used to build up the correct bit pattern for each symbol. Usually an accumulator register would appear to be the most sensible choice for this operation but the powerful indirect addressing mode of the DSP allows the bit pattern to be built up efficiently from right to left in the "R0" address register. This process is repeated for all the used symbols to produce a look-up table of Huffman code patterns. This look-up table can then be used with indirect addressing to efficiently encode the symbol stream.

Figure 3.18 shows how the relatively complex operation of constructing the tree in Figure 3.17 can be executed very rapidly by using the bit-reversed addressing mode that most DSPs support. Usually this addressing mode is only used in fast transform

algorithms such as the Fast Fourier Transform (FFT) algorithm to descramble the output of the transform. Here, the bit-reversed addressing mode is used to rapidly build the code patterns for the Huffman codes. In normal binary addition the carry bit that results from adding '1' to '1' is propagated to the left. With bit-reversed addressing the carry bit is propagated to the right, this is very useful when producing the Huffman code bit patterns.

The procedure illustrated in Figure 3.18 shows that the "N0" offset register which is used with indirect addressing to update the "R0" address register can be configured to use bit-reversed addition. Once the "N0" register is initialised to produce the correct increment for codes of a desired length (10b for length two codes, 100b for length three codes, etc.), repeated additions of "N0" to "R0" produces the desired sequence of bit patterns for symbols of the same length. "N0" is updated as the length of the symbols increase. This simple algorithm rapidly produces the same bit pattern assignment as shown in the tree structure of Figure 3.17.

Initialise R0	00000000 (R0)
Set N0 to increment for 2 bit codes	00000010 (N0)
Write out code for 2-bit symbol 'E'	00 (R0)
Bit-reversed add N0 to R0	00000010 (R0)
Set N0 to increment for 3 bit codes	00000100 (N0)
Write out code for 3-bit symbol 'A'	010 (R0)
Bit-reversed add N0 to R0	00000110 (R0)
Write out code for 3-bit symbol 'I'	110 (R0)
Bit-reversed add N0 to R0	00000001 (R0)
Write out code for 3-bit symbol 'T'	001 (R0)
Bit-reversed add N0 to R0	00000101 (R0)
Set N0 to increment for 4 bit codes	00001000 (N0)
Write out code for 4-bit symbol 'N'	0101 (R0)
Bit-reversed add N0 to R0	00001101 (R0)
Write out code for 4-bit symbol 'O'	1101 (R0)
Etc...	

Symbol	Length	Code
E	2	00
A	3	010
I	3	110
T	3	001
N	4	0101
O	4	1101
Etc...	Etc...	Etc...

Normal binary addition:

00001001
+ 00001000
00010001 ← Carry bit propagation

Bit-reversed binary addition:

00001001
+ 00001000
00000101 → Carry bit propagation

Figure 3.18 – Procedure using bit-reversed addressing to build Huffman codes

3.3.3.2 Delta coding of Huffman code tables

Research into compacting the representation of optimal Huffman code tables resulted in a new method that produces a very low overhead transmission of the table. This work [3.3] has been published at the IEEE International Conference on Image Processing in 1998 (ICIP-98) and the paper is included as an appendix to this thesis. The method combines predictive coding (simple delta coding) with run-length coding and Huffman coding, to compactly represent the optimal Huffman code table for an image. By reducing the typical table overhead down to about 300 bits, the effective table overhead for an 4.8 kbyte image is less than 0.8%.

3.3.3.3 Efficient concatenation of variable length codes

Optimal Huffman coding is implemented in two stages, as the actual entropy encoding cannot begin until the symbol entropy distribution has been calculated. While the coefficients are being run-length coded, a symbol frequency table is updated to reflect the distribution of output symbols. When all the coefficients have been coded, an optimal Huffman table is constructed. The second stage of the coding process applies the optimal Huffman table to the run-length coded coefficient symbols. The DSP has two 56-bit wide accumulators, a barrel shifter and a specialised "`insert`" op-code [3.1]. The "`insert`" op-code allows a variable number of bits from one register to be inserted into a specified bit range in an accumulator. This single cycle instruction is very powerful for this common style of variable length coding. Figure 3.19 shows an example of actions taken by the "`insert A1,Y1,B`" assembly instruction. In this example the first 5 bits of the 24-bit register `Y1` are inserted into bits 13-17 of the 56-bit `B` accumulator. All of the other bits in the `B` accumulator are left untouched.

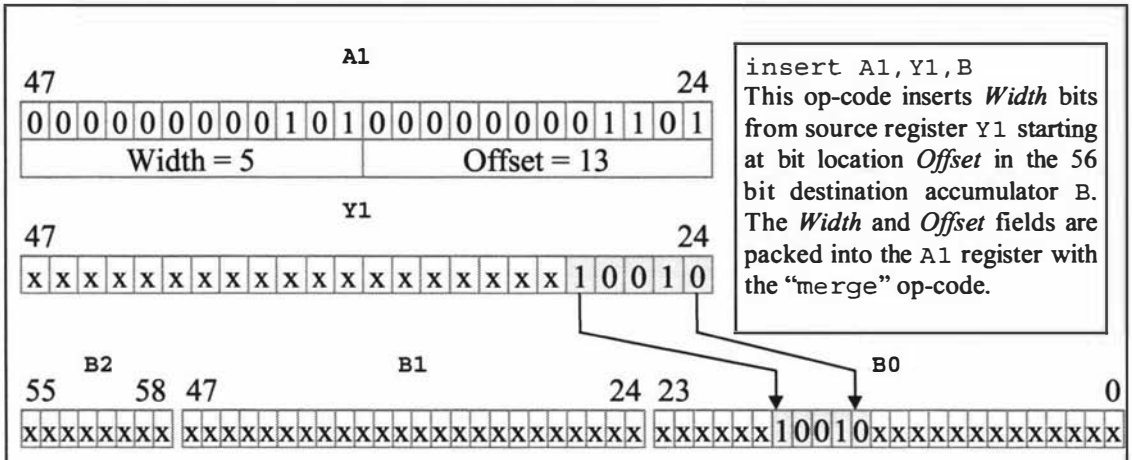
The assembly code given in Figure 3.19 shows a subset of the reasonably optimal inner loop operation of an algorithm that concatenates variable length codes representing a symbol stream into an output bitstream. An average of 18.8 machine cycles are required by the algorithm to encode each input symbol.

3.3.3.4 Summary

Huffman coding is a well established technique for entropy encoding a symbol stream in a lossless fashion. It is computationally inexpensive and performs well when compared to more sophisticated techniques such as arithmetic coding [3.3]. Several architectural features of the DSP have been exploited to ensure the implementation is executed efficiently:

- Bit-reversed addressing: By using only the transmitted symbol bit lengths, a unique allocation of bit patterns can be performed at both the encoder and decoder using bit-reversed addressing.
- Indirect addressing with dedicated "`insert`" and "`merge`" op-codes: Table look-ups are used to rapidly obtain the symbol bit length and bit pattern. The two dedicated op-codes together with the wide 56-bit accumulators are used to concatenate the variable length Huffman codes in a single machine cycle.

A new compact table representation has been developed to minimise the overhead associated with using the optimal Huffman table with each image. This has reduced the typical overhead for a 4.8 kbyte file from 2,048 bits (5%) down to around 300 bits (0.8%). The reduction in overhead has allowed the simple optimal Huffman coding technique to be used competitively in place of the computationally more sophisticated adaptive arithmetic technique. Even though adaptive arithmetic coding is at least an order of magnitude more computationally complex, it has been shown that the technique is less efficient compressing short symbol streams.



insert A1,Y1,B
This op-code inserts *Width* bits from source register Y1 starting at bit location *Offset* in the 56 bit destination accumulator B. The *Width* and *Offset* fields are packed into the A1 register with the "merge" op-code.

```
HUFFCODE move    #HUFFCODL+128,R5   ;Point to ZERO symbol for length codes
         move    #HUFFCODD+128,R6   ;Point to ZERO symbol for data codes
         move    #HUFFOUT,R7        ;Point to Huffman output stream in Y:
         move    #INBUF,R0          ;Point to symbols to be coded in X:
         clr     A
         do      #LEN,HUFFLOOP      ;Process LEN symbols
✂. . . . . .
         move    X:(R5+N5),Y0       ;Length
         move    Y:(R6+N6),Y1       ;Data
         merge   Y0,A               ;Put length (Y0) into bits 47-36
                                    ; of A (top-A1)
         insert  A1,Y1,B            ;Insert top-A1 number of bits from Y1
                                    ; data into B at offset bottom-A1
✂. . . . . .
         blt     NOTOVER            ;Is B0 full of bits now?
         move    B0,Y:(R7)+         ;If B0 is full then write 24 bits of B0
         asr     #24,B,B            ;Shift B1 into B0
         sub     #24,A              ;Modify offset to keep in the
NOTOVER  nop                        ; range of 0..23
HUFFLOOP
```

Summary of inner loop operation:

- Not all instructions within the inner loop take one cycle to execute:
  - "move X:(R5+N5),Y0" and "move X:(R5+N5),Y0" take 2 cycles due to pre-increment
  - "blt NOTOVER" takes 4 cycles if branch is taken or 5 cycles if branch not taken
- Each input symbol takes either 18 or 22 cycles to process. The longer cycle count occurs about 20% of the time when an excess of 24 bits of variable length codes have been concatenated together. This is based on the assumption that the average variable length code is 5 bits long.
- A weighted average based on the above assumptions gives 18.8 cycles per symbol.

Figure 3.19 – Inner loop operation of Huffman code concatenation using the "insert" op-code

3.4 Efficient use of memory

The usage of both the on-chip and off-chip memory resource varies as the algorithm proceeds through its three main stages; wavelet transformation, coefficient coding and Huffman coding. There are four main pools of memory to consider in the system. Internal to the DSP is the program memory (`P:`), X data memory (`X:`) and Y data memory (`Y:`). The fourth pool of memory is the external SRAM. Figure 3.20 shows the sizes of each of the internal pools of memory, whilst the external memory is assumed to be as large as is required. It is important to minimise the amount of external SRAM in a practical system to reduce the overall system cost. A 48 kword frame buffer is present in the external memory pool which contains a 512 x 288 pixel monochrome image. As the memory is organised in terms of 24-bit words, there are three 8-bit pixels stored in each frame buffer word.

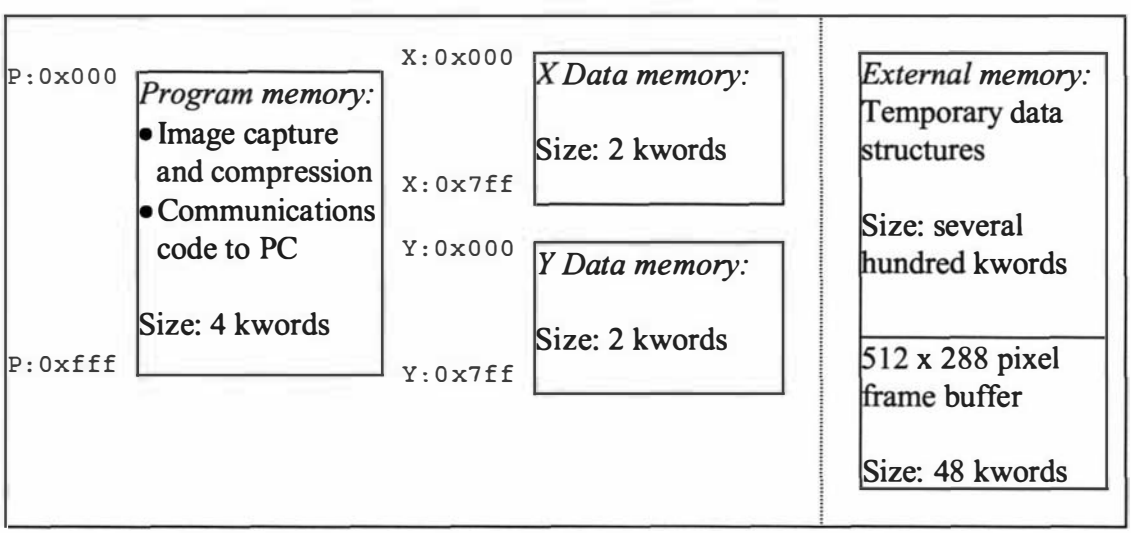


Figure 3.20 – Memory pools available for image compression algorithm

The program memory pool is used to contain the image capture and compression algorithms as well as a communications kernel to support the reception of setup parameters to the camera as well as the transmission of compressed images to a PC. The last quarter of the Y: data memory is used to contain constants and system variables. The following sections will detail the allocation of X and Y data memory pools as well as external memory for each algorithmic stage.

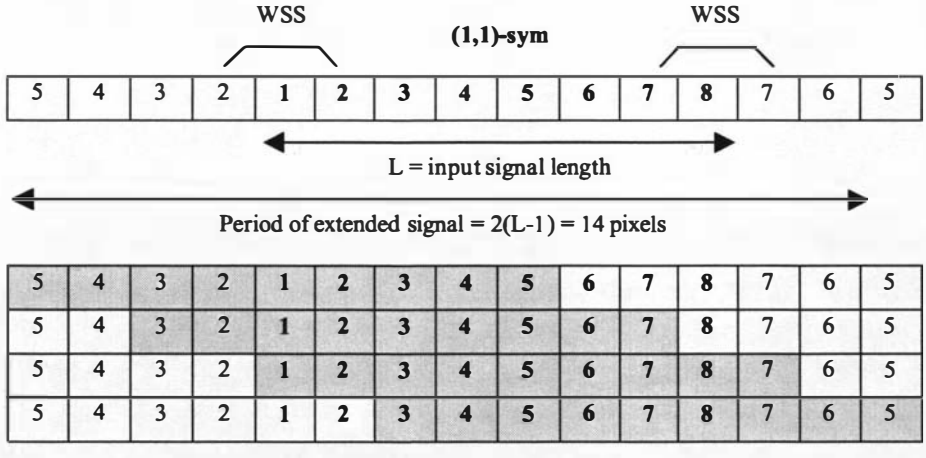
3.4.1 Stage 1: Wavelet transformation

Two input buffers each of 768 words are allocated in the X: memory space to contain alternate rows and columns of data to be transformed. The maximum amount of data that these buffers ever contain is a single row of the raw image with a symmetric edge extension for a nine element filter bank. Seven additional locations are required for the symmetric extension so the largest amount of data to occupy the input buffers is 519 words which easily fits into the 768 word buffers. The wavelet filter coefficients occupy 18 words in the top quarter of the Y: memory space which is reserved for constant data.

The two output buffers are each 768 words in length and are allocated in the Y: memory space. Each of these buffers is split into two halves, one to contain the low-pass filtered output data and the other for the high-pass data. The maximum amount of output data to occupy the 384 word half buffers is one half of the raw image scanline which is 256 words. Figure 3.21 shows the usage of the internal memory of the DSP for double buffering both the input and output streams of data to be processed by the DSP core.

Implementation detail of symmetric edge extension in wavelet decomposition

Input signal of even length (L) = 8 pixels
Quadrature Mirror Filter bank of odd length = 9 coefficients
Therefore use Whole Sample Symmetry (WSS), no repeat of the end value, at *both* end of the signal for decomposition, denoted as **(1,1)-sym** [3.5]



Overlay of the 9 element filter banks for the four convolution outputs from each filter. This produces a non-expansive decomposition with 4 elements in both the approximation and detail output vectors.

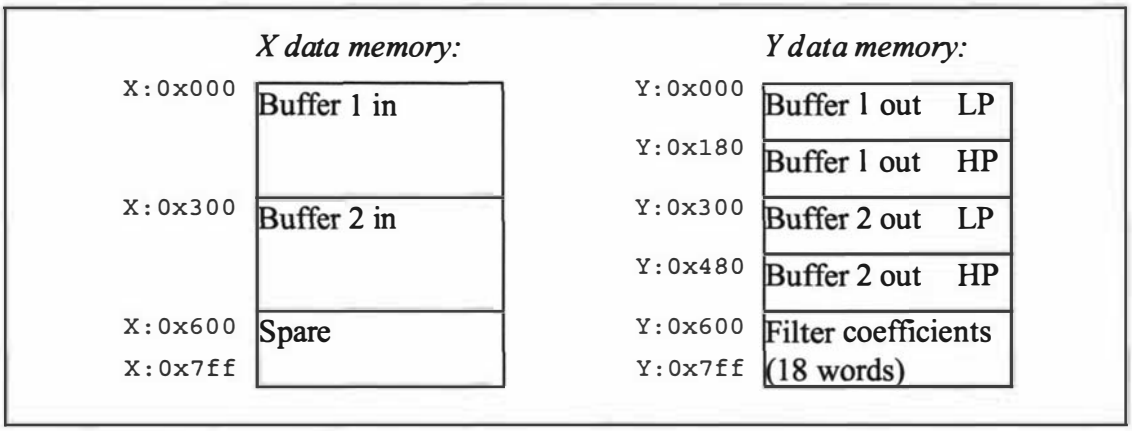


Figure 3.21 – Internal memory usage for wavelet transform

The use of the external memory pool is more complicated than the static allocation of memory used for the internal memory pools. This is because large intermediate data structures must coexist with the more permanent transformed data structures as the four level wavelet transformation is applied to the raw image. Figure 3.22 shows the external memory usage for the first level of the wavelet transform. The raw image located in the frame buffer is transformed along its rows to produce two 72 kword intermediate data structures in the first column of the figure. The high-pass data structure is then filtered to produce two 36 kword level 1 detail structures (Horizontal and Diagonal). Finally the low-pass data structure is filtered to produce the last two 36 kword level 1 structures (Vertical detail and Approximation). To conserve external memory, the Vertical detail overwrites the no longer needed, high-pass intermediate data.

It should be noted that 216 kwords of external memory is required to perform a full wavelet transform of an image that can be stored in 48 kwords of memory. The majority of the memory is required to store the two intermediate sets of coefficients during the first level of the wavelet transformation. The final set of 24-bit coefficients that make up the full wavelet transform is stored in a total of only 144 kwords.

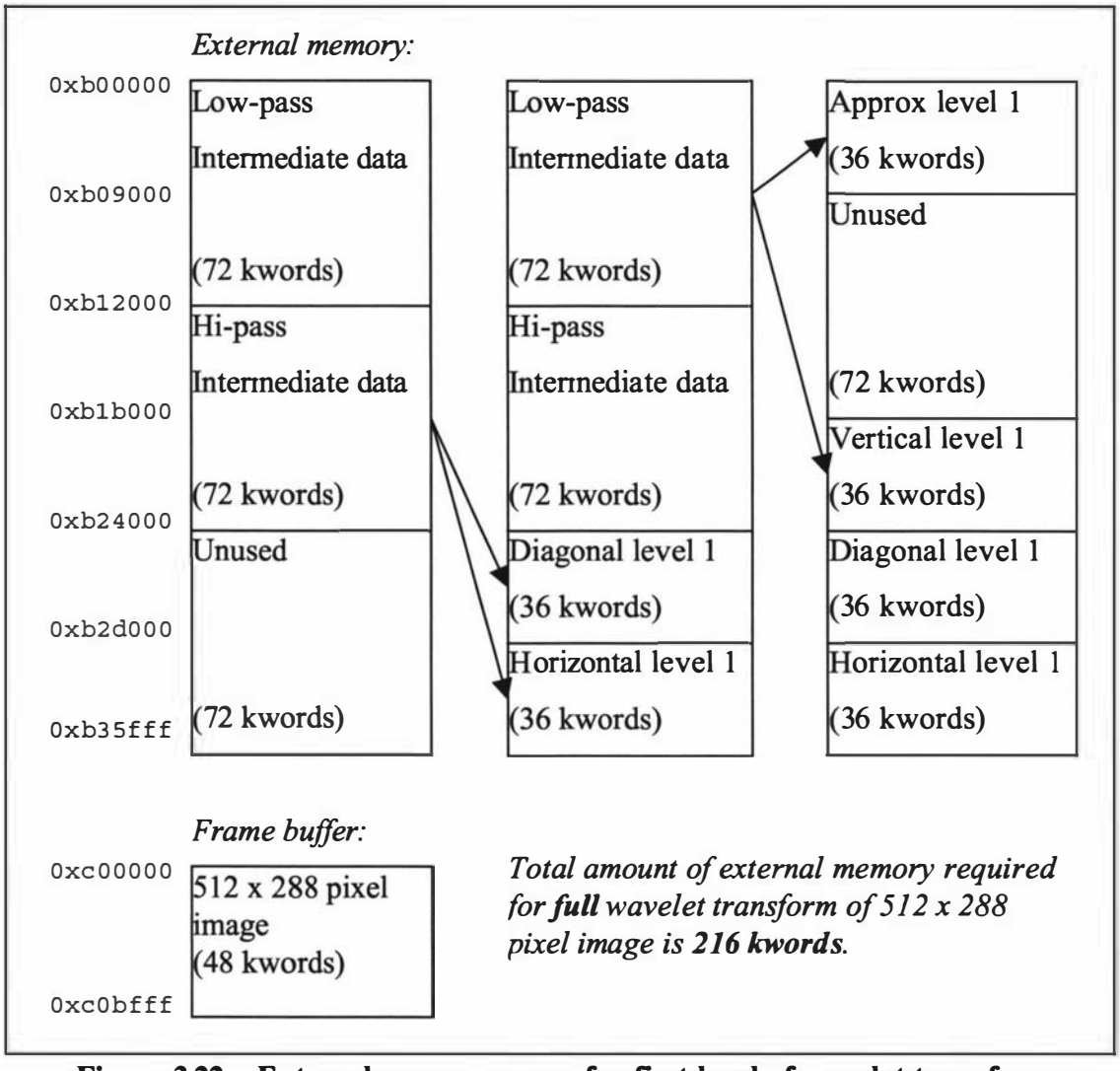
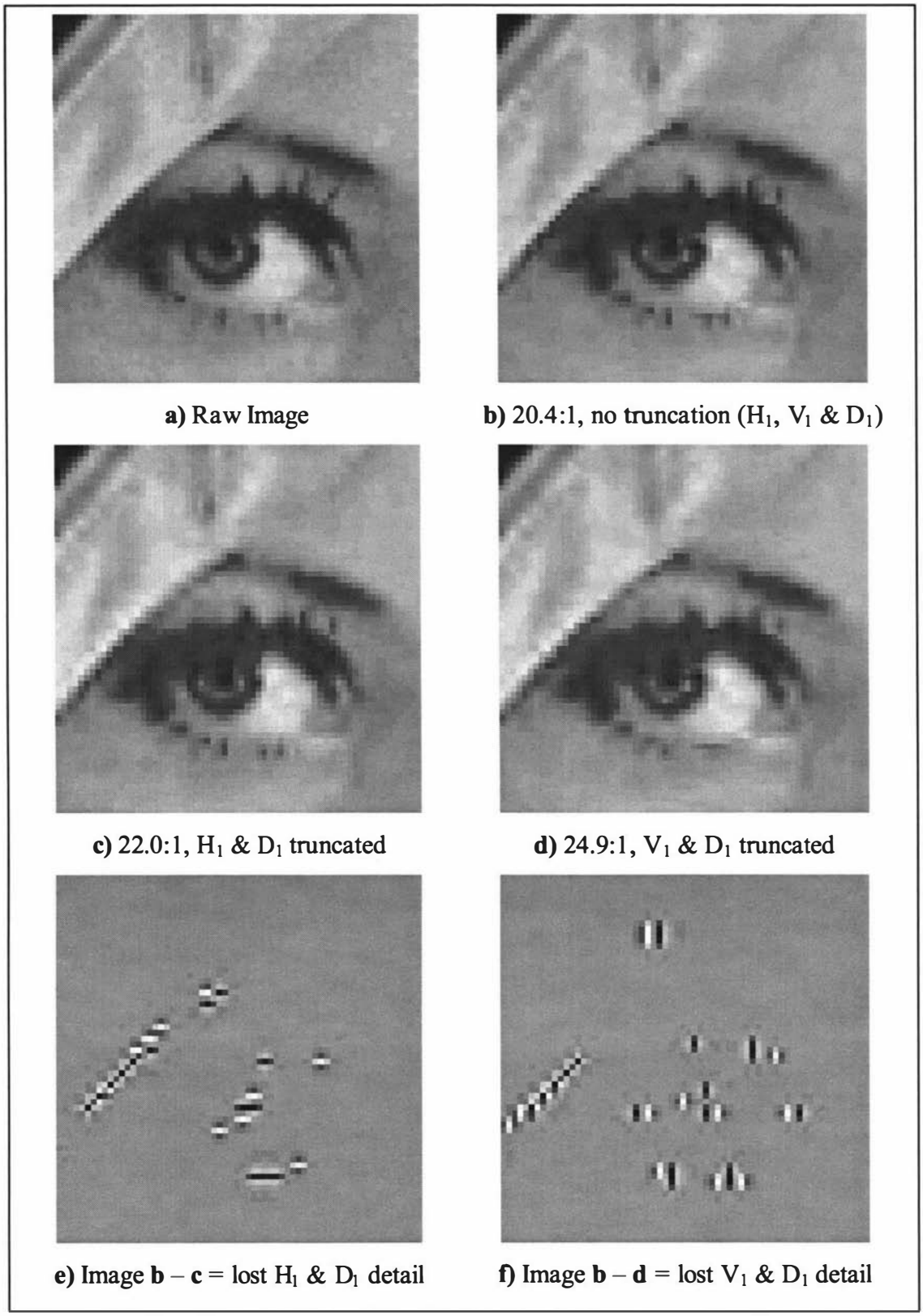


Figure 3.22 – External memory usage for first level of wavelet transform

The amount of external memory required to complete the wavelet transform can be halved by not calculating or storing the H_1 and D_1 coefficient sets. This is similar to the approach taken by Analog Devices who created a Wavelet/Scalar Quantisation (WSQ) integrated circuit [3.6]. Their decomposition omits the calculation of the V_1 and D_1 coefficients for similar memory saving reasons. It is reasonable to truncate the decomposition in these ways if high compression with the consequent loss of some detail is acceptable. For natural images there is little information in the highest frequency coefficient spaces and when coarse quantisation is applied to these spaces, most of the coefficients are rounded to zero.



a) Raw Image

b) 20.4:1, no truncation (H_1, V_1 & D_1)

c) 22.0:1, H_1 & D_1 truncated

d) 24.9:1, V_1 & D_1 truncated

e) Image **b** – **c** = lost H_1 & D_1 detail

f) Image **b** – **d** = lost V_1 & D_1 detail

Figure 3.23 – Non-critical nature of high frequency information in a natural image

To illustrate the non-critical nature of the truncated high frequency coefficient spaces, Figure 3.23 shows a 64 x 64 pixel portion of the Lena image in **a)**, its original raw form; **b)**, compressed at 20.4:1 without any truncation; **c), e)**, where truncation of H_1 & D_1 detail has occurred and **d), f)**, where truncation of V_1 & D_1 detail has occurred. The two truncation methods improve the overall compression ratio achieved for the image and do not severely reduce the visual quality of the compressed image.

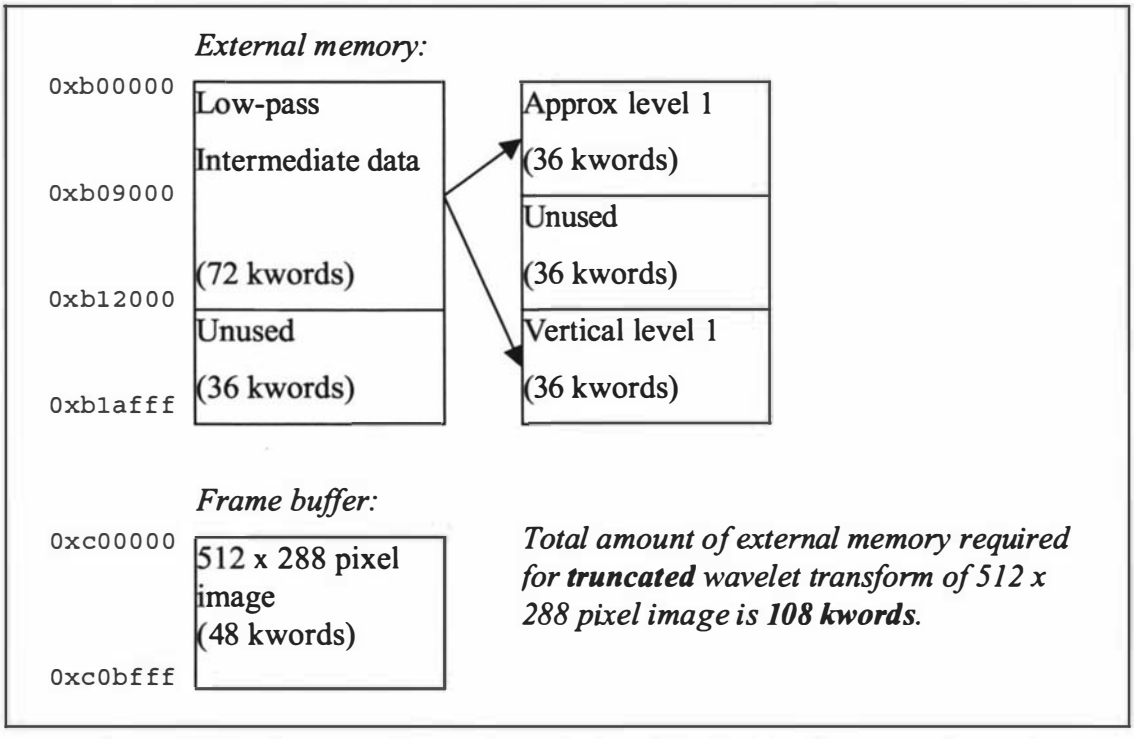


Figure 3.24 – External memory usage for first level of truncated wavelet transform

The truncated wavelet transform uses half the amount of computation as well as half the amount of memory compared to the full wavelet transform, see Figure 3.24. The remaining levels of the transform proceed without any further truncation. The final layout in memory of the truncated wavelet transform, after four levels have been calculated, is shown in Figure 3.25. There is a 36 kword memory overhead associated with calculating 72 kwords of wavelet coefficients in the *truncated* wavelet decomposition. This is a considerable improvement over the 72 kword overhead associated with calculating 144 kwords of wavelet coefficients in the *full* wavelet decomposition where a large proportion of the additional coefficients will be quantised to zero.

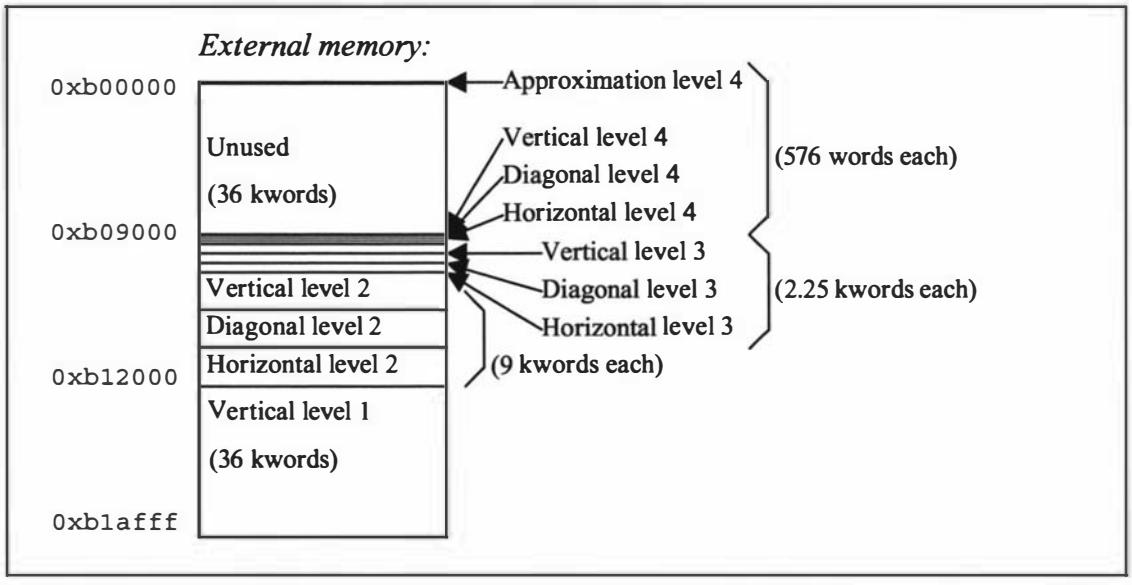


Figure 3.25 – External memory usage after four levels of the initially truncated wavelet transform

3.4.2 Stage 2: Coefficient coding

The coefficient coding stage makes use of a number of memory structures located within the DSP core. Figure 3.26 shows the locations of these structures which include:

- Double buffered input buffers to contain the raw coefficients prior to quantisation and tree building (768 words each).
- Double buffered output buffers to store temporary results and the run-length coded zerotrees (1024 words each).
- A table of quantisation scalars, one for each level of the wavelet decomposition, to allow for a variation in the compression ratio by altering the coarseness of the quantisation process (4 words).
- Three scanning pattern tables to perform horizontal (2 words), vertical (84 words) and diagonal (84 words) scans through the zerotree structure prior to run-length coding.
- A symbol probability table to maintain the usage counts for each symbol as the zerotrees are run-length coded (256 words).

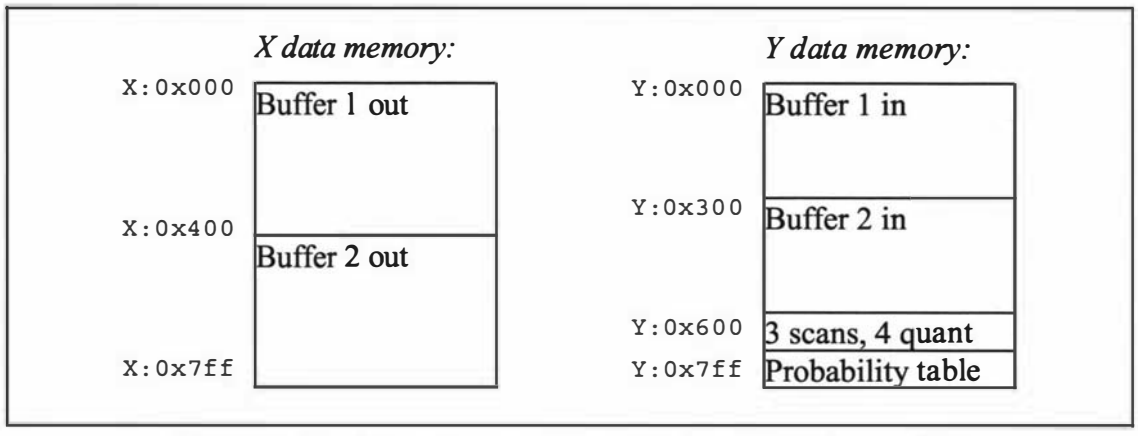


Figure 3.26 – Internal memory usage for coefficient coding

The two input buffers are 768 words each and a single zerotree which is the basic working unit of the coefficient coding algorithm, is only 85 words long. This allows for eight zerotrees to be constructed and coded at a time, reducing the number of DMA setup operations by a factor of eight. The previously unused 36 kwords in external memory located between the approximation coefficients and the beginning of the detail coefficients is used to store the run-length coded zerotrees as they are produced in groups of eight. The external memory usage is shown in Figure 3.27. The actual amount of memory consumed out of the 36 kword block is dependant on the image content and the quantiser settings.

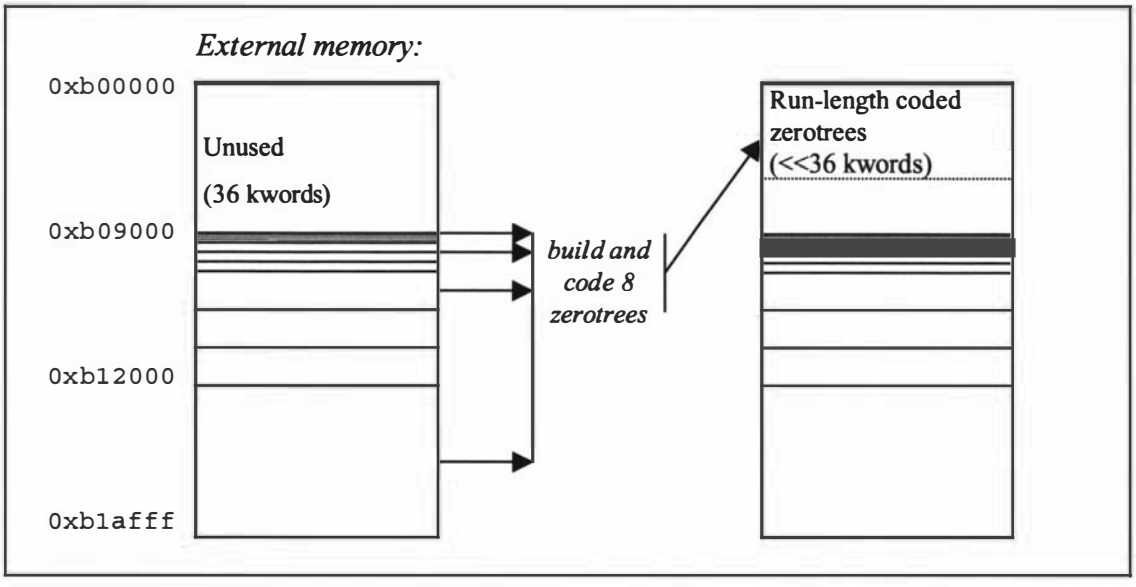


Figure 3.27 – External memory usage for coefficient coding

3.4.3 Stage 3: Huffman coding

The Huffman coding stage makes use of a number of memory structures located within the DSP core. Figure 3.28 shows the locations of these structures which include:

- A table of used symbols and their counts in the image generated from the probability table (up to 513 words).
- A table of symbol lengths for every node in the Huffman tree (up to 511 words).
- Temporary data structures used to assign unique bit patterns to the leaf nodes in the Huffman tree (320 words).
- A table of symbol bit patterns for each leaf node in the Huffman tree (up to 256 words).
- A table of up to 255 new symbol structures each 3 words in size. The combined structures describe the Huffman tree in terms of the symbol values of the two child nodes and the combined count of the two child nodes (up to 765 words).
- A symbol probability table to maintain the usage counts for each symbol as the zerotrees are run-length coded (256 words).

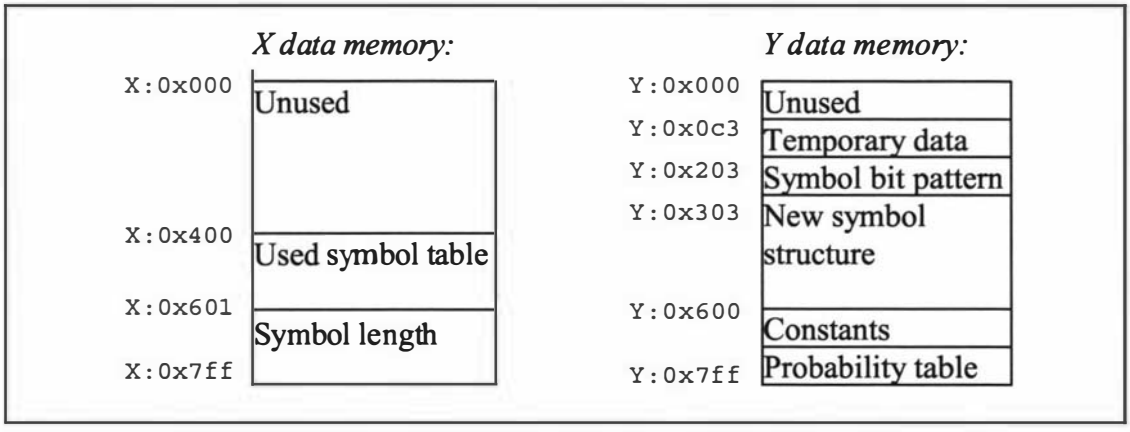


Figure 3.28 – Internal memory usage for Huffman coding

The algorithm initially uses the above internal memory structures to construct an optimal Huffman tree from the symbol probability table for the current image. Next, the bit pattern and length for each of the used symbols are calculated and stored in two

lookup tables. Finally, the 1,728 run-length coded zerotrees and DPCM coded (see section 3.1) approximation image (576 symbols) are Huffman coded. No double buffering or DMA is used to pipeline the external data through the DSP core for this stage of the algorithm. This is due to the fact that the majority of the data compression has already taken place in the "front-end" lossy stages of coding. The number of symbols to be encoded is much fewer for the entropy encoding stage. The wait state penalty for directly accessing data in the external memory is minimal in this final "back-end" entropy coding stage and the lack of DMA usage simplifies the algorithm. The external memory usage is shown in Figure 3.29. The actual amount of memory used out of the remainder of the 36 kword block by the Huffman bitstream is dependant on the image content and the quantiser settings. It is the Huffman bitstream along with the Huffman table that is transmitted to the PC for decompression and display.

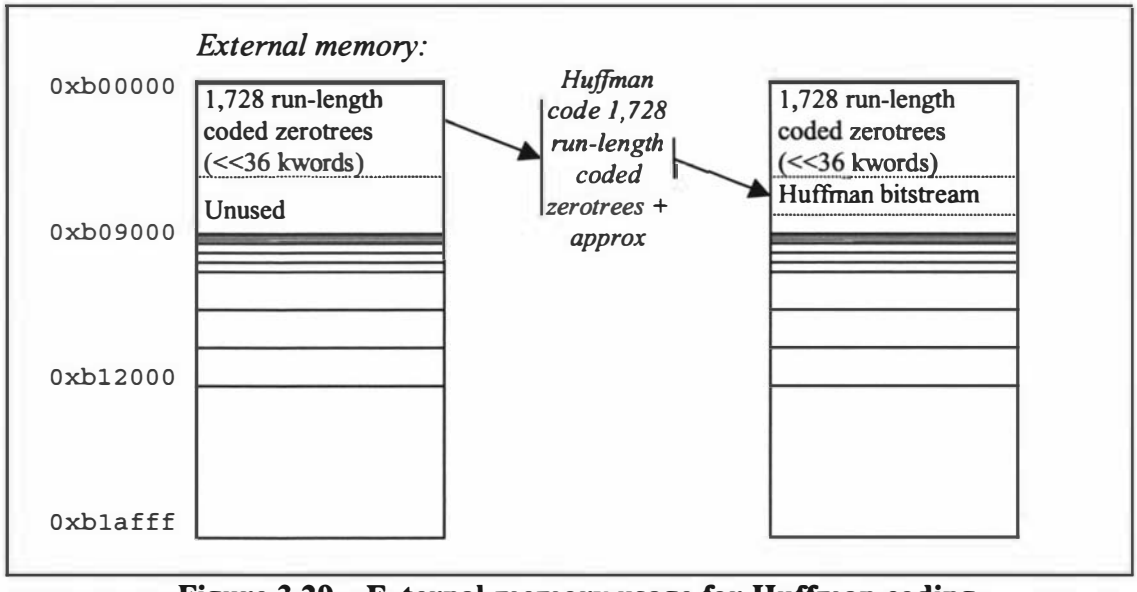


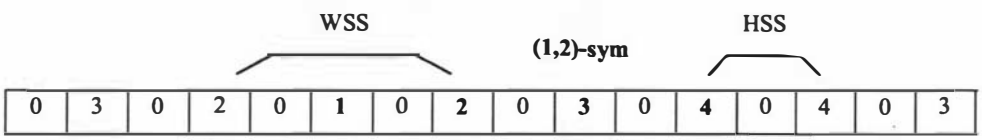
Figure 3.29 – External memory usage for Huffman coding

3.5 Compression algorithm performance

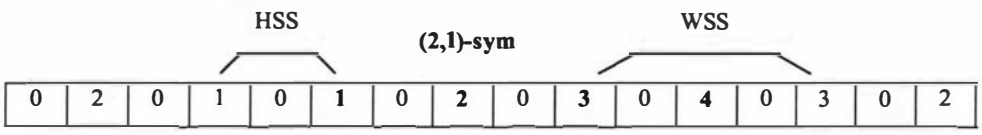
This section will examine the performance of the wavelet based compression algorithm in terms of its speed. The speed of the algorithm can be normalised to clock cycles per pixel for the wavelet transformation and zerotree coding stages. For the Huffman coding stage of the algorithm where the original image is now represented as a set of symbols, the speed of the algorithm can be normalised to clock cycles per symbol. The main reason for normalising the performance of each stage of the algorithm is to allow a comparison to be made between the various stages in the compression and decompression algorithms. The compression algorithm is written in assembly language on the Motorola 56303 DSP and the decompression algorithm is written in C++ on a Pentium based PC.

Implementation detail of symmetric edge extension in wavelet reconstruction

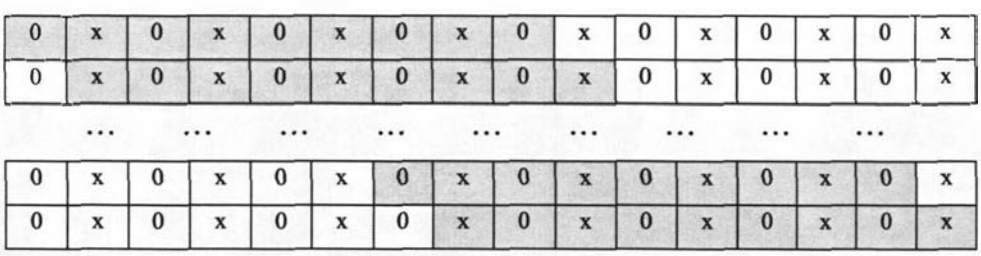
Two input signals (A, D) of even length (L/2) = 4 coefficients each
Quadrature Mirror Filter bank of odd length = 9 coefficients
Therefore use a combination of Whole Sample Symmetry (WSS) and Half Sample Symmetry (HSS) at the ends of the two signals for perfect reconstruction, denoted as **(1,2)-sym** for approximation (A) and **(2,1)-sym** for detail (D) [3.5]



Zero-padded (up-sampled) and extended approximation signal (A)



Zero-padded (up-sampled) and extended detail signal (D)



Overlay of the 9 element filter banks for the eight convolution operations required from each filter.
The sums of each pair of convolution outputs produce 8 output pixels.

The PC implementation makes use of the Intel Signal Processing Library (SPL) version 4.0 [3.7] which provides hand-optimised assembly language implementations of the FIR filters required for efficient wavelet reconstruction. In January 1997 Intel added signal processing extensions to its Pentium processors. These extensions are designed to achieve some of the performance that DSPs have with respect to FIR filter implementations. These signal processing or multimedia extensions are known as "MMX" by Intel and both MMX and non-MMX usage is compared in section 3.5.2.

With the advent of signal processing extensions to general-purpose processors (GPPs) such as the Intel Pentium series of processors, the need for an additional DSP in a PC may not appear to be warranted to perform efficient signal processing operations. There are however a number of fundamental architectural differences between a typical GPP and a DSP that significantly affect the implementation efficiency of signal processing algorithms [3.8]:

- Signal processing requires *efficient multiplication performance* and a significant proportion of the DSPs silicon area is dedicated to single cycle multiply and accumulate (`mac`) operations. Early GPPs and some modern GPPs still take several cycles to execute a multiply operation. Additional accumulation and data move operations take extra instructions for most GPPs but a DSP can perform all these operations concurrently in a pipeline.
- DSPs feature *specialised addressing modes* that are not found in GPPs. Modulo addressing allows for the creation of circular buffers that can contain delay lines or sets of filter coefficients. GPPs need to perform additional checks when addressing circular buffers to ensure that the end of the buffer has not yet been reached. These additional instructions can add a significant overhead to short loops that need to be iterated through thousands of times.
- *Zero-overhead looping* improves the execution efficiency of tight loops often found in signal processing applications. DSPs offer hardware support to execute a single instruction or a block of instructions a certain number of times. Complex branch prediction is used by high performance GPPs to reduce some of the software overhead in looping but this does not remove the overhead entirely.

- The *memory architecture* of a DSP is different from a GPP. DSPs use a *Harvard* memory architecture where there are at least two separate memory spaces to allow concurrent access to both program code and data each clock cycle. This effectively doubles the DSP memory bandwidth and is crucial to keeping the processor core fed with new data and instructions. Traditionally GPPs use a *von Neumann* architecture where both program code and data reside in the same memory space. This architecture works perfectly well for many computing applications where the memory bandwidth is sufficient to supply the processor with instructions and data in an interleaved fashion. Signal processing applications generally require a higher memory bandwidth to support the single cycle `mac` instruction. This instruction requires at least three memory accesses (one program and two data) per cycle. Modern high performance GPPs require multiple accesses to program code and data memory each cycle to support their superscalar instruction execution, (described in the next bullet point). Program memory and data memory caches with separate buses are used to supply the necessary memory bandwidth to support high clock speeds in excess of 200 MHz as the external memory cannot be accessed at this high speed. Even though the physical memory architecture of modern GPPs appear Harvard in nature, the logical view perceived by the application programmer is still von Neumann.
- Modern GPPs often use *superscalar execution* to increase their performance in terms of the number of instructions executed per cycle [3.9]. Depending on the sequence of instructions to be executed, up to four instructions may be executed in parallel on separate processing units within the GPP core. The DSP achieves a similar performance by utilising *specialised instruction op-codes* such as the `mac` instruction. These specialised op-codes also specify data read and write operations as well as data scaling (via a barrel shifter). Each of these operations are executed in parallel by dedicated execution units in the DSP core. The application programmer explicitly controls these operations to occur in parallel on a DSP whereas complex and dynamic instruction scheduling occurs internally within the GPP core to achieve similar results.
- The *predictability* of an algorithm's run-time performance differs between the DSP and GPP architectures. DSPs are best used when there are hard real-time constraints in the application. DSP manufacturers often provide simulation tools

that accurately model the timing of the DSP instructions so that an exact figure can be placed on the number of clock cycles to execute a particular sequence of instructions. High performance GPPs incorporate complicated algorithms that use branching history to predict whether a branch is likely to be taken. This leads to speculative execution of instructions based on a prediction. A side effect is that the same section of code may consume a different number of cycles depending on events that have taken place beforehand.

3.5.1 Profile of DSP compression algorithm

The following timings relate to the compression of 512 x 288 pixel images using an 80 MHz Motorola 56303 DSP. The algorithm requires at least 156 kwords of external SRAM (which includes the 48 kword frame buffer) and the memory used was rated with a 15ns access time. The external SRAM requires at least two wait states to be reliably accessed by the DSP. During implementation it was found that the wavelet transformation and encoding stages of the algorithm would only run reliably with an eight wait state access to the external SRAM. This large wait state penalty has been attributed to the long address and data buses on the initial prototype DSP board. A production quality board (considered in chapter 5 of this thesis) has an improved layout to compensate for this wait state deficiency, reducing the number of wait states to two. The timeline in Figure 3.30 shows the amount of time spent in each section of the algorithm.

Extensions to the Intel Pentium family of processors

Intel introduced MMX extensions to its Pentium family of processors in January 1997. The maximum clock speed of the processor was 200 MHz at this time and the experimental work performed for this thesis was based on this processor. Since then, the Pentium II and III have been released which both support the additional 57 MMX instructions and have clock speeds up to 600 MHz at the end of 1999. The Pentium III features 70 additional instructions known as "Streaming SIMD Extensions" (Single Instruction, Multiple Data) [3.10]. These extensions provide a performance improvement for filter operations that operate on packed single-precision floating point data. There are additional enhancements over MMX for SIMD integer arithmetic as well as new "cacheability control" instructions. The latter enable the application programmer to have better control of what data should stay in the processor's data cache. Intel report a 29% increase in performance for multimedia applications using the new streaming SIMD extensions.

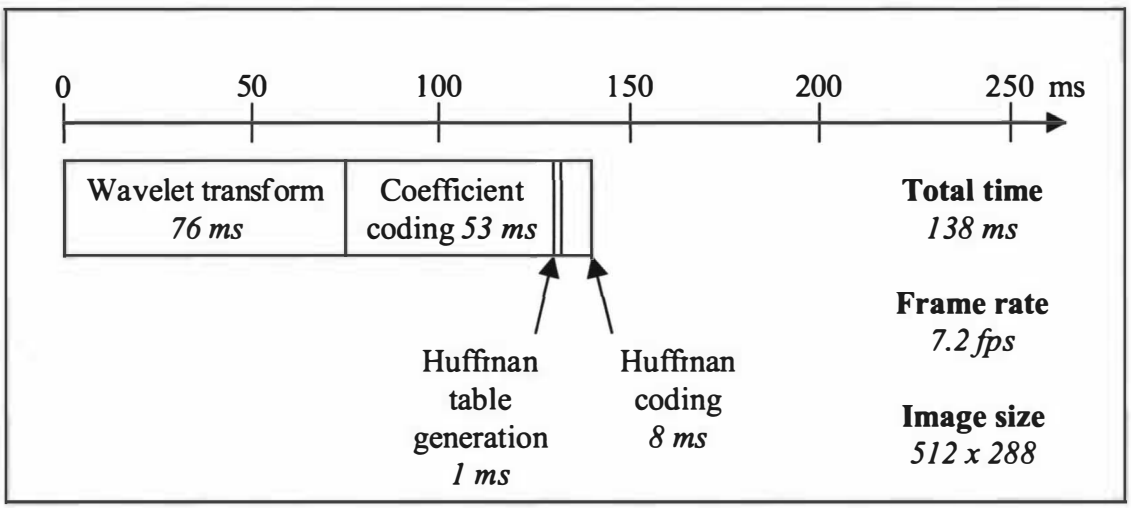


Figure 3.30 – Time spent in each stage of the wavelet based algorithm

A breakdown of the timings for the various stages of the compression algorithm is given in Table 3.1. Just over half of the total execution time is spent in the wavelet decomposition stage and a frame rate of 7.2 frames per second is achievable with the DSP clocked at 80 MHz. The 76 ms time for the wavelet transform is constant for all images decomposed to four levels with the truncated wavelet transform but the other two stages are slightly dependent on the quantiser settings as well as image content. A total of 11 million DSP clock cycles are required to compress a single image.

Stage	Time	Percentage of time	Normalised cycles
Wavelet transform	76 ms	55%	41 cyc/pix
Coefficient coding	53 ms	38%	28 cyc/pix
Huffman table and coding	9 ms	7%	49 cyc/sym
Total	138 ms	= 7.2 frames/second	11 million cycles

Table 3.1 – Performance of 80 MHz Motorola 56303 DSP compression algorithm

3.5.2 Profile of Pentium decompression algorithm

A complete system for remote activity monitoring requires an image decompression algorithm to decode the images sent from the remote cameras. A PC was chosen as a suitable platform for decompressing images for display to an end user. This section provides approximate timings for the various stages of the algorithm used to decompress the images received from the remote camera. The wavelet based image compression algorithm is symmetric in execution. This means that the type and amount of calculation required to decompress the image is very similar to that required to compress the image. The algorithm symmetry allows a comparison to be made in section 3.5.3 between the signal processing ability of a 200 MHz Pentium based PC and an 80 MHz DSP.

When profiling the Pentium implementation of the wavelet decompression algorithm, a great deal of variability was observed in the timing of each stage of the algorithm. The timings stated in the previous tables for the Pentium implementation represent the best timing observed for each stage over a number of decompressed frames. Most of this variability can be explained by the fact that the decompression algorithm is running on a pre-emptive multitasking operating system (Microsoft Windows 95). The operating system will often switch between tasks running on the PC which further reduces the predictability of the algorithm's performance.

Stage	Time	Percentage of time	Normalised cycles
Wavelet transform	460 ms	88%	620 cyc/pix
Coefficient coding	< 50 ms	10%	68 cyc/pix
Huffman table and coding	< 10 ms	2%	150 cyc/sym
Total	520 ms	= 1.9 frames/second	104 million cycles

Table 3.2 – Performance of 200 MHz Pentium non-MMX decompression algorithm

A breakdown of the decompression algorithm timings is given in Table 3.2 for an implementation that does *not* use the MMX extensions. Almost 90% of the total execution time is spent in the wavelet reconstruction stage and a frame rate of only 1.9 frames per second is achievable compared to the 7.2 frames per second achieved by the DSP. A total of 104 million Pentium clock cycles are required to decompress a single image. Only the wavelet reconstruction stage of the algorithm uses the Intel Signal Processing Library but without the MMX extensions enabled on the processor, the Pentium cannot efficiently implement the FIR filter calculations. Table 3.3 shows the timing of the decompression algorithm for an MMX enabled 200 MHz Pentium processor. Only the wavelet reconstruction stage of the algorithm is improved by using the Intel Signal Processing Library but this stage is executed 2.6 times faster with the MMX extensions. The overall frame rate for the decompression algorithm also improves from 1.9 frames per second to 4.2 frames per second. Also the number of clock cycles required to decompress an image drops from 104 million to 48 million.

Stage	Time	Percentage of time	Normalised cycles
Wavelet transform	180 ms	75%	240 cyc/pix
Coefficient coding	< 50 ms	21%	68 cyc/pix
Huffman table and coding	< 10 ms	4%	150 cyc/sym
Total	240 ms	= 4.2 frames/second	48 million cycles

Table 3.3 – Performance of 200 MHz Pentium MMX decompression algorithm

3.5.3 Comparison of algorithm performance

The normalised timing for the various algorithm stages are best used for comparing the efficiency of the three implementation platforms (DSP, MMX Pentium and non-MMX Pentium) considered in the last two sections. These timings are collated in Table 3.4 and clearly show that for the coefficient and Huffman coding stages of the algorithm, the DSP implementation is about three times more efficient than either of

the two Pentium implementations. The wavelet transform stage however exhibits a much wider range of efficiencies. The DSP implementation of the wavelet transform is six times more efficient than the MMX enhanced Pentium implementation and 15 times more efficient than the non-MMX Pentium implementation. Figure 3.31 shows the proportion of time each processor spends in each stage of the algorithm.

Stage	Pentium non-MMX 200 MHz		Pentium MMX 200 MHz		Motorola 56303 DSP 80 MHz	
Wavelet transform	460 ms 88%	620 cyc/pix	180 ms 75%	240 cyc/pix	76 ms 55%	41 cyc/pix
Coefficient coding	< 50 ms 10%	68 cyc/pix	< 50 ms 21%	68 cyc/pix	53 ms 38%	28 cyc/pix
Huffman table and coding	< 10 ms 2%	150 cyc/sym	< 10 ms 4%	150 cyc/sym	9 ms 7%	49 cyc/sym
Total time	520 ms = 1.9 fps		240 ms = 4.2 fps		138 ms = 7.2 fps	
Total clock cycles	104 million		48 million		11 million	

Table 3.4 – Performance comparison between Pentium and DSP architectures

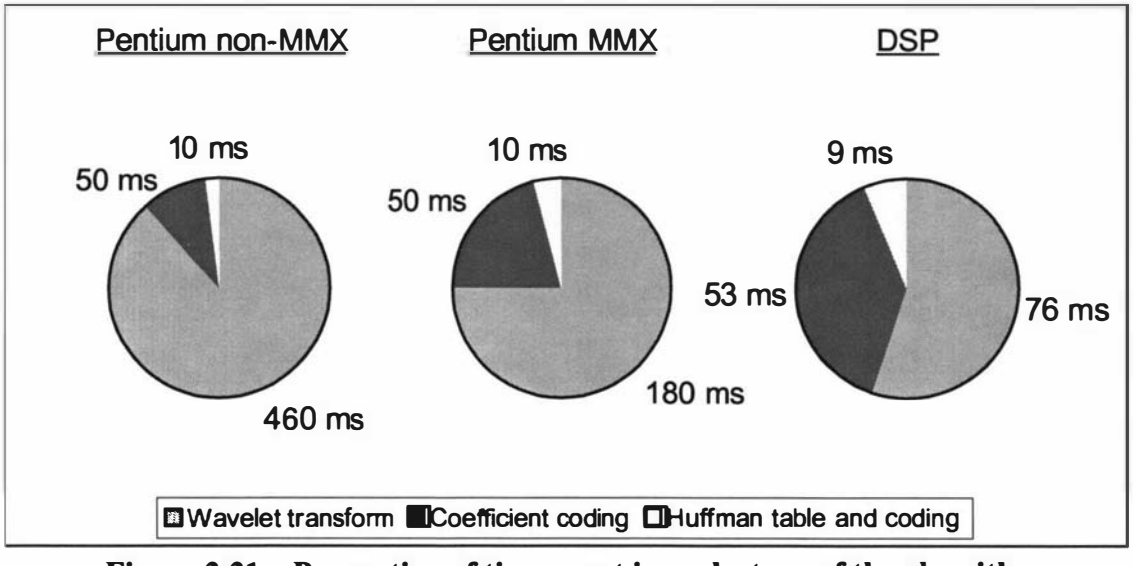


Figure 3.31 – Proportion of time spent in each stage of the algorithm

Even though there are significant factors weighing against the Pentium processor's ability to perform signal processing operations as efficiently as a DSP, the MMX

extensions to the processor do speed up the execution of the wavelet filter banks. The speed increase is by a factor of 2.6 and this can be attributed to the 57 MMX instructions added to the base Pentium architecture:

- The MMX instructions use a SIMD (single instruction, multiple data) model, operating on several values at a time [3.11]. As the MMX registers are 64 bits wide, they can store two 32-bit words, or four 16-bit words or 8 bytes at a time. The MMX arithmetic instructions can operate on these packed data structures to perform parallel addition and multiplication within these registers greatly increasing the throughput of the processor for integer arithmetic.

Theoretically such architectural improvements to the Pentium should allow multiplication of 16-bit quantities to occur at a greater rate than one per clock cycle. This ideally would occur when the data to be multiplied is located in the data cache of the processor core. Most of the impressive timings given in the Intel Signal Processing Library literature are qualified with the following statement: "...results averaged over many repetitions of each function call. Data is in cache." [3.7]. Unfortunately, most DSPs do not have or *require* a data cache because DSP data is typically *streaming*. This means that the DSP processor performs computations with each data sample and then discards the sample with little reuse [3.8].

While the FIR filtering operations in the wavelet reconstruction algorithm utilise the MMX instructions, it is likely that many clock cycles are wasted as the Pentium needs to stream new data from the slow external memory into the data cache to perform the filtering operation. This is essentially the same problem the DSP implementation of the FIR filter bank addresses by using DMA when reading data from the slow external SRAM into the fast internal memory of the DSP core. With the DSP implementation, DMA is used explicitly under the programmer's control to marshal the required data into the core. The Pentium processor relies on smart algorithms built into the data cache controller to "hopefully" minimise cache misses. The Pentium application programmer has little to no control over what data resides in the Pentium data cache.

3.6 Comparison with a JPEG like image compression algorithm

The purpose of this section is to compare the wavelet based compression system with the well-known JPEG algorithm. There are three specific attributes that are briefly compared and these are the algorithm structure and memory usage; the algorithm execution speed on the Motorola 56303 DSP; and image quality at a given compression ratio.

3.6.1 Algorithm structure and memory usage

The upper portion of Figure 3.32 shows the basic algorithm structure for a JPEG like algorithm based upon the use of the Discrete Cosine Transform (DCT) working on 8 x 8 pixel image blocks. Each transformed block is then processed with a Sequential Baseline Coder which zig-zag scans, run-length codes and then Huffman codes the resultant symbol stream. The total amount of temporary storage required for the DCT based algorithm is 64 24-bit words to contain the intermediary transformed image block.

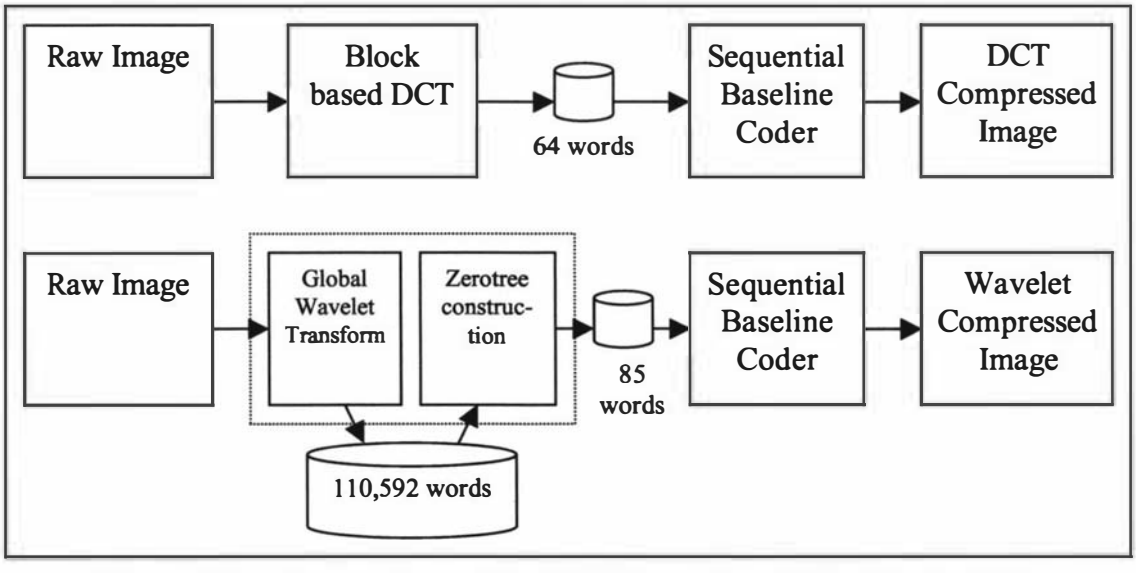


Figure 3.32 – Comparison between DCT and wavelet based algorithms

The amount of memory required for temporary data structures is far greater for the global wavelet transform and this is shown in the lower portion of Figure 3.32. For a 512 x 288 pixel image, 110,592 24-bit words are required to hold the structures needed to calculate the truncated wavelet transform of the image. Once the entire image has been transformed, it is processed in 85 word blocks (a zerotree) with the Sequential Baseline Coder algorithm similar to that used by the DCT based algorithm.

Although the basic building blocks of the two image compression algorithms are similar, the critical difference is the transform used. The DCT based algorithm only works on an 8 x 8 pixel image block at a time and requires very little intermediary storage. The wavelet based algorithm requires that the entire image is transformed with a global transform before the zerotrees are constructed and encoded. The amount of memory required for the wavelet based algorithm is proportional to the image size. As a lot of memory is essential, the cost of an embedded system that utilises the wavelet based algorithm will be higher. Section 3.6.2 will show that the execution speed of the algorithm is not adversely affected by the large transform and large amount of memory used. Section 3.6.3 will show that the compression ratio achievable with the wavelet based algorithm exceeds that of the DCT algorithm and produces images of a higher quality for a given compression ratio.

3.6.2 Execution speed

An efficient, hand-optimised, assembly language implementation of the Fast-DCT was used as a substitute for the wavelet transform and zerotree construction code. A moderate compression ratio of 15:1 was chosen and the algorithm was timed in two stages. The first stage consisted of the time taken to transform and encode 2,304 8 x 8 pixel blocks that make up a 512 x 288 pixel image. This initial stage corresponds to the first two stages of the wavelet based algorithm. The second main stage includes the calculation of the optimal Huffman table and the application of the table to the resultant symbol stream. This second stage corresponds to the third stage of the wavelet based algorithm.

Figure 3.33 shows the amount of time spent in each stage of the DCT based algorithm. This can be compared with Figure 3.30 where the first two stages coincidentally add up to 129 ms and the combined Huffman table generation and Huffman coding take 9 ms. Although the timings are slightly image dependent it can be seen that the execution speed of the two algorithms is very similar. In this instance the main reason that the DCT based algorithm takes about twice as long to Huffman encode is that there are twice as many symbols to be encoded. For a given image quality there are likely to be considerably more symbols to be Huffman coded under the DCT scheme. This is because the energy compaction property of the block based DCT is not as good as that of the global wavelet transform.

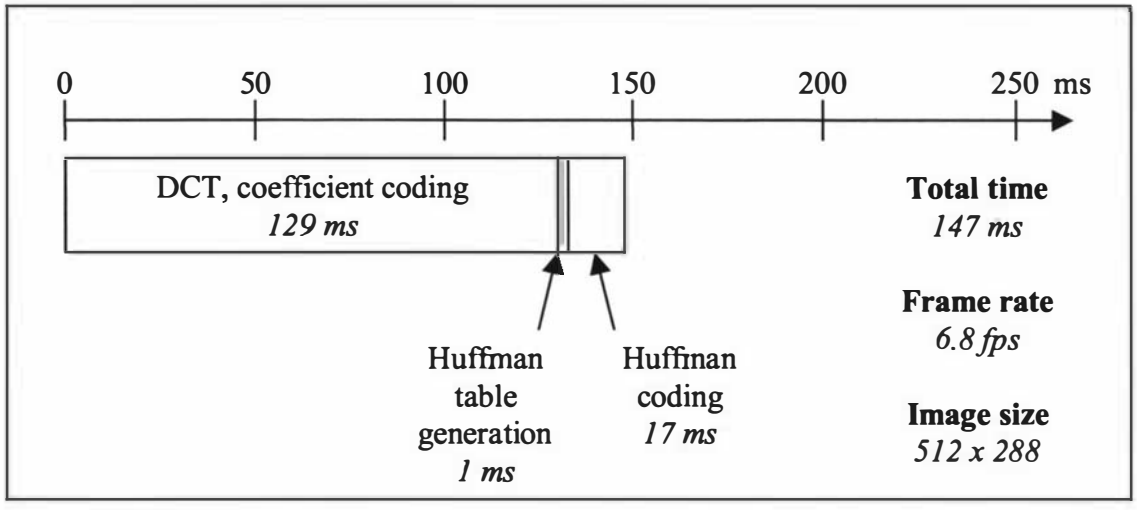


Figure 3.33 – Time spent in each stage of the DCT based algorithm

JPEG2000

The JPEG image compression standard has existed since 1991. In its current form the only transform kernel available in the standard is the 8 x 8 pixel Discrete Cosine Transform (DCT). The first major enhancement to this standard has reached final committee draft status in mid 2000 and will become an international standard in early 2001 [3.12]. The new standard will be known as JPEG2000. The new standard supports several transform kernels including the global wavelet transform with a variety of wavelet filter banks including the biorthogonal-7.9 wavelet filter used in this thesis. Given the popularity of the current JPEG image compression technique for image interchange across the Internet and the virtual absence of any common wavelet based image compression scheme, JPEG2000 should provide the general public with the first wavelet based image compression scheme. Internet users will benefit from the new standardised compression scheme as images on web pages will be able to be compressed at a higher ratio and will download much quicker without any further visual degradation.

3.6.3 Image quality

With a target compression ratio of 25:1, Figure 3.34 and Figure 3.35 show a comparison between zoomed in sub-sections of the original Lena image and reconstructed images using the global wavelet transform and the block based DCT. It is clear from these images that the wavelet transform produces a higher quality image when compared with a DCT based algorithm at a high compression ratio. The JPEG algorithm is DCT based and was used to generate the 25:1 compressed DCT image. The application used was Paint Shop Pro set to a quality factor of 80 on a scale of 0 to 100 where 0 is the best quality setting. A quality setting of "5" from Table 5.1 (page 203) was used to configure the quantisers in the wavelet compression algorithm.

Throughout the development work that produced the wavelet based image codec, subjective measures have been informally made when evaluating the quality of the compressed images. Exact measures of image quality have not been the focus of the research. Objective quality measures such as Root Mean Square Error (RMSE) are often used to automate the comparison between different image codecs. Unfortunately such measures do not usually correlate well with subjective measures produced by human observers so they are often of little use.

A more sophisticated objective measure called "Image Activity Weighting with HVS response" [3.13] has been applied to the full Lena image compressed at 25:1 with both the wavelet and DCT based compression schemes. This measure includes the model of the Human Visual System (HVS) as well as a simple measure of the *global activity* within the image to theoretically produce a quality value close to the CCIR 500 recommendation for subjective assessment of image quality using a five-point impairment scale. This scale was discussed in section 2.2 of this thesis and is repeated below for convenience.

1. Very annoying
2. Annoying
3. Slightly annoying
4. Perceptible, but not annoying
5. Imperceptible

A summary of three quality measures applied to the 25:1 compressed full Lena image is given in Table 3.5. The two "subjective" measures were taken at an image viewing size of 200 x 200 mm at a viewing distance of 600 mm.

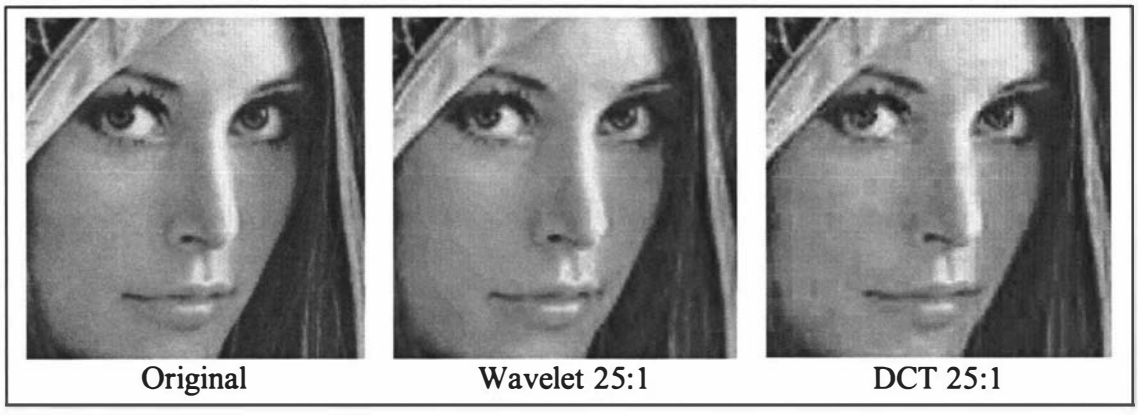


Figure 3.34 – 160 x 160 pixel subsection from 512 x 512 pixel Lena image

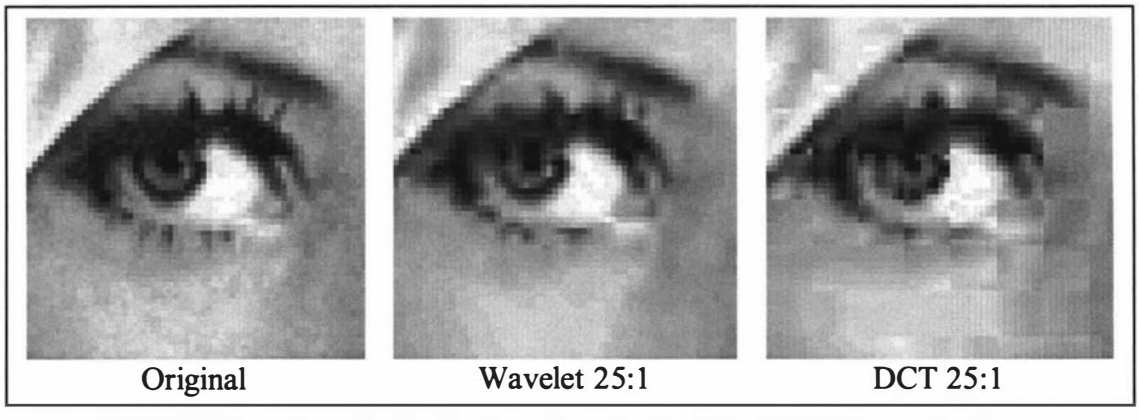


Figure 3.35 – 60 x 60 pixel subsection from 512 x 512 pixel Lena image

Compression algorithm	Root Mean Square Error [0 → ∞] (Lower = better)	Image Activity Weighting with HVS response [1 → 5] (Higher = better)	Subjective Mean Observer Score (average of 10 students) [1 → 5] (Higher = better)
Wavelet 25:1	5.59	1.1	3.8
DCT 25:1	5.83	1.1	3.0

Table 3.5 – Objective and subjective image quality measures

The RMSE and human subjective measures both favoured the wavelet based compression algorithm in terms of measured image quality at a compression ratio of 25:1. The "Image Activity Weighting with HVS response" rated both the images the same, (to two significant figures). A rating of 1.1 on the five-point impairment scale indicates that the artefacts are "Very annoying" to the "automated" observer. With this particular image, the "human" observers only found the artefacts slightly annoying. It is concluded that the creation of an automated algorithm that can be reliably used to emulate human observers is still an ongoing challenge to the image processing community!

In summary, it can be seen that the type of artefacts that are likely to appear when pushing a compression scheme to its upper limits is related to the transform kernel in the two algorithms examined. Block based algorithms tend to produce artefacts that emphasise the boundaries between neighbouring blocks and this particular artefact is absent from the global based wavelet transform. Another artefact which is common to both algorithms, is high frequency ringing which appears as a ripple effect about sharp edges in the image (Gibb's phenomena [3.14]). Both image compression algorithms reduce the amount of data to be encoded by suppressing high frequency information in the original image and this leads to the observed ringing artefact.

3.7 Discussion

This chapter has shown that an efficient DSP implementation of the wavelet based image compression algorithm has been achieved. The Motorola 56303 DSP has several architectural features that map well onto the data flow graphs that define the data flows and operation of the wavelet compression algorithm:

- The dual accumulators and the powerful "`mac`" op-code are used to efficiently calculate the Quadrature Mirror Filter at the heart of the wavelet decomposition. The two on-chip data spaces (`X:` and `Y:`) are used to provide sufficient bandwidth to the filter calculation by storing the coefficients in the `Y:` space and the pixel data in the `X:` space.
- Three DMA channels are used to efficiently marshal image data and coefficients to and from the DSP during the wavelet decomposition. Five of the six on-chip DMA channels are used during the zerotree construction algorithm. The automated transfer of two-dimensional structures is supported by the DMA controller which is essential for working efficiently with image data.
- Advanced indirect addressing modes are used to efficiently scan the zerotree structure for run-length coding.
- Bit-reversed addressing is used to create the bit patterns for the Huffman codes. The "`insert`" and "`merge`" op-codes and the 56-bit wide accumulators are used to efficiently concatenate the Huffman codes to form the compressed bitstream.

A useful comparison between the DSP architecture and the Intel Pentium architecture has been presented to illustrate the implementation efficiency of the wavelet transform on different processors. The use of the MMX signal processing extensions of the Intel Pentium processor provides a considerable improvement to the implementation efficiency of the wavelet reconstruction algorithm. When this measure is normalised to clock cycles per pixel, the DSP implementation is still six times more efficient than the MMX accelerated Pentium processor.

This chapter has made a comparison between the implementation efficiency of the 80 MHz Motorola 56303 DSP and the 200 MHz Intel Pentium MMX processor. These two hardware platforms were chosen in early 1997 and represent what was state of the art at that time. Over the past three years both processors have been superceded by newer variants of each processor, Table 3.6 and Table 3.7 briefly compare the capabilities of DSPs in 1997 and 1999 where the cost of the processor in small quantities was constrained to be less then US$50. The BDTImark is a composite score (higher = better) based on the DSP's performance on a group of typical DSP application kernels and these figures have been sourced from Berkeley Design Technology, Inc. [3.15].

Manufac-turer	DSP	Data / Program word width	Clock speed (MHz)	DMA channels	Accu-mulator width	Memory access per cycle	External address	BDTI-mark
TI	TMS320C54x	16/16	100	0	40	3	8M	25
Motorola	DSP563xx	24/24	80	6	56	3	16M	20
Analog Devices	ADSP-21xx	16/24	50	2	40	3	16k	13

Table 3.6 – State of the art DSPs under US$50 at start of 1997

Manufac-turer	DSP	Data / Program word width	Clock speed (MHz)	DMA channels	Accu-mulator width	Memory access per cycle	External address	BDTI-mark
TI	TMS320C54x	16/16	100	6	40	3	8M	25
Motorola	DSP563xx	24/24	100	6	56	3	16M	25
Analog Devices	ADSP-21xx	16/24	75	2	40	3	16k	19

Table 3.7 – State of the art DSPs under US$50 at end of 1999

The highlighted cells in Table 3.7 indicate areas of change in the DSP families over the 3 years. In 1997 Texas Instruments (TI) had the powerful TMS320C54x just under US$50 but by 1999 TI's new TMS320C62xx processor is still over US$100 in quantity thus there is no significant change in processor offerings from TI over the three year period. Motorola have also not offered any processors more powerful than the 100 MHz DSP56303 in the past three years. Most of the enhancements to Motorola's DSP line have involved adding peripherals to the DSP for telephony algorithms as well as adding more on-chip SRAM. Analog Devices was the only

other company to offer DSPs under US$50 that were reasonably powerful yet they still fall short in capability when compared with TI and Motorola.

The provision of multiple DMA channels on the Motorola 56303 DSP and the data bus width of 24 bits are significant factors when comparing Motorola and TI DSPs for image processing algorithms. The 24-bit bus allows for a greater memory bandwidth and a greater precision in arithmetic operations. The sophisticated six channel DMA controller allows the Motorola DSP to efficiently pipeline two-dimensional blocks of data to and from external SRAM. The wavelet algorithm uses five of the six DMA channels as the zerotrees are constructed which further confirms that the Motorola 56303 was a good choice of processor for the wavelet based compression algorithm. Interestingly, the 1999 variant of the TI DSP now has 6 DMA channels just like the Motorola DSP.

A brief comparison between a block based DCT algorithm and the global wavelet based algorithm is made to compare the wavelet algorithm with the well-known JPEG style image compression system. This comparison shows that the implementation efficiency is nearly identical when the achievable frame rate is compared, yet the image quality and compression ratio achievable is far superior for the wavelet based algorithm. The main disadvantage of the wavelet based algorithm is the large amount of storage required to compute the global wavelet transform.

The next chapter will look at image processing algorithms that work co-operatively with the wavelet based image compression system. These algorithms add value to the digital camera as they offer image authentication and motion detection services with very little computational overhead.

3.8 References

[3.1] **Motorola,** *DSP56300 24-bit Digital Signal Processor Family Manual,* 1994. Motorola.

[3.2] **Huffman, D. A.,** *"A method for the construction of minimum-redundancy codes,"* Proceedings of the IRE, vol. 40, no. 9, pp.1098-1101, 1952.

[3.3] **Body, N. B. and Bailey, D. G.,** *"Efficient Representation and Decoding of Static Huffman Code Tables in a Very Low Bit Rate Environment,"* Proceedings of IEEE International Conference on Image Processing (ICIP-98), Chicago, USA, pp.90-94, vol. 3, 4-7 October, 1998. ISBN 0-8186-8821-1.

[3.4] **Nelson, M. and Gailly, J.,** *The Data Compression Book,* Ed. 2, 1996. M&T Books, ISBN 1558514341.

[3.5] **Brislawn, C.,** *WSQ Gray-scale Fingerprint Image Compression Specification* [Web Page]. 1993. Available at: http://www.c3.lanl.gov/~brislawn

[3.6] **Analog-Devices,** *Low Cost Multiformat Video Codec - ADV601 Datasheet* [Web Page]. 1997. Available at: http://www.analog.com/pdf/2011_0.pdf.

[3.7] **Intel,** *Intel Signal Processing Library Reference Manual,* 1997. Intel Corporation.

[3.8] **Eyre, J. and Bier, J.,** *DSP Processors Hit the Mainstream*; Computer, pp.51-59, August 1998.

[3.9] **Halfhill, T. R.,** *AMD K6 takes on Intel P6*; Byte, pp.67-72, January 1996.

[3.10] **Intel,** *FIR and IIR Filtering using Streaming SIMD Extensions,* Ed. 1.1, 1999. Intel Corporation.

[3.11] **Gwennap, L.,** *Microprocessor Report - Intel's MMX Speeds Multimedia* [Web Page]. 1996 Mar 5. Available at: http://www.chipanalyst.com/report/articles/mmx/mmx.html.

[3.12] **JPEG,** *JPEG2000 Committee Drafts* [Web Page]. 2000. Available at: http://www.jpeg.org/CDs15444.htm.

[3.13] **Lowe, D. and Ginige, A.,** *"Image Quality Assessment using an Image Activity Weighting and the HVS Response,"* Image and Vision Computing New Zealand, New Zealand, pp.169-176, 1993.

[3.14] **Weeks, A. R.,** *Fundamentals of Electronic Image Processing,* Ed. 1, 1996. IEEE Press.

[3.15] **Blalock, G.,** *The BDTImark: A Measure of DSP Execution Speed* [Web Page]. 1997. Available at: http://www.bdti.com.

4

Motion Detection and Image Watermarking

The development of an intelligent remote activity monitoring system is the goal of this thesis. A digital camera alone does not require any intelligence to simply relay compressed images to a remote computer. There is however the inherent potential to embed an intelligence or a decision making capacity into a digital camera. The wavelet decomposition of the image performed by the DSP is a powerful technique when used to analyse an image. By performing certain image processing functions, in addition to image compression, at the camera this has two main advantages over performing the same analysis at a central computer. The required interconnection network bandwidth is reduced as the decision to send or not send an image can be made at the camera. Also there is no longer a need for a centrally located processing resource powerful enough to process all the images from all the cameras in a system in real-time.

This chapter will examine two image processing algorithms that were implemented for real-time execution within the digital camera. These algorithms are for motion detection and image watermarking. The addition of these algorithms adds value to the camera and they can be implemented with a low computational cost. Both algorithms have been specially designed to make use of the wavelet decomposition computed in the first stage of the image compression algorithm. Both algorithms can therefore be described as being co-operative with the image compression algorithm described in chapters 2 and 3.

4.1 Motion detection

Motion detection is often used in video compression systems to increase the compression ratio by exploiting the redundant data present between individual frames. In typical video sequences, many pixels stay at the same intensity level over a group of frames and image subtraction or interframe image analysis techniques can be used to detect which pixels correspond to stationary or moving objects in a scene. In order to make good use of motion detection in an image coding system, extra memory is generally required to store short image sequences for analysis. A large amount of computation is also required to analyse the interframe difference images to allow the images to be encoded efficiently. The MPEG (Motion Pictures Expert Group) family of international standards provide a framework for producing interchangeable bitstreams that can be decoded by a standardised MPEG decoder [4.1-3]. The encoding of a set of images into an MPEG bitstream is computationally expensive as interframe motion is not only detected but is also compensated for by quantifying the magnitude and direction of the motion. This information is represented as a field of motion vectors across the image.

The image compression system used in the digital camera described in chapters 2 and 3 is intraframe based and does not rely on detecting motion to achieve compression. One approach to image sequence compression is to signal change between image frames which is similar to detecting interframe motion. In the context of surveillance, a motion detection capability is needed to enable the automatic monitoring of activity at remote locations without the need for a human operator. The motion detection algorithm has been designed to work co-operatively with the intraframe image compression to ensure an efficient implementation. The following section describes the motion detection requirement.

4.1.1 Requirement for motion detection

In an activity monitoring system, the presence of movement in a scene is usually associated with an event that requires human attention. In the context of surveillance there are likely to be long periods of time where there is little change in a scene. It is very tedious to expect a human operator to observe a static scene for long periods of time and still to be able to reliably detect a sudden, but brief change in a scene. Image processing techniques can be used to detect motion in a scene as this is an essential step in an automated activity monitoring system. By adding automated motion detection at the camera, only image sequences that contain significant motion need to be highlighted by the system to the human operator. Also as the image sequences are inherently digital they can be easily stored for later retrieval and analysis by a human operator.

The Passive Infra-Red (PIR) sensor is a standard sensor that is used in the surveillance industry to detect the presence of motion in an enclosed area. The sensor is constructed so that it can monitor changes in location of infra-red heat sources including the human body. The sensor usually comprises of one or two single element infra red sensors that simply detect a change in the infra red radiation in the room over time as viewed through a fresnel lens. The sensor can only report the presence of motion, not the location or trajectory of the motion. These limitations lead to the problem of alarm verification especially if the location of the monitoring station is remote from the scene. The first step in developing a "smart" camera is to add a motion detection capability. The camera will perform the function of a PIR sensor, it will also detect motion in the visible spectrum as well as the near infra-red spectrum if used with the correct lighting. Secondly the camera can be used to provide remote alarm verification by the generation of a sequence of digitised images from the scene both before and after the initial detection of motion.

The use of digital images to detect motion allows the flexibility to set regions of interest within the camera field of view. Some regions of the field of view include unwanted sources of motion such as trees moving in the wind. To prevent false

alarms these regions can be specified by an operator and then masked out from the motion detection algorithm.

4.1.2 Review of motion detection techniques

There are a number of image processing techniques that are used for scene analysis for sequences of images. These techniques are often used in automated inspection systems to relieve human operators of the tedium of inspection tasks. The following sections briefly describe three techniques for analysing the motion content of image sequences.

4.1.2.1 Root Mean Square Error measurement

The simplest technique involves detection of motion over the entire image. Sequential images can be subtracted from each other and the resulting differences are then squared, accumulated and the square root is taken to produce a single Root Mean Square Error (RMSE) measurement for the pair of images [4.4]. In a real image capture system there will be noise present in the images so that even in a scene where there is no motion between sequential images, there will still be a residual RMSE between the images. The RMSE value can be plotted over time and a suitable threshold can be set to distinguish between system noise and real sources of motion.

4.1.2.2 Optical flow analysis

As an object moves across the field of view of a camera, the pixels that the object is composed of generate a collection of two-dimensional velocities. A complete collection of pixel velocities between two image frames is called a two-dimensional motion field. The measurement of this field can be achieved in a number of ways and is called the optical flow [4.5]. It is generally not possible to precisely compute the

velocity of every pixel in the image due to the complex mapping of three-dimensional objects onto a two-dimensional image plane.

The region based matching technique [4.4] of optical flow analysis is similar to the motion compensating techniques used in MPEG encoders. The image is divided into small blocks, typically 16 x 16 pixels in size, and the location of each of the blocks is searched for in the previous image in the sequence. The resulting motion vectors form a vector field that describes the apparent motion observed in the scene by assuming that objects move in simple planar-motion [4.4]. The vector fields can be combined over several frames to model the apparent trajectory of the objects and camera through the scene. The matching can be performed in the spatial domain as in the MPEG encoder or can be performed in the transform domain. Transform domain processing can be used to measure both the energy magnitude and the phase changes between frames to quantify approximate motion. A hierarchical search [4.4] is a useful technique that can be applied to reduce the computational complexity of the problem. This technique initially uses a low resolution version of the image to achieve an approximate template match followed by localised searches on higher resolution images to refine the accuracy of the process. The low resolution images can be obtained from the original high resolution through a series of low pass filtering and downsampling operations. This is similar to the processes used in the wavelet decomposition to produce a set of multiresolution approximation images.

4.1.2.3 Object segmentation and tracking

Object segmentation uses a single image as a basis to determine the location of significant objects within the scene. This is often performed by first intensity thresholding the grey scale image to separate objects from the image background [4.4]. Once the boundaries of several objects have been identified in a scene, their location can be compared in subsequent images of the scene to determine if motion is present. Other features of the detected objects such as their size or shape can be used to mask out objects that may not be of interest. Object tracking algorithms are required to determine which objects in a frame are most likely to correspond to

objects in a subsequent frame. Objects can be occluded or appear suddenly which further complicates the problem of object tracking.

4.1.2.4 Motion detection for surveillance

The three motion detection techniques described above have significantly different computational complexity. The Root Mean Square Error (RMSE) calculation works well with a wide variety of images due to its simple model of image change. Optical flow and image segmentation are significantly more sophisticated techniques for analysing object motion within a scene. These algorithms are more complicated and the amount of computation required is much greater than simple measurements such as RMSE. The robustness of the optical flow and image segmentation algorithms is very dependent on scene content. These techniques are often used to best effect in controlled scenes such as machine vision inspection systems.

In a surveillance application it is often sufficient to simply report that there is significant motion in a scene and the RMSE method is suitable for this purpose. Coarse localisation of motion can be achieved by dividing the image into a number of regions-of-interest. The RMSE for each of these regions can be calculated separately to improve the reliability of the system to only detect significant motion in significant areas of the image.

4.1.3 Wavelet decomposition and motion measurement

Rather than using a full resolution image (of 512 x 288 pixels) to compute the root mean square error between regions in subsequent frames, it is possible to use a downsampled image. It was subsequently found that the amount of computation required is dramatically reduced without compromising the reliability of motion detection. The amount of memory required to store the previous frame is also reduced when a low-resolution version of the frame is stored. Wavelet decomposition produces a series of low-pass filtered and downsampled approximation images at each

level of the decomposition. At the fourth level of decomposition, the image is reduced by a factor of 16 (to a size of 32 x 18 pixels) and can be stored in a 256^{th} of the space required to store the full size image. The reduction is size also reduces the amount of computation required by the same factor. A further advantage of using low-pass filtered downsampled data to perform the root mean square error calculation is that image system noise is reduced prior to the image subtraction calculation.

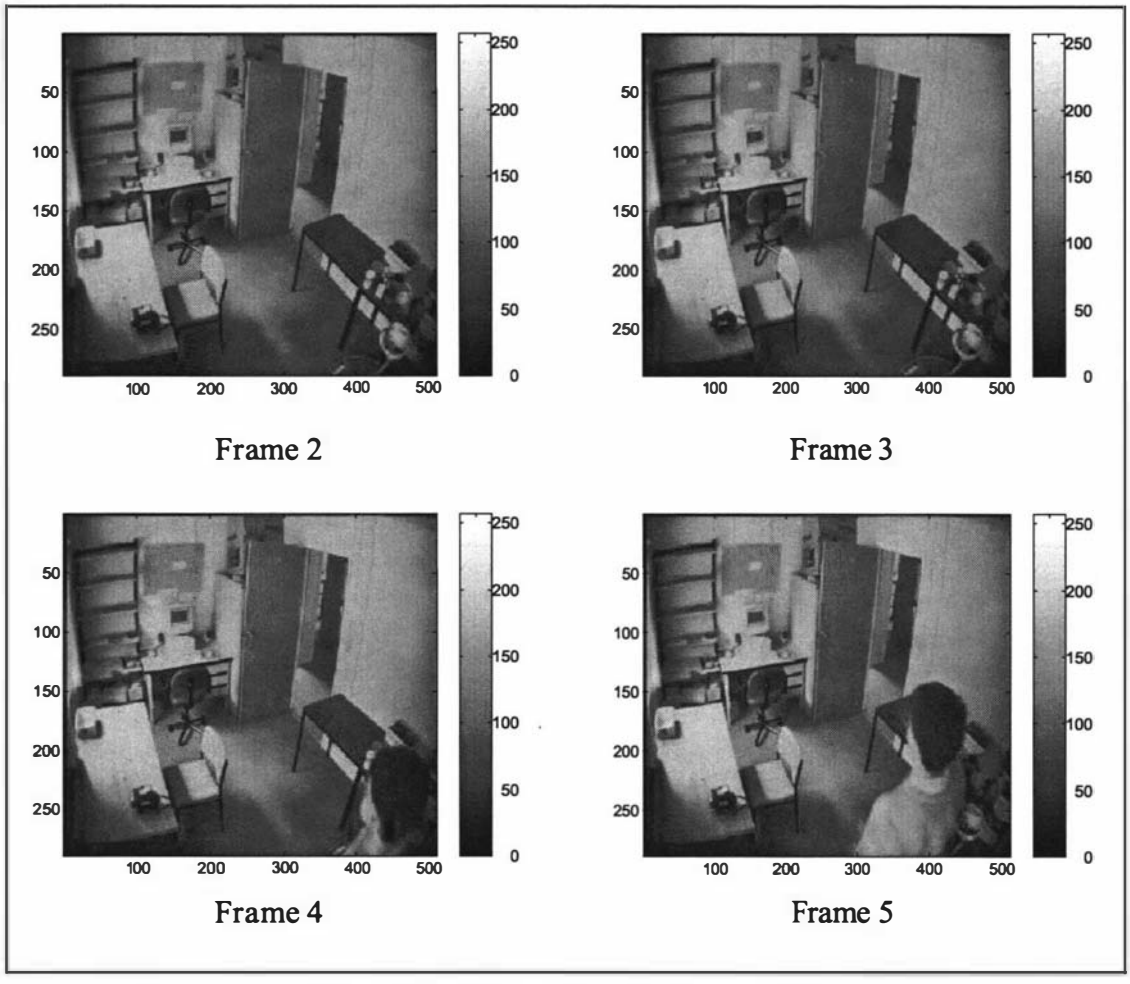


Figure 4.1 – 512 x 288 pixel raw images from 140 frame sequence

Figure 4.1 shows four frames at full resolution from a sequence of 140 images taken at a frame rate of 2 frames per second. The images were taken with the digital camera and have been moderately compressed using the wavelet based compression algorithm described in chapters 2 and 3. The sequence shows the author entering the camera's field of view from the lower right corner of the frame. The full sequence is summarised in Figure 4.4 as a set of 32 x 18 pixel approximation images.

Figure 4.2 shows difference images produced at full resolution between the four frames shown in Figure 4.1. Frame 3-2 mainly shows the high frequency noise present in the system as there is no significant motion other than a slight flickering of the computer monitor in the scene. Frame 4-3 contains significant energy in the lower right hand corner of the image as the author's head occludes the desk in the foreground. Frame 5-4 also locates significant motion in the lower right hand corner of the frame as the author moves through the scene. Frame 3-2 is repeated in the lower right hand corner of Figure 4.2 with enhanced contrast to reveal the high frequency noise more clearly. From a surveillance point of view, the initial frames 2 and 3 contain insignificant motion whereas frames 4 and 5 contain motion that is significant and should trigger an alarm or alert an operator.

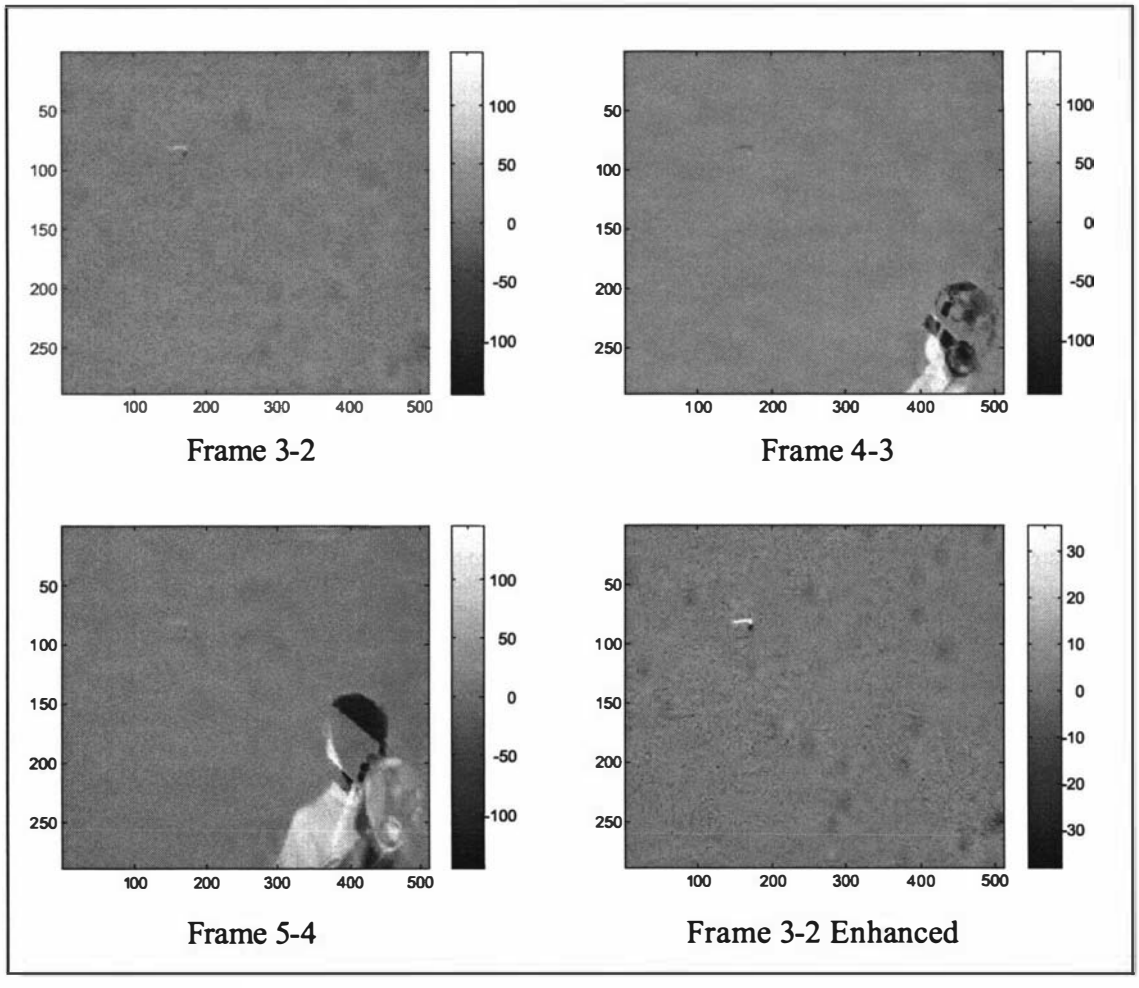


Figure 4.2 – 512 x 288 pixel raw difference images from 140 frame sequence

The images in Figure 4.3 are the first five frames of the sequence downsampled to 32 x 18 pixels with the four difference images. These downsampled images are the 4th level approximation images produced by the wavelet decomposition and the greyscale ranges from 0 to 255. The last three difference images correspond to the three full resolution difference images shown previously in Figure 4.2. The greyscale for the difference images ranges from −128 to +128 and the contrast is constant across the set of images. Intuitively it can be seem that there is sufficient resolution in the downsampled images to still enable a reliable detection of significant motion in the scene. The two intensity histograms shown in Figure 4.2 for the lower right hand region of the frame confirm that the measured energy will be significantly different when there is motion to be detected.

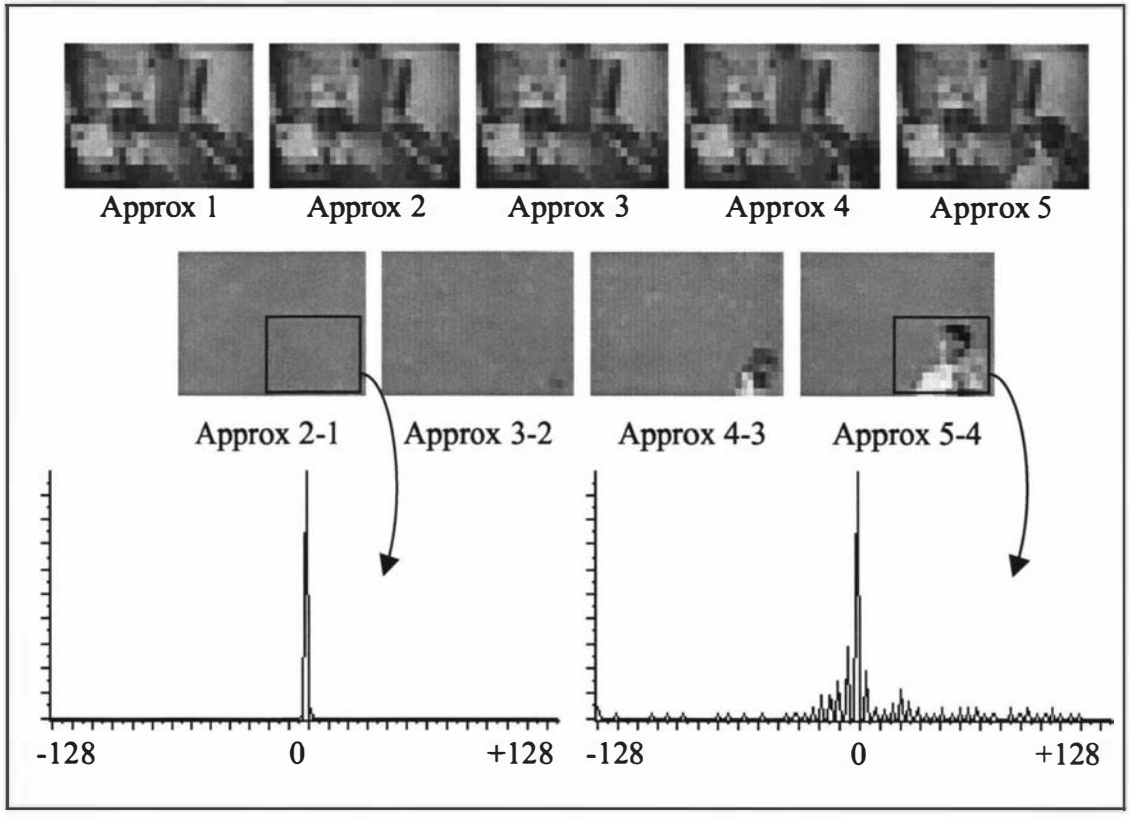


Figure 4.3 – 32 x 18 pixel approximation images from 140 frame sequence

Figure 4.4 – Full set of 32 x 18 pixel approximation images

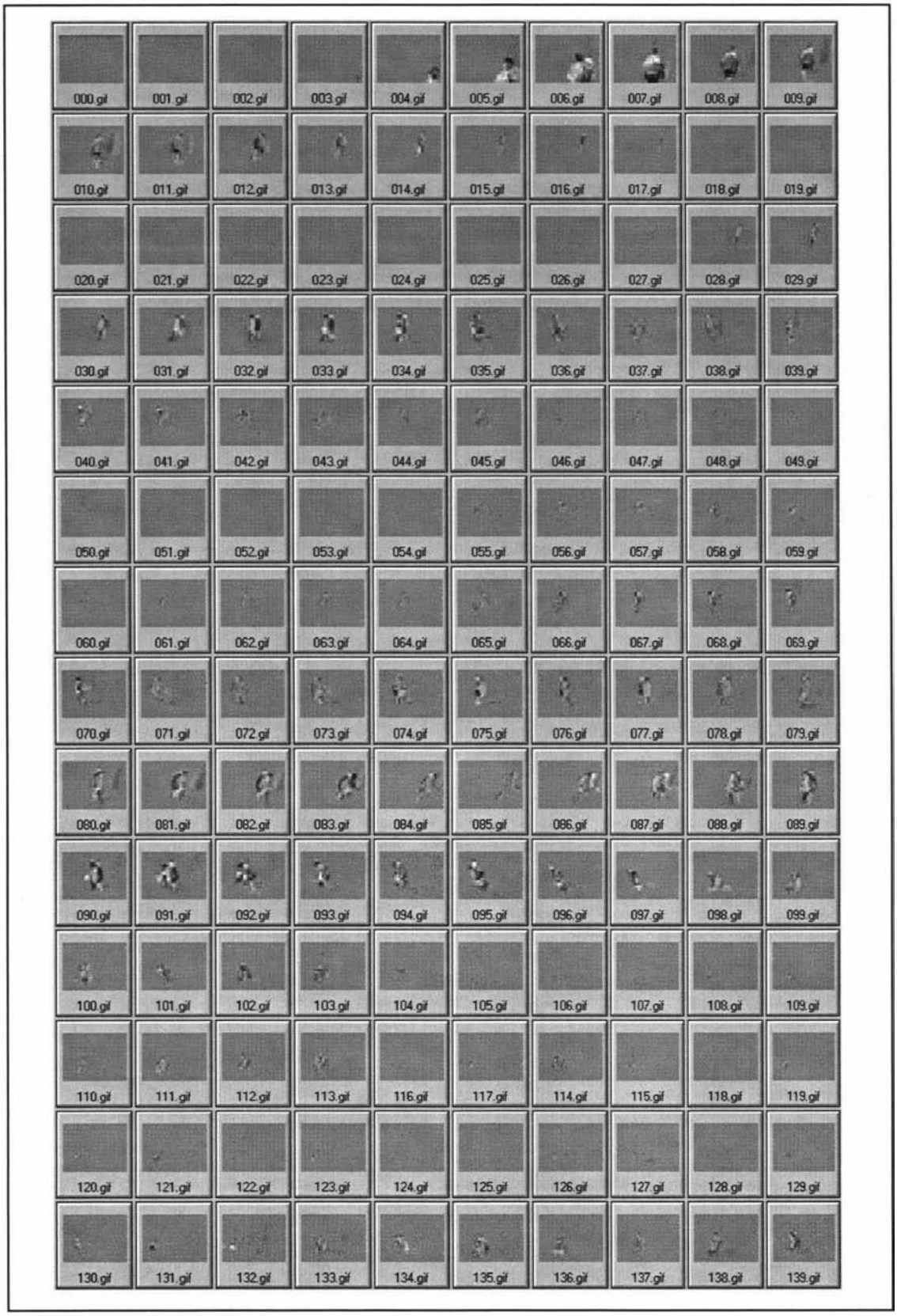


Figure 4.5 – Full set of 32 x 18 pixel difference images

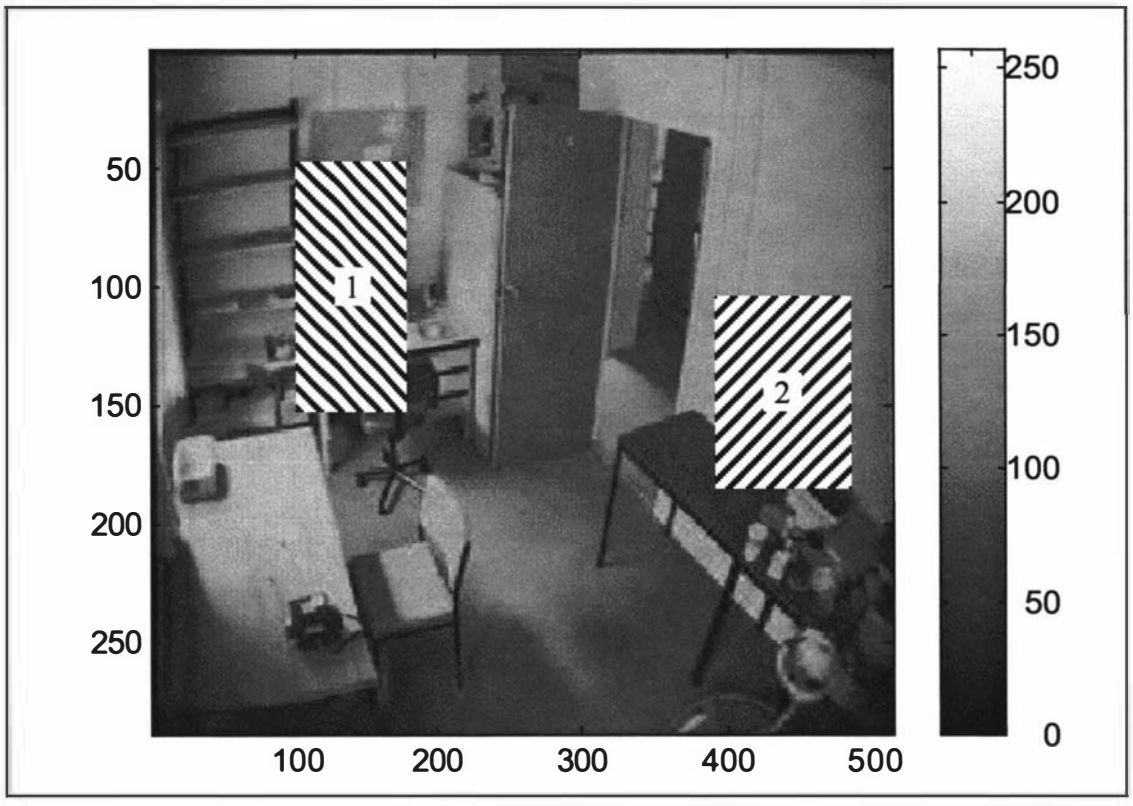


Figure 4.6 – Two regions of interest where motion is to be measured

The complete set of 140 low-resolution difference images shown in Figure 4.5 are derived by interframe subtraction from the 140 frames shown in Figure 4.4. It can be seen that over time (indicated by the frame sequence number) the amount of motion in the scene varies in both magnitude and location. In a surveillance environment it is useful to be able to specify one or more regions of interest within the frame. Each region represents an area where the presence of motion is particularly important to the operator monitoring the system. Figure 4.6 identifies two regions of interest in the frame. The first of these regions is located about the computer in the top left corner of the frame. The second region is above the table to the right of the frame. For each of the two regions, the amount of energy [4.4] is calculated in each difference image for the entire sequence. The energy is normalised by dividing the Sum of Squared Error (SSE) by the number of pixels in each region to form a Root Mean Square Error value (RMSE). The RMSE values for each region is graphed over time in Figure 4.7 and Figure 4.8. By choosing a suitable threshold, for example in this scene an RMSE value of 15, it is possible to use the regionised RMSE value in each frame to produce a binary decision about whether there is significant motion in a region at the time.

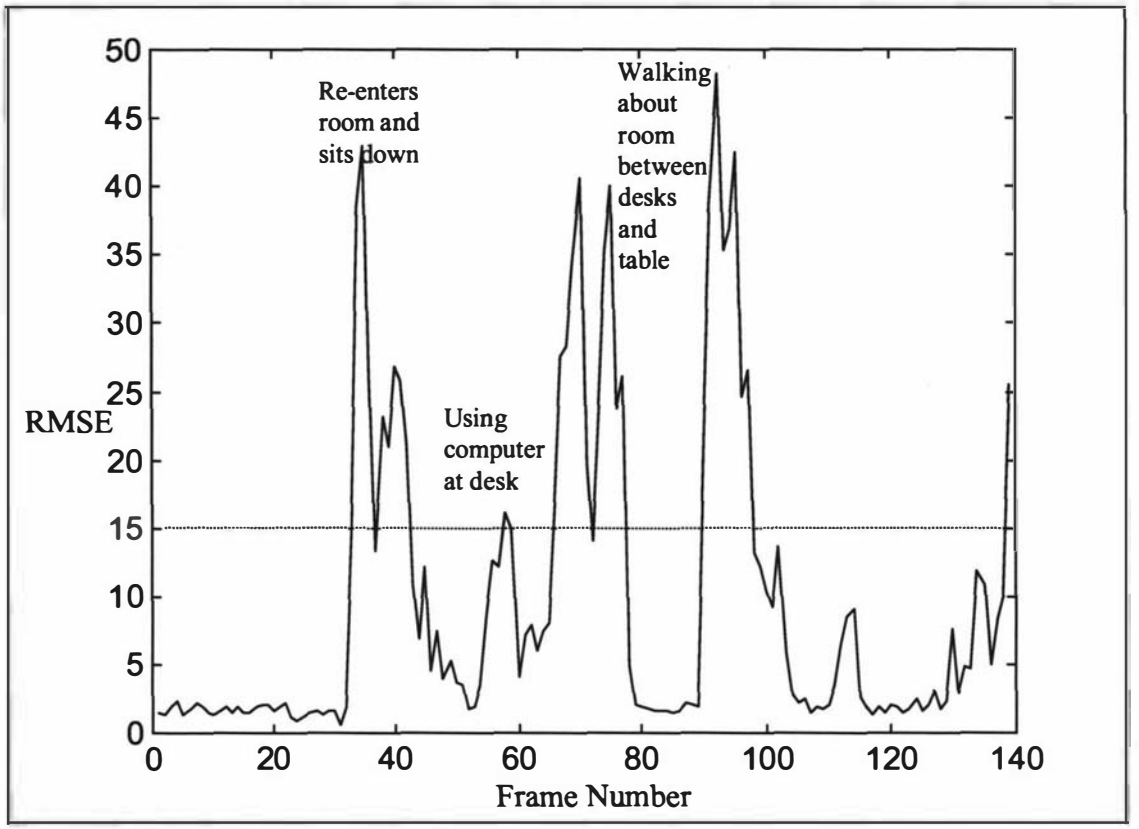


Figure 4.7 – Root Mean Square Error over 140 frames for region 1

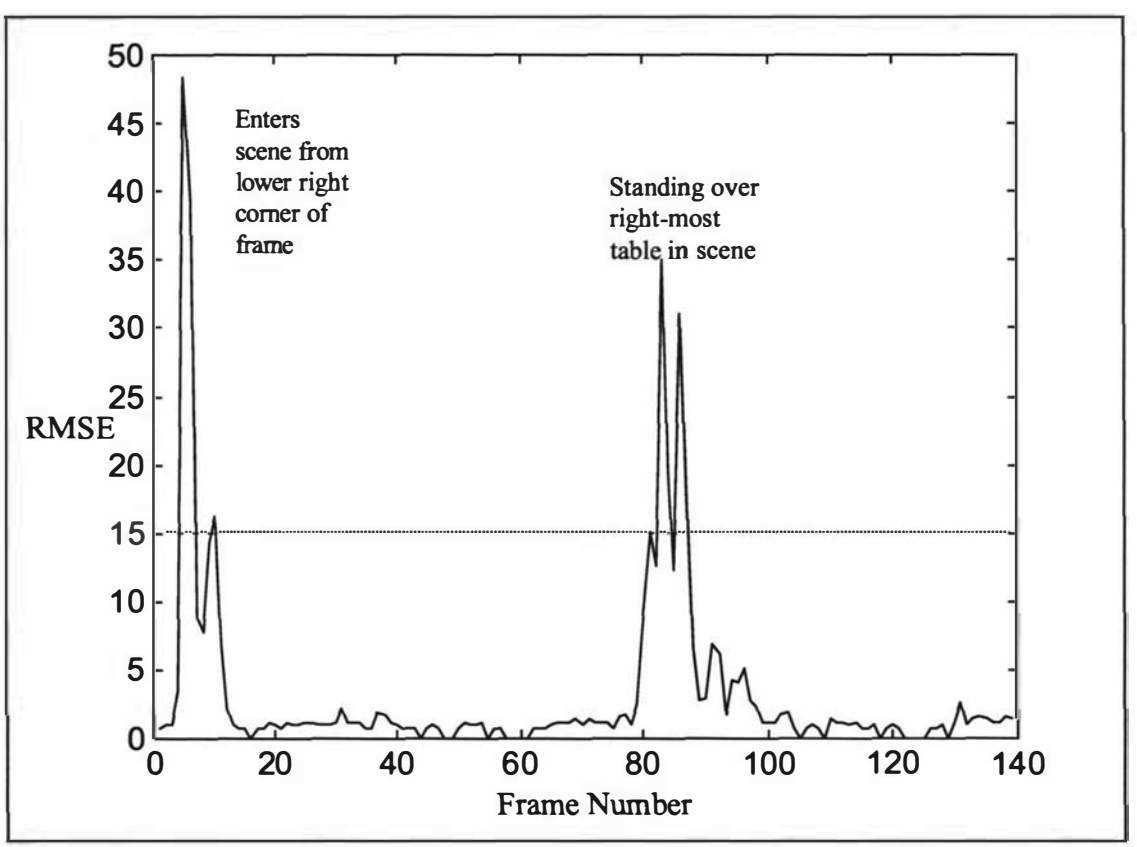


Figure 4.8 – Root Mean Square Error over 140 frames for region 2

4.1.4 Efficient DSP implementation of the motion metric

It is generally computationally expensive to perform unnecessary division and square root operations on a DSP when a simple multiplication operation can be performed instead. Calculating the Sum of Squared Error is an attractive alternative to calculating the Root Mean Square Error. By first removing the square root calculation the RMSE metric becomes the Mean Square Error (MSE). The relationship between the MSE metric and the SSE metric is shown in equation (4.1) for a region of dimension *height* x *width*. By calculating only the SSE, the final division is not performed to normalise the metric. The SSE can still be compared to a normalised threshold (MSE_{thresh}) such as the MSE value of 225 used in Figure 4.7 and Figure 4.8 (corresponding to an RMSE of 15). The only difference is that the threshold needs to be pre-scaled by the number of pixels in the region prior to the comparison being performed as shown in equation (4.2).

$$MSE = \frac{SSE}{height \cdot width} = \frac{\sum_{y=0}^{height-1} \sum_{x=0}^{width-1} \left(current_{x,y} - previous_{x,y}\right)^2}{height \cdot width} \tag{4.1}$$

$$SignificantMotion = \begin{cases} FALSE : SSE \leq MSE_{thresh} \cdot height \cdot width \\ TRUE : SSE > MSE_{thresh} \cdot height \cdot width \end{cases} \tag{4.2}$$

The calculation of the Sum of Squared Error initially requires the subtraction of corresponding pixels in the previous frame from the current frame to establish the interframe "error" due to possible motion. The second operation required is the summation of the squared pixel differences over each of the pixels within a region of interest[1]. Rectangular regions have been chosen for convenience to model the operators' regions of interest within a frame.

4.1.4.1 Implementing the SSE metric

Figure 4.9 shows the signal processing flow graph for the two main processing elements required to implement the SSE metric, an iteration of subtraction and

accumulated square operations. The two input streams of data are stored as two blocks of 576 words representing the current and previous downsampled approximation images (A_4). These images can be compactly stored in off-chip memory (Memory Unit 1) as the images are only 32 x 18 pixels in size. For each pixel in a given region of interest, the DSP calculates the pixel difference and then squares and accumulates the result. Two nested loops are used to iterate over each of the pixels that form the region of interest. The final result shown in Figure 4.9 is a single unsigned SSE value stored in a 24-bit word in on-chip memory (Memory Unit 2) that can be later used to determine whether significant motion is present in the region just considered.

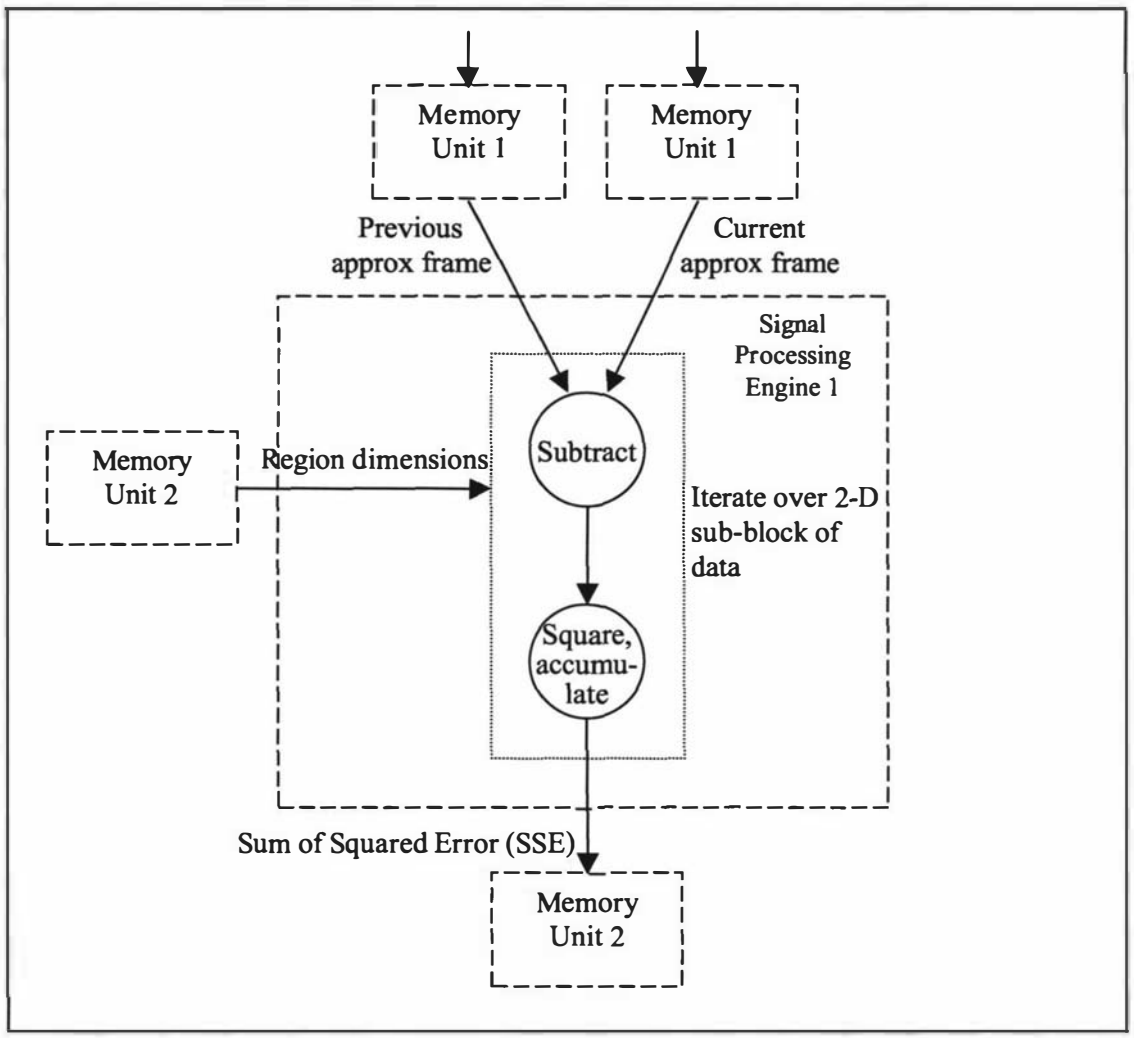


Figure 4.9 – Signal processing flow graph showing SSE calculation for a single region

[1] An alternative metric is the Sum of Absolute Differences (SAD) which does not require the squaring of differences. On the DSP, the square-accumulation can be performed more efficiently than the accumulation of absolute values due to the `mac` instruction so this inferior metric was not considered.

The computation of the SSE of a two-dimensional region of interest is discussed below. An excerpt of the DSP assembly language code is included in Figure 4.10 to show the kernel that performs the accumulation of squared differences. This operation can be performed in a pair of nested tight loops efficiently on the DSP. Prior to executing this code the following register assignments have been made (see Figure 3.8 and Figure 3.9 in chapter 3 for the function of each of these registers). The R0 register points to the top-left pixel of the region of interest in the current frame and R1 points to the corresponding pixel in the previous frame. The Y1 and A1 registers contain the width and height of the region to be processed. The N0 and N1 offset registers contain the necessary jump required to move from the end of each row in the region, to the beginning of the next row. Finally the B accumulator is used to accumulate the Sum of Squares Error.

```
   clr    b              ;B used to accumulate Sum Square Error (SSE)
   do     A1,SSE1        ;A1 contains height
   do     Y1,SSE2        ;Y1 contains the width
   move   Y:(R0)+,A      ;A4 coefficients between $004000 and $400000 (22 bits)
   move   Y:(R1)+,Y0     ;R0 = current A4 image, R1 = previous A4 image
   sub    Y0,A           ;Difference in A
   nop                   ;Pipeline stall
   move   A,X0           ;Difference ready to be squared from X0
   mac    X0,X0,B        ;Creates a 45 bit product accumulated up to 32*18=576
SSE2                     ; times = 10 extra bits required
   move   (R0)+N0        ;This will still fit into a 56 bit accumulator
   move   (R1)+N1        ; (55 bits)
SSE1                     ;Update address registers to start of next row
   asr    #7,B,B         ;Scale 55 bit unsigned product back to 48 bits
   nop                   ;Pipeline stall
   move   B1,Y:BLKSSE    ;Store output in BLKSSE as an unsigned 24 bit number
```

Figure 4.10 – Computation of SSE for a single region

The hardware-looping feature of the DSP is used to create two nested loops to traverse the rectangular block of data. The hardware assisted looping feature of the DSP architecture allows the two loops to execute with zero-overhead. Inside the nested loops, the A accumulator is used to perform the pixel subtraction and the result is move to the X0 register. The multiply and accumulate (mac) instruction is used to square the value in the X0 register and accumulate the result in the B accumulator in a single cycle. For each pixel in each row of data, six clock cycles are required to accumulate the SSE attributed to that location in the image. At the end of each row

two additional clock cycles are used to make the `R0` and `R1` registers point to the start of the next row of data.

4.1.4.2 Arithmetic overflow considerations

Throughout this process of subtracting, squaring and accumulating the 24-bit approximation image pixels, the possibility of arithmetic overflow is a critical factor to be considered. Scaling in the wavelet filter bank calculations causes the approximation pixels at the fourth level of the wavelet decomposition to have a large dynamic range. The pixel values range between two to the power of 14 (0x004000) and two to the power of 22 (0x400000). When the coefficients are subtracted and squared, a 45-bit product will result. Additionally if the entire image is chosen to be a region of interest then there would be up to 32 x 18 accumulation operations performed on the `B` accumulator. At most, 10 additional bits are required to prevent an overflow when summing 576 numbers. When each number is potentially up to 45 bits wide, a total of 55 bits is required. It is concluded that the two 56-bit accumulators in the DSP can handle the worse case scenario without arithmetic overflow, yet still have enough precision to handle small differences in small regions of interest.

The final accumulated result is shifted right by seven bits to ensure that the most significant bits of the SSE fit within the lower 48 bits of the `B` accumulator. The `B1` portion of the accumulator (representing bit numbers 24 to 47) can then be used to extract the scaled, 24-bit unsigned SSE which is written out to on-chip memory for later processing.

4.1.4.3 System considerations

The process describe above is repeated for each of the regions of interest in the image to produce a separate SSE value for each region. Each SSE value is compared with the MSE_{thresh} value scaled by the number of pixels in each region to determine whether

or not there has been significant motion in the last two frames for each region. The binary decision can then be transmitted to a centrally located computer if significant motion is detected. Also the control software in the camera will decide whether or not to transmit a stored image sequence to describe to the detected motion event.

4.2 Image watermarking

Image watermarking is a technique that can be applied to digital images. It can be used to authenticate and manage ownership of digitally stored images. An image watermark can be either invisible or visible. The Alexander Turnbull Library in New Zealand [4.6] offers historic images held in its collection over the Internet with a visible library logo embedded over the image as shown in Figure 4.11. Invisible watermarks are usually added to images and subsequently detected and verified by the use of specially developed software. A wide range of techniques are possible and potentially useful. Such watermarks are designed to alter the original image in ways that are not easily detected by human observers.

The following two sections examine the reasons for applying image watermarking techniques as well as background on the techniques available. Section 4.2.3 describes a new watermarking technique designed to be implemented in the wavelet domain. This technique has been designed to be co-operative with the image compression algorithm described in chapters 2 and 3. The performance of this new algorithm is examined in section 4.2.4. Section 4.2.5 details how the image watermarking algorithm can be efficiently implemented on a DSP.

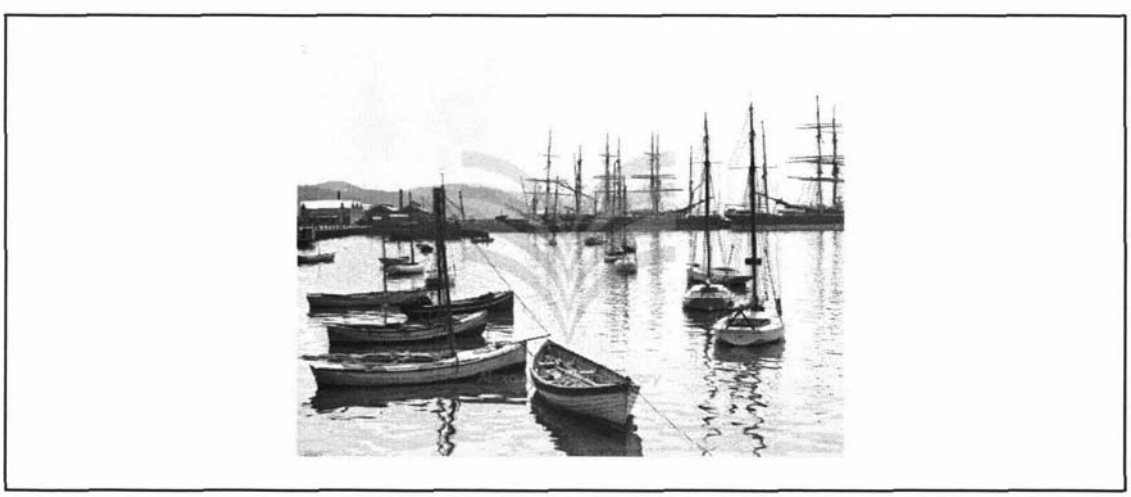


Figure 4.11 – Visible image watermark

4.2.1 Reasons for image watermarking

There are several motivations for image watermarking. They all stem from the fact that digital images are easily altered on a computer. Image watermarking can be used to authenticate the visual integrity of an image. An authentic image is validated as being free of any tampering. Image watermarking can also be used to embed image ownership information. This information can be used to assert who the legal owner of the image is.

4.2.1.1 Image authentication

Once an image is stored on a computer it is very easy to alter it at the whim of the user. This ease of alteration can be a problem when the image is to be used as evidence. Even conventional 35mm photography produces a negative that is more difficult to tamper; specialist equipment is required rather than just a computer. To protect the integrity of digital photography, invisible digital image watermarks can be used. An image watermark is often spread throughout the entire image, thus protecting the entire image from unauthorised alteration. Any attempt to tamper with the image results in the hidden watermark becoming corrupted and thus signalling the presence of an alteration.

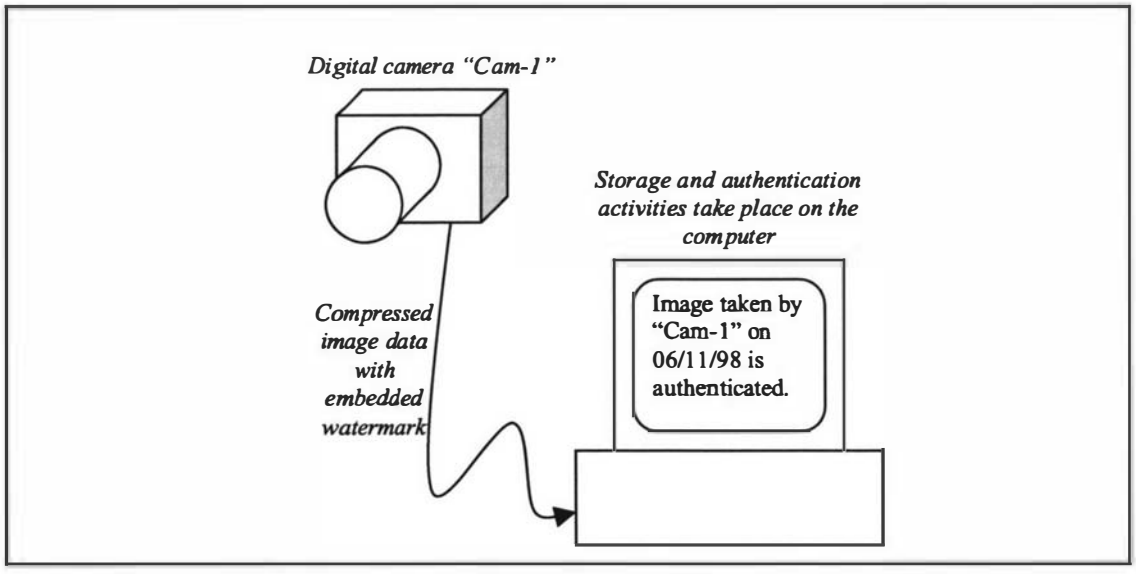


Figure 4.12 – Digital camera sends compressed image to computer for storage

In order to detect whether tampering has occurred on an image protected by an invisible watermark, a computer algorithm is required to process the image to recover the hidden watermark information. "Steganography", defined as the art of hiding messages inside images [4.7], is closely related to digital image watermarking. In the context of surveillance there are typically three significant pieces of information that an image watermark aims to protect and communicate. They are the authenticity of the image, a time stamp and the identity of the source of the image. The digital image watermark is used to provide a "seal of authenticity" for the image that it is protecting. Figure 4.12 shows a typical application of image authentication of where a digital camera sends a compressed image with an embedded watermark to a remote computer. The image on the remote computer can be stored and authenticated.

Figure 4.12 also illustrates the locations on the complete system where tampering with the image may occur. With this information a list of potential image "attacks" can be considered:

- The raw image inside the digital camera is the most vulnerable aspect of the system. Any tampering with the image prior to the compression and watermarking processes will succeed. To counter this the camera can be constructed in such a way that the opening of the camera enclosure can be detected by sensors inside the camera and the state of the camera enclosure can be reported to the controlling computer.
- The compressed image data with the embedded watermark could be intercepted between the camera and the computer. In order to alter the image data, the attacker would need to know two pieces of proprietary information. They would need to know how to decompress and re-compress the image and also how to extract the embedded watermark information. The attacker could also alter the reconstructed image, but would need to be able to compress the image and embed the correct watermarking information. If such an attack were to succeed in a "live" system, these activities would need to occur within the short time interval between transmitted frames. If these tasks were not executed in real-time, the controlling computer would detect the absence of image data from the camera as an abnormal condition.

- A variation of the previous attack could be the situation where the tampering of the compressed image data is performed on historic image files archived on the computer. Again proprietary information is required to perform this attack.

With adequate protection of the proprietary algorithms and the format of the compressed data structures, the above attacks on image integrity can be largely thwarted. The embedded watermark data can be additionally encrypted using a secret key unique to each manufactured camera. This will provide additional security to the image integrity.

4.2.1.2 Image ownership

Image watermarking is used to better manage ownership and rights to digitally stored images [4.8]. With the advent of the Internet it has become very simple to transmit perfect digital copies of images stored in digital form. This is an advantage for the creator of the image if the creator wishes to widely distribute the image. It does however pose a problem if the creator wishes to maintain and assert ownership of the image.

Image watermarking offers the owner of an image the ability to embed either an invisible or visible message inside the image. An owner of an image may wish to embed an invisible watermark that contains a unique serial number that identifies the image as belonging to the owner, this information can be repeated many times throughout the image. If at a later date a third party reuses even a small part of this image to create a new image it may be possible for the original owner to establish the claim that part of the original image has been used without the owner's permission.

There are a number of commercial packages such as SysCOP, FBI and Digimarc that can perform digital watermarking [4.9]. Whilst image ownership issues and copyright protection are an important application of watermarking [4.10], this thesis focuses on the issue of image authentication as this is an important aspect of security applications.

4.2.2 Review of image watermarking techniques

This section will examine some of the techniques used to watermark images. The process of embedding a watermark in an image and then detecting and decoding the watermark can be regarded as a digital communications problem [4.11]. To encode the watermark information that can be reliably transmitted via the image, a digital modulation technique is required. The modulated image must not contain any visible artefacts. Encryption and error control coding techniques are required to ensure reliable and secure communication of the watermark.

The next section examines the difference between watermarking techniques that require access to the original unwatermarked (clear) image and those that are self-referencing. Then the following two sections consider watermarking in the spatial (pixel) domain and watermarking in the transform domain.

4.2.2.1 Original-referencing verses self-referencing watermarking

Depending on the intended application of the digital watermark, it may be advantageous to use a watermarking technique that is based on the image difference between the original image and the watermarked image. This has the consequence that the watermark cannot be detected or decoded without having access to the original unwatermarked image. In other applications it may not be possible to have access to the original image after the watermarked image has been produced. In this case the watermarked image must contain a form of self-referencing watermark that allows the watermark to be recovered with only reference to the watermarked image [4.8].

The amount of watermarked data that can be encoded into an image is very different for the two schemes. When an original-referencing scheme is used, very small perturbations can be introduced into the watermarked image to signal a message. As the perturbations can be as small as a single greyscale level change in a single pixel, a dense message bitstream can be embedded into the image without creating any visual

artefacts that would betray the presence of the watermark. A self-referencing scheme must create a "carrier" perturbation on which it can modulate a lower bit-rate watermark message. This dramatically reduces the density of the embedded message bitstream and the magnitude of the perturbations are generally much larger to ensure reliable detection of the watermark. The difference in watermark capacity is shown in Figure 4.13 to Figure 4.15 for a one-dimensional signal such as an image scanline.

Figure 4.13 **a)** shows the original "clear" image 100 pixel scanline used to illustrate the original-referencing and self-referencing watermark techniques. Figure 4.13 **b)** represents the watermarked image which has had 25 bits of watermark information added to it. This information can only be recovered from the watermarked image if the original image is available as the two images need to be differenced. Figure 4.13 **c)** is the difference signal that clearly contains the 25 watermark bits.

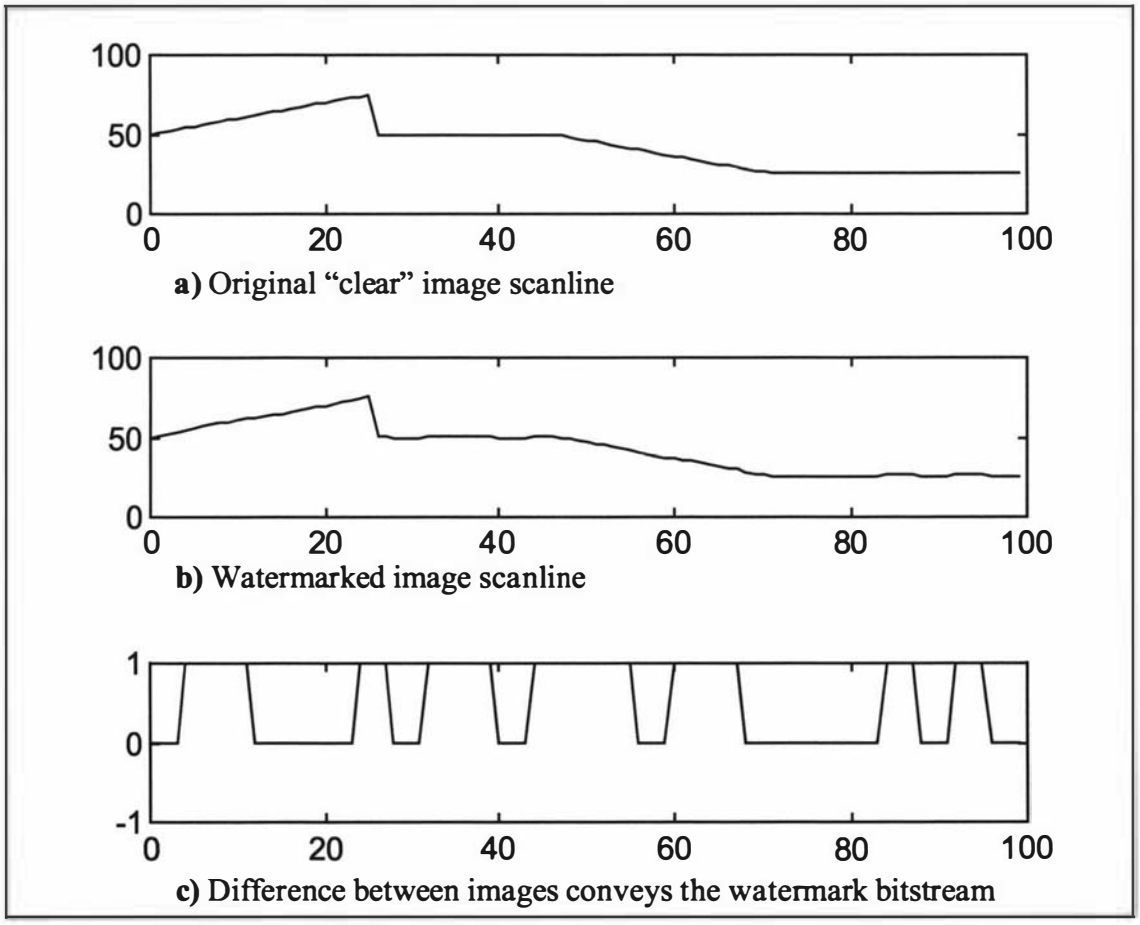


Figure 4.13 a)-c) – Original-referencing watermark example

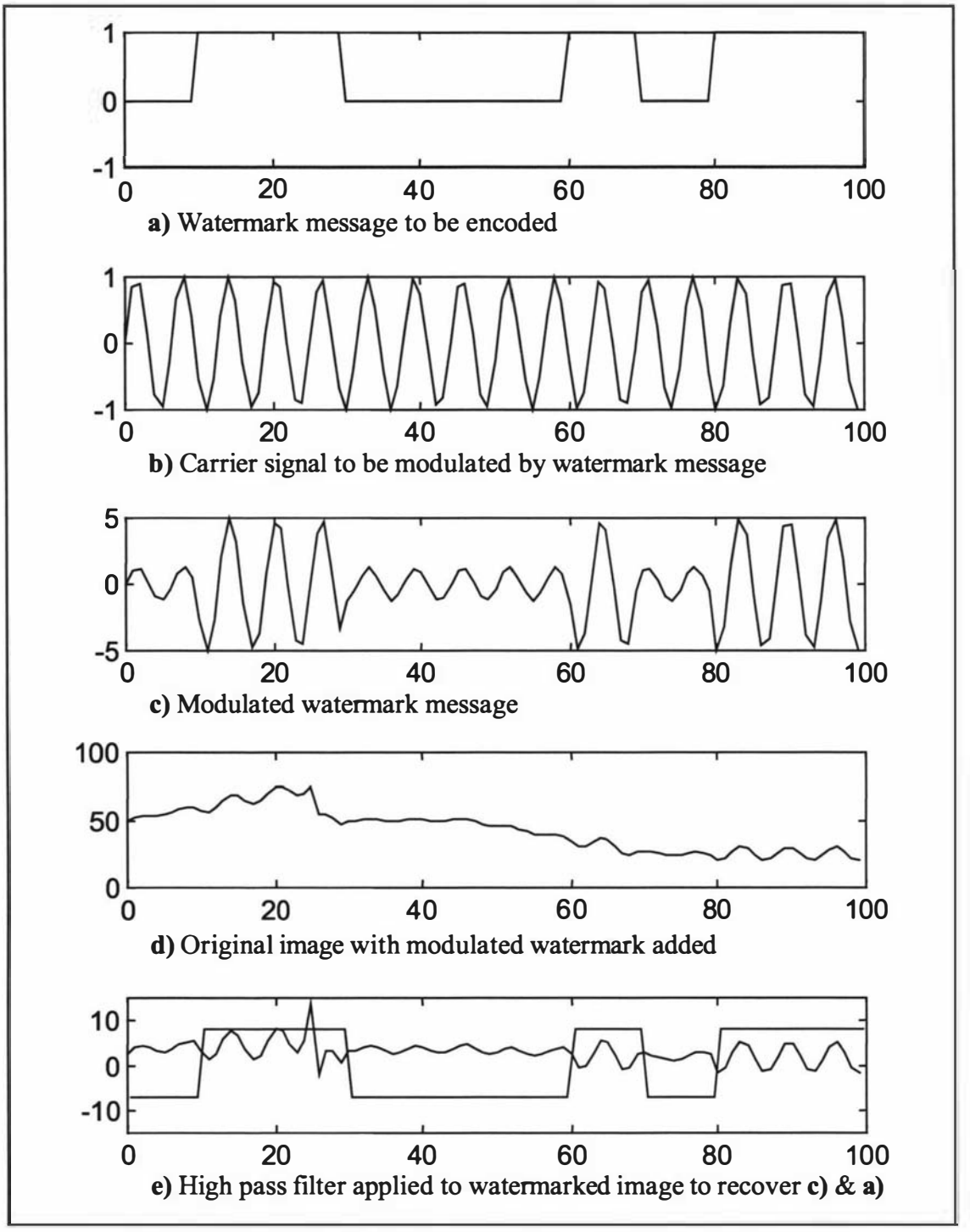


Figure 4.14 a)-e) – Self-referencing image watermark example

Figure 4.14 **a)** shows a 10-bit watermark message that is to be embedded into the same scanline using a self-referencing watermark technique. As the watermark must be recoverable from the image without access to the original image, a carrier signal, Figure 4.14 **b)**, must be used to convey the watermark information. The modulated carrier signal shown in Figure 4.14 **c)** is embedded into the image and causes some

distortion that is likely to be visible to the human observer. When the watermarked image in Figure 4.14 **d)** is passed through a high pass filter, the modulated carrier is recovered but is contaminated with some of the high frequency content present in the original image. Figure 4.14 **e)** shows the output from the filter and the overlaid watermark bitstream that is inferred from the amplitude of the signal.

Figure 4.15 shows the greyscale representations of the original clear scanline and the two watermarked scanline images. The original-referencing technique allows a greater number of watermark bits to be embedded and they can be embedded very discretely so that the human observer cannot tell whether Figure 4.15 **a)** or **b)** is the watermarked image. These advantages come at the cost that the original image must be available to recover the watermark information.

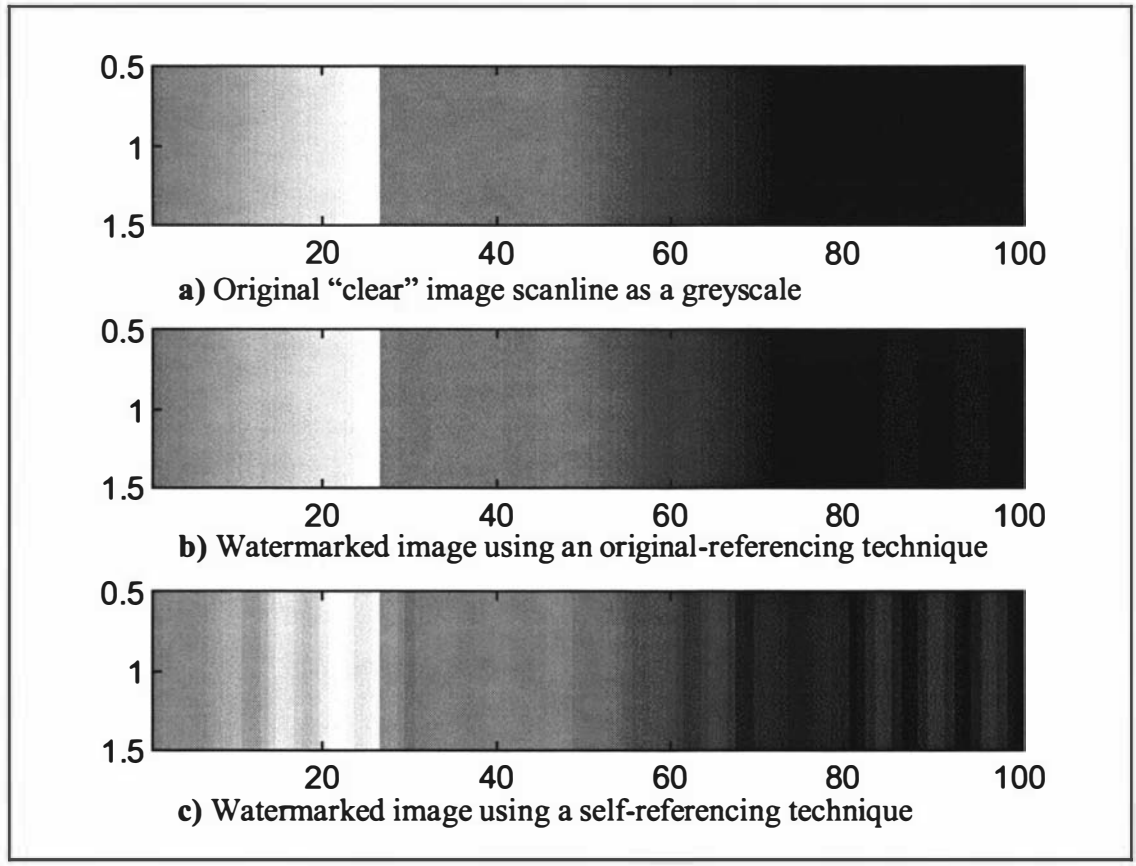


Figure 4.15 a)-c) – Original- and self-referencing image watermark examples

The self-referencing watermarked image shown in Figure 4.15 **c)** contains many visible artefacts from the watermark embedding process. Varying the amplitude of

the modulated carrier signal can control the visibility of these artefacts. If the amplitude is too small then there is a higher probability that the signal will be corrupted by the high frequency content in the image. If the amplitude is too large then the watermark will be easily recovered at the expense of visible distortions being introduced into the image.

4.2.2.2 Pixel based watermarking techniques

A simple way of encoding a watermark into an image is to alter the grey scale value of a pixel or block of pixels in an image in a predictable way. The magnitude of the alteration can be used to signal a hidden message. As long as the magnitude of the image pixel perturbations do not become excessive the human observer will not be able to detect the presence of the hidden watermark. There is always a compromise between the amount of watermark information that can be embedded in an image and the desired robustness of the communicated message. In an ideal communication medium, where the image is digitally transmitted without error, every pixel in the image could be used to transmit at least one bit of watermarking information. The least significant bits (LSBs) of the grey side pixel values could be used to encode the watermark information. If however the image was to be compressed using a lossy technique such as JPEG or a wavelet based algorithm, the LSBs of the image are typically scrambled by the compression process. This is because the LSBs of the pixel values are visually insignificant and are not encoded precisely by a lossy image compression algorithm. To allow the watermark to survive a lossy compression process the image perturbations have to be made more significant, yet must remain invisible to the human observer. This is generally achieved by spreading a single watermark bit over a block of pixel values to create a visually insignificant change that will still survive a lossy image compression process. The spreading of a single watermark bit over multiple pixels improves the robustness of the message whilst reducing the amount of data that can be embedded into the watermarked image.

4.2.2.3 Transform based watermarking techniques

A transform based watermarking technique makes use of perturbations made in the transform domain rather than the pixel domain. The perturbation of a transform coefficient is used to signal the presence and value of a watermark bit. This has the advantage that a model of the spatial frequency response of the human visual system can be taken into account when choosing which transformed coefficients should be perturbed. A single perturbation of a transformed coefficient generally results in a block of image pixels being altered. This result assists in making the technique much more robust to other sources of error such as image compression artefacts. Another advantage of transform based watermarking techniques is that the number of watermarking bits per image region can be varied according to local image content. In flat image regions the human eye is more sensitive to small changes thus minimal watermarking should occur in these regions. In heavily textured regions of an image several transform coefficients can be perturbed in order to encode multiple watermarking bits. Section 4.2.3 details a watermarking technique designed by the author for the wavelet domain. This technique is designed to make use of the wavelet decomposition that is used to perform lossy compression of the raw image and digital watermarking.

4.2.3 Watermarking in the wavelet domain

The image compression algorithm described in chapter 3 makes use of a wavelet transform of the image as the first step in the algorithm. The watermarking technique described in this section is based on perturbing the coefficients produced by the wavelet transform in order to embed a digital watermark into the image. The perturbations must be small compared to the amplitude of the wavelet coefficients or the effect of the perturbation will become visually significant. To produce a highly compressed image, the image compression algorithm quantises the coefficients and thus perturbs their value. One of the challenges in designing a co-operative watermark algorithm is devising a way to be able to distinguish quantisation noise from deliberate watermarking perturbations in the transmitted coefficient stream. To achieve maximum compression using a given technique, the quantisation noise should be set to cause slightly less than a "just noticeable difference" (JND) between the original image and the compressed image. This implies that any additional distortion caused by embedding a digital watermark is likely to become visible to the observer.

From an information theory point of view, watermarking and image compression are not readily compatible. Lossy image compression techniques remove much of the redundant data from an image so that only the visually important information is encoded. An invisible watermark added to an image effectively increases the information content of the image.

4.2.3.1 Proposed co-operative scheme

Watermarking algorithms that are designed to work independently of a lossy image compression algorithm are required to be robust in the presence of the quantisation noise produced by an image compression algorithm. The proposed watermarking scheme is designed to work in conjunction with the specific image compression algorithm described in chapter 3. The scheme is called an "Image Authentication Watermark" (IAW). This is because the scheme offers the ability to encode a hidden

watermark message into the image as well as providing a simple image tamper detection ability.

The IAW algorithm is designed to work only in a zero Bit Error Rate (BER) environment. The algorithm is designed to protect the integrity of the compressed image data at the application layer of the network. This means that any bit errors introduced in the communications network have already been detected and corrected and the compressed image data is ready for decompression and authentication.

As the image is compressed using the wavelet based image compression scheme, the image data is transformed into a series of domains as shown in Figure 4.16. Within any of these domains it is possible to embed watermark data that is then able to be detected as the image is decompressed.

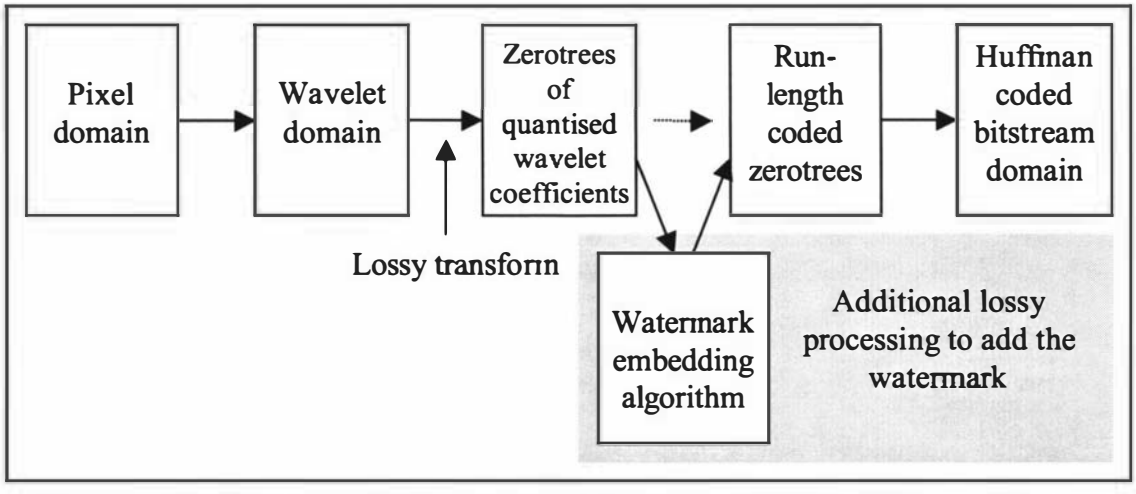


Figure 4.16 – Domains the image data is transformed into during compression

Within the image compression process, the only operation that visually distorts the appearance of the image is when the wavelet coefficients are quantised. The proposed watermarking scheme perturbs the already quantised wavelet coefficients to modulate the desired watermark data into the quantised coefficient stream. This process invariably causes additional distortion in the image especially when the desired image compression ratio is high.

The complete watermark message is embedded into the quantised wavelet coefficient zerotree structures with uniform density. A single watermark data bit is encoded into every group of four zerotrees. Each zerotree contains 85 quantised wavelet coefficients that are each generally small signed integers within the range of ± 16 when a high compression ratio is in use. The 85 coefficients in each zerotree are split into four frequency bands: 1 low frequency coefficient, 4 low-mid frequency coefficients, 16 mid-high frequency coefficients and 64 high frequency coefficients. In a typical image most of the mid-high and high frequency coefficients are quantised to zero as most of the energy in the image is packed into the low and low-mid frequency coefficients.

The method of modulation used to encode a single watermark data bit into a set of four zerotrees is based on the modulo-8 checksum that is calculated over the 340 coefficients. In an unwatermarked compressed image, the modulo-8 checksum of each set of four zerotrees will be a random number between '0' and '7'. By perturbing a small number of the 340 quantised coefficients, either up or down by a single quantisation step value, enables the modulo-8 checksum to be used to communicate the state of the encoded watermark data bit.

If the modulo-8 checksum after perturbation is a '0' then the watermark data bit is inferred to be a '0'. If the modulo-8 checksum after perturbation is a '1' then the watermark data bit is also inferred to be a '1'. But if the modulo-8 checksum is calculated to be a value between '2' and '7' then the watermark detection algorithm signals that this particular set of four zerotrees contains an invalid checksum. The inclusion of the six invalid checksum values, (out of the eight possible values obtainable), enables the checksum to detect data corruption within the zerotree structures. A potential cause of such data corruption could be attributed to the unauthorised alteration of the compressed image data. Thus a basic form of tamper detection is possible using this modulation scheme.

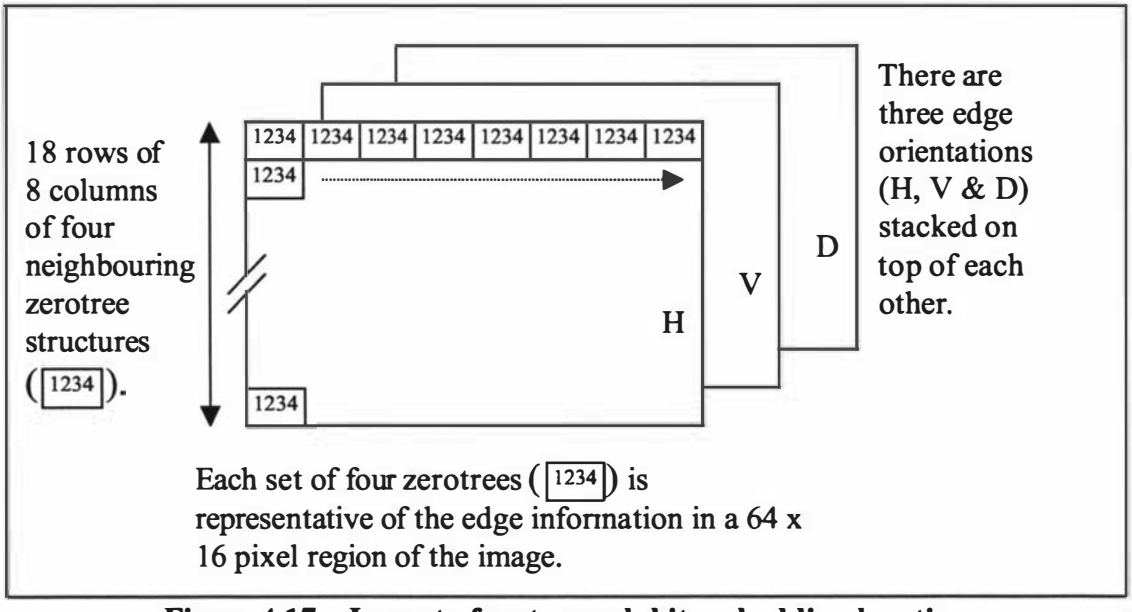


Figure 4.17 – Layout of watermark bit embedding locations

In a 512 x 288 pixel image there are 432 sets of four zerotrees that enable a moderate amount of data (54 characters) to be embedded throughout the image. Equation (4.3) can be used to calculate the watermark information capacity in bits for an arbitrary size image with dimensions divisible into 64 x 16 pixel blocks. Figure 4.17 shows the layout of the groups of four zerotrees for a 512 x 288 pixel image. The reason the four zerotrees are grouped in horizontal bands of four is related to the order that the zerotrees are grouped and pipelined through the DSP core. This process was discussed in section 3.4.2 of chapter 3 where it was shown that the DSP core can process 8 sequential zerotrees at a time in its internal core memory. The following paragraphs describe how the coefficients to be perturbed are chosen from the full set of 340 coefficients in each set of the four zerotrees.

$$capacity = 3\frac{x}{64}\cdot\frac{y}{16} \tag{4.3}$$

In order to correct a modulo-8 checksum to encode either a '0' or a '1' watermark data bit, at most four coefficients need to be incremented or decremented by a single quantisation step value. The first 21 coefficients from each of the four zerotrees are searched to locate the four coefficients with the largest magnitude. Then the perturbations described in Table 4.1 are made to these four coefficients in order to

correct the modulo-8 checksum. Figure 4.18 shows a typical set of four zerotrees where only the magnitudes of the first 21 coefficients are shown. In this example, the modulo-8 checksum of the full set of 340 coefficients is '4' and the watermark data bit to be encoded is a '1'. By using the line highlighted in italics in Table 4.1, it is necessary to subtract one from each of the first three largest coefficients to correct the modulo-8 checksum of the full set of 340 coefficients to a '1'.

Existing checksum	**Bit to be encoded**	**1st**	**2nd**	**3rd**	**4th**
		largest coefficient magnitude			
0	0				
1	0	Subtract 1			
2	0	Subtract 1	Subtract 1		
3	0	Subtract 1	Subtract 1	Subtract 1	
4	0	Add 1	Add 1	Add 1	Add 1
5	0	Add 1	Add 1	Add 1	
6	0	Add 1	Add 1		
7	0	Add 1			
0	1	Add 1			
1	1				
2	1	Subtract 1			
3	1	Subtract 1	Subtract 1		
4	*1*	*Subtract 1*	*Subtract 1*	*Subtract 1*	
5	1	Add 1	Add 1	Add 1	Add 1
6	1	Add 1	Add 1	Add 1	
7	1	Add 1	Add 1		

Table 4.1 – Perturbations to be made to correct modulo-8 checksum

The largest coefficients are chosen to be perturbed because the net percentage error introduced by altering the largest coefficients by a single quantisation step value is minimised. This in turn produces less visual distortion in the reconstructed image that contains the embedded image watermark. In uniform regions of the image, all the

quantised coefficients will be set to zero. In this case the first low frequency coefficient in each of the four zerotrees are used as candidates for perturbation. The next section will illustrate the performance of the co-operative scheme.

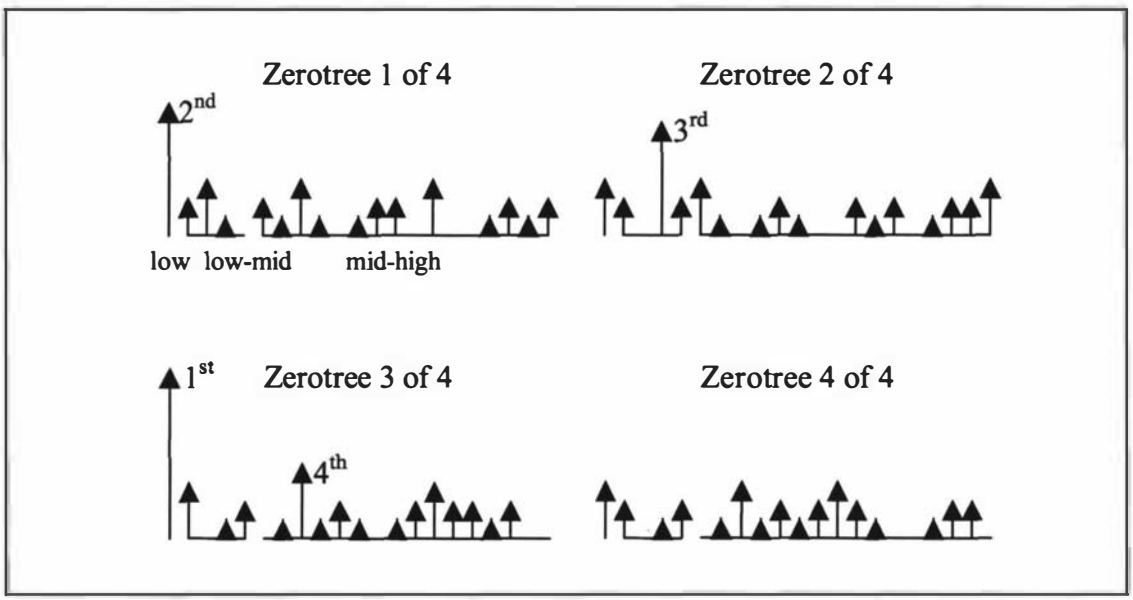


Figure 4.18 – Four largest coefficient magnitudes from a set of four zerotrees

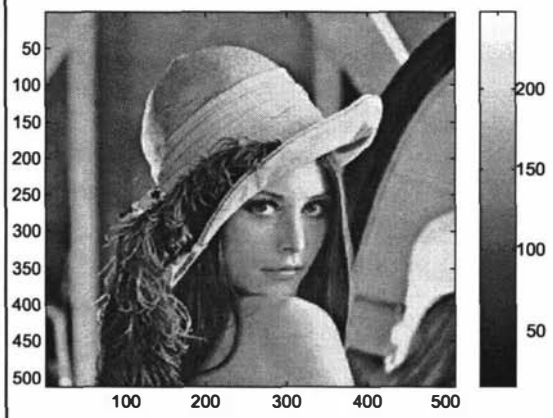
4.2.4 Performance of co-operative scheme

In this section the Lena image is used to illustrate the impact on visual quality that the co-operative watermarking scheme has. Both moderately compressed (14:1) and highly compressed (43:1) images are tested with the "Image Authentication Watermark" (IAW) scheme.

Figure 4.19 **a)** shows the Lena image compressed with the wavelet based compression algorithm at a ratio of 14:1. At this moderate compression ratio there is visually no loss in image quality even when the original image is viewed on screen. When 768 bits of watermark information are added into the 14:1 compressed image using the IAW scheme, there is no significant change to the image as shown in Figure 4.19 **b)**.

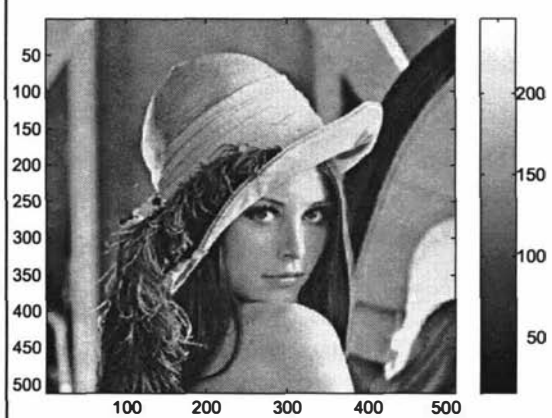
In order to see what the embedded watermark actually looks like, a scaled difference image between Figure 4.19 **a)** and Figure 4.19 **b)** is shown in Figure 4.19 **c)**. Note that the contrast has been enhanced in Figure 4.19 **c)** to highlight the differences that would otherwise be hard to see in Figure 4.19 **b)**. The watermark contains sharp features around the sharp edges in the image and smooth changes in the uniform image regions. The watermark is effectively invisible to the human observer and yet can be completely recoverable when the appropriate detection algorithm is applied to the compressed image. The RMSE introduced into the compressed image by the process of image compression ($RMSEc_{low}$) at a "low" ratio of 14:1 is 4.72. The RMSE created by the embedding of the IAW ($RMSEw_{low}$) into the already compressed image is 1.25. As the error caused by the compression process is effectively invisible to the human eye, it is not surprising that the smaller error introduced by the IAW is also not visible.

The compression ratio obtained and the measured RMSE relate specifically to the Lena image. The results of two other standard test images are considered later on in this section. When the image is compressed with a low compression ratio, the IAW adds very little error to the image.

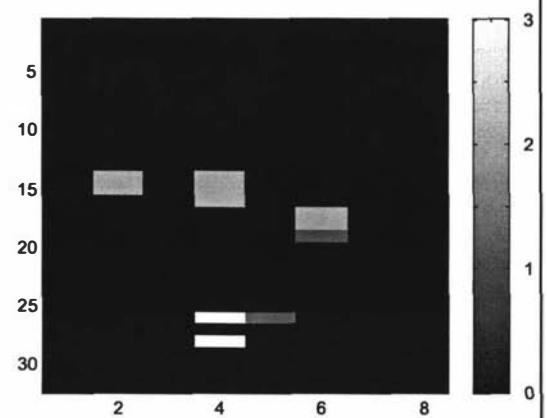


a) 14:1 Wavelet compressed Lena image without watermark

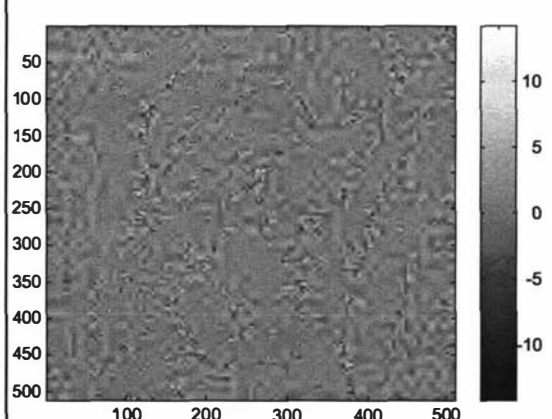
d) Tampered compressed image with original watermark



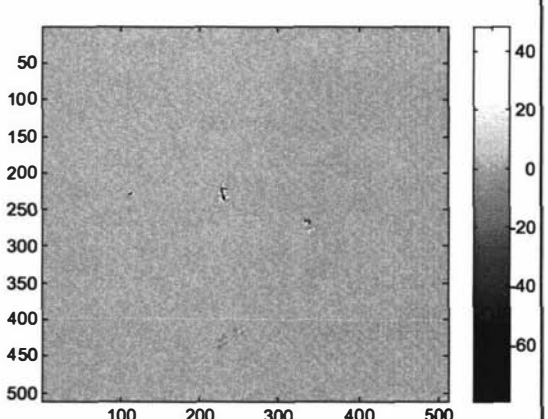
b) Image after watermark has been added



e) Output from watermark detection algorithm showing where tampers have been detected



c) Scaled difference between **a)** and **b)** revealing the actual watermark



f) Scaled difference between **b)** and **d)** revealing the actual tampering that occurred to validate **e)**

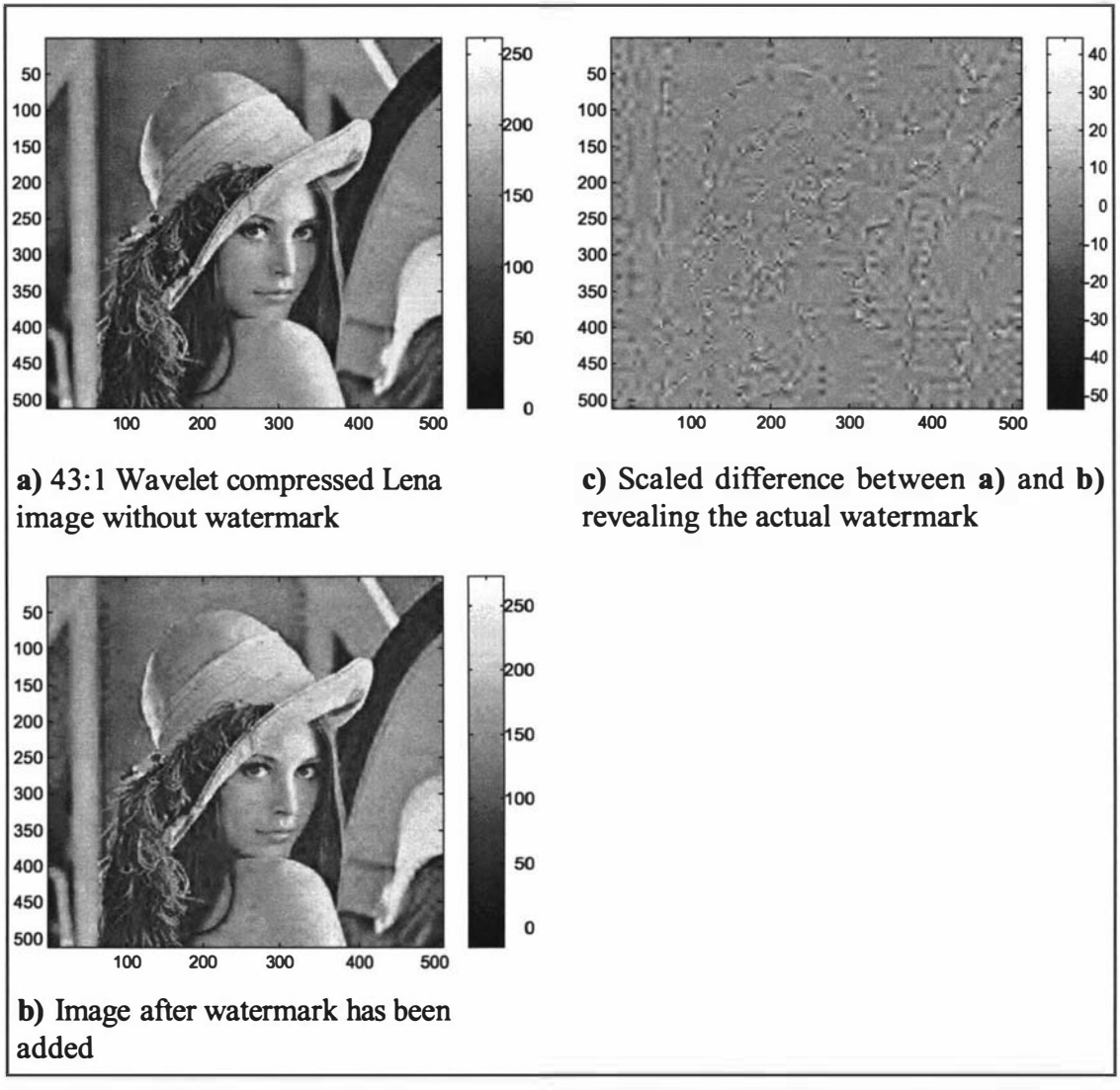
Figure 4.19 a)-f) – Watermark and tamper detection with 14:1 compressed image

To demonstrate the tamper detection properties of the IAW, Figure 4.19 **d)** shows the Lena image after it has been tampered with. The tampers are purposefully minor to illustrate the sensitivity of the tamper detection. The following procedure lists the steps required to produce the tampered image. The list illustrates the amount of proprietary knowledge required to tamper a compressed image and still not defeat the tamper detection scheme:

1. The compressed image file is decompressed to a bitmap that is then exported to an image editing program such as Paint Shop Pro. *This process requires the ability to decompress an image using the proprietary image decompression algorithm.*
2. The image is edited as a bitmap, this process will unwittingly disturb the local frequency distribution in the edited image region which will alter the wavelet decomposition of the image. The alteration of the wavelet decomposition is likely to disrupt the modulo-8 checksums of the wavelet zerotree structures when the image is re-compressed.
3. The altered bitmap is then re-compressed so that it can be inserted back into the compressed image archive. *This process requires the ability to compress an image using the proprietary image compression algorithm.* It is assumed at this point that the tamper detection scheme based on modulo-8 checksums is not known to the attacker.
4. The tampered compressed image is retrieved from storage and decompressed. The decompressed image is shown in Figure 4.19 **d)**.

When the watermark detection algorithm is used to authenticate the tampered image in Figure 4.19 **d)**, a tamper detection image is produced in Figure 4.19 **e)**. This image shows the location and number of watermark bits that show evidence of tampering in each region. A tampered region is detected when the modulo-8 checksum does not equal 0 or 1. Each rectangular region shown in Figure 4.19 **e)** corresponds to a 64 x 16 pixel block in the original image. Each region contains three watermark bits, one for each of the Horizontal, Vertical and Diagonal detail coefficient zerotree sets. A single rectangular region can record up to three tampered bits and this information is represented by the grey level in Figure 4.19 **e)**. The image in Figure 4.19 **f)** reveals

the locations where the actual tampers to the image occurred. This is a scaled difference image as the actual tampers are minor but occur in four separate areas of the image.



a) 43:1 Wavelet compressed Lena image without watermark

c) Scaled difference between **a)** and **b)** revealing the actual watermark

b) Image after watermark has been added

Figure 4.20 a)-c) – Increased watermark visibility with 43:1 compressed image

Figure 4.20 **a)** shows the Lena image compressed with the wavelet based compression algorithm at a ratio of 43:1. At this high compression ratio there is some loss of high frequency detail in the image. When 768 bits of watermark information are added into the 43:1 compressed image using the IAW scheme, the smooth areas of the image now show the presence of the watermark signal clearly. This can be seen in Figure 4.20 **c)**, a contrast enhanced difference image between Figure 4.20 **a)** and **b)**. The main reason why the watermark signal is visible in the highly compressed image is that the amplitude of the watermark signal has increased from ± 15 to ± 50. This

increase in amplitude is directly proportional to the change in quantiser step size, which has resulted in the higher compression ratio. The RMSE introduced into the compressed image by the process of image compression ($RMSEc_{high}$) at a ratio of 43:1 is 6.96 (was 4.72 when the image was compressed at a ratio of 14:1). The RMSE created by the embedding of the IAW into the already compressed image ($RMSEw_{high}$) is 4.42 (was 1.25 when the image was compressed at a ratio of 14:1). The error due to compression at 43:1 is 50% greater than the error at 14:1 and this error is more visible to the human eye. The additional error introduced by the IAW creates a total RMSE of 8.16 which is almost twice the total error at a compression ratio of 14:1.

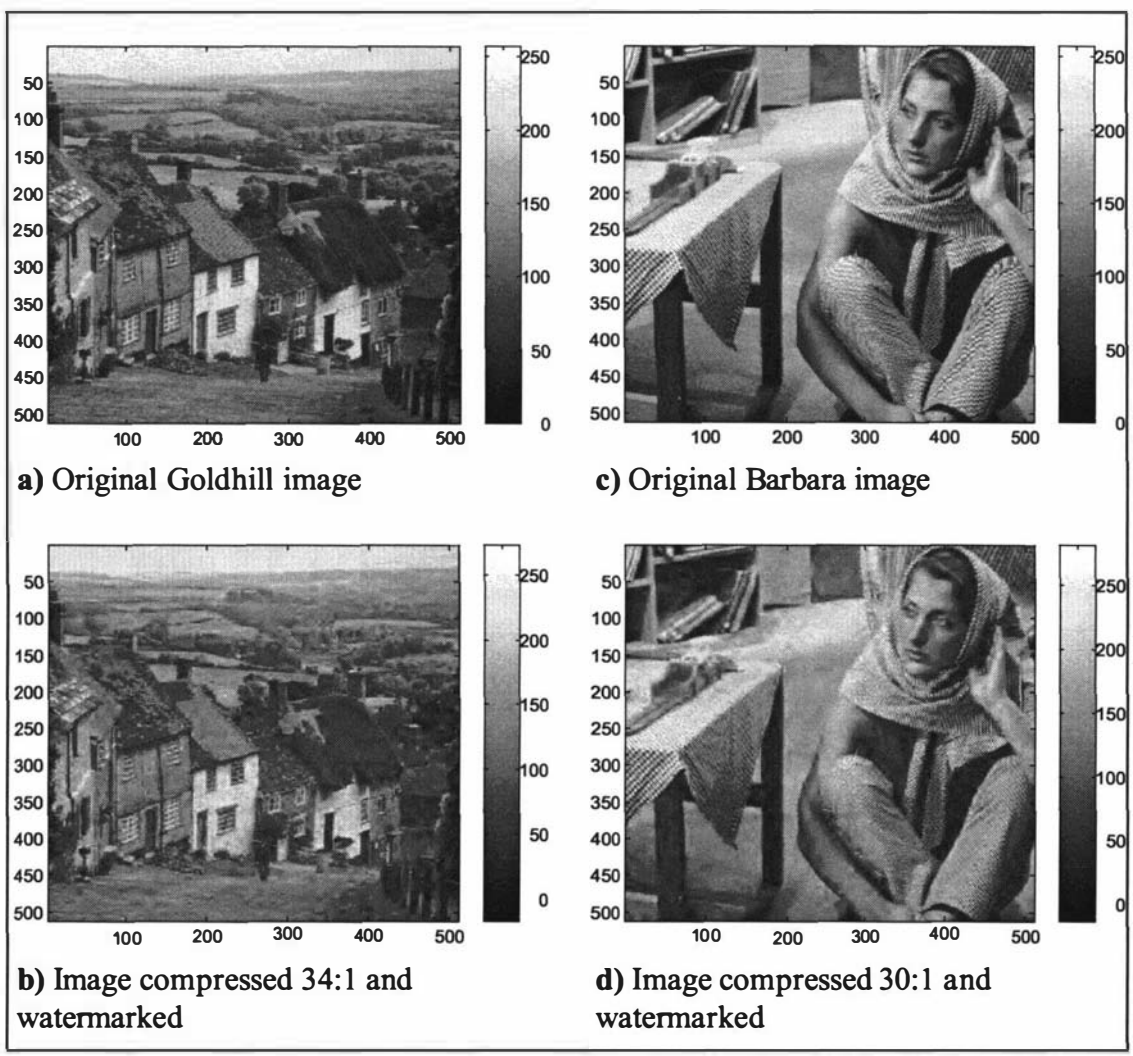
Table 4.2 and Table 4.3 compare the ratio of the RMSE caused by the compression process (RMSEc) over the RMSE caused by the IAW process (RMSEw). Three standard images using two sets of quantiser levels (corresponding to "low" (CR_{low}) and "high" (CR_{high}) compression) are considered. The total RMSE error caused by both processes is listed in the RMSEw+c column. The low compression images all have IAW errors that are many times smaller than the errors induced by the low compression process (3.8 to 10 times smaller), which is also visually insignificant. The highly compressed images all have IAW errors that are of a similar magnitude to the error induced by the high compression process (only 1.6 to 3.0 times smaller), which is already quite visible. The achievable compression with the "low" and "high" quantiser settings varies over the different images. The "Barbara" and "Goldhill" images shown in Figure 4.21 **a)-d)** have larger areas of complex texture that require more non-zero coefficients to encode the texture. Figure 4.21 **a)** and **c)** show the original images, and Figure 4.21 **b)** and **d)** show the images after the high compression quantiser set is used to compress and watermark the images.

Image	CR_{low}	$RMSEc_{low}$	$RMSEw_{low}$	$RMSEw+c_{low}$	$\frac{RMSEc_{low}}{RMSEw_{low}}$
Lena	14:1	4.72	1.25	4.88	3.8
Goldhill	8.7:1	5.85	1.25	5.98	4.7
Barbara	9.8:1	12.9	1.24	13.0	10

Table 4.2 – RMSE errors induced by low compression and IAW processes

Image	CR_{high}	$RMSEc_{high}$	$RMSEw_{high}$	$RMSEw+c_{high}$	$\frac{RMSEc_{high}}{RMSEw_{high}}$
Lena	43:1	6.96	4.42	8.16	1.6
Goldhill	34:1	9.01	4.65	10.1	1.9
Barbara	30:1	14.4	4.77	15.2	3.0

Table 4.3 – RMSE errors induced by high compression and IAW processes



a) Original Goldhill image

c) Original Barbara image

b) Image compressed 34:1 and watermarked

d) Image compressed 30:1 and watermarked

Figure 4.21 a)-d) – Original and high compression Goldhill and Barbara images

A more thorough analysis of the effect on image quality when varying the quantiser step size is given in section 5.2 of chapter 5 for the three images considered briefly in this section. Section 5.3 of chapter 5 also analyses the performance of the image watermark visibility over a wide range of compression ratios for the Lena image.

4.2.5 Efficient DSP implementation of the image watermark

There are several factors that allow the implementation of the IAW on the DSP to be efficient. The main factor is that the watermark is embedded directly into the quantised wavelet coefficients *as* the image is being compressed. Secondly the actual process of embedding the watermark data is computationally simple. The most computationally intensive process involves searching for the largest four coefficients from a small set of 84 candidates.

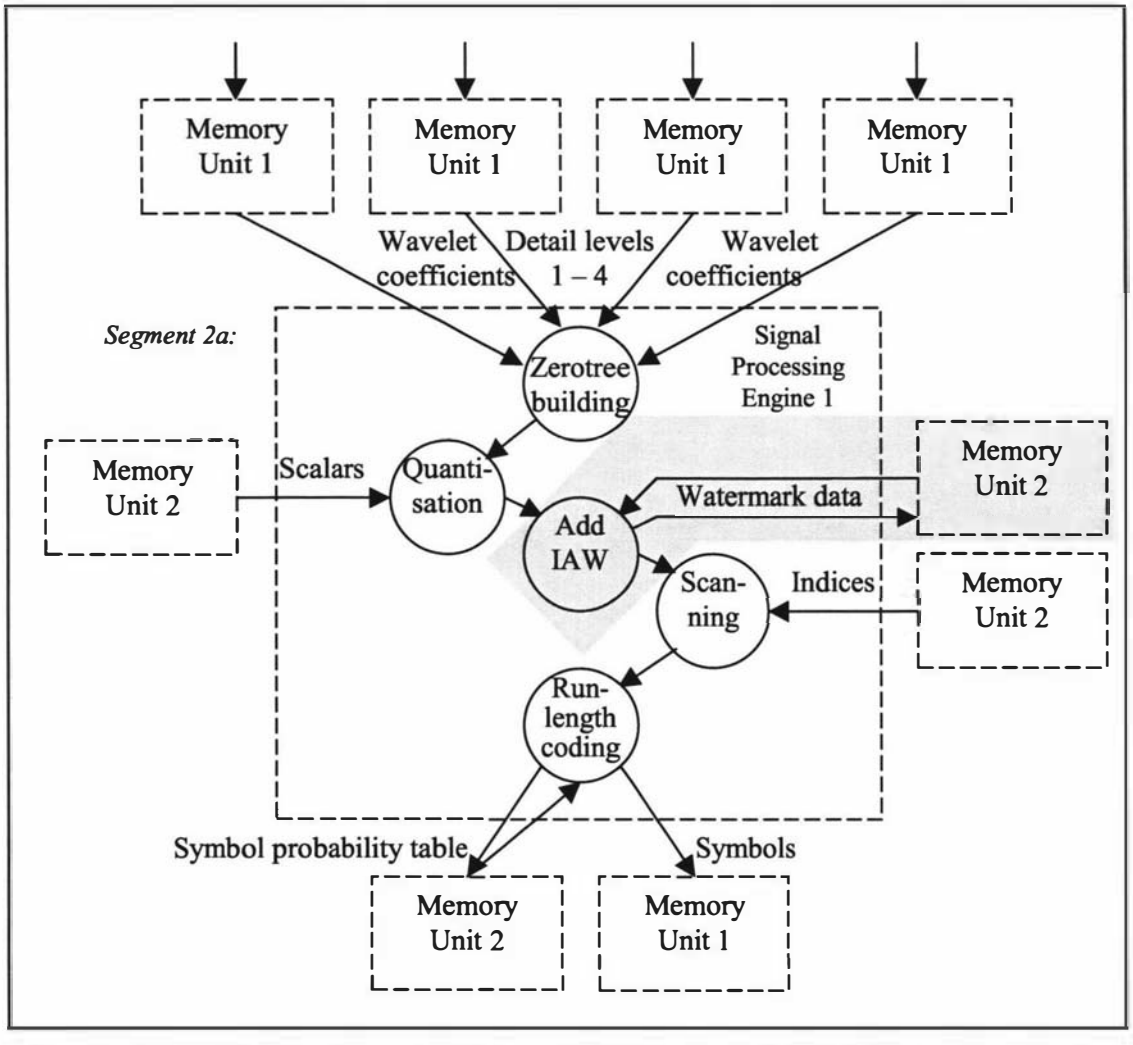


Figure 4.22 – Signal processing flow graph showing where IAW is added during image coding

Figure 4.22 is a modified version of the signal processing flow graph introduced in chapter 3 as Figure 3.5. The highlighted "Add IAW" process is placed between the

"Quantisation" and "Scanning" tasks. The basic working block of input data for the "Add IAW" process of the compression algorithm is four zerotrees (or 340 coefficients). Figure 4.22 shows an additional stream of data labelled "Watermark data" which is consumed at a rate of one bit per set of four zerotrees processed. The watermark data consists of the camera serial number and date-stamp, represented as a 96-bit code repeated throughout the image.

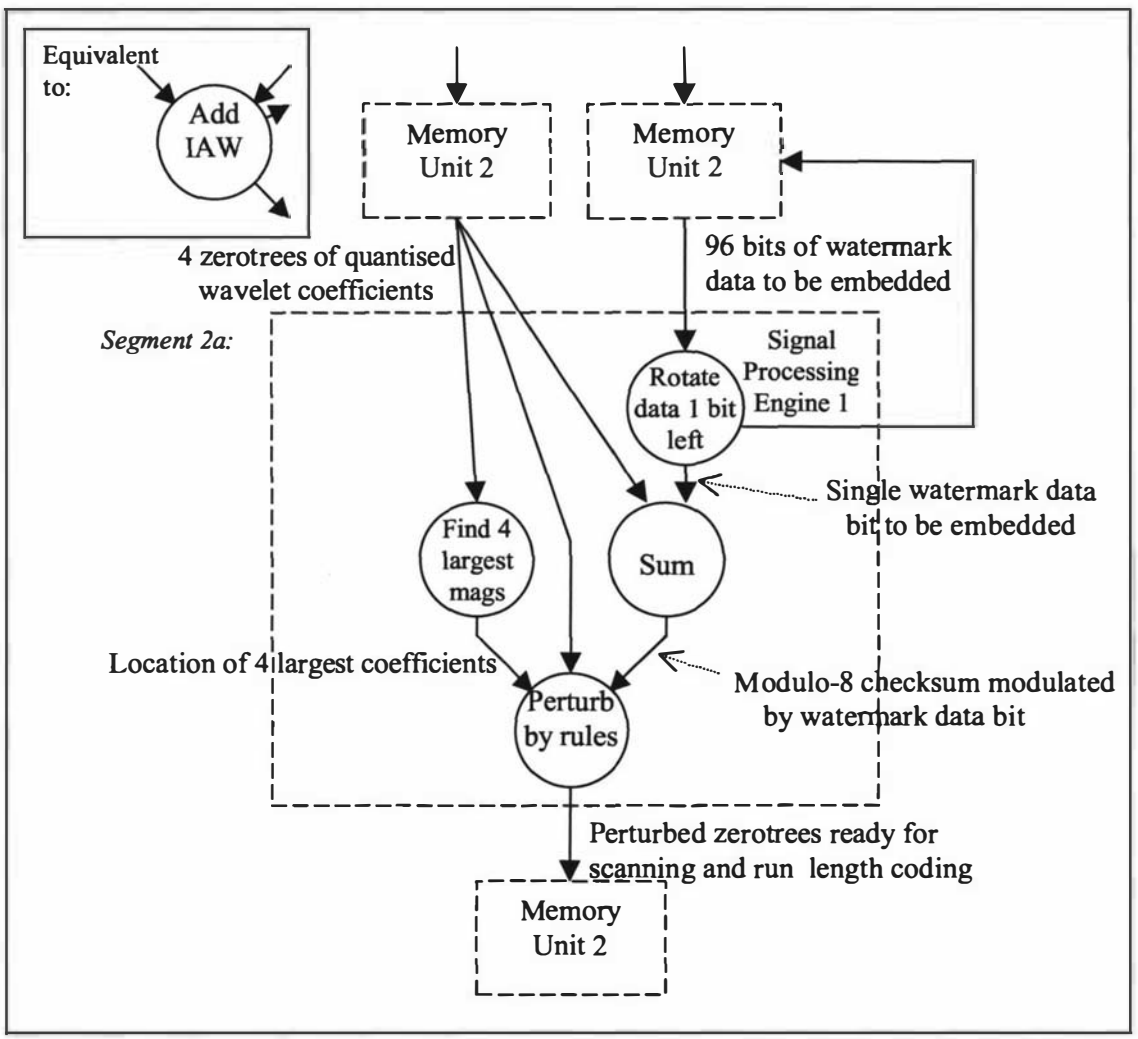


Figure 4.23 – Signal processing flow graph showing processing required to add IAW

The "Add IAW" process is further detailed in Figure 4.23. Four processes are implemented in the DSP to embed the watermark data, these are listed below:

1. A rotation of the 96-bit watermark to extract the left most bit.

2. The modulo-8 summation of the 340 coefficients and the watermark bit. This results in an integer between –4 and +3 which represents how many coefficients need to be perturbed and the direction of the perturbation.
3. An insertion sort to find the four greatest coefficient magnitudes and their locations in a single pass through the first 21 coefficients of the four zerotrees.
4. The perturbation of up to four coefficients to correct the modulo-8 checksum.

Each of these four processes are examined in closer detail with DSP assembly language excerpts included to illustrate the efficiencies gained by effective use of the DSP instruction set. This directly relates to one of the main themes of this thesis.

4.2.5.1 Watermark bit rotation

The watermark data is stored in the on-chip SRAM in four 24-bit word locations. The DSP rotate left opcode "`rol`" is used to rotate each location, 24 bits at a time, through the carry flag so that the full 96-bit watermark can be rotated efficiently to generate the next watermark bit to be encoded. Figure 4.24 shows the DSP assembly language instructions used to extract the bit to be encoded and to rotate the watermark data. A no-operation "`nop`" instruction is included after the rotate instruction to mark the pipeline stall. This is because the results of the "`rol`" instruction are written back to the "`B`" accumulator in the next instruction cycle and the upper part of the "`B`" accumulator ("`B1`") cannot be written back to memory until the next instruction cycle. In summary, 4 instruction cycles are required for every 24 bits rotated and this includes the reading and writing operations to internal SRAM.

```
;Rotate the 96 bit watermark left by one bit
   move   #IAW4,R0      ;Point R0 to right most word of watermark
   btst   #23,Y:IAW1    ;Get left most bit of 96 bit watermark into Carry Flag
   move   Y:(R0),B      ;Get First 24 bits of watermark
   rol    B             ;Rotate B1 left through the Carry Flag
   nop                  ;Wait for B accumulator to be written to
   move   B1,Y:(R0)-    ;Put First 24 bits of watermark, decrement R0
   move   Y:(R0),B      ;Get Second 24 bits of watermark
   rol    B             ;Rotate B1 left through the Carry Flag
   nop                  ;Wait for B accumulator to be written to
   move   B1,Y:(R0)-    ;Put Second 24 bits of watermark, decrement R0
(last four lines repeated two more times)
```

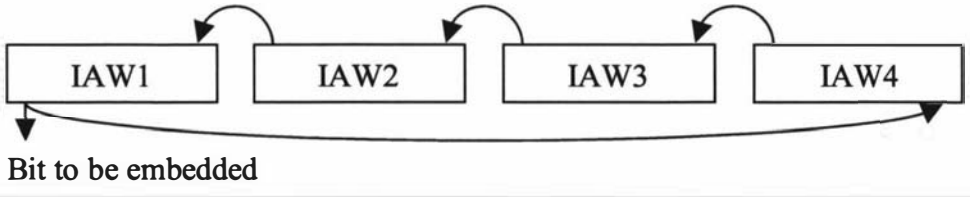


Figure 4.24 – Rotation of 96-bit watermark data

4.2.5.2 Modulo-8 checksum

The next process involves calculating the modulo-8 summation of the 340 quantised wavelet coefficients and the single watermark data bit to be embedded. The DSP can sum a block of memory rapidly with only two cycles per element required. Once the "A" accumulator contains the summation, the sum is converted to a modulo-8 value which is then offset to an integer between –4 and +3. Figure 4.25 shows the combination of logic and conditional branching instructions that are used to produce this result.

```
;Sum up the first 4 trees, calculate number of coefficients to perturb
   clr    A
   do     #85*4,SUMIAW
   move   X:(R3)+,X0    ;Two clock cycles per element in loop
   add    X0,A
SUMIAW
   add    #>4,A         ;Always add 4 to checksum
   btst   #0,Y:IAW4     ;Get right most bit of watermark into the Carry Flag
   bcc    NOADDBIT
   add    #>1,A         ;Conditionally add 1 if watermark bit is a 1
NOADDBIT
   and    #7,A          ;Take Mod base-8
   move   #0,A2         ;Clear sign extension to ensure range of [0..7]
   sub    #>4,A         ;Subtract 4 to get a difference in the range [-4..+3]
```

Figure 4.25 – Calculating modulo-8 checksum of four zerotrees and the watermark bit

4.2.5.3 Insertion sort

The most complicated algorithm of the IAW procedure, is that for the location of the four coefficients with the greatest magnitude. As there are four sets of 21 coefficients to search and there are 64 high frequency coefficients between each set, two nested loops are used to traverse the 84 coefficients in a single pass. The "R4" register points to the current coefficient being considered and the "N4" register is set to 64. This enables the high frequency coefficients to be skipped as each zerotree is processed. The "B" accumulator contains the magnitude of the current coefficient. A (truncated) insertion sort algorithm is used [4.12], each of the 84 coefficients is inserted sequentially into a sorted list that stores the four coefficients having greatest magnitude. The insertion sort terminates early after the four largest coefficients are found. The ranked list of the four coefficients is maintained in the DSP core registers to minimise the need to move results to and from SRAM. Table 4.4 shows the allocation of DSP pointer and value registers to the four highest ranked coefficients considered so far. Figure 4.26 contains the assembly language code developed to efficiently implement the truncated insertion sort in a *single pass* through the set of coefficients. Other sorting approaches that could have been implemented such as bubble-sort would require up to 84 passes through the data in order to find the four largest coefficients.

Magnitude	Pointer to location	Value of coefficient
Fourth greatest value	R0	X0
Third greatest value	R1	X1
Second greatest value	R2	Y0
Greatest value	R3	Y1

Table 4.4 – DSP register usage for efficient insertion sort

```
;Find the four largest absolute elements in the four trees' first 21
; elements
   move   #64,N4
   do     #4,IAWOUTR
   do     #21,IAWINR
   move   X:(R4),B      ;Get next value to be insertion sorted
   abs    B             ;Compare the magnitude of the coefficient
   cmp    X0,B
   ble    NOINSRTW      ;If new value is less than or equal to fourth greatest
                        ; value then no insertion needed
   cmp    X1,B          ;New value is greater than fourth greatest value
   bgt    INSRT1
   move   B,X0          ;New value is less than or equal to third greatest
                        ; value: overwrite least
   move   R4,R0         ;OLD: (3) 2 4 6 8   NEW: (3) 3 4 6 8
   bra    NOINSRTW      ;Finished sort
INSRT1                  ;OLD: (5) 2 4 6 8   NEW: (5) 4 5 6 8
   move   X1,X0         ;New value is greater than third greatest value
   move   B,X1
   move   R1,R0
   move   R4,R1
   cmp    Y0,B          ;Test with second greatest value
   ble    NOINSRTW      ;New value is less than or equal to second greatest
                        ; value: no more insertion
INSRT2                  ;OLD: (7) 4 7 6 8   NEW: (7) 4 6 7 8
   move   Y0,X1         ;New value is greater than second greatest value
   move   B,Y0
   move   R2,R1
   move   R4,R2
   cmp    Y1,B          ;Test with the greatest value
   ble    NOINSRTW      ;New value is less than or equal to the greatest
                        ; value: no more insertion
INSRT3                  ;OLD: (9) 4 6 9 8   NEW: (9) 4 6 8 9
   move   Y1,Y0         ;New value is greater than the greatest value
   move   B,Y1
   move   R3,R2
   move   R4,R3

NOINSRTW
   move   (R4)+         ;Process next coefficient

IAWINR                  ;End of inner loop
   move   (R4)+N4       ;Skip over the 64 high frequency coefficients
IAWOUTR                 ;End of outer loop
```

Figure 4.26 – Insertion sort to find location of four largest coefficient magnitudes

4.2.5.4 Coefficient perturbation

The final process is to apply the appropriate number of coefficient perturbations to correct the modulo-8 checksum across the four zerotrees. Figure 4.27 contains the assembly language code to perform this task. The "X0" register is used to store the perturbation to be made to each of the coefficients (either –1 or +1). The "A1" portion of the accumulator contains the number of coefficients to be perturbed between 0 and 4. The four address registers from "R3" down to R0" contain the addresses of the four largest coefficients available for perturbation. After each

perturbation, the next coefficients to be perturbed are moved up the list so that the "R3" register always points to the address of the next largest coefficient.

At this point the single IAW data bit has been embedded into the four zerotrees and the remainder of the image compression algorithm proceeds without any further modification.

```
;Perform the perturbation
   move   X1,A          ;Recover direction & number of perturbations [-4..+3]
   move   #>1,X0
   tst    A
   bge    IAWPOS
   move   #>-1,X0       ;X0 contains direction of perturbations
IAWPOS
   abs    A
   nop
   do     A1,IAWPERT    ;While more perturbations are required, perturb the
   move   X:(R3),A      ;greatest coefficient in the list and then bump the
   sub    X0,A          ;smaller coefficients up on the list.
   nop
   move   A,X:(R3)
   move   R2,R3
   move   R1,R2
   move   R0,R1
   nop
IAWPERT
```

Figure 4.27 – Application of the perturbation rule to correct the modulo-8 checksum

4.3 Discussion

In this chapter both motion detection and image watermarking routines have been added to the image compression algorithm described in chapter 3. The computational complexity of these two algorithms has been kept low by making extensive use of the data structures already calculated during the image compression process. The calculation of the wavelet decomposition is computationally expensive, but has the following advantage. The resulting wavelet decomposition enables these simple routines to achieve good results with minimal extra computation.

The motion detection algorithm described in this chapter can be used to evaluate the interframe motion in separate or overlapping rectangular regions within the frame. The calculations are performed on the four-time low-pass filtered and downsampled image pixel data. This process virtually eliminates all high frequency noise prior to the frame difference being calculated. The Root Mean Square Error metric is then used to evaluate the energy in each region of interest. In the two frames per second motion sequence shown in this chapter, reliable detection of significant motion has been demonstrated in arbitrarily chosen regions of interest.

It has been shown that watermarking techniques can be made to work co-operatively with a wavelet based image compression algorithm. It is however important to investigate the limiting factors when developing a new algorithm. The visibility of the watermark in uniform image regions has been shown to be related to the amount of compression applied to the original image. The optimum amount of compression to apply to an image which is to be watermarked can be chosen so that the watermark is just noticeable when compared with the unwatermarked image. The relationship between watermark visibility and quantiser setting is studied more thoroughly in section 5.3 of chapter 5.

The design of the IAW algorithm relies on a uniform spread of watermark bits throughout the compressed image. This has the consequence that watermark bits that are hidden in uniform regions of a highly compressed image become visible to the

human eye as artefacts. A better approach, to be recommended as future work, is to use an adaptive spread of watermark bits throughout the image. The scheme currently developed embeds a uniform number of bits per region across the image. An adaptive algorithm could better use the information in the wavelet decomposition to estimate the "smoothness" of the region and then determine the number of bits to embed. The smooth regions of an image are likely to contain little detail and are therefore unlikely to be candidates for tampering.

The tamper detection property of the IAW is sensitive to small tampers, which is a good result. The computational simplicity of the scheme has the advantage that it is practical to implement in real world systems that produce digital images.

4.4 References

[4.1] **Brown, C. W.,** *Graphics File Formats: reference and guide,* 1995. Manning Publications.

[4.2] **Chiariglione, L.,** *"The development of an integrated audiovisual coding standard: MPEG,"* Proceedings of the IEEE, vol. 83, February, pp.151-157, 1995.

[4.3] **Lin, D. L.,** *"Video on phone lines: technology and applications,"* Proceedings of the IEEE, vol. 83, February, pp.175-193, 1995.

[4.4] **Jain, A. K.,** *Fundamentals of Digital Image Processing,* 1989. Prentice-Hall.

[4.5] **Jähne, Bernd,** *Digital Image Processing: concepts, algorithms and scientific applications,* 1991. Springer-Verlag Berlin. Heidleberg.

[4.6] **Alexander Turnbull Library,** *Sample image from the National Library of New Zealand* [Web Page]. 1998. Available at: http://timeframes.natlib.govt.nz/.

[4.7] **Johnson, Neil F,** *Exploring Steganography: seeing the unseen*; IEEE Computer, pp.26-34, February 1998.

[4.8] **Zeng, Wenjun,** *"A Statistical Watermark Detection Technique Without Using Original Images for Resolving Rightful Ownerships of Digital Images,"* IEEE Transactions on Image Processing, vol. 8, no. 11, pp.1534-1548, 1999.

[4.9] **Chudy, P.,** *Handcuff digital thieves*; Byte, pp.5-8, April 1996.

[4.10] **Prestage, Michael,** *Tell-tale pictures can be candid about cameras*; The Times, Interface: p.3, January 15, 1997.

[4.11] **Boland, F. M.,** *"Watermarking digital images for copyright protection,"* Image Processing and its Applications, pp.326-330, 1995.

[4.12] **Press, W. H.,** *Numerical recipes in C : the art of scientific computing,* Ed. 2, pp. 329-346, 1992.

5

The Complete Remote Activity Monitoring System and its Performance

In this penultimate chapter the performance of the complete remote activity monitoring system is discussed. The "system" consists of one or more smart digital cameras located in the client's premises, a "Server" PC that communicates via Ethernet to the cameras and stores image sequences, and one or more "Workstation" PCs that can be used to view stored or live image sequences.

The previous chapters have focussed on specific components of a system that have been designed to fulfil the requirements of remote activity monitoring. This chapter describes the development and performance of the smart digital camera and its associated PC based "front-end" control software. The identification of bottlenecks in the hardware and software components is discussed as well as the inherent latency due to pipelined processing. The performance of the image compression algorithm is then compared with algorithms published at around the same time as this algorithm was developed in 1996.

Image watermarking is an important requirement of a digital image based monitoring system. The watermark is used to authenticate the image as being free from tampering as well as establishing the identity of the smart digital camera used and the date and time of capture of the image. The image degradation due to watermarking at

various compression ratios is tabulated and graphed to describe the performance of the embedded watermarking algorithm.

5.1 System evolution

At the middle of the year 2000, the smart digital camera is being sold commercially as an integrated part of a well-established building access control and alarm management solution. The initial feasibility studies and development of prototypes started in the middle of 1996. State-of-the-art digital hardware was chosen at that time knowing that the cost of such hardware would be expected to be significantly cheaper at the time the product would be manufactured and sold in commercial quantities. The following sub-sections describe the initial prototype system on which most of the research was conducted and then the final design that went into the production system. The prototype and production system descriptions include discussions about the hardware and software architecture followed by a discussion about the processing bottlenecks and latency in the architecture.

5.1.1 Prototype system

Initially, evaluation boards from Motorola were used to test the performance of signal processing algorithm kernels on the chosen 66 MHz Motorola 56303 DSP core. Algorithms such as the 8 x 8 DCT for a JPEG style algorithm and the QMF algorithm for a wavelet based algorithm were tested. Then the use of DMA and double buffering techniques were investigated as means to minimise the amount of data marshalling that the DSP core needed to perform. A more complete image compression algorithm was developed and tested with small images manually downloaded to the evaluation hardware. A specialist in electronic design from the industry partner designed and built a multi-layer circuit board as a prototype of the smart digital camera. This prototype was used to determine the feasibility of the chosen hardware to produce a real-time image compression system with full-scale images. This prototype used a faster processor and increased the amount of SRAM

available. The new board included a CCD sensor and control chip giving the capability of capturing a 512 x 288 pixel digital image in the SRAM of the DSP.

The next two sections specify the hardware and software that were used to assess the performance of the image processing algorithms developed by the author. The last section provides performance measurements of the complete prototype system. These results were used to identify bottlenecks in the system and provide a measure of latency between image capture, transmission and display.

5.1.1.1 Hardware architecture

Figure 5.1 shows a block diagram of the smart digital camera internal hardware layout including the two communication interfaces to the development system and the camera user interface developed for use on the PC.

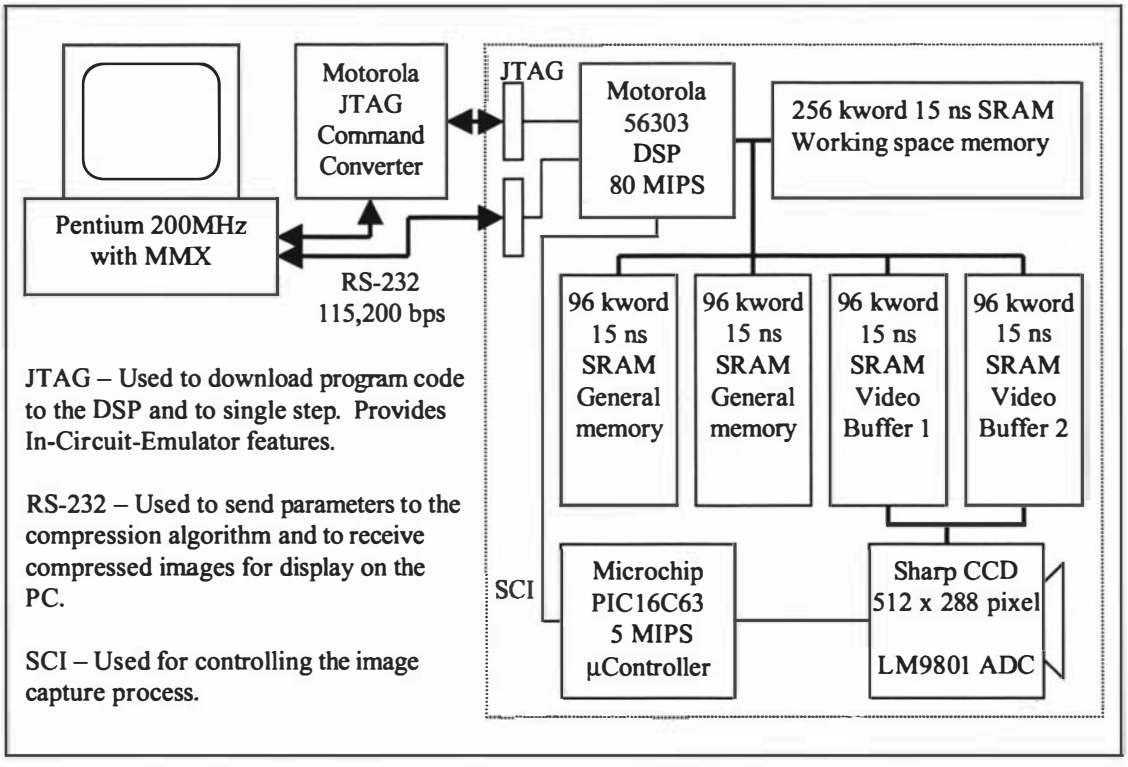


Figure 5.1 – Block diagram of prototype hardware for smart digital camera

The custom-built evaluation board was populated with 1.875 MBytes of fast (15 ns) SRAM to enable a full sized digital image to be stored and processed with fast access to the external memory. 144 kBytes is sufficient to contain the raw image of 512 x 288 pixels, but the actual working space needed for the final truncated wavelet decomposition at 24-bit precision (as described in chapter 3) is 175 kwords or 324 kBytes. The additional 1,452 kBytes of memory above these requirements was used to trial different algorithms as well as to buffer sequences of compressed images for transfer to a PC. Unfortunately the length and layout of the address and data buses on the board was such that several additional wait-states were required to *reliably* use the fast SRAM. For this reason the algorithms running on the DSP ran more slowly whenever the external memory was directly accessed. This circuit board layout deficiency was largely overcome through the use of DMA to access data from external memory. Note that the DSP is still able to process data with zero wait-states for internal memory access.

An 80 MHz Motorola 56303 DSP was used rather than the slower 66 MHz processor that had been used on the Motorola evaluation board. An additional micro-controller, the Microchip PIC16C63 was added to the smart digital camera. The DSP communicates with the micro-controller via an SCI protocol (Synchronous Communications Interface). The sole task of the micro-controller is to control the acquisition of a digital image from the CCD sensor and a moderately slow "line-scan" sensor ADC (Analogue to Digital Converter). Three neighbouring pixels are packed into a single 24-bit word address in the memory space of the DSP. The packing of the pixel data effectively increases the pixel bandwidth to the DSP during the initial wavelet decomposition as three pixels can be read every machine cycle. An entire image can be acquired in 70ms due to the slow ADC used, but this was more than adequate given that the target frame rate for transmitted images was only 2 fps.

There are two main communication channels between the development PC and the embedded hardware of the smart digital camera. These two channels are the JTAG (Joint Test Action Group – IEEE 1149.1 test access port) and the RS-232 interfaces as shown previously in Figure 5.1. The developed assembly language algorithms for the DSP are downloaded to the smart digital camera via the JTAG interface and the

debugging software running on the PC can be used to control the execution of the programs running on the smart digital camera. The RS-232 interface is used to provide a modest bandwidth of 115,200 bps between the smart digital camera and the PC. This bandwidth is mostly used to transmit the compressed images from the smart digital camera to the PC, but is also used to allow the user of the PC to configure the parameters for image capture and compression in the camera.

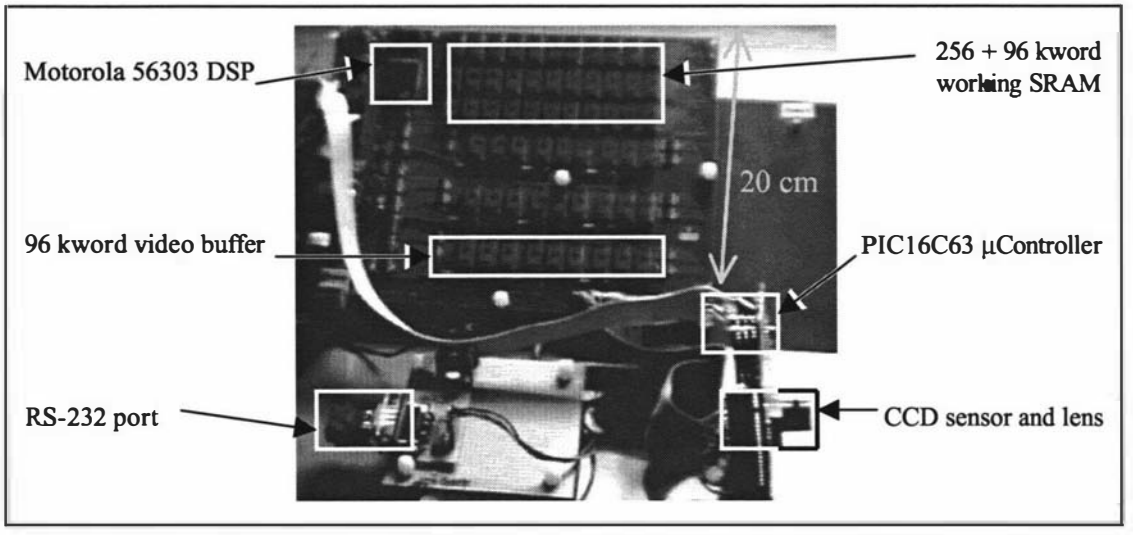


Figure 5.2 – Photo of prototype hardware for smart digital camera

Figure 5.2 shows a photo of the prototype board where the main board measures 25 x 20 cm. A separate smaller board contains the micro-controller, ADC and CCD sensor in the lower right corner of the photo.

5.1.1.2 Software architecture

The software components used on the PC and the smart digital camera are shown in Figure 5.3. The highlighted components were written by the author and contain the actual image compression and decompression algorithms. Motorola, Intel and the industry partner provided the other proprietary software components. When combined, the software provides the user with mechanisms to download to and debug code in the smart digital camera via the JTAG interface and also the ability to control the image capture and compression process.

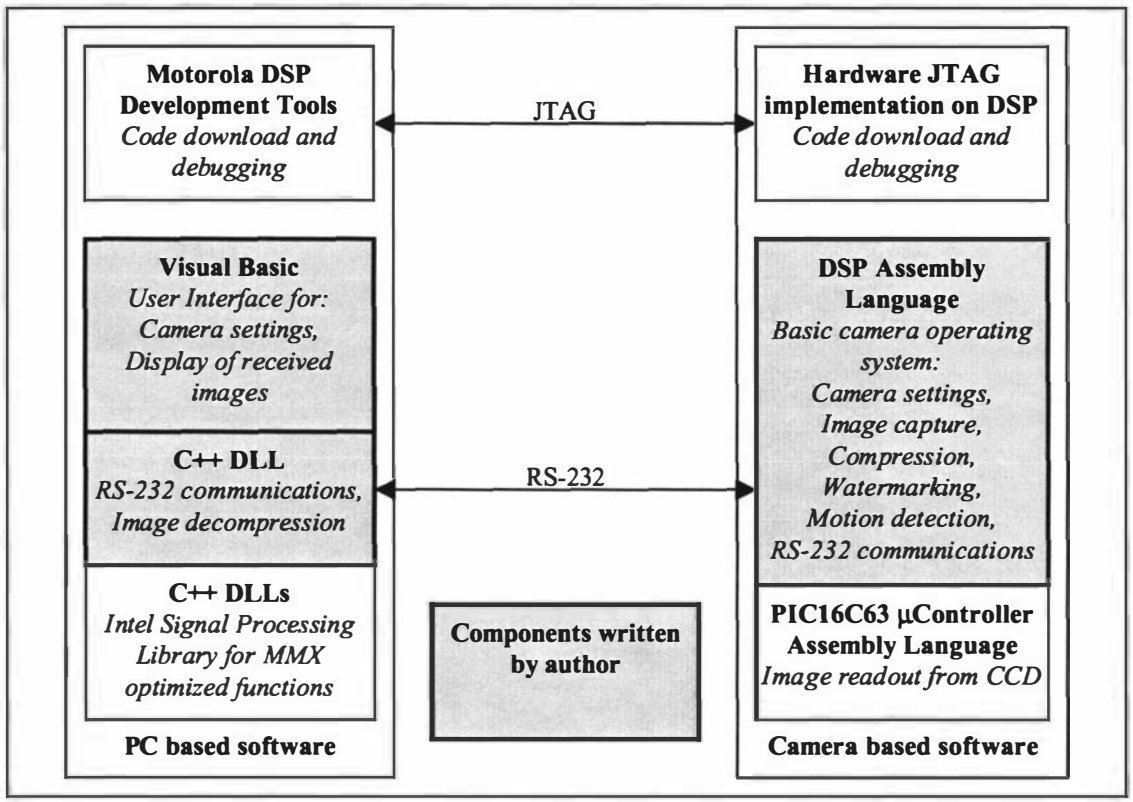


Figure 5.3 – Software architecture for prototype smart digital camera

Figure 5.4 and Figure 5.6 show the two main windows and the graphing window of the "Camera Viewer" user interface written in Visual Basic to allow images to be viewed and the smart digital camera to be controlled. The "Camera Viewer" application was written to support the capture and display of a single image at a time. A second Visual Basic application called the "Video Viewer" was developed to demonstrate the system using a fully pipelined sequence of operations. Figure 5.5 shows the "Video Viewer" user interface. This particular figure shows the motion detection function in use. The Mean Square Error calculated between the previous and current images is used to highlight the rectangular region near the centre of the image in proportion to the amount of motion detected.

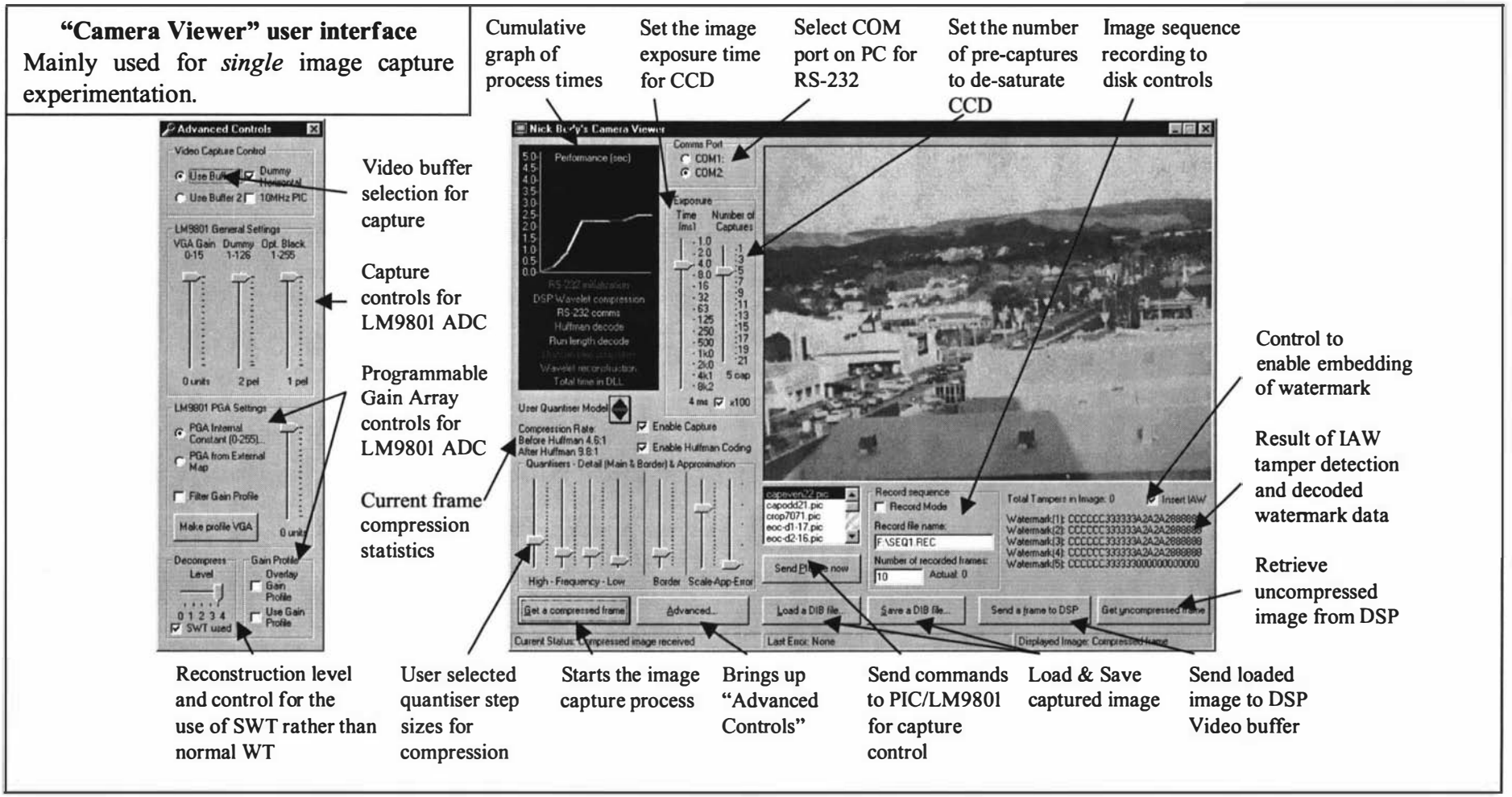
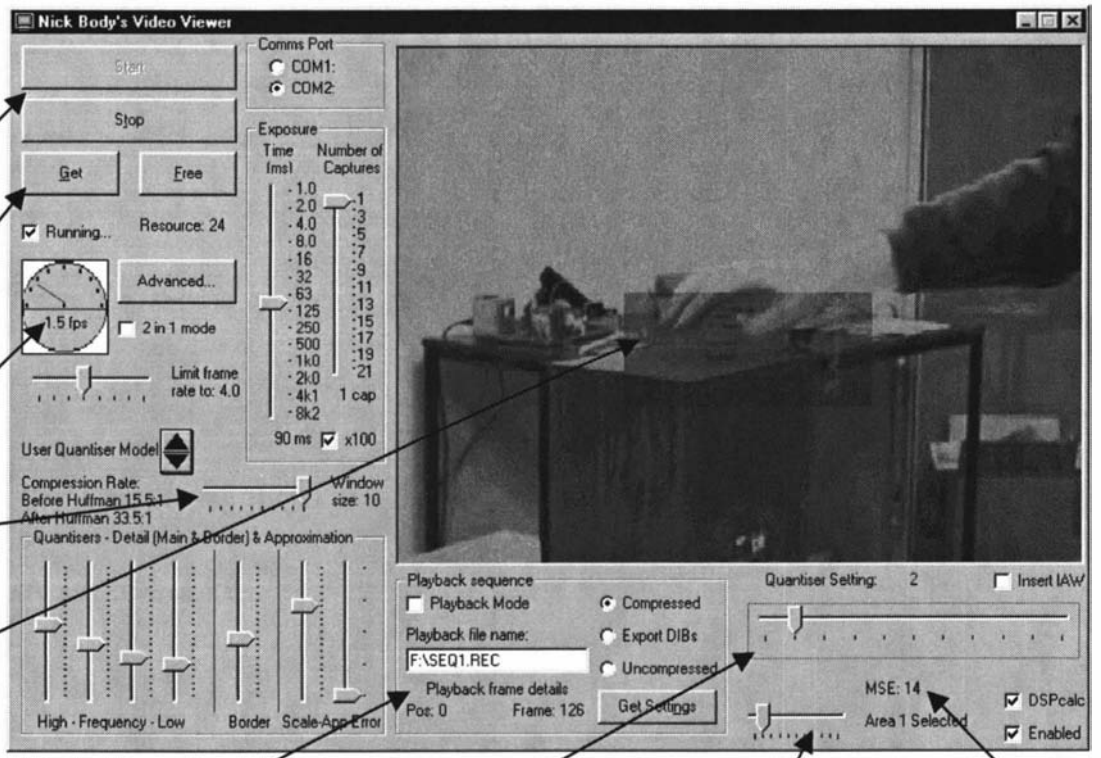


Figure 5.4 – Visual Basic "Camera Viewer" user interface for prototype smart digital camera

"Video Viewer" user interface
Used for viewing live and recorded Image sequences. Also for motion detection evaluation.



"Start" capturing images in a continuous pipeline until "Stop" is pressed

Initially allocate PC memory and COM port until resources are freed

Current frame rate and control to limit maximum frame rate

Scaling used for displayed image to see what reduced images look like

Rectangular regions where motion is detected are highlighted in proportion to the amount of MSE measured

Playback from disk of sequences recorded in "Camera Viewer"

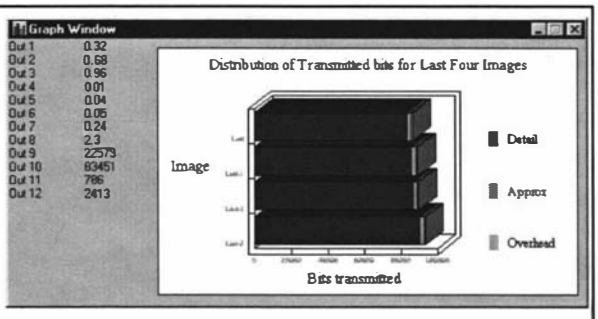
20 graduated image quality settings to select quantiser step sizes automatically

Motion detection in multiple rectangular regions controls

Scaled MSE metric calculated by the DSP between last two frames for current region

Figure 5.5 – Visual Basic "Video Viewer" user interface for prototype smart digital camera

Decompression output parameters
1. RS-232 setup time (320 ms)
2. DSP capture + compression (680 ms)
3. RS-232 transmission (960 ms)
4. Huffman decode (10 ms)
5. Run-length decode (40 ms)
6. Unscan tree structures (50 ms)
7. Wavelet decomposition (240 ms)
8. Total time in C++ DLL (2,300 ms)
9. Decoded symbols (22,579 sym)
10. Total number of bits for image (83,451 bits)
11. Bits for Huffman table (786 bits)
12. Bits for approximation image (2,413 bits)



The statistics shown relate to the capture of the image shown in Figure 5.4. This image has been compressed by a ratio of 9.8:1. The image contains 576 bits of watermark data. The graph shows the distribution of bits between the image detail, approximation and Huffman table.

Figure 5.6 – Visual Basic display of statistics for prototype smart digital camera

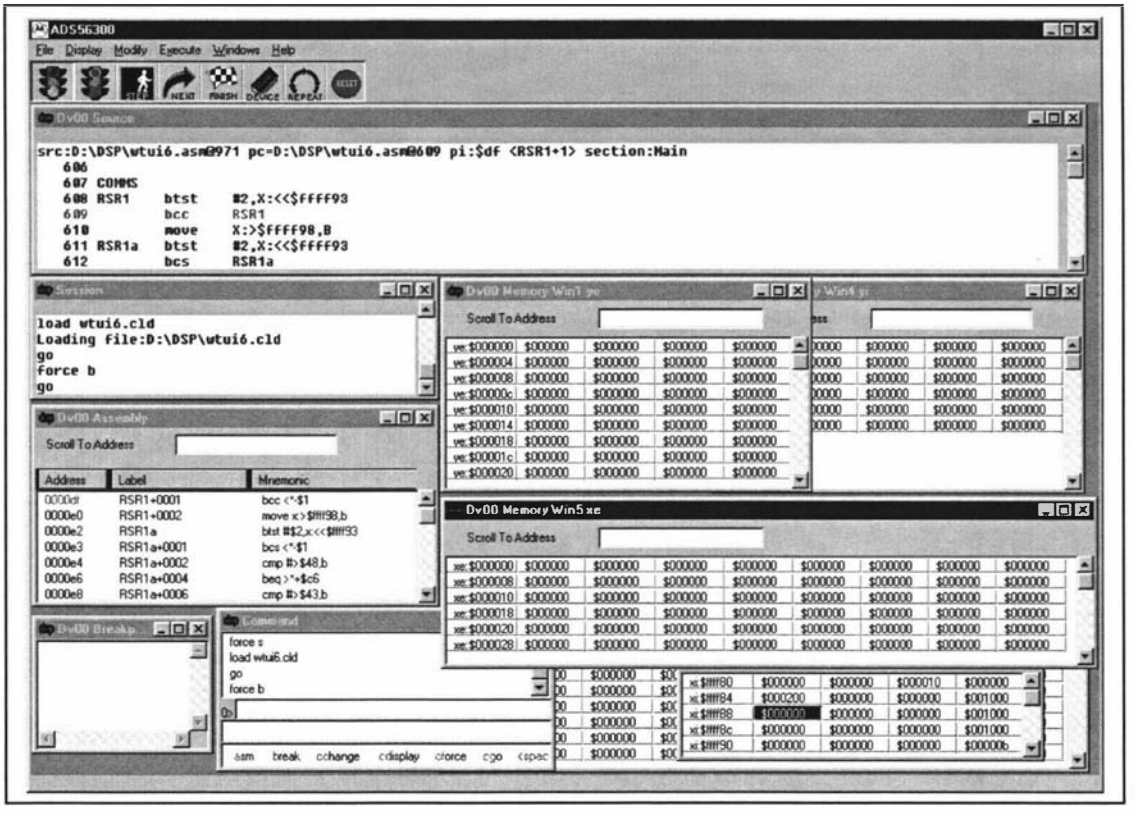


Figure 5.7 – Motorola development environment for debugging the DSP

Figure 5.7 shows the Motorola development interface used to download and debug code on the DSP via the JTAG interface. Memory can be inspected, breakpoints can be set and the program can be single stepped.

5.1.1.3 Bottlenecks and latency

The process of compressing an image for the purpose of transmission and display on a remote PC involves several stages of processing that can be viewed as a pipeline. Figure 5.8 details the main processes from left to right with the vertical axis measuring the time involved for each process. The horizontal arrows in the diagram indicate inter-process dependencies where one process cannot start until the previous process has completed. The specific processing for two separate images, ***A*** and ***B*** are labelled throughout the diagram to emphasise the pipeline. The main sequence of processing is listed below, the bracketed figures are indicative process times:

1. The PC initiates the capture of image ***A*** and provides the necessary parameters to specify exposure time and compression ratio. *(50 ms)*
2. The image is exposed, read out, and stored into SRAM under control of the PIC micro-controller. *(70 ms)*
3. The DSP compresses the image. The speed of compression is largely *independent* of scene content and compression ratio. *(140 ms)*
4. The DSP communicates the compressed image size and the PC replies with a request for the data. The PC also specifies the settings for the next image (***B***) to be captured and compressed while image ***A*** is being transmitted. *(50 ms)*
5. The compressed image is transmitted by the DSP to the PC. This is performed by an ISR (Interrupt Service Routine) on the DSP so that the PIC and DSP can continue processing the next image in the foreground. The time to transmit the image is *dependent* on the achieved compression ratio, which is scene and quantiser setting dependent. In this diagram it is assumed that the image has been compressed to a 4.5 kByte file, corresponding to a compression ratio of 32:1. At 115,200 bps (with one stop and one start bit) this transmission takes 400 ms to complete. *(400 ms)*
6. A further 50 ms is spent communicating the settings for the next image as well as requesting the transmission of compressed image ***B***. *(50 ms)*
7. The received image ***A*** is decompressed using software in the C++ DLL (Dynamic Link Library) and the Intel Signal Processing Library. To reconstruct the image to the full 512 x 288 pixel resolution takes 400 ms on a Pentium 200 MHz PC with

MMX capabilities. This reconstruction time is independent of the compression ratio. *(400 ms)*

8. The C++ DLL returns a bitmap to the Visual Basic user interface that is scaled to correct for the aspect ratio and is then displayed to the user. *(20 ms)*

By the time image ***A*** has been displayed, image ***B*** has been received by the PC ready for decompression, image ***C*** has been compressed by the DSP and is waiting for transmission, and image ***D*** is about to be captured. The typical time between image capture and display for image ***B*** onwards is 1,320 ms (α). The time between frames (δ) is 450 ms corresponding to a frame rate of 2.2 fps. This frame rate can only be sustained if at least a 32:1 compression ratio is achieved. The frame rate cannot be improved on as the 200 MHz PC always takes 400 ms to decompress the image. The image decompression and the RS-232 bandwidth are both limiting bottlenecks in performance in this example. A faster PC would improve the image decompression time and would then make the RS-232 bandwidth the only bottleneck in the system if the compression ratio remained at 32:1. The PC software also uses interrupt driven RS-232 communication to ensure that the PC can decompress a previously received image whilst receiving the next image at the same time in the background.

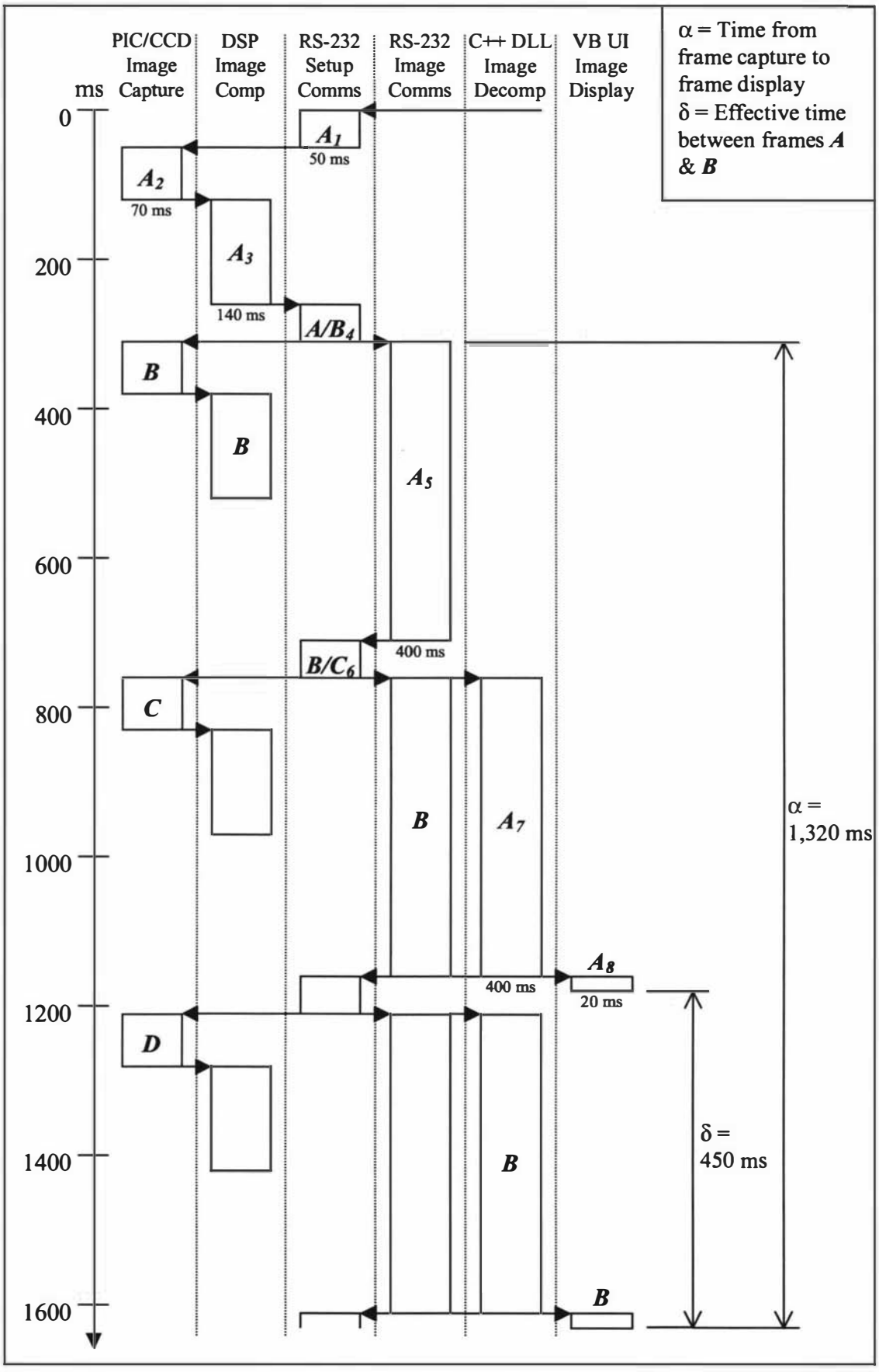


Figure 5.8 – Process timing diagram for prototype smart digital camera

5.1.2 Production system

The final production system measures 12 x 12 x 6 cm and is shown in Figure 5.9 in its case. The processing power of the hardware is similar to that of the prototype board, however the layout is more compact. The SRAM memory package density is four times as dense as that used in the prototype camera, this reduces the number of memory chips required by a factor of four. The image size has also been increased by 33% vertically to 512 x 384 pixels and a CMOS based camera "system-on-a-chip" sensor [5.1] is used instead of the CCD based sensor in the prototype. Six infra-red LEDs can be controlled to provide supplementary illumination in poor light.

The following sections describe the differences in the hardware and software architectures from that of the prototype system. The larger image size increases the time taken to compress and transmit the image data. This is partially compensated for by the improved communications network that allows a single smart digital camera to be viewed by many Workstation PCs simultaneously.

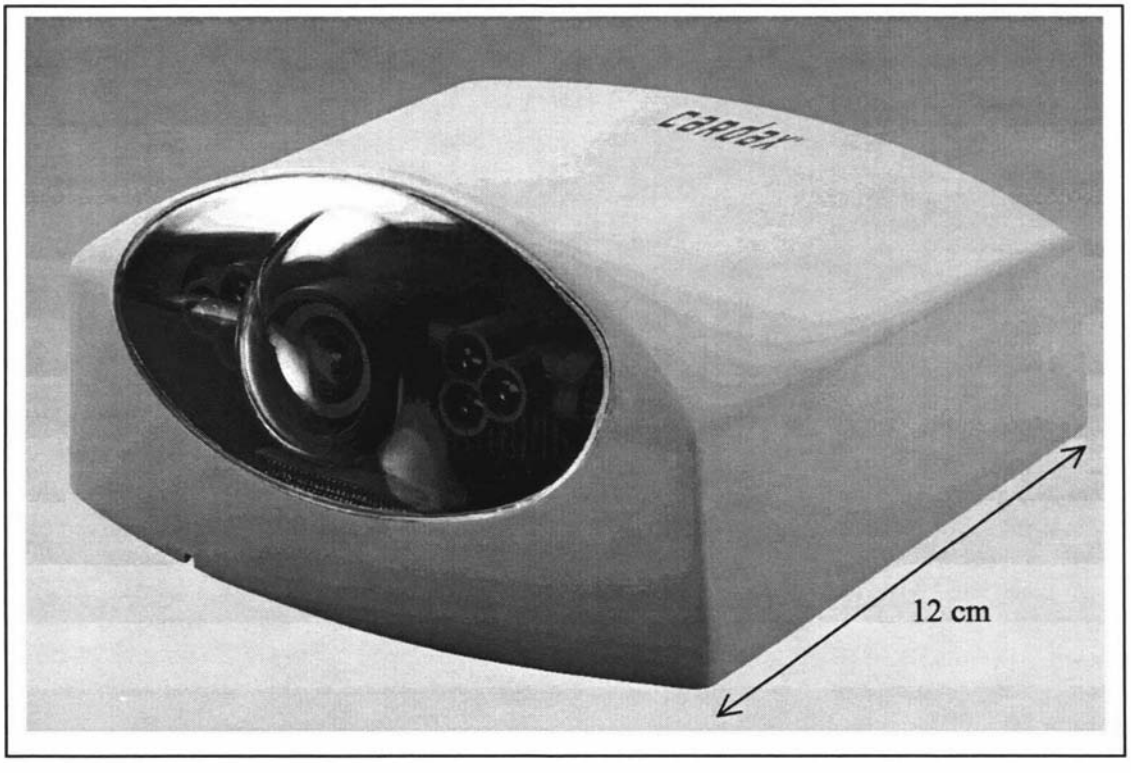


Figure 5.9 – Photo of production hardware for smart digital camera

5.1.2.1 Hardware architecture

The production hardware block diagram is shown in Figure 5.10. This diagram shows that the production system has two additional controllers between the DSP and the PC. The Zilog Z80181 micro-controller is integral to the smart digital camera and provides a buffered interface between the parallel "Host Port" on the DSP and the serial "RS-485" based local network that the camera and other surveillance devices use for communication. At the centre of the RS-485 communications network is an i386EX33 based embedded micro-computer known as the "Controller" that is capable of buffering image sequences. The other main function of the micro-computer is to facilitate communications between a Server PC and multiple Workstation PCs on an Ethernet based network and the devices on the RS-485 local network such as the smart digital camera. Images may be streamed from multiple smart digital cameras to multiple Workstation PCs using the UDP protocol (User Datagram Protocol) on the Ethernet network. Configuration and software can be downloaded to the smart digital camera via its Ethernet connection using the TCP protocol (Transmission Control Protocol).

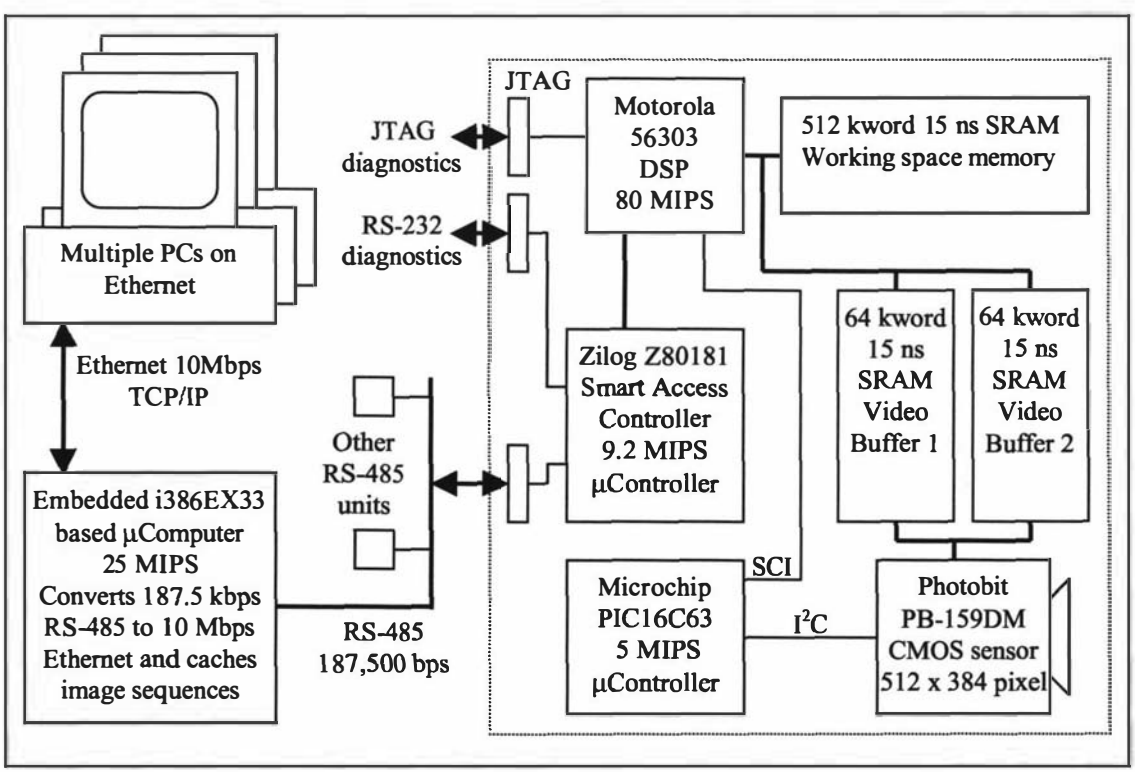


Figure 5.10 – Block diagram of production hardware for smart digital camera

The JTAG and RS-232 interfaces are still available in the production system but only when the casework is opened. These interfaces are primarily used for debugging purposes and for retrieving images directly from the smart digital camera to a laptop during maintenance operations. This feature is useful when mounting the smart digital camera on a wall as it allows the installer to aim the camera correctly prior to completing the installation of the entire system.

The final production board uses 1.875 MBytes of (compact footprint) 15ns SRAM. The main board shown in Figure 5.11 and measures only 10 x 10 cm. Because of its compact size, there are no address and data bus layout problems so the external memory can be accessed at full speed with just a single wait state. Of the total memory, 192 kBytes are required for the 33% larger raw image of 512 x 384 pixels, and the working space for the final truncated wavelet decomposition at 24-bit precision is 233 kwords or 700 kBytes. The additional 1,028 kBytes of memory above these requirements is used to store the larger smart digital camera operating system code as well as to buffer sequences of up to 100 compressed images that are transferred to the main controller when bandwidth is available.

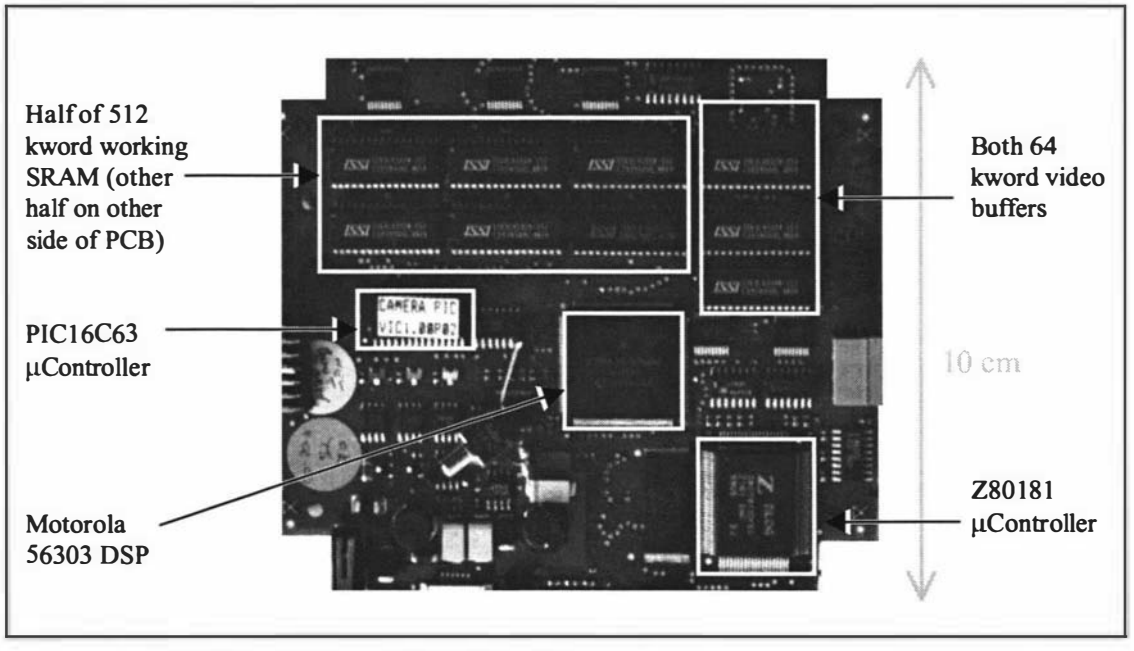


Figure 5.11 – Photo of production hardware main circuit board for smart digital camera

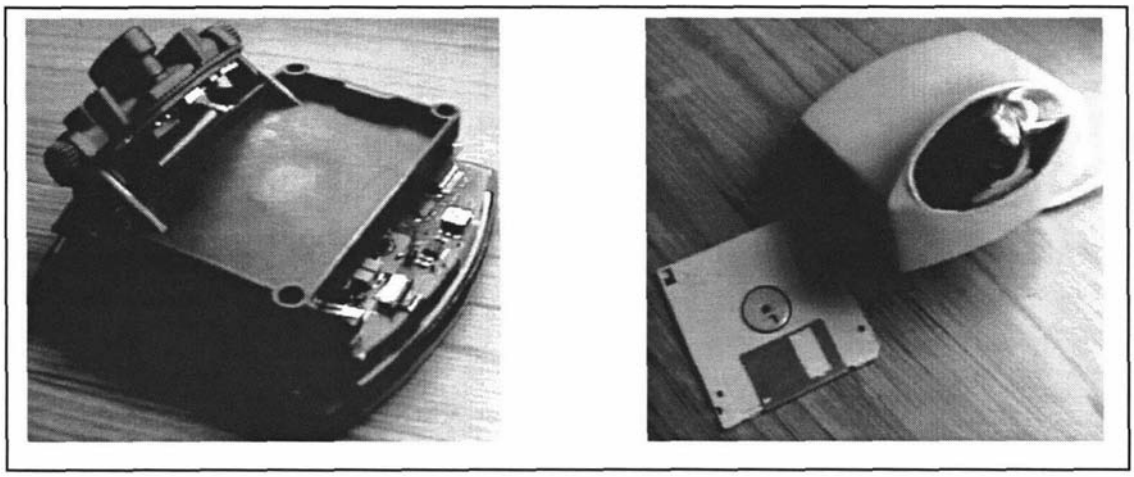


Figure 5.12 – Photos of production hardware for smart digital camera

The Photobit PB-159DM CMOS image sensor and the lens assembly are mounted on a separate circuit board measuring 7 x 3 cm. This circuit board is attached to an adjustable platform to enable the sensor to be pointed at the desired field of view as shown in the left photo of Figure 5.12. A flexible cable attaches this circuit board to the main circuit board. The Photobit sensor is described by its manufacturers as a digital camera "system-on-a-chip" [5.1]. The single chip contains the CMOS image sensor array as well as the ADC and control circuitry. The imaging chip directly provides a digital stream of 8-bit pixel values that are ready to be stored in the SRAM based video buffer. The overall size of the camera is compared to that of a 3.5" floppy disk in the right photo of Figure 5.12.

In summary, the main differences in the production smart digital camera hardware are:

- Higher density memory used to reduce size of board by a factor of four.
- 33% larger image area for a higher vertical resolution.
- Use of a CMOS based camera-on-a-chip running at a continuous 20 fps exposure rate rather than a CCD sensor exposed on demand by the DSP.
- Interface to 187,500 bps RS-485 local bus network of devices rather than to a 115,200 bps RS-232 port on a single PC.
- The addition of a micro-computer based Controller external to the smart digital camera to interface the RS-485 network to a standard TCP/IP based broadband computer network.

5.1.2.2 Software architecture

The software architecture for the production smart digital camera is similar to that of the prototype system. The architecture is shown in Figure 5.13, the algortihm used for the image compression and decompression is identical to that of the prototype system. The production system software uses a C++ custom operating system running on the DSP and Z80181 processor. The operating system controls when the image compression algorithm is run.

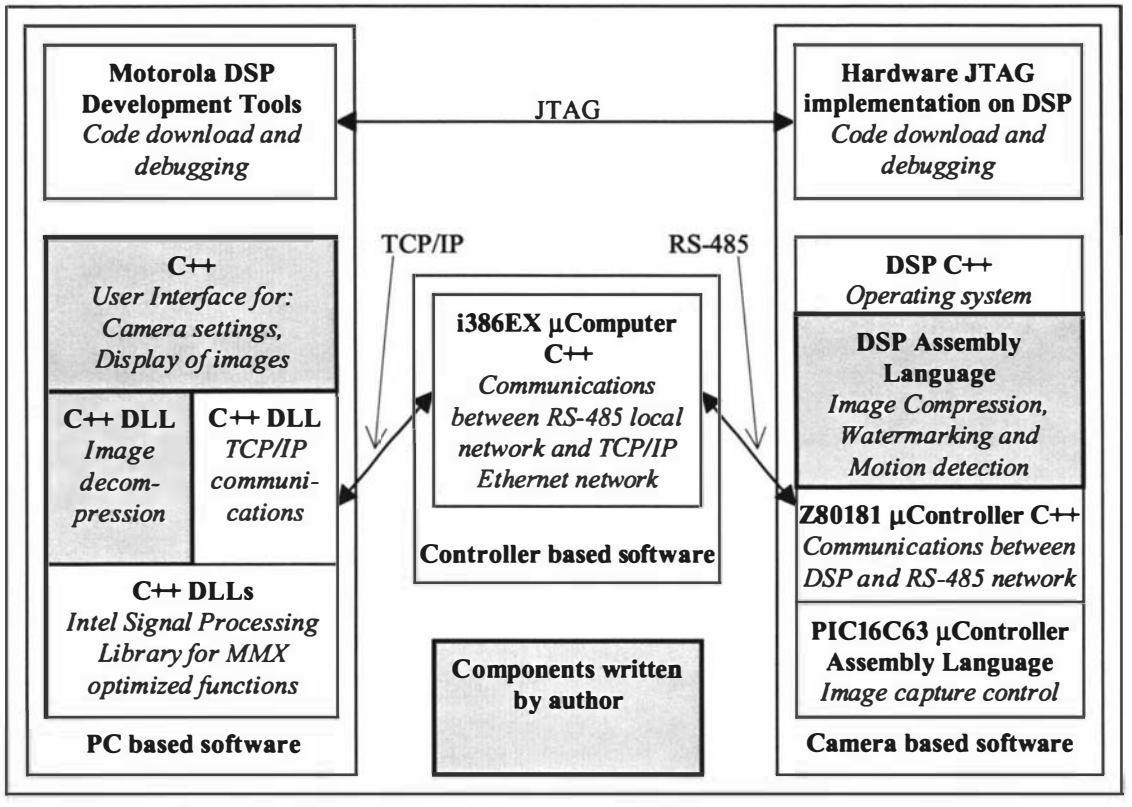


Figure 5.13 – Software architecture for production system

The smart digital camera operating system is responsible for managing the additional functional requirements of the production camera. These requirements include:

- The automated downloading of new software for the Z80181 and DSP processes from the Server PC without the need to use the JTAG interface.
- The management of a circular buffer of 100 images available to be sent to the Server PC on demand as an archive sequence associated with a specific event.
- The configuration of motion detection regions in the field of view.

- The configuration of image compression quality.
- Response to "heartbeat polls" from the Controller. These polls are used by the Controller to detect that the smart digital camera is able to communicate with the Controller when required.

The main role of the Controller is to manage and configure the network of RS-485 devices and to serve image sequences over the TCP/IP network to multiple Workstation PCs. If a smart digital camera detects significant motion in one of its motion detection regions, or another surveillance device is triggered then the camera can be configured to send a set of pre-event and post-event images to the Server PC. These short event sequences can be recorded at up to 4 fps even though they can only be transmitted at a rate of 2 fps. The Server PC then archives the image sequence for later retrieval and also associates the image sequence with the event that triggered the sequence. This feature is essential to the provision of an alarm verification service where a short image sequence is used to verify the cause of an alarm.

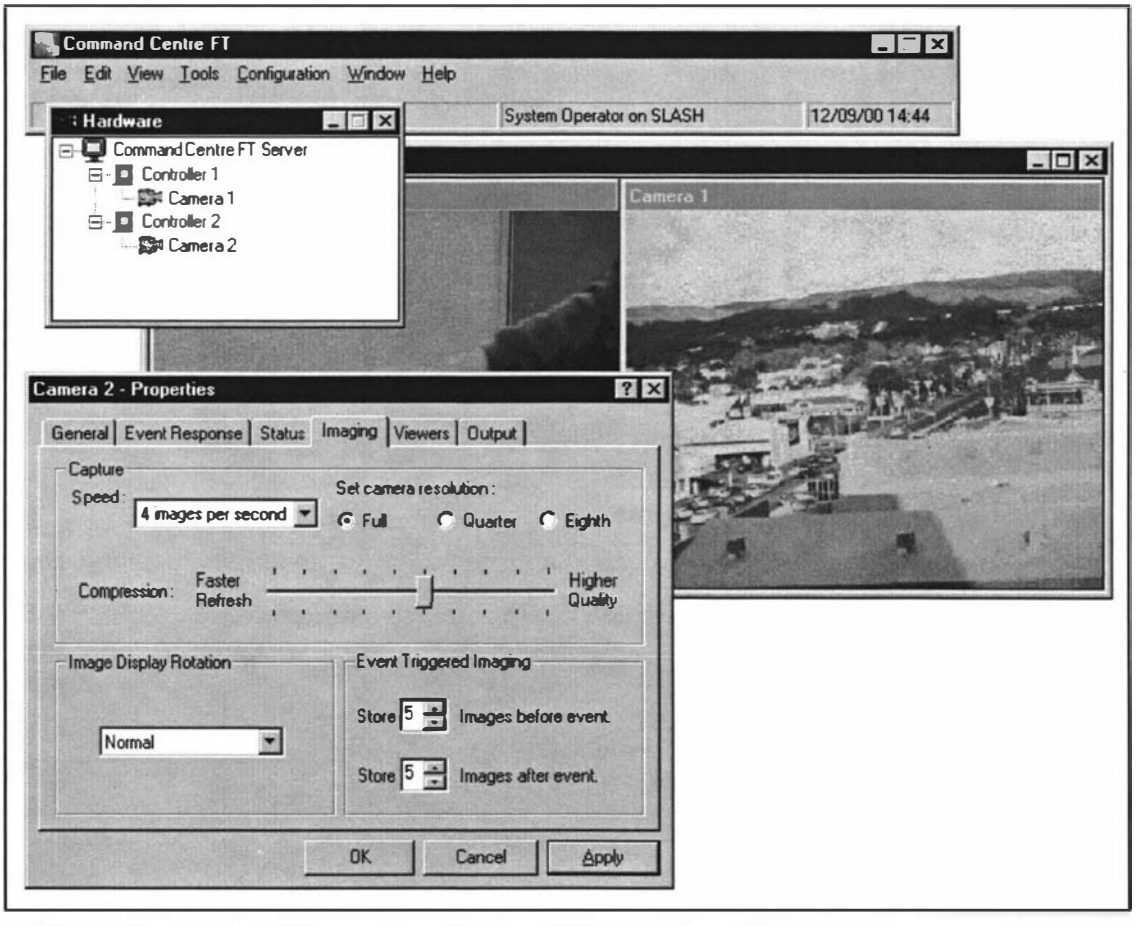


Figure 5.14 – C++ "Command Centre FT" user interface for production system

The user interface relevant to the smart digital camera is shown in Figure 5.14. The smart digital camera is part of a much larger surveillance product. The same basic user interface is used to configure the entire system. A "Property Page" of one of the two smart digital cameras is shown in the figure. The user can alter the settings for each smart digital camera using the property pages and the "Hardware" window shows the physical connections between the Server PC, the Controllers and the smart digital cameras. Up to four "live" smart digital cameras can be viewed at two fps from each Workstation PC. Additionally an archived image sequence associated with a particular event can be viewed from any Workstation PC as they are stored in a central database on the Server PC.

In summary, the main differences in the production smart digital camera software are:

- The smart digital camera related software is only a small part of a much larger surveillance system. The original prototype system was a stand-alone unit.
- The smart digital camera has a higher degree of self-management. The camera operating system controls when captured images are compressed and also manages a circular buffer of images ready for retrieval by the Controller.
- The real-time communication over the RS-485 local network is managed by a separate processor to ensure that the overhead of communicating with the Controller does not adversely impact the operation of the DSP.
- The Workstation PCs can communicate with multiple Controllers to allow simultaneous decompression and display of multiple live image sequences.
- The Server PC is capable of associating events in the surveillance system with archived image sequences that can be stored in a central database for later retrieval by Workstation PCs.

5.1.2.3 Bottlenecks and latency

The processing pipeline for the production system is shown in Figure 5.16. There are fewer stages in the overall pipeline than that shown for the prototype system in Figure 5.8. The production unit has its own operating system and is more autonomous than

the prototype camera. The prototype smart digital camera relied on the host PC to co-ordinate the image capture and compression processes followed by a request for transmission of the compressed image. The production unit simply captures images continuously and maintains its own circular buffer of images that can be asynchronously requested by the Controller. The process diagram shows the smart digital camera capturing images at **both** the rates of two frames per second (fps) (images ***A*** and ***B***) **and** four fps (images ***B*** through ***E***). There is only enough bandwidth to transmit two fps at a compression ratio of 32:1. For general live viewing of the images, the two fps frame rate is appropriate. If an event triggers the capture of an archive image sequence then the smart digital camera can switch to the four fps capture and compression mode of operation. The higher frame rate improves the temporal resolution of the captured sequence about the time of the event as shown in Figure 5.15.

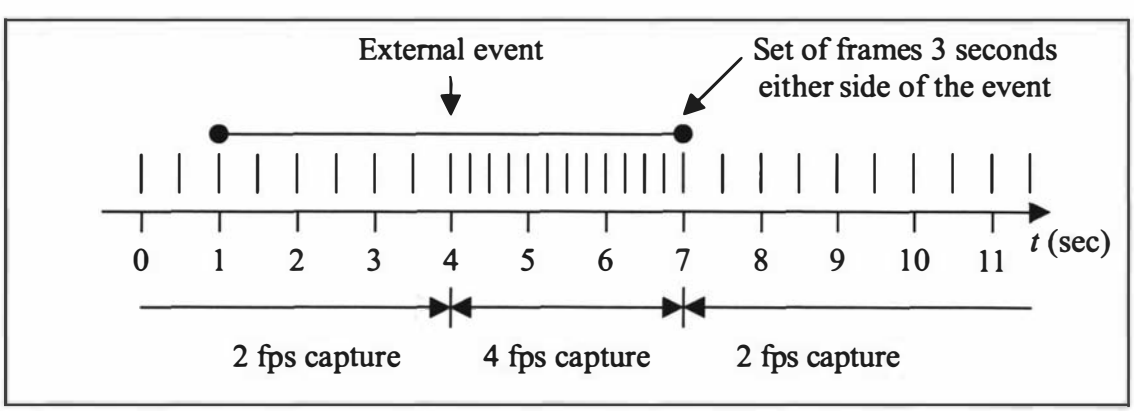


Figure 5.15 – Change in frame rate when an event triggers an archive sequence

The specific processing for two separate images, ***A*** and ***B*** are labelled throughout the diagram to emphasise the pipeline. The main sequence of processing is listed below, the bracketed figures are indicative process times:

1. The Photobit sensor continually captures frames at the rate of 20 fps. The DSP commands the PIC micro-controller to selectively gate the next frame to the SRAM capture buffer when a new image is required. The Photobit sensor automatically adjusts the image exposure time using an in-built algorithm. *(50 ms)*
2. The DSP compresses the image. The speed of compression is largely *independent* of scene content and compression ratio. This process takes approximately 33%

longer than the prototype as the image size is also increased by this factor. The DSP generally compresses every fifth or tenth captured image to produce an effective frame rate of four or two fps. *(180 ms)*

3. The DSP stores the compressed image in a circular buffer that can contain 100 compressed images. The Zilog micro-controller uses DMA to retrieve the compressed image data in 256 byte packets. The DSP is free to begin compressing the next image according to the frame rate it has been configured for. Image ***B*** is chosen 500 ms after image ***A*** was captured to achieve a 2 fps frame rate. The Zilog micro-controller sends the packetised image along the RS-485 local bus network at an effective bit-rate of 95 kbps as the total network bandwidth of 187.5 kbps is shared among other units on the network. An image compressed at 32:1 of size 512 x 384 pixels will take 520 ms to be transmitted. *(520 ms)*
4. The packetised image data arriving at the i386EX based micro-computer is converted one packet at a time to UDP/IP for simultaneous transmission to one or more Workstation PCs on the Ethernet LAN. *(520 + 20 ms)*
5. The received image ***A*** is decompressed using software in the C++ DLL and the Intel Signal Processing Library. To reconstruct the image to the full 512 x 384 pixel resolution takes 520 ms on a Pentium 200 MHz PC with MMX capabilities. This reconstruction time is independent of the compression ratio. *(520 ms)*
6. The C++ DLL returns a bitmap to the production system C++ user interface that is scaled to correct for the aspect ratio and is then displayed to the user. *(20 ms)*

The minimum time between image capture and display for an image is 1,310 ms (α) in the production system at a compression ratio of 32:1. The time between frames (δ) is 520 ms corresponding to a frame rate of 1.9 fps. A 200 MHz PC always takes 520 ms to decompress the image and this bottleneck would severally limit the number of simultaneous smart digital cameras that could viewed concurrently. The production system utilises either Pentium II or Pentium III based Server and Workstation PCs running at 500MHz or more. These PCs are at least 5 times more powerful that the original PC used for the prototype system and are well suited to viewing multiple image sequences at a time. Four smart digital cameras can be viewed simultaneously, each at 2 fps, using a modern day PC.

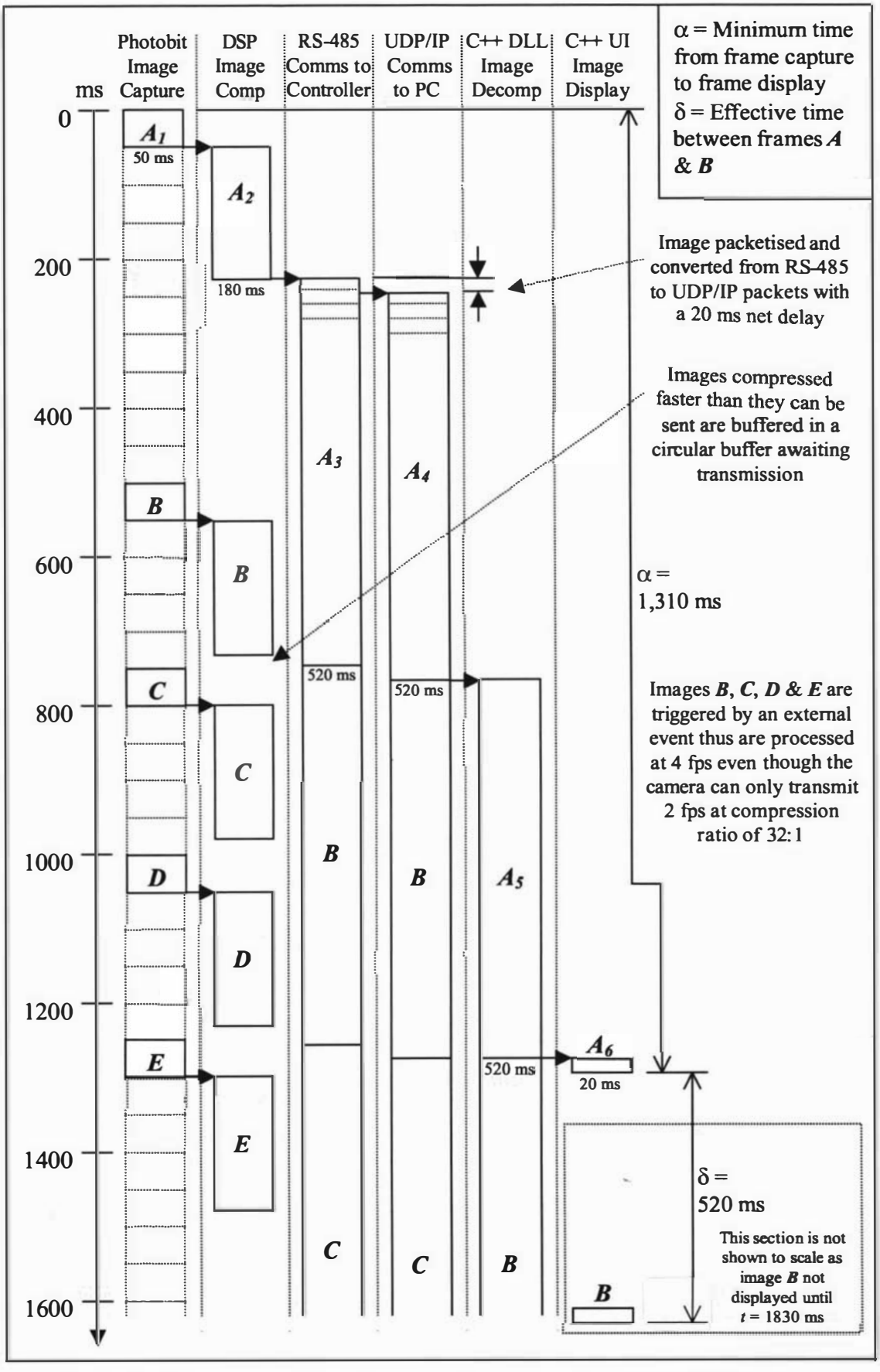


Figure 5.16 – Process timing diagram for production smart digital camera

In summary, the main differences in the production smart digital camera performance are:

- The smart digital camera uses a higher resolution image that results in a larger compressed image to be transmitted.
- There is slightly reduced bandwidth available to transmit images from the camera to the Controller.
- The camera has better buffering ability to ensure images destined for archives are transmitted at a later time when there is sufficient bandwidth available. This usually occurs at the expense of the live viewing frame rate which is decreased whilst the archived images are transmitted to the Server PC.
- Faster PCs are used to ensure that multiple images can be decompressed at 2 fps on a single Workstation PC. The use a modern PC reduces the net time between image capture and display (α) from 1,310 ms to 900 ms in the production system.

5.2 Compression algorithm performance

The SWT (Symmetric Wavelet Transform) based image compression algorithm described in chapters 2 and 3 was designed in 1996 using techniques that were current in the literature but were not yet used in commercial image compression standards. At about the same time a group of graduate students from the University of Waterloo (Ontario) provided a useful Internet based resource for comparing the performance of state-of-the-art image compression algorithms [5.2]. The resource is called the "Waterloo BragZone" as it provides an independent way of comparing the performance of various image coders in the literature. One of the goals of the BragZone authors is to provide a uniform set of test results from various image coders using a diverse set of images. Various test images are compressed at a large number of compression ratios and several objective mathematical distortion measures are tabulated for each image coder.

This section graphs the RMSE (Root Mean Squared Error) metric verses compression ratio for the main classes of image coders that existed in 1995 as well as the SWT based coder described in this thesis. The RMSE metric is graphed against the compression ratio as this metric provides a closer match to visual difference than the more often used PSNR metric [5.2]. The 512 x 512 pixel Lena, Goldhill and Barbara 8-bit greyscale images from the BragZone Internet site are used to compare the performance of the SWT based coder with other coders at various compression ratios. The five image coders compared from best to worst are:

1. SAPB – Said and Pearlman Progressive, embedded, wavelet based coder with real valued wavelet filters, arithmetic entropy coding [5.3-5].
2. TRNA – Fisher Fractal based 3-level quadtree coder, Huffman entropy coding [5.6].
3. EPIC – Simoncelli and Adelson Pyramidal Image coder, Huffman entropy coding [5.7].
4. IJPG – Independent JPEG group coder with optimised multiple pass Huffman entropy coding [5.8].

5. JPEG – Commercial JPEG coder from Iterated Systems with standard Huffman entropy coding [5.9].

5.2.1 Target SWT compression ratios and quantiser settings

The target compression ratios examined for comparison were "Low" – 20:1; "Medium" – 30:1; and "High" – 60:1. To achieve a variety of compression ratios, 10 predefined image "quality" settings were defined by using a linear range of quantiser multipliers. The quantiser multipliers (Q_{mult}) are inversely proportional to the quantiser step size (Q_{step}) by the ratio of $8{,}192/Q_{mult} = Q_{step}$. The Q_{mult} values range from 512 (corresponding to a "fine" quantiser step size of 16) being visually lossless down to as low as 8 (corresponding to a "coarse" quantiser step size of 1,024). The two low frequency quantisers were set to be less coarse than the two high frequency quantisers to approximately model the human visual system's spatial frequency sensitivity [5.10]. The table of quantiser multipliers is shown in Table 5.1 and they are graphed in Figure 5.17.

Quality	High Freq	Mid-High Freq	Mid-Low Freq	Low Freq
1 (Poor)	8	16	32	64
2	64	71	85	114
3	120	126	139	164
4	176	181	192	213
5	232	236	245	263
6	288	292	299	313
7	344	347	352	363
8	400	402	405	412
9	456	457	459	462
10 (Good)	512	512	512	512

Table 5.1 – Quantiser multipliers for 10 quality settings

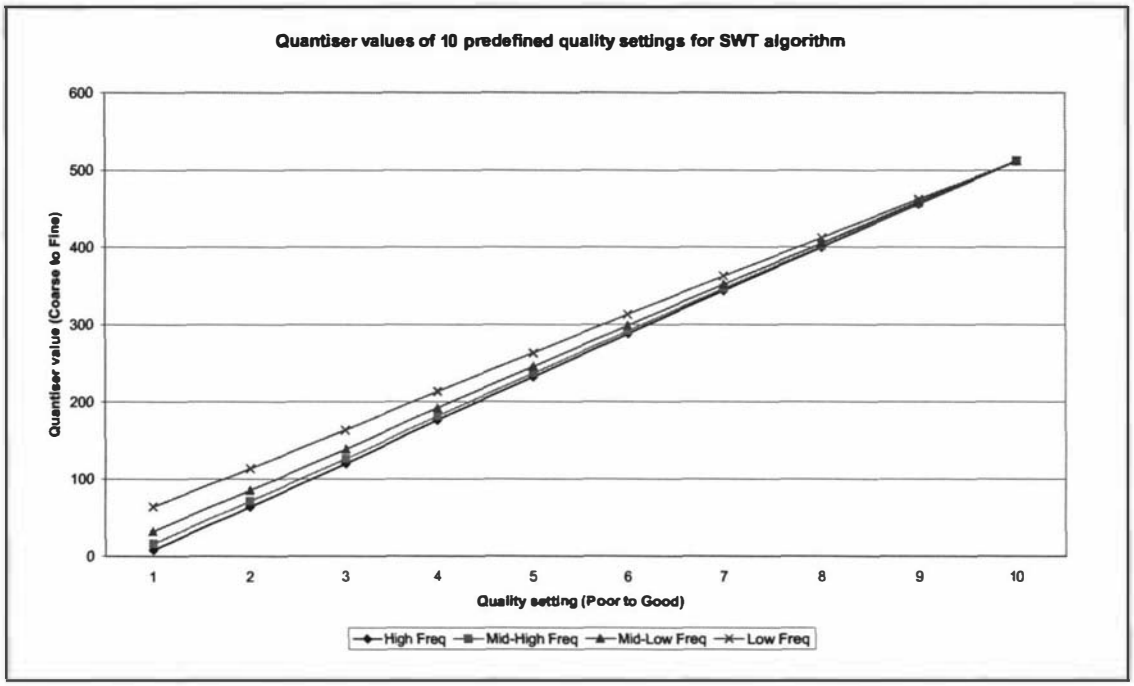


Figure 5.17 – Quantiser multiplier values used for the 10 SWT quality settings

5.2.2 Reference images

The three reference images used to evaluate the compression algorithms differ in scene complexity. This means the amount of compression achievable for each image differs for a single quality setting. The Lena image has mainly smooth regions without much texture except for the feather in the hat. This image compresses well, as there is not much edge information to represent. The Barbara image also has a number of smooth regions in the image but also has large regions of structured "checkerboard" texture. The structured texture easily demonstrates, to the human observer, quantisation errors that occur in most image coders at high compression ratios. The Goldhill image contains medium scale texture and virtually no large uniform areas and is a good image to test the ability of the image coder to hide quantisation errors in the textured regions where they remain invisible to the human observer. These three images are shown in Figures 5.19, 5.21 and 5.23.

5.2.3 Results of compression performance test

Table 5.2 contains the achieved Compression Ratio (CR) and RMSE for each of the three reference images at the 10 different image quality settings. The highlighted cells in the table represent the closest compression settings that achieve the target compression ratios of 60:1, 30:1 and 20:1. The images produced at these three target compression ratios are shown on the page following each of the algorithm performance graphs. To the right of each of the compressed images is the error image. This image is produced by subtracting the original image from the compressed image and scaling the intensity by a factor of 4. There is a commentary beneath each of the algorithm performance graphs and the SWT based coder is ranked amongst the 6 image coders for each of the three images. The ranking is based solely on the algorithm performance graphs by examining the performance over the 20:1 to 60:1 range. The complete ranked order is shown as the order of the coders in the legend beneath each graph.

Quality	$Lena_{CR}$	$Lena_{RMSE}$	$Barbara_{CR}$	$Barbara_{RMSE}$	$Goldhill_{CR}$	$Goldhill_{RMSE}$
1	190.1	18.2	157.9	24.0	213.4	18.2
2	63.5	8.8	44.1	15.8	57.8	11.0
3	41.2	6.9	27.2	14.3	30.1	8.9
4	31.3	6.1	20.6	13.7	20.7	7.8
5	25.5	5.6	16.8	13.4	16.2	7.1
6	21.6	5.3	14.4	13.2	13.4	6.7
7	19.0	5.1	12.8	13.1	11.7	6.4
8	16.7	4.9	11.4	13.0	10.3	6.2
9	15.2	4.8	10.6	13.0	9.5	6.0
10	13.7	4.7	9.8	12.9	8.7	5.9

Table 5.2 – Compression Ratio and RMSE values at 10 quality settings

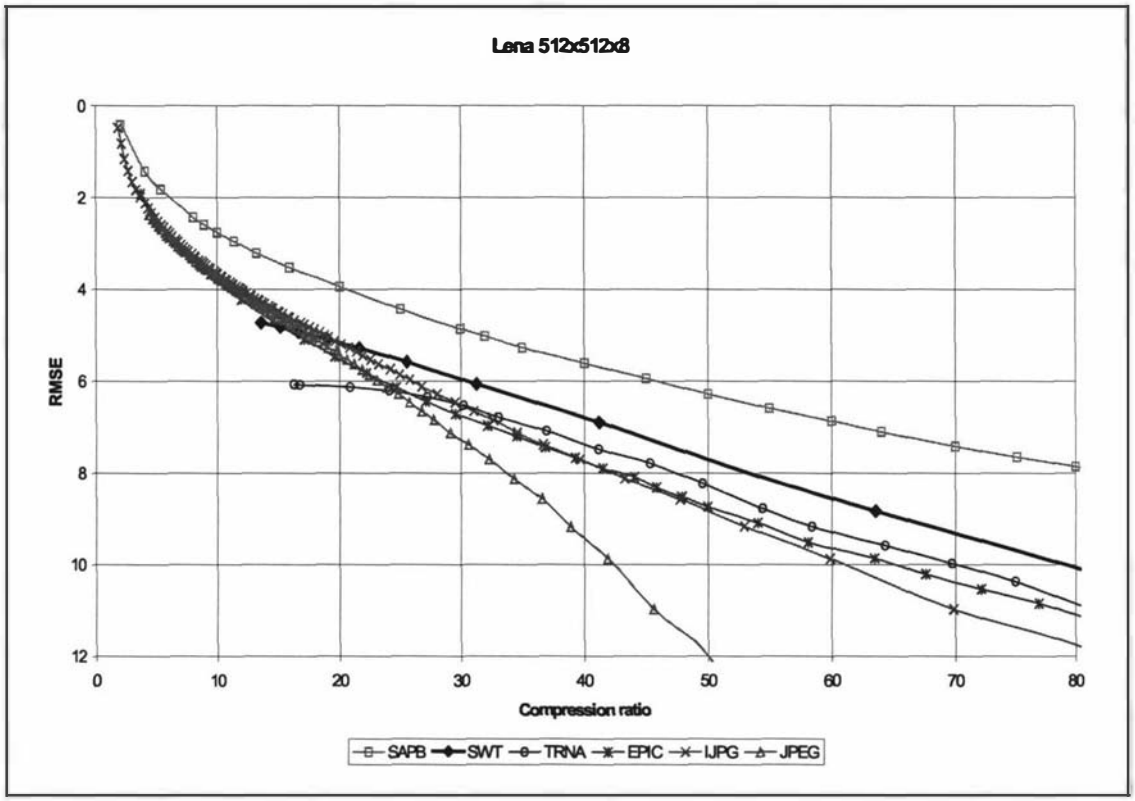


Figure 5.18 – SWT algorithm performance against its peers on Lena image

The Lena image compresses well with all image coders with a low RMSE. The SWT based coder performs particularly well and is ranked 2nd from the set of 6 image coders tested for this image. The error images on the next page show the highest amplitude errors occur about the edges in the image and the textured feature region. The heavily textured regions disguise the quantisation errors well. There are some ringing effects about the sharp edges in the image caused by the coarse quantisation of the higher frequencies in the image.

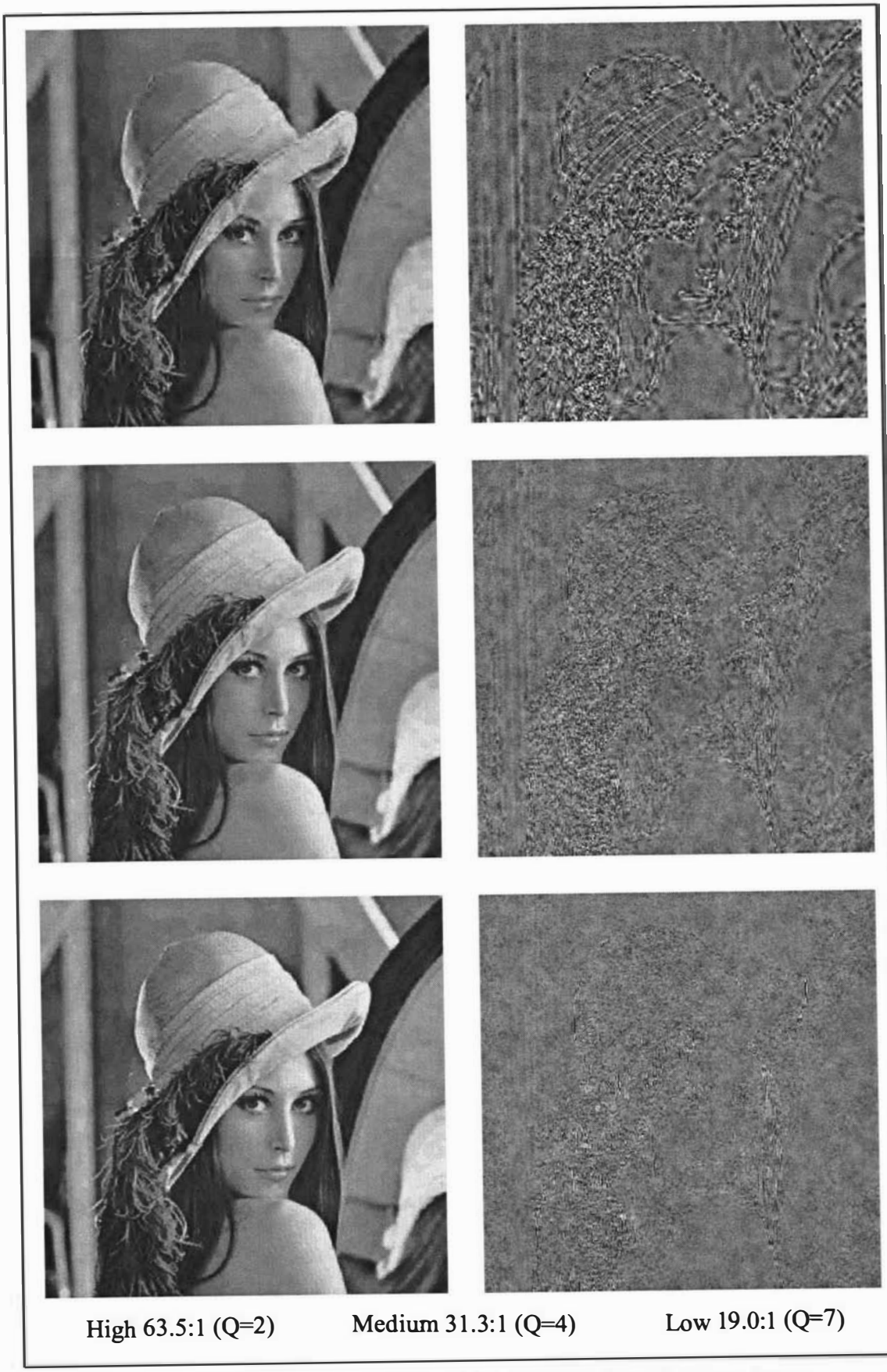


Figure 5.19 – High, medium and low compression on Lena image

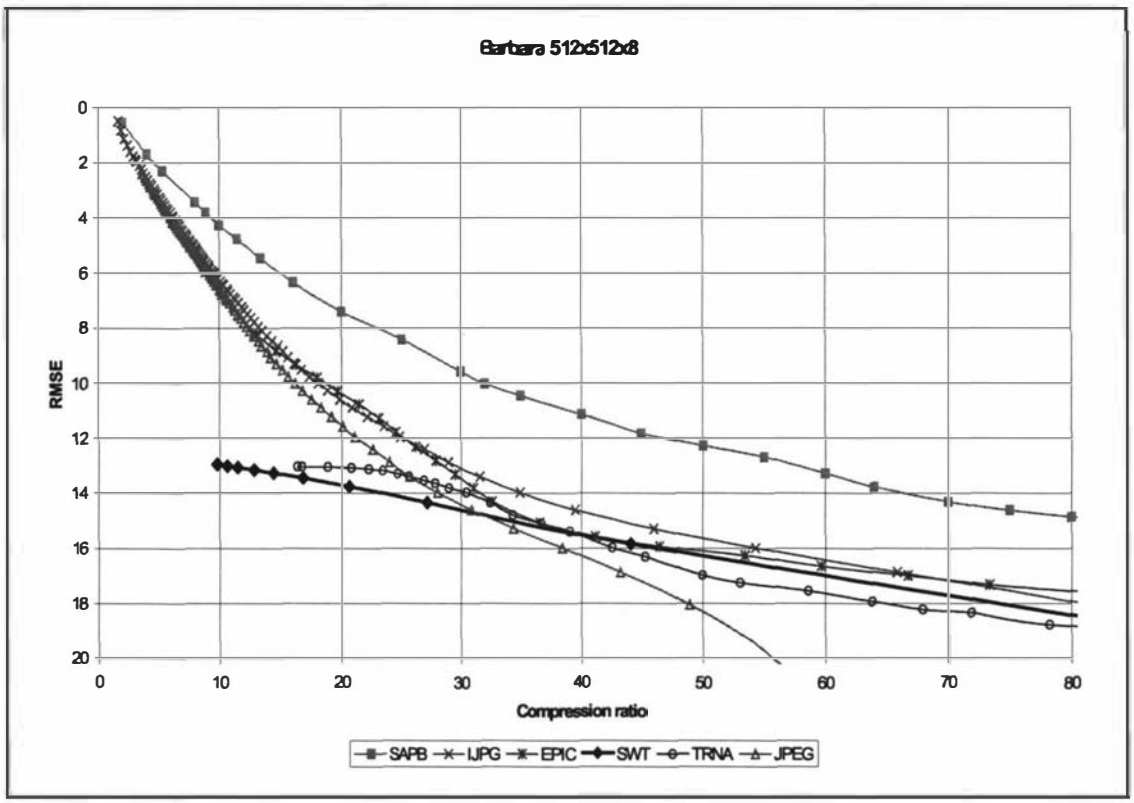


Figure 5.20 – SWT algorithm performance against its peers on Barbara image

The checkerboard regions in the Barbara image reveal compression artefacts even at the low compression ratio of 20.6:1. The SWT based coder discards all the vertical and diagonal high frequency information in the first decomposition of the image to reduce computation and memory usage. These same frequencies are used to define the fine checkerboard pattern and the pattern is blurred when these frequencies are removed. The SWT based coder is ranked 4th from the 6 image coders tested for this image. This image does not compress as well as the Lena and Goldhill images due to the large amount of high frequency information required to represent the image.

Figure 5.21 – High, medium and low compression on Barbara image

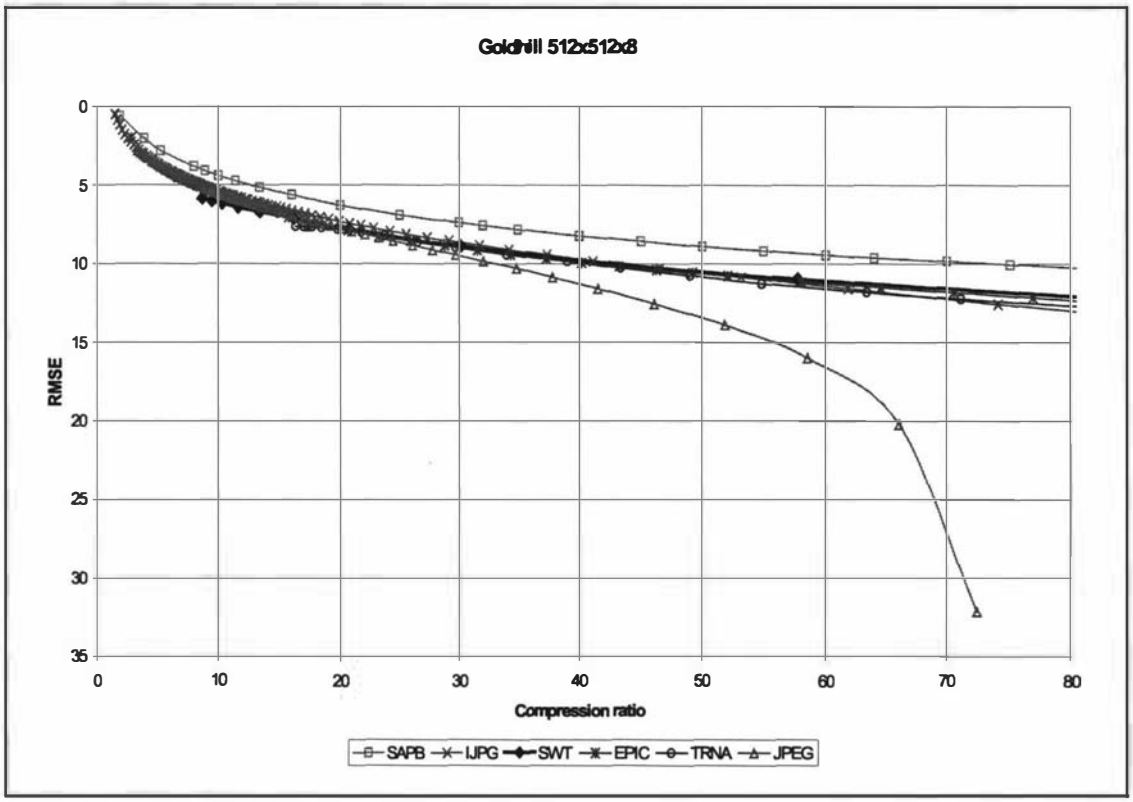


Figure 5.22 – SWT algorithm performance against its peers on Goldhill image

The Goldhill image compresses well with only a slight blurring of texture at 30:1. The fine texture is virtually eliminated in the 60:1 image, yet the main structure of the image is still intact. The SWT based coder is ranked 3rd from the set of 6 image coders tested for this image. There are very few smooth regions in the image so this image does not compress as well as the Lena image at high quality levels. At low quality levels, most of the fine texture has been reduced to smooth regions which then allows the compression ratio to improve.

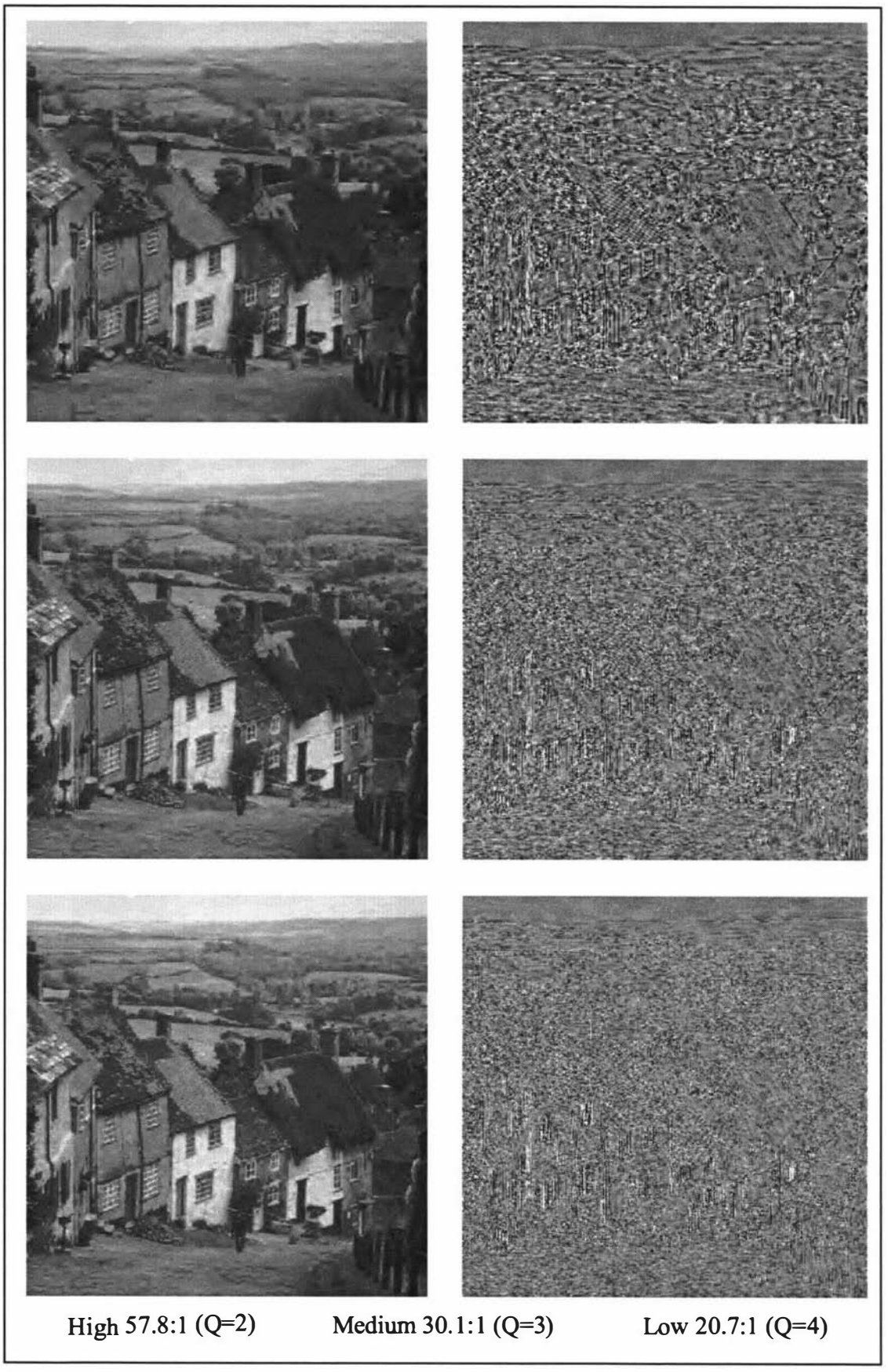


Figure 5.23 – High, medium and low compression on Goldhill image

5.2.4 Discussion of compression performance test

The SWT based algorithm generally performed as well as the middle-ground image coders denoted as "IJPG", "EPIC" and "TRNA". The "SAPB" algorithm consistently performed the best and "JPEG" the worst. The author considers this a good result for the following reasons:

- The other five image coders were all configured to produce the best results achievable given that time and memory were not constrained. This allows the coders to optimise the quantisation and entropy encoding to achieve higher compression ratios. The SWT based coder has no such image quality optimisations as it has been designed to work in "real-time" with a minimal memory footprint. The SWT algorithm implemented in this thesis is, strictly speaking, a reduced complexity (truncated transform), fixed-point implementation of the Symmetric Wavelet Transform.
- The "SAPB" coder utilises arithmetic entropy coding which is guaranteed to produce better compression ratios with no additional loss in image quality. Unlike Huffman coding, patent restrictions apply to the use of arithmetic coding so this form of lossless coding was not used in the SWT based coder. The Lena image compressed at quality level Q=10 produces 40,655 symbols to be entropy encoded. Huffman coding uses 152,224 bits to represent the symbol stream. Arithmetic coding would compress the same symbol stream to its true entropy of 150,970 bits saving 0.8 % on the file size. The compression ratio would increase from 13.70 to 13.89 through the use of arithmetic coding. If adaptive arithmetic coding was used then there would be an additional gain in compression ratio. The arithmetic and adaptive arithmetic algorithms are at least an order of magnitude more computationally complex than Huffman coding. The additional processing required would affect the real-time performance of the algorithm if implemented.
- The "IJPG" coder uses the 8 x 8 pixel DCT as its transform kernel and at medium to high compression ratios the blocking artefacts of this coder are likely to be quite visible. This type of visual artefact is not measured by a simple objective error

metric such as RMSE, yet would be quite noticeable to a human observer in a subjective test.

In summary, the SWT based image compression algorithm described in this thesis has been shown to be competitive with algorithms that appeared in the literature at the same time as the algorithm was first designed. More importantly, the SWT based algorithm implementation is efficient in its processor usage and memory footprint making it suitable for real-time execution. The type of artefacts likely to be seen in compression ratios greater than 30:1 are a blurring of fine textures and slight ringing about hard edges in the image. The annoying block artefacts inherent to JPEG style algorithms are not present in the wavelet based algorithm used in this thesis.

5.3 Watermark algorithm performance

In this section, the effect of adding the Image Authentication Watermark (IAW) to a compressed image for a range of compression ratios is examined. In all cases the same 768 bits of watermark information is added to the image. The 512 x 512 pixel Lena image is used with image quality levels 2 to 10, as defined in section 5.2.1. The IAW is inserted into the compression process during the zerotree construction. The individual watermark data bits are coded by modulating the quantised coefficients prior to run-length coding and Huffman coding. This process alters the effective compression ratio as both the run-length coding and Huffman coding processes use a slightly different set of input data once the IAW has been inserted.

The other side effect of embedding the IAW into the quantised coefficient data stream is that additional quantisation noise is introduced into the decompressed image. When the image is compressed at a low compression ratio, the additional quantisation noise caused by the IAW is invisible to the human observer. As the compression ratio is increased, the additional quantisation noise will become more visible. The additional quantisation noise is measured as an increase in the RMSE metric between the original image and the IAW protected compressed image.

5.3.1 Results of watermark performance test

Table 5.3 shows the results of embedding the IAW into the Lena image for the nine selected image compression quality settings. The $\mathrm{Lena}IAW_{CR}$ column contains the new compression ratio achieved and the $\mathrm{Lena}IAW_{CR} - \mathrm{Lena}_{CR}$ column contains the change in compression ratio after the IAW is added. The unwatermarked Lena_{CR} values can be found in Table 5.2 in section 5.2.3. In all cases the change is positive indicating that the compression ratio has *increased* slightly with the *addition* of the extra IAW embedded data which is a surprising result. This result will be discussed in the next section. The $\mathrm{Lena}IAW_{RMSE}$ column contains the new RMSE error measure between the original Lena image and the compressed and watermarked image. The

LenaIAW_{RMSE} – $Lena_{RMSE}$ column contains the amount the RMSE metric has increased due to the embedding of the watermark data. In all cases this change is positive, which is the expected result as additional quantisation errors have been introduced into the compressed image in order to communicate the IAW.

Quality	LenaIAW_{CR}	LenaIAW_{CR} – $Lena_{CR}$	LenaIAW_{RMSE}	LenaIAW_{RMSE} – $Lena_{RMSE}$
2 (Poor)	65.1	+ 1.61	11.5	+ 2.64
3	41.9	+ 0.69	8.2	+ 1.31
4	31.6	+ 0.33	6.9	+ 0.83
5	25.7	+ 0.21	6.2	+ 0.57
6	21.8	+ 0.14	5.7	+ 0.41
7	19.1	+ 0.09	5.4	+ 0.32
8	16.8	+ 0.10	5.2	+ 0.23
9	15.3	+ 0.08	5.0	+ 0.21
10 (Good)	13.8	+ 0.07	4.9	+ 0.16

Table 5.3 – Change in CR and RMSE after IAW added for 9 quality settings

Figure 5.24 shows that the increase in the RMSE metric due to the embedded IAW has a strong linear relationship with the increase in compression ratio. This is an expected result as the quantiser step size increases in a linear relation with the compression ratio and the embedded IAW alters selected coefficients in the compressed image by ± 1 quantiser step. Figure 5.25 shows the decompressed Lena images containing the embedded IAW for quality levels 2, 4 and 6. The watermark artefacts are clearly present in the smooth regions of the left most image which is compressed at 65.1:1. The error image below also shows the presence of both compression artefacts and watermark artefacts. As the image is compressed with finer quantiser steps in the middle and rightmost images, the watermark artefacts disappear as the amplitude of each artefact decreases. In the case of the Lena image it could be concluded that the IAW can be employed without causing any noticeable visual disturbance for compression ratios up to about 30:1. This corresponds to an increase in the RMSE of approximately 0.8 over the unwatermarked compressed image. At

compression ratios greater than 30:1 both compression artefacts and watermark artefacts will become visible to the human observer.

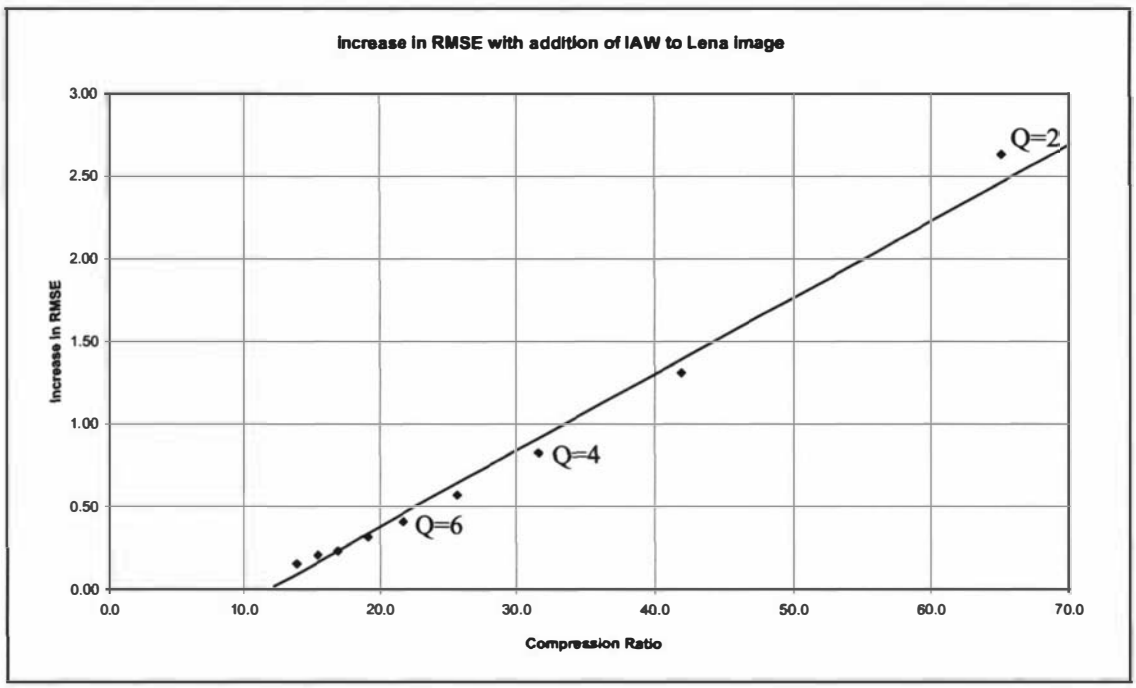


Figure 5.24 – Increase in RMSE error after IAW added to Lena image

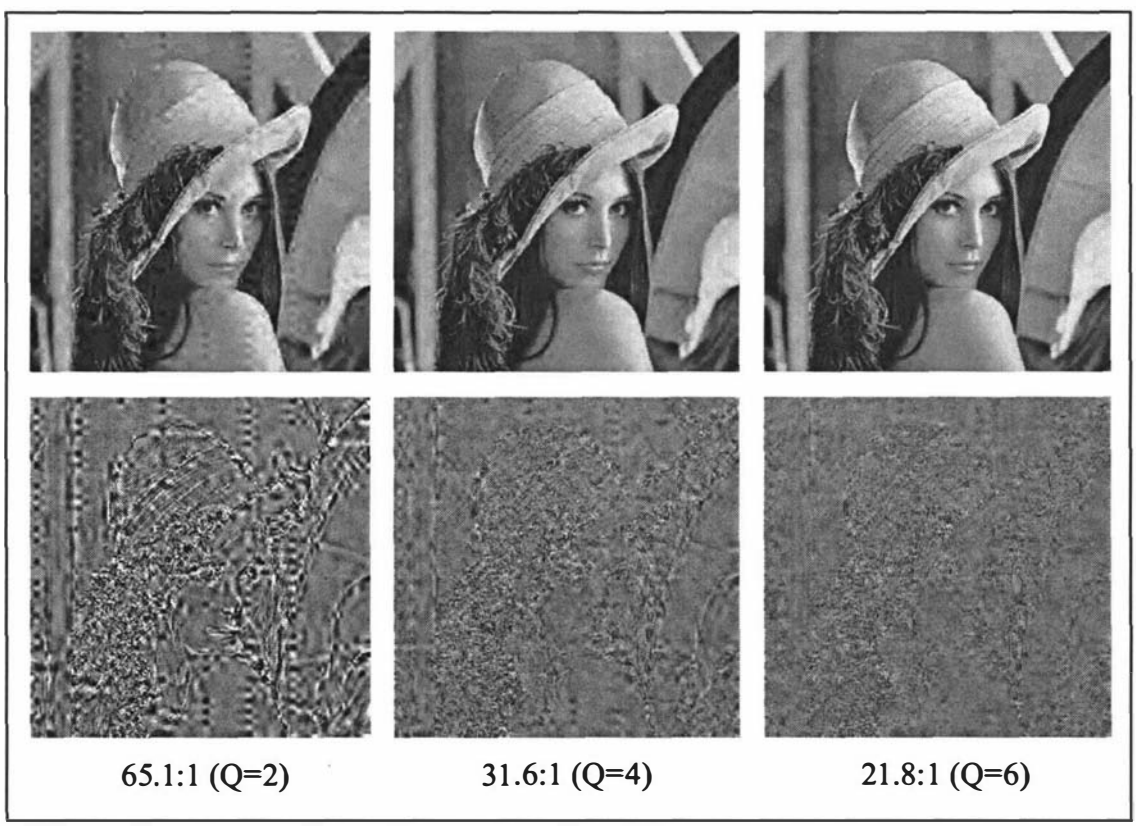


Figure 5.25 – Visibility of the IAW in Lena image at high compression ratios

5.3.2 Discussion of watermark performance test

The IAW algorithm performs co-operatively with the SWT based image compression algorithm and this fact is reflected in the RMSE error metric results. As the image compression ratio is increased, the visibility of both compression artefacts as well as watermark artefacts increases proportional to the changing quantisation step size. In the Lena image, the IAW is only visible in the smooth image regions of the greater than 30:1 compressed images. The addition of the watermark to textured regions is well hidden, as the induced quantisation error is less noticeable in these regions.

One unexpected result was that the compression ratio increases slightly with the addition of the IAW. This result is surprising as the addition of the watermark theoretically increases the information content of the compressed data stream as the data now represents the image (in a lossy format) *and* 768 bits of losslessly encoded watermark data. There has been no change in the quantiser step size, yet the compressed data stream is significantly smaller *after* the IAW has been added.

The actual size of the compressed Lena image file for the poor quality setting "Q=2" is 4,129 bytes when the IAW is *not* added ($\text{Lena}_{CR} = 63.5$). After adding the IAW, the new image file size is 4,027 bytes ($\text{Lena}IAW_{CR} = 65.1$) which is 102 bytes smaller than first file size. To summarise, the watermarked image contains 96 bytes (768 bits) of losslessly encoded watermark data, but is 102 bytes smaller than the unwatermarked image. A similar calculation can be performed on the good quality setting "Q=10". In this case the image file decreases in size by 97 bytes with the addition of the watermark from an original size of 19,093 bytes.

To explain why the compression ratio increases is not simple. The number of symbols to be Huffman encoded decreases by about 200 in these images after the watermark is added. The most likely reason is that for every bit of the watermark data, the IAW embedding algorithm targets up to four of the largest valued coefficients, in order to perturb them by either ± 1 quantiser step. In many of the image regions where there are few edges, the corresponding zerotrees are nearly empty with only a few

coefficients that are either +1 or –1. By perturbing these coefficients, many of them are eliminated and this favours the run-length coding compression step that follows. Up to two symbols can be saved per zerotree each time a coefficient is eliminated due to it being perturbed to 0. The net effect of the embedding of the 768 bits of IAW data is that the file size is likely to decrease slightly and the RMSE will increase due to the additional quantisation error involved.

The visibility of the IAW at moderate to high compression ratios over 30:1 could be argued as being a *desirable* feature of the algorithm. Just as the tamper proof metallic strip and *real* watermark are quite visible in paper currency, the visibility of the IAW artefacts subtly could communicate to the casual observer that the image may be protected in some way. The most noticeably artefacts occur in the smooth regions of the image where there is generally no visual information of interest to the human observer anyway.

In summary, the IAW image watermarking algorithm is capable of embedding a useful amount of losslessly encoded data into already highly compressed images. The visibility of the watermark is controlled by the compression ratio selected as the compression algorithm and watermarking algorithm work co-operatively. The watermark also provides tamper resistance to the compressed image file as alterations to the compressed image file without complete knowledge of the watermarking and compression algorithms is likely to cause decoding errors when the embedded watermark data is recovered.

5.4 Discussion

This chapter has examined the performance of the complete remote activity monitoring system described in this thesis. Several different aspects of performance have been considered. The timing of each of the processes involved between the capture of the image at the smart digital camera to the display of the image at the Workstation PC have been analysed. The performance of the SWT based compression algorithm has been compared with other first-generation techniques with regard to the quality of the image for a range of compression ratios. The Image Authentication Watermark (IAW) was applied at various compression ratios to determine the visibility of the watermark in relation to the compression ratio used.

All of the research and development work described in this thesis was implemented on the prototype system described in this chapter. In addition the author developed two Microsoft Windows based applications. These applications have powerful graphical user interfaces that give control of the capture and compression processes on the smart digital camera. The image compression, motion detection and watermarking software written by the author, has been directly incorporated into the production version of the smart digital camera described in this chapter. The user interface has been significantly changed in the production system. The production unit is quite different in appearance from the prototype camera and is less than quarter the size.

The SWT based compression algorithm has been shown to be competitive with other first-generation techniques and to be consistently better than the JPEG standard for image compression. The medium target compression ratio of 30:1 produces images that contain only minor compression artefacts and are compact enough to be transmitted by the system at the rate of 2 frames per second. The IAW watermarking algorithm has also been shown to create only minor artefacts in the compressed image for compression ratios greater than about 30:1. Most of the visible artefacts lie within otherwise smooth regions of the image where critical image detail is not present. The slight visibility of the watermark could communicate to the casual technical observer that the image contains some form of protection and thus could deter attempts to

tamper the digital image. The addition of 768 bits of watermark data to the already compressed 512 x 512 pixel image has been shown to slightly *decrease* the compressed image size at the expense of an *increase* in RMSE error. This unexpected result has been shown to be explained by consideration of the run-length coding that follows the embedding of the image watermark data.

5.5 References

[5.1] **Photobit,** *CIF-size Sensors* [Web Page]. 2000. Available at: http://www.photobit.com/Products/CIF-size_Sensors/cif-size_sensors.htm.

[5.2] **Kominek, J.,** *Waterloo BragZone* [Web Page]. 1995. Available at: http://links.uwaterloo.ca/BragZone/bragzone.readme.txt.

[5.3] **Said, A. and Pearlman, W. A.,** *"A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees,"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp.243-249, 1996.

[5.4] **Said, A. and Pearlman, W. A.,** *SPIHT - Image Compression with Set Partitioning in Hierarchical Trees* [Web Page]. 1996. Available at: http://ipl.rpi.edu/SPIHT/.

[5.5] **Said, A. and Pearlman, W. A.,** *SAPB progressive coder* [Web Page]. 1995. Available at: ftp://ipl.rpi.edu/pub/EW_Code/codetree.tar.z.

[5.6] **Fisher, Y.,** *Fractal Image Compression: Theory and Application* [Web Page]. 1995. Available at: ftp://links.uwaterloo.ca/pub/Fractals/Programs/Fisher/.

[5.7] **MIT Media Labs,** *Efficient Pyramid Coder* [Web Page]. 1995. Available at: ftp://whitechapel.media.mit/pub/epic.tar.z.

[5.8] **Independent JPEG Group,** *Free JPEG C source code* [Web Page]. 1995. Available at: ftp://ftp.uu.net.graphics/jpeg/jpegsrc.v5.tar.z.

[5.9] **Iterated Systems,** *Image Incorporated Commercial JPEG coder,* rel. 1995.

[5.10] **Jain, A. K.,** *Fundamentals of Digital Image Processing,* 1989. Prentice-Hall.

6

Discussion and Conclusions

The discussion and conclusion sections of this final chapter aim to integrate the themes of chapters 2 through to 5. These chapters demonstrate that DSP technology is an appropriate and effective base for implementing image processing algorithms to produce a remote activity monitoring system.

6.1 Thesis discussion

Each of the following sub-sections summarise the findings of the main chapters in this thesis. Chapter 2 is discussed in section 6.1.1 and covers the reason why low bit-rate coding is required and how the Symmetric Wavelet Transform (SWT) based algorithm was selected for the system. Section 6.1.2 summarises chapter 3 and shows how the various components of the image compression algorithm were mapped onto the architecture of the DSP for an efficient implementation. The material from chapter 4 on the motion detection and image watermarking algorithms that work co-operatively with the image compression process, is reviewed in section 6.1.3. Finally section 6.1.4 discusses the findings of chapter 5 which analyses the performance of both the smart digital camera hardware and the algorithms that are run on the camera.

6.1.1 Low bit-rate image coding

Central to a remote activity monitoring system is the mechanism by which remote digital images are conveyed to the human system operators. The system developed in this thesis is designed with real-world constraints that clearly dictate that some form of digital image compression is required. The effective bandwidth available to each camera is set to be no more than 100 kbps. Typical digital imaging requires a horizontal resolution of at least 512 pixels, this means compression ratios exceeding 30:1 are required to achieve a desired transmission rate of two frames per second. Chapter 2 provides foundational material for techniques used to compress images at high compression ratios.

To achieve image compression at high ratios requires the use of low bit-rate image coding techniques. These techniques require some loss in image quality to achieve the low bit-rate. The image compression techniques examined in chapter 3 are known as first-generation techniques because the coder does not adapt to local image statistics (second-generation), or analyse the image for objects in the scene (third-generation). The image compression is achieved by exploiting data redundancy in the two-dimensional structure of pixel values that make up a single image. The advantage of using a first-generation technique is that it is well suited to real-time implementation. Another advantage is that the execution time of the algorithm is predictable over a wide variety of images. Even in the year 2000, four years after the design of the prototype system began; the newest commercial image coding standards, such as JPEG 2000, still utilise first-generation techniques. Some of these techniques include wavelet based algorithms similar to that used in this thesis. Practical second-generation image compression techniques based on scene analysis, that are able to be implemented in real-time, are not likely to be available for some time.

Chapter 2 concluded that a suitable low bit-rate image compression technique could be based on the discrete Symmetric Wavelet Transform (SWT). Digital Signal Processors (DSPs) excel at efficient implementation of filters and the SWT is entirely based on the Quadrature Mirror Filter (QMF). The transform stage of the image

compression algorithm is important as it influences the type of compression artefacts visible when using high compression ratios. The global wavelet transform does not suffer from the blocking artefact that plagues the standard JPEG technique when it is used for compression ratios greater than about 16:1. The blocking artefact is caused at the transform stage of the JPEG algorithm that is based on the 8 x 8 pixel Discrete Cosine Transform (DCT).

The "back-end" of the image coder consists of the post-transform stages that are responsible for discarding the redundant data and minimally encoding the remaining data to achieve the overall lossy image compression. The construction of zerotrees followed by Sequential Baseline Coding (SBC) was chosen as a suitable combination of existing techniques. These techniques were chosen to achieve the high compression ratios required without too much computational overhead.

6.1.2 Efficient implementation on a DSP

Data Flow Graphs (DFGs) are a useful way of describing a signal processing algorithm and section 3.1 of chapter 3 details the entire SWT based image compression algorithm as a set of connected and nested DFGs. Chapter 3 also introduces the Motorola 56303 DSP and section 3.2 describes the internal architecture of the DSP. It is important to fully understand the DSP architecture design as it is a specialised processor. Once the design is understood, it is possible to write short sequences of assembly language instructions (a "kernel") that enable efficient processing. The kernel usually processes only a small number of data points in the image at a time and is then repeated thousands of times in order to cover the entire image. A well written DSP kernel will perform two or three data processing operations for each elapsed clock cycle. For this reason although the processor may be running at a clock speed of 80 MHz, 160 to 240 million "operations" will be occurring each second.

Sections 3.3 and 3.4 of chapter 3 detail how the various stages of the SWT based image compression algorithm are mapped from the DFGs to the DSP architecture.

Where specific features of the DSP architecture are used to obtain a particularly efficient implementation of the algorithm, these kernel implementations have been detailed and explained. Specifically the following architectural features of the DSP are significant:

- The dual accumulators and the powerful "`mac`" op-code are used to efficiently calculate the QMF at the heart of the wavelet decomposition. The two on-chip data spaces (`X:` and `Y:`) are used to provide sufficient bandwidth to the filter calculation by storing the coefficients in the `Y:` space and the pixel data in the `X:` space.
- Three DMA channels are used to efficiently marshal image data and coefficients to and from the DSP during the wavelet decomposition. Five of the six on-chip DMA channels are used during the zerotree construction algorithm. The automated transfer of two-dimensional structures is supported by the DMA controller which is essential for working efficiently with image data.
- Advanced indirect addressing modes are used to efficiently scan the zerotree structure for run-length coding.
- Bit-reversed addressing is used to create the bit patterns for the Huffman codes. The "`insert`" and "`merge`" op-codes and the 56-bit wide accumulators are used to efficiently concatenate the Huffman codes to form the compressed bitstream.

A new way to compactly transmit the optimal Huffman code table for an arbitrary bitstream is detailed in section 3.3.3 and in the ICIP-98 conference paper reprinted in the appendix. This original algorithm allows the optimal Huffman code table to be transmitted with the image with a low and reduced overhead.

Section 3.5 of chapter 3 compares the Intel Pentium-I (MMX) architecture with that of the Motorola DSP. The SWT based algorithm is used to compare the performance of the two architectures as the compression and decompression components of the algorithm are symmetric in complexity. This comparison reveals that when the clock cycle difference is accounted for, the DSP is six times more efficient than the MMX accelerated Pentium-I processor.

Finally, section 3.6 of chapter 3 makes a comparison between a block based DCT algorithm and the global wavelet based algorithm. This comparison shows that the time to compute the algorithms, when efficiently implemented on the DSP, is nearly identical when the maximum achievable frame rates are compared. The main difference between the two algorithms is in the image quality and compression ratio achieved, with the wavelet based algorithm showing superior performance. The main disadvantage of the wavelet based algorithm is the large amount of temporary storage required to compute the global wavelet transform.

Chapter 3 concludes that an efficient DSP implementation of the wavelet based image compression algorithm has been achieved with the Motorola 56303 DSP. The SWT based algorithm is concluded to be a good choice for a real-time low bit-rate image compression system for two main reasons:

- It is computational simple and predictable.
- It provides a significant compression ratio and quality improvement over the well-known JPEG technique.

Both of these factors are important when the image compression algorithm is to be implemented on a real-time embedded DSP system. Embedded systems typically do not have a lot of memory and the fast access memory required for a real-time system usually costs as much as the processor itself. For a cost-sensitive application, the amount of memory and the speed of the processor need to be minimised.

6.1.3 Motion detection and image watermarking

Chapter 4 looked at image processing algorithms that work co-operatively with the wavelet based image compression system. These algorithms add value to the smart digital camera as they offer motion detection and image authentication services with very little computational overhead. The ability to detect motion within defined areas in the camera field of view enables the smart digital camera to be configured to raise an alarm when the amount of motion exceeds a threshold. This alarm can be

configured to initiate an archive sequence consisting of a set of images before and after the motion event.

The image wavelet decomposition computed by the DSP provides a powerful way to analyse an image. Both the motion detection and the Image Authentication Watermark (IAW) are computed in the wavelet domain rather than the "raw" pixel domain. In both cases the wavelet decomposition reduces the amount of data that needs to be processed in order to evaluate the amount of interframe motion or insert the IAW. In a conventional Closed Circuit Television (CCTV) system, analogue cameras are used in the remote locations. The images from the cameras are digitised and processed at a central location by expensive Automate Video Surveillance (AVS) systems. By locating the processing power within the smart digital camera, decisions whether or not to send an image sequence can be made directly by the camera. This processing architecture eliminates the need for a centrally located processing resource powerful enough to process, in real-time, all the images from all the cameras of a system.

The IAW algorithm described in section 4.2 of chapter 4 is an original design by the author. The algorithm encodes a uniform number of data bits per image area. This unfortunately has the consequence that in smooth regions of the image the perturbations of the coefficients become noticeable especially at high compression ratios. Where the image region contains edges or texture, the perturbation of the coefficients is much less noticeable. The IAW algorithm serves two purposes, firstly to embed a data stream throughout the image and secondly to provide a fragile tamper detection mechanism. Together these two facets of the IAW algorithm are used to authenticate an image as being free from tampering as well as verifying the time and identity of the camera that took the image.

6.1.4 Performance of complete system

Section 5.1 of chapter 5 analyses the performance of the complete remote activity monitoring system and describes how the prototype camera has evolved into a

production smart digital camera that is part of an integrated security system. The core software developed by the author to efficiently compress images on the Motorola 56303 DSP has been used in the production system without any changes. The image decompression software, also developed by the author, is used in the production system with some user interface changes to ensure that the camera integrates well into the security system product.

The individual processes that form a pipeline to convey the image from the camera to the Workstation PC are listed and profiled to identify bottlenecks and the latency in both the prototype and production systems. Three years ago in 1997, the bottlenecks were both the communication network and the speed of a 200 MHz Intel Pentium-I (MMX) based PC used to decompress an image. With modern PCs such as the 500 MHz Intel Pentium-III used to decompress an image, the bottleneck is now only the communication network that limits the bandwidth to under 100 kbps per camera.

Section 5.2 of chapter 5 compares the performance of the SWT based image compression algorithm with other first-generation algorithms described in the literature. The SWT based algorithm competes well with existing algorithms and is much better than the JPEG standard for image compression. Algorithm performance graphs of Root Mean Square Error (RMSE) verses compression ratio are shown for three widely used test images. These performance graphs allow a direct comparison to be made between the different image compression algorithms. The compressed images and their corresponding error images reveal the location and type of compression artefacts produced by the lossy SWT based algorithm.

In section 5.3 of chapter 5, the effect of adding the IAW watermark algorithm to the same image at a range of compression ratios is examined. This is done to determine the change in compression ratio and RMSE error introduced when the IAW watermark is added. In all cases the compression ratio is slightly improved and the RMSE error increases in proportion to the quantiser step size which is closely related to the compression ratio. The artefacts from both the compression algorithm and the IAW watermark process are well hidden for compression ratios up to about 30:1. At higher compression ratios, the image lacks high frequency texture and detail. There

are also ringing artefacts around any sharp edges in the image. The IAW watermark also becomes more visible in smooth regions of the image as the compression ratio increases past 30:1.

6.2 Thesis conclusions

The remote activity monitoring system described in this thesis succeeds in taking the first steps towards the embedding of intelligent decision making abilities into a digital camera. The Motorola 56303 DSP has been shown to contain a number of significant architectural features that make it well suited to the task of image compression even though the processor was primarily designed for audio signal processing. The DSP is not powerful enough to process video at television frame rates but is more than capable of meeting the two to four frames per second processing rate required for a surveillance application.

The mental drudgery associated with the use of the remote activity monitoring system is likely to be alleviated by the motion detection capabilities of the smart digital camera. This capability enables the camera to automatically evaluate motion within the scene and then alert the human operator that there is something worth observing. The smart digital camera internally maintains a circular buffer of the last 100 images so that it can provide an archival sequence of images before and after an alarm. This short image sequence is then recorded to disk on the Server PC, an important feature known as alarm verification. An operator can then quickly review the short sequence of recorded images to determine the likely cause of the alarm. This allows the operator to either dismiss the alarm as false or enables the operator to take an appropriate action. The short recorded and watermarked image sequence can be used as evidence to detail the cause of the alarm.

The addition of an Image Authentication Watermark (IAW) is important if images are to be used as evidence that an event took place. The ease, with which digital images can be edited, requires that safeguards be put in place to deter and detect any alteration to an image. Both the motion detection and IAW watermark processes have been

designed to work co-operatively with the Symmetric Wavelet Transform (SWT) based image compression algorithm. The co-operative nature of this arrangement has the following advantages:

- The algorithms use the wavelet decomposition of the image to efficiently perform the designated tasks. The wavelet decomposition is already computed for the image compression algorithm. By making use of the decomposition, the watermark can be embedded discretely and the motion detection algorithm can be made robust in the presence of noise in the image.
- Simple algorithms for motion detection and watermarking can be used effectively in the wavelet domain reducing the overall computational complexity of the algorithm implementations. Low computational complexity is important in a cost-sensitive, real-time embedded system.

The original algorithms developed in the course of this study are:

1. A compact Huffman table coding scheme to suit the low computation and low bit-rate requirements of modern embedded compression engines.
2. An Image Authentication Watermark algorithm designed to embed a self-referencing watermark into the compressed symbol stream within an image coding algorithm.

In conclusion, this thesis has shown that a modern DSP architecture can provide a cost-effective platform for implementing a real-time remote activity monitoring system. It is important to carefully map each of the components of the image processing algorithm onto the most appropriate features of the DSP architecture. A second consideration is to ensure that the data can be marshalled to and from external memory at a rate fast enough to ensure the DSP core does not stall. An efficient algorithm implementation has resulted from appropriate algorithm mapping and data marshalling. This in turn has enabled the author to implement software for a powerful and cost effective smart digital camera. This camera is well suited to the task of industrial remote activity monitoring.

6.3 Future work

The production camera has been designed to be software upgradable whilst it is installed at a client's premises. This potentially future proofs the smart digital camera for at least a couple of years. The processing power of the camera is not simply upgradable so any future image compression algorithm would need to work within the processor and memory constraints of the existing hardware. The following list of potential enhancements could be considered in the future:

1. There is potential in using adaptive components in the quantisation process of the image compression algorithm. Finer quantisation could be used around human faces if they could be reliably and efficiently detected. This enhancement could improve the quality of the images without reducing the overall compression ratio. The IAW watermark algorithm would potentially work more discretely if a non-uniform distribution of watermark data bits were embedded throughout the image using an adaptive algorithm. This would reduce the watermark artefacts currently seen in smooth image regions.

2. In the current system the motion detection algorithm is very simple and only uses the low resolution approximation image from the wavelet decomposition. It would be possible to use other components of the wavelet decomposition to detect motion from the edge information in the image. It may also be possible to discriminate between different types of motion in a complex image and only trigger alarms on certain types of motion.

3. The current SWT based image compression algorithm produces a sequence of standalone still images. A future algorithm could use interframe differencing with motion compensation similar to MPEG image sequence compression [6.1]. This style of image sequence compression was not considered appropriate for low frame rate compression as objects in the scene can move considerable distances in half a second. The large distances involved make it difficult to track

objects from frame to frame and the amount of computation required to search for related objects in the previous frame is considerable.

6.4 References

[6.1] **Fogg, C.,** *MPEG-2 FAQ v3.7* [Web Page]. 1995. Available at: http://www-plateau.cs.berkeley.edu/mpegfaq/MPEG-2-FAQ.html.

Bibliography

Alexander Turnbull Library, *Sample image from the National Library of New Zealand* [Web Page]. 1998. Available at: http://timeframes.natlib.govt.nz/.

Analog-Devices, *Low Cost Multiformat Video Codec - ADV601 Datasheet* [Web Page]. 1997. Available at: http://www.analog.com/pdf/2011_0.pdf.

Antonini, M.; Barlaud, M.; Mathieu, P., and Daubechies, I., *"Image Coding Using Wavelet Transform,"* IEEE Transactions on Image Processing, vol. 1, no. 2, pp.205-220, 1992.

Barreto, C. S. and Mendonça, G. V., *"Enhanced Zerotree Wavelet Transform Image Coding Exploiting Similarities Inside Subbands,"* Proceedings 1996 IEEE International Conference on Image Processing, Lausanne, Switzerland, pp.549-551, 1996.

Barthel, K. U., *"A new image coding technique unifying Fractal and transform coding,"* ICIP-94, pp.112-116, 1994.

Bates, R. H. T. and McDonnell, M. J., *Image Restoration and Reconstruction,* Ed. 1, 1986. Oxford.

Blalock, G., *The BDTImark: A Measure of DSP Execution Speed* [Web Page]. 1997. Available at: http://www.bdti.com.

Body, N. B., *"Digital Camera for Low Bit-Rate Transmission of Slow Frame-Rate Video,"* Proceedings of the Third NZ Conference of Postgraduate Students in Engineering and Technology, University of Canterbury, Christchurch, New Zealand, pp.467-468, 1996. ISBN 0-473-03889-7.

Body, N. B.; Page, W. H., and Hodgson, R. M., *"Application of the Wavelet Packet Transform to Very Low Bit Rate Coding of Images,"* The NZ Communication Research Workshop 1997, Wellington Town Hall, Wellington, New Zealand, 1997.

Body, N. B.; Page, W. H.; Khan, J. Y., and Hodgson, R. M., *"Efficient Mapping of Image Compression Algorithms on a Modern Digital Signal Processor,"* Proceedings of the 4th Annual New Zealand Engineering and Technology Postgraduate Students Conference, University of Waikato, New Zealand, pp.59-64, 1997. ISBN 0-473-04578-8.

Body, N. B.; Page, W. H.; Khan, J. Y., and Hodgson, R. M., *"Efficient Digital Signal Processor Implementation of a Wavelet Transform Based Image Compression Algorithm,"* Proceedings of Digital Image and Vision Computing: Techniques and Applications (DICTA/IVCNZ-97), Albany Campus, Massey University, Auckland, New Zealand, pp.71-76, 10-12 December, 1997. ISBN 0-473-04947-3.

Body, N. B.; Page, W. H.; Khan, J. Y.; Hodgson, R. M., and Collins, F. A. H., *"Efficient Wavelet Image Coding on a Digital Signal Processor Based Digital Camera,"* Proceedings of International Conference on Signal Processing Applications & Technology (ICSPAT-98), Toronto, Canada, pp.782-786, 13-16 September, 1998.

Body, N. B. and Bailey, D. G., *"Efficient Representation and Decoding of Static Huffman Code Tables in a Very Low Bit Rate Environment,"* Proceedings of IEEE International Conference on Image Processing (ICIP-98), Chicago, USA, pp.90-94, vol. 3, 4-7 October, 1998. ISBN 0-8186-8821-1.

Body, N. B.; Page, W. H., and Hodgson, R. M., *"An Image Authentication Watermark for Wavelet Compressed Digital Images,"* Proceedings of Image and Vision Computing '99 New Zealand (IVCNZ-99), Canterbury University, Christchurch, New Zealand, August 30-31, 1999. ISBN 0-478-09333-0.

Boland, F. M., *"Watermarking digital images for copyright protection,"* Image Processing and its Applications, pp.326-330, 1995.

Bracewell, R. N., *The Fourier Transform and its Applications,* 1986. McGraw-Hill.

Brislawn, C., *WSQ Gray-scale Fingerprint Image Compression Specification* [Web Page]. 1993. Available at: http://www.c3.lanl.gov/~brislawn

Brislawn, C., *How Wavelet/Scalar Quantisation Works* [Web Page]. 1996. Available at:
http://www.c3.lanl.gov/~brislawn/FBI/OverCompress/HowItWorks/howitworks.html.

Brown, C. W., *Graphics File Formats: reference and guide,* 1995. Manning Publications.

CCIR, *"Method for the Subjective Assessment of the Quality of Television Pictures,"* Rec. 500-1, 1978.

Chiariglione, L., *"The development of an integrated audiovisual coding standard: MPEG,"* Proceedings of the IEEE, vol. 83, February, pp.151-157, 1995.

Chudy, P., *Handcuff digital thieves*; Byte, pp.5-8, April 1996.

da Silva, E. A. B. and Ghanbari, M., *"On the Performance of Linear Phase Wavelet Transforms in Low Bit-Rate Image Coding.,"* IEEE Transactions on Image Processing, vol. 5, no. 5, pp.689-704, 1996.

El-Sakka, M. R. and Kamel, M. S., *"Adaptive Image Compression Based on Segmentation and Block Classification,"* ICIP, Chicago, USA, p.TP3.01, 1998.

Eyre, J. and Bier, J., *DSP Processors Hit the Mainstream*; Computer, pp.51-59, August 1998.

Fisher, Y., *Fractal Image Compression: Theory and Application* [Web Page]. 1995. Available at: ftp://links.uwaterloo.ca/pub/Fractals/Programs/Fisher/.

Fogg, C., *MPEG-2 FAQ v3.7* [Web Page]. 1995. Available at: http://www-plateau.cs.berkeley.edu/mpegfaq/MPEG-2-FAQ.html.

Ghafourian, M. A., *"Comparison between several adaptive search vector quantisation schemes and JPEG standard for image compression,"* IEEE Transactions on Communications, vol. 43, February-April, pp.1308-1312, 1995.

Girod, B. and Stuhlmuller, K. W., *"A Content-Dependant Fast DCT for Low Bit-Rate Video Coding,"* ICIP, Chicago, USA, p.WA3.01, 1998.

Gwennap, L., *Microprocessor Report - Intel's MMX Speeds Multimedia* [Web Page]. 1996 Mar 5. Available at: http://www.chipanalyst.com/report/articles/mmx/mmx.html.

Halfhill, T. R., *AMD K6 takes on Intel P6*; Byte, pp.67-72, January 1996.

Huffman, D. A., *"A method for the construction of minimum-redundancy codes,"* Proceedings of the IRE, vol. 40, no. 9, pp.1098-1101, 1952.

Independent JPEG Group, *Free JPEG C source code* [Web Page]. 1995. Available at: ftp://ftp.uu.net.graphics/jpeg/jpegsrc.v5.tar.z.

Intel, *Intel Signal Processing Library Reference Manual,* 1997. Intel Corporation.

Intel, *FIR and IIR Filtering using Streaming SIMD Extensions,* Ed. 1.1, 1999. Intel Corporation.

Iterated Systems, *Image Incorporated Commercial JPEG coder,* rel. 1995.

Jähne, Bernd, *Digital Image Processing: concepts, algorithms and scientific applications,* 1991. Springer-Verlag Berlin. Heidleberg.

Jain, A. K., *Fundamentals of Digital Image Processing,* 1989. Prentice-Hall.

Jayant, N., *"Signal compression based on models of human perception,"* Proceedings of the IEEE, vol. 81, October, pp.1383-1422, 1993.

Johnson, Neil F, *Exploring Steganography: seeing the unseen*; IEEE Computer, pp.26-34, February 1998.

JPEG, *JPEG2000 Committee Drafts* [Web Page]. 2000. Available at: http://www.jpeg.org/CDs15444.htm.

Kim, J. S., *"A fast feature-based block matching algorithm using integral projections,"* IEEE Journal on Selected Areas of Communications, vol. 10, June, pp.968-971, 1992.

Kominek, J., *Waterloo BragZone* [Web Page]. 1995. Available at: http://links.uwaterloo.ca/BragZone/bragzone.readme.txt.

Kunt, M., *Object based image and image sequence coding,* 1990. Elsevier Science.

Lewis, A. S. and Knowles, G., *"Video Compression Using 3D Wavelet Transforms,"* Electronics Letters, vol. 26, no. 6, pp.396-398, 1990.

Lewis, A. S. and Knowles, G., *"Image Compression Using the 2-D Wavelet Transform,"* IEEE Transactions on Image Processing, vol. 1, no. 2, pp.244-250, 1992.

Lin, D. L., *"Video on phone lines: technology and applications,"* Proceedings of the IEEE, vol. 83, February, pp.175-193, 1995.

Lookabaugh, T., *"Variable rate vector quantisation for speech, image, and video compression,"* IEEE Transactions on Communications, vol. 41, January, pp.186-199, 1993.

Lowe, D. and Ginige, A., *"Image Quality Assessment using an Image Activity Weighting and the HVS Response,"* Image and Vision Computing New Zealand, New Zealand, pp.169-176, 1993.

Martucci, S. A. and Sodagar, I., *"Zerotree Entropy Coding of Wavelet Coefficients for Very Low Bit Rate Video,"* Proceedings 1996 IEEE International Conference on Image Processing, Lausanne, Switzerland, pp.533-536, 1996.

Martucci, S. A.; Sodagar, I.; Chiang, T., and Zhang, Y. Q., *"A Zerotree Wavelet Video Coder,"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1, pp.109-118, 1997.

McGoldrick, P., *Video-Compression Chip is the First to use Wavelets*; Electronic Design, pp.150-152, August 1996.

McLeod, A., *"The impact and effectiveness of low-cost automated video surveillance systems,"* IEEE International Carnahan Conference on Security Technology, Lexington, Kentucky, USA, pp.204-211, 1996.

Meyer, M.; Hotter, M., and Ohmacht, T., *"A New System for Video-Based Detection of Moving Objects and its Integration into Digital Networks,"* IEEE International Camahan Conference on Security Technology, Lexington, Kentucky, USA, pp.105-110, 1996.

Misiti, M.; Misiti, Y.; Oppenheim, G., and Poggi, J. M., *Wavelet Toolbox User's Guide,* 1996. The MathWorks, Inc.

MIT Media Labs, *Efficient Pyramid Coder* [Web Page]. 1995. Available at: ftp://whitechapel.media.mit/pub/epic.tar.z.

Motorola, *DSP56300 24-bit Digital Signal Processor Family Manual,* 1994. Motorola.

Moura, J. M. F., *"Video over wireless,"* IEEE Personal Communications, February, pp.44-54, 1996.

Nelson, M. and Gailly, J., *The Data Compression Book,* Ed. 2, 1996. M&T Books, ISBN 1558514341.

Nguyen, H., *"Concentric-shell partition vector quantisation with application to image coding,"* IEEE Transactions on Communications, vol. 42, February-April, pp.1911-1918, 1994.

Oehler, K., *"Digital video compression,"* TI Technical Journal, March-April, pp.27-38, 1996.

Pearson, D. E., *"Developments in model-based video coding,"* Proceedings of the IEEE, vol. 83, June, pp.892-905, 1995.

Photobit, *CIF-size Sensors* [Web Page]. 2000. Available at: http://www.photobit.com/Products/CIF-size_Sensors/cif-size_sensors.htm.

Prestage, Michael, *Tell-tale pictures can be candid about cameras*; The Times, Interface: p.3, January 15, 1997.

Qiu, G., *"Image coding based on visual vector quantisation,"* Image Processing and its Applications, pp.301-305, 1995.

Ramchandran, K., *"Wavelets, subband coding, and best bases,"* Proceedings of the IEEE, vol. 84, April, pp.541-560, 1996.

Said, A. and Pearlman, W. A., *SAPB progressive coder* [Web Page]. 1995. Available at: ftp://ipl.rpi.edu/pub/EW_Code/codetree.tar.z.

Said, A. and Pearlman, W. A., *"A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees,"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp.243-249, 1996.

Said, A. and Pearlman, W. A., *SPIHT - Image Compression with Set Partitioning in Hierarchical Trees* [Web Page]. 1996. Available at: http://ipl.rpi.edu/SPIHT/.

Salembier, P., *"Region-based video coding using mathematical morphology,"* Proceedings of the IEEE, vol. 83, June, pp.843-857, 1995.

Shapiro, J. M., *"Embedded Image Coding Using Zerotrees of Wavelet Coefficients,"* IEEE Transactions on Signal Processing, vol. 41, no. 12, pp.3445-3462, 1993.

Simon, B., *How Lossy Compression Shrinks Files*; PC Magazine, pp.371-382, July 1993.

Strang, G. and Nguyen, T., *Wavelets and Filter Banks,* 1996. Wellesley-Cambridge Press.

Weeks, A. R., *Fundamentals of Electronic Image Processing,* Ed. 1, 1996. IEEE Press.

Witten, I. H.; Radford, M. N., and Cleary, J. G., *"Arithmetic Coding for Data Compression,"* Communications of the ACM, vol. 30, no. 6, pp.520-540, 1987.

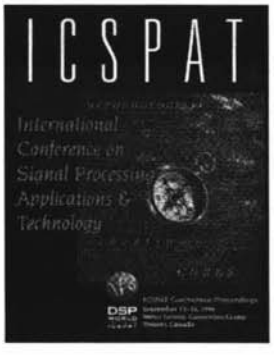
Wright, A., *Squeezing into the picture*; Electronics World + Wireless World, pp.663-666, August 1992.

Zeng, Wenjun, *"A Statistical Watermark Detection Technique Without Using Original Images for Resolving Rightful Ownerships of Digital Images,"* IEEE Transactions on Image Processing, vol. 8, no. 11, pp.1534-1548, 1999.

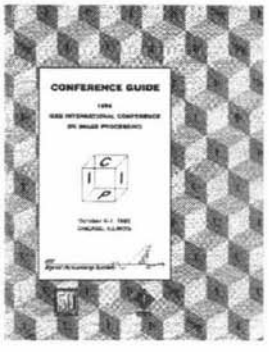
Zhaoyang, L., *Application of Delaunay triangulation in image coding,* 1990. World Scientific.

Appendix

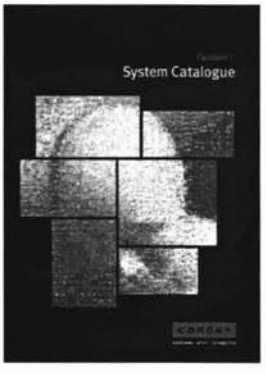
Body, N. B.; Page, W. H.; Khan, J. Y., and Hodgson, R. M., *"Efficient Digital Signal Processor Implementation of a Wavelet Transform Based Image Compression Algorithm,"* Proceedings of Digital Image and Vision Computing: Techniques and Applications (DICTA/IVCNZ-97), Albany Campus, Massey University, Auckland, New Zealand, pp.71-76, 10-12 December, 1997. ISBN 0-473-04947-3.



Body, N. B.; Page, W. H.; Khan, J. Y.; Hodgson, R. M., and Collins, F. A. H., *"Efficient Wavelet Image Coding on a Digital Signal Processor Based Digital Camera,"* Proceedings of International Conference on Signal Processing Applications & Technology (ICSPAT-98), Toronto, Canada, pp.782-786, 13-16 September, 1998.



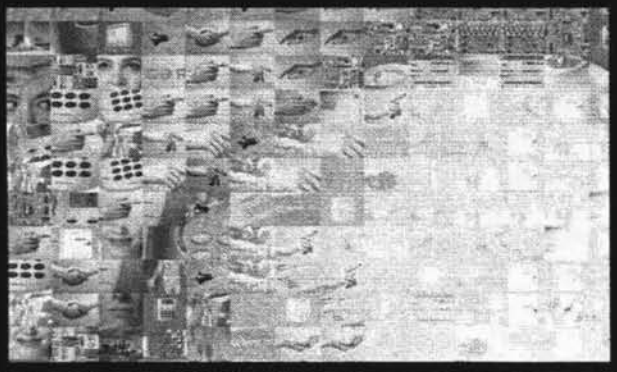
Body, N. B. and Bailey, D. G., *"Efficient Representation and Decoding of Static Huffman Code Tables in a Very Low Bit Rate Environment,"* Proceedings of IEEE International Conference on Image Processing (ICIP-98), Chicago, USA, pp.90-94, vol. 3, 4-7 October, 1998. ISBN 0-8186-8821-1.

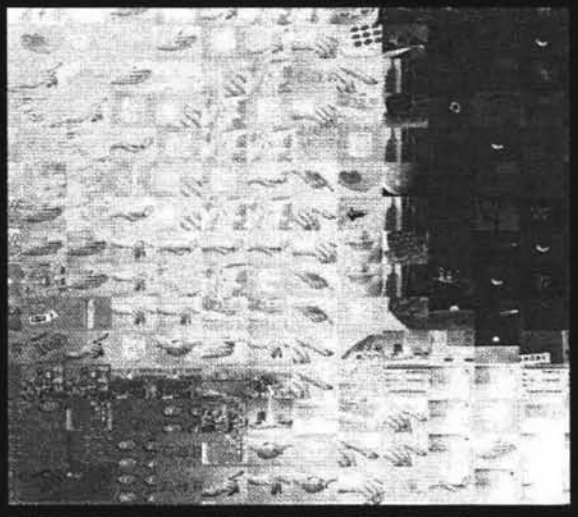


Cardax International, *"Cardax FT System Catalogue,"* 2000.

CardaxFT

System Catalogue

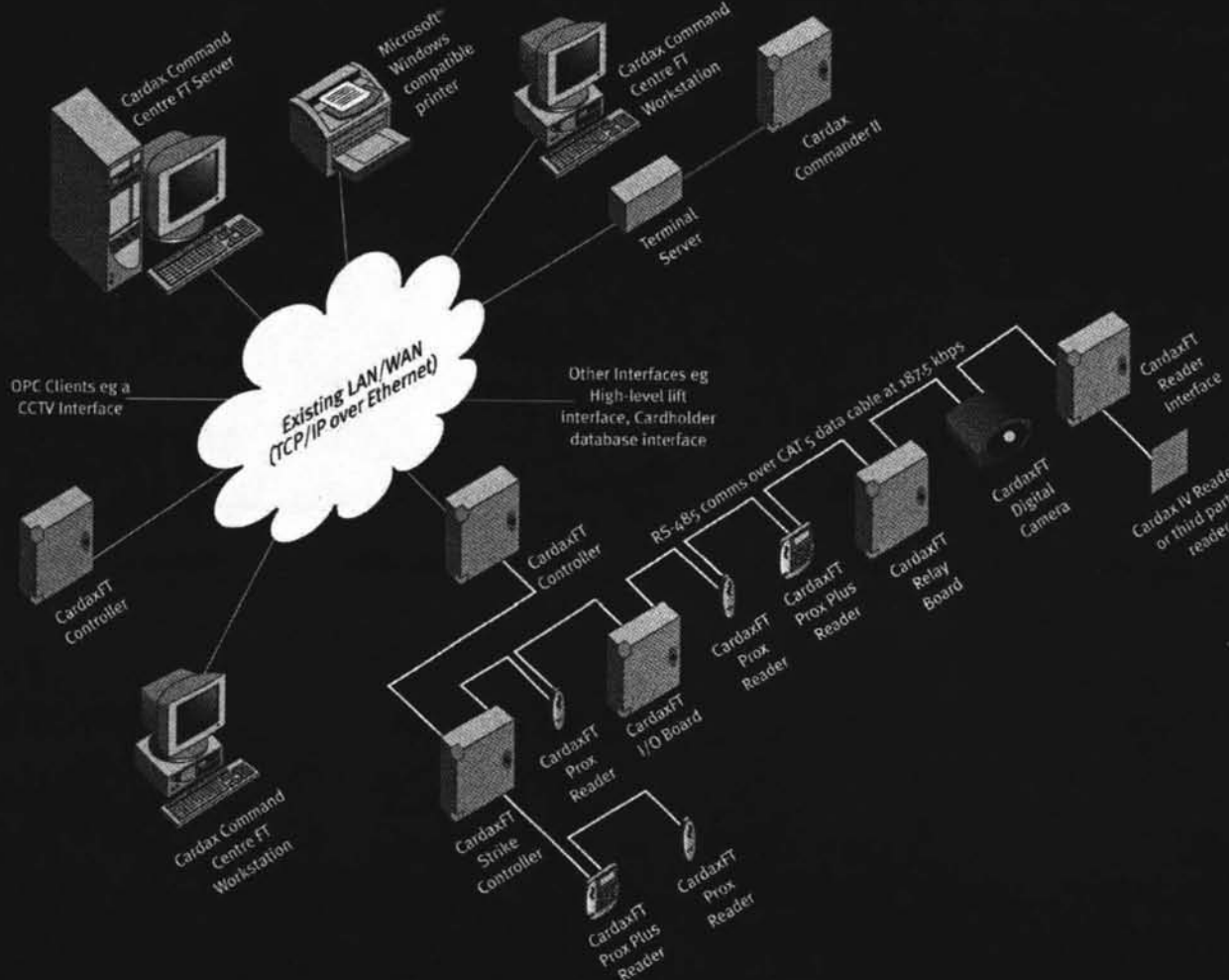




cardax

systems with integrity

CardaxFT System



Standard Configuration

Up to 500 CardaxFT Controllers

Up to 5000 CardaxFT Field Devices

Up to 50 Operator Workstations

Up to 250 CardaxFT Digital Cameras

Up to 16 CardaxFT field devices per CardaxFT Controller (Max 1 CardaxFT Digital Camera per CardaxFT Controller)

Up to 500 CardaxFT Prox Plus Readers with licensed digital Intercom

CardaxFT can be configured to manage systems larger than the standard configuration. Please contact Cardax for information.

Introducing CardaxFT

CardaxFT is a comprehensive Microsoft WindowsNT® based security system. It integrates access control and alarms management, intruder alarm monitoring, remote digital imaging, intercom and PhotoID badging.

The system is very flexible, enabling it to meet the needs of small single sites right through to very large global enterprise-wide installations.

It is a network-friendly system, designed to use minimal bandwidth. The system's communications, which are encrypted, can utilise an organisation's existing Intranet and are Internet compatible.

The NT server component uses the Microsoft® SQL database. It is OPC (OLE for Process Control) enabled, facilitating the sharing of selected system information with other OPC enabled process applications.

The CardaxFT system includes:

- Cardax Command Centre FT
- CardaxFT Controllers
- CardaxFT Readers, including CardaxFT Plus Readers with Intercom
- CardaxFT Digital Cameras
- A range of CardaxFT Input and Output Devices

CardaxFT also supports backwards compatibility with Cardax Commander II technology.

For further information about CardaxFT, please email cdxsales@cardax.com, visit our web site at www.cardax.com or contact your nearest Cardax office or Cardax distributor.

The features described in this brochure are being progressively added to the CardaxFT system during 2000/2001.
Please contact Cardax for further information.

Cardax Command CentreFT

Cardax Command Centre FT is the central management tool for the CardaxFT system.

Consisting of server and workstation software, Cardax Command Centre FT enables system operators to perform the following:

- Configure the site and all system components
- Manage cardholders including their access
- Monitor and interact with the system
- Retrieve and report on stored system information

It is a powerful, yet operator-friendly security system. Installed on a Pentium PC server, Cardax Command Centre FT resides on a Microsoft® WindowsNT® operating platform.

It can reside on a company's existing Intranet or Wide Area Network (WAN), or reside on its own dedicated network.

Cardax Command Centre FT links to CardaxFT Controllers over Ethernet using TCP/IP, a world standard network and Internet transmission protocol. Using this protocol means the system communications are Internet compatible.

Cardax Command Centre FT is a security system built using the latest technologies. It is the Cardax platform for ongoing future development.

Key system features include

- System division, enabling the system to be divided amongst security operators for management and monitoring purposes
- The ability to view live and stored images taken by CardaxFT Digital Cameras
- The ability to communicate with people at intercom-enabled CardaxFT readers and record the conversations that take place
- PhotoID to record images within each card holder record and produce photo identification badges
- The ability to share status and event data via the system's OPC interfaces
- The ability to create customised interfaces with the system's cardholder database
- Full alarms management with the ability to channel all alarms through the system for appropriate action
- Encryption of data communications to a very high standard

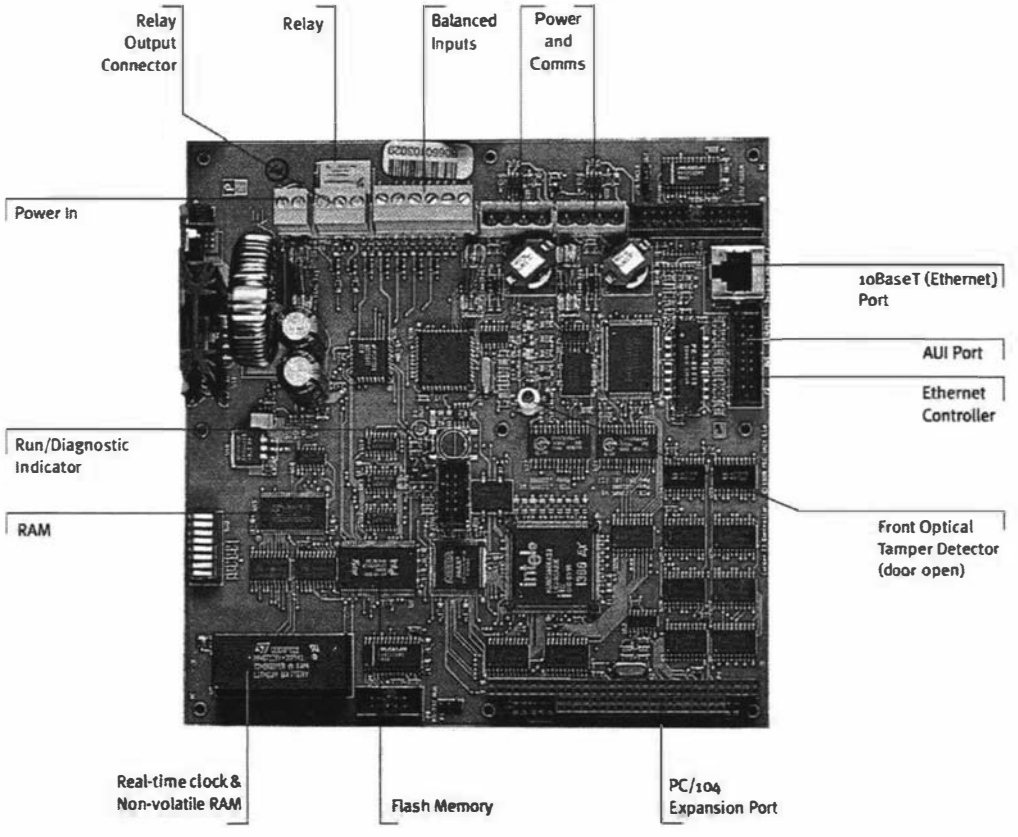
CardaxFT Controller

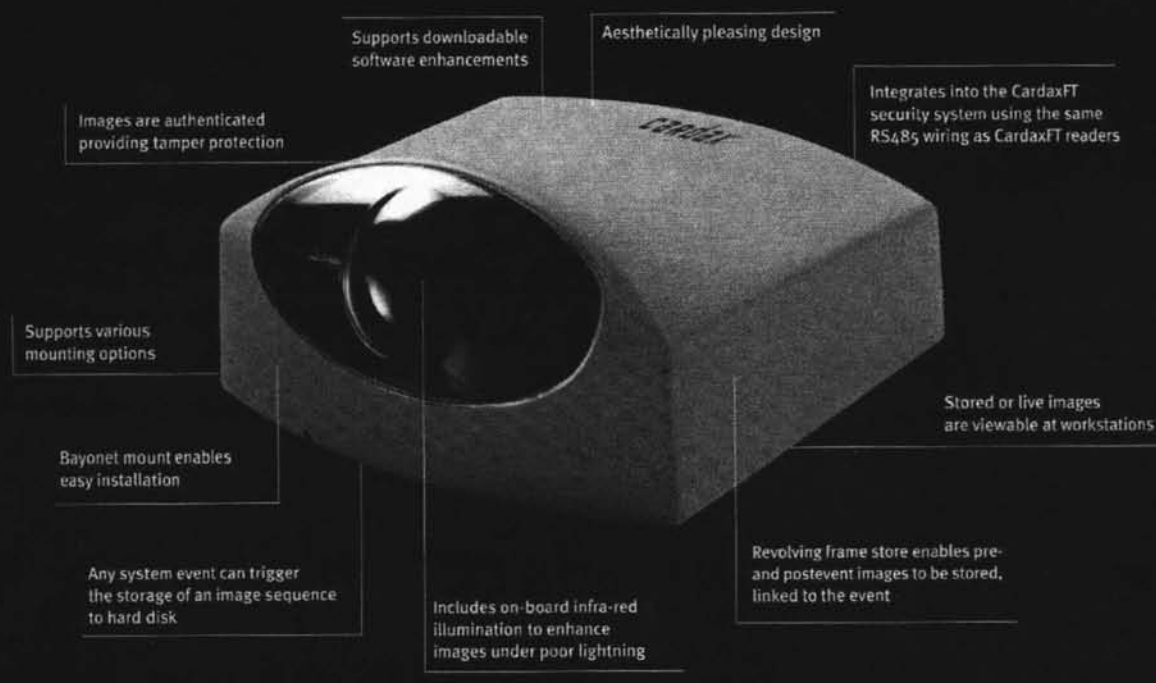
The CardaxFT Controller is the intelligent field controller in the CardaxFT system.

The CardaxFT Controller:

- Includes a 32 bit Intel® microprocessor and the Pharlap operating system for reliable and efficient performance
- Supports up to 16 CardaxFT field devices including one CardaxFT Digital Camera
- Includes a database capable of supporting 30,000 card records and a 5,000 events buffer in its default configuration for access control functions – increasing to 200,000 card records with expanded memory
- Communicates with the Cardax Command Centre FT using TCP/IP over an Ethernet network – data is encrypted to a very high standard
- Supports dial-up access for cost effective monitoring of remote sites and to provide dual-redundant communications
- Includes full Intruder Alarm functionality with the ability to set and unset alarms, and incorporate entry and exit delays
- Accommodates local time zone and daylight savings
- Can be enhanced in the future over the network via downloadable software upgrades

Key features




Supports downloadable software enhancements
Aesthetically pleasing design
Integrates into the CardaxFT security system using the same RS485 wiring as CardaxFT readers
Images are authenticated providing tamper protection
Supports various mounting options
Stored or live images are viewable at workstations
Bayonet mount enables easy installation
Any system event can trigger the storage of an image sequence to hard disk
Includes on-board infra-red illumination to enhance images under poor lightning
Revolving frame store enables pre- and postevent images to be stored, linked to the event

CardaxFT: Extending the boundaries

Cardax has been designing and manufacturing access control systems and hardware since the early 1980s. In that time, we have established a reputation for high quality systems and hardware design.

CardaxFT builds on this, extending the boundaries to a fully integrated security system. Providing flexible configuration and integration options, CardaxFT offers opportunities to tailor the system to meet the site's specific requirements.

CardaxFT: Site References and System Demonstrations
For site references and system demonstrations please contact Cardax. We would be happy to arrange.

CardaxFT: Backwards Compatibility
Cardax values long term customer relationships. CardaxFT includes options to migrate existing Cardax sites to the CardaxFT system. Backwards compatibility with Cardax Commander II technology is achieved through the Cardax Command Centre FT graphical user interface. Cardax IV readers, including the Cardax SmartSwipe, SmartProx and Smart Card Prox readers, can also be integrated into the CardaxFT system directly through the CardaxFT Reader Interface.

CardaxFT: the Cardax platform for future development.

cardax

http://www.cardax.com

Cardax (International) Ltd
Kahikatea Drive
Private Bag 3026
Hamilton
New Zealand
Telephone +64-7 838-9851
Fax +64-7 838-9801
E-mail cdxsales@cardax.com

Cardax Offices are located in:
Australia
China
New Zealand
South Africa
Thailand
United Kingdom
United States of America

CardaxFT Digital Camera

The CardaxFT Digital Camera provides totally integrated, remote digital imaging in the CardaxFT security system. It offers a unique combination of features, setting it apart from CCTV systems.

- The CardaxFT Digital Camera is an intelligent imaging device, designed to perform the following functions:
  - Motion detection to detect and report activity
  - Image security using "encrypted watermarking" to authenticate images
  - Image compression for the transmission of images over the network
- In-built infrared illumination provides enhanced digital images of poorly lit areas up to 2m from the camera
- It has an on board rotating frame store of 100 images. This enables the capture and recording of both pre-event and post-event images
- Any system event (including alarms) can trigger image storage on the event log
- The CardaxFT Digital Camera captures images at 4 frames per second and transmits them at 2 frames per second. Image data is only transmitted when:
  - an operator wants to view live images on an authorised workstation
  - a system event triggers image transmission
- If the event is an alarm, both pre-event and post-event images can be accessed via the Alarm Details window at authorised workstations
- To locate images long after the event occurred, an operator simply retrieves the event or alarm record and views the images associated with it

The CardaxFT Digital Camera enables remote sites to be monitored centrally and effectively, and uses minimal bandwidth on existing LANs or WANs. It connects to the same communications cable as the other field devices in the CardaxFT security system.

Supporting downloadable software, future feature upgrades to the CardaxFT Digital Camera will be easy to implement over the network.