Multiwavelets and Scalable Video Compression

THAM JO YEW

(B.Eng. (Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF PH.D. OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2002

In memory of my beloved

grandfather and grandmother

who had always given us their very best.

Acknowledgments

I am most grateful to Buddha for his blessings and unfailing answering of my prayers. His presence has been an abundant spiritual reservoir that constantly supplies me with the strength of mind and body to ride through many bumpy chapters along my life journey.

To my Ph.D. supervisor, Associate Professor Ranganath Surendra, for his patience, wisdom, and understanding. He is such an exemplary teacher whose guidance is admirable and greatly appreciated since my undergraduate days. His sincerity and willingness to offer help when needed are just among his hallmark qualities that I can count on.

To my thesis co-supervisor, Associate Professor Ashraf Kassim, for his many advices and help throughout my postgraduate candidature. As an energetic and dynamic person, he has always been a source of motivation and encouragement to me.

To my thesis co-supervisor, Professor Lee Seng Luan, for his selfless mentoring and persistency to help develop the best in myself. His frankness and tactfulness are traits of an honest person from whom I have learned a great deal, and also a trusted person who I regard more like a father than a mentor.

I am also overwhemingly thankful to my following friends and co-workers: A/P Tan Hwee Huat and Dr. Shen Lixin, with whom I have learned a lot of interesting mathematics through our joint research and publications; Dr. Liu Bao, Mr. Wang Pei Jie, Mr. Shang Fuchun, Dr. Xia Tao, and Ms. Jolyn Wong, who have helped me in many ways to accomplish this mission in a friendly environment, even when I was working in the United States; Ms. Anny Liauw Chon Yap, and Mr. Yap Soon Seong, for their continuous moral support and encouragement; and to all the many other people who have helped me in many different ways.

Last, but definitely not least, this dissertation is specially dedicated to my parents who have never failed to shower me with all their loves and support, and also to my brothers, sisters, nephew, and niece who I have not been spending as much time with as I would very much like to.

Contents

1 Introduction **1**

1.1 Video Compression, Scalability, and Wavelets 1

1.2 Organization of the Thesis 3

2 Multiwavelets, Filter Design, and Applications **6**

2.1 Introduction 6

2.2 Preliminaries of Multiwavelet Theory 8

2.3 Multiwavelet Filter Design and Construction 12

2.3.1 Multifilter System and Equivalent Scalar Filter Bank System 12

2.3.2 Good Multifilter Properties 15

2.3.3 Symmetric-Antisymmetric Orthonormal Multifilters 19

2.3.4 Symmetric-Antisymmetric Biorthogonal Multifilters 25

2.4 The Proposed Discrete Multiwavelet Transforms 29

2.4.1 Some Problems of Multiwavelet Initialization 29

2.4.2 Vector-Valued Multiresolution Analysis and Synthesis 31

2.4.3 Generalized Pre-Analysis and Post-Synthesis Multirate Filters . . . 33

2.4.4 Integrated Discrete Multiwavelet Transforms 36

2.5 Experimental Results and Discussions 39

2.5.1 Performance Analysis of Various Multifilter Design Criteria 39

2.5.2 Performance Analysis of Various Pre-Filtering Techniques 43

2.5.3 Computational Complexity Analysis of Transforms 45
2.6 Conclusion 48

3 Video Scalability and Scalable Video Architecture 49

3.1 Introduction 49
3.2 Scenarios for Video Scalability and Communication 50
3.3 Desirable Video Scalability Features 53
3.3.1 Bit Rate Scalability 54
3.3.2 Distortion Scalability 54
3.3.3 Spatial Resolution Scalability 55
3.3.4 Temporal Resolution Scalability 55
3.3.5 Alphabet Scalability 56
3.3.6 Hardware Scalability 56
3.3.7 Complexity Scalability 57
3.3.8 Object-Based Scalability 58
3.4 A Multi-scalable Video Compression Framework 58
3.4.1 Bit Rate Scaling and Some Related Problems 59
3.4.2 Spatial Resolution Scaling and Some Related Problems 62
3.4.3 Temporal Scaling and Some Related Problems 64
3.4.4 Alphabet and Complexity Scaling 67
3.4.5 Generation and Organization of Multi-Scalable Video Bit Stream . . 68
3.5 Conclusion 71

4 Fast Block Motion Estimation and Compensation 73

4.1 Introduction 73
4.2 A Review of Block Matching Algorithms 75
4.3 Unrestricted Center-Biased Diamond Search Algorithm 80
4.3.1 Algorithm Development of UCBDS 80

4.3.2 Theoretical Analysis of Fast UCBDS 84
4.4 Fast Block Matching in Wavelet Domain 87
4.4.1 Non-Translational Invariance of Wavelet Transform 87
4.4.2 Wavelet-Based Multiresolution UCBDS Algorithm 90
4.5 Experimental Results and Discussions 95
4.5.1 Performance Analysis of Monogrid UCBDS 95
4.5.2 Performance Analysis of Multiresolution Wavelet-based UCBDS . . 101
4.6 Conclusion 106

5 Multi-scalable Video Compression Platform 110

5.1 Introduction 110
5.2 Overview, Terminology, and Notation 112
5.3 Multi-scalable Video Encoder 119
5.3.1 Exploitation of Redundancies 119
5.3.2 Intra-Frame Scalable Encoding Algorithms 121
5.3.2.1 Recursive Direct Splitting Strategy in Segmentation Phase 124
5.3.2.2 Recursive Overlay Mapping Strategy in Segmentation Phase 128
5.3.2.3 Progressive Precision Enhancement in Refinement Phase . 134
5.3.3 Inter-Frame Scalable Encoding Algorithms 136
5.4 Multi-scalable Video Decoder 141
5.4.1 Intra-Frame Scalable Decoding Algorithms 142
5.4.2 Inter-Frame Scalable Decoding Algorithms 147
5.4.3 Examples of Scalable Decoding Specifications 150
5.5 Simulation Results and Promising Video Applications 153
5.5.1 Simulation Results of Multi-Scalable Video Codec 154
5.5.2 Some Promising Applications of Multi-scalable Video Codec 162
5.6 Conclusion 164

6 Conclusions and Future Research Directions **169**

6.1 Conclusions 169

6.2 Suggested Future Research Directions 171

A Proofs **173**

A.1 Proof of Theorem 2.1 173

A.2 Proof of Proposition 2.3 174

B Examples of Multifilters **175**

B.1 Parameterization of Length-4 SAOMF Family. 175

B.2 Parameterization of Length-6 SAOMF Family. 176

B.3 Parameterization of **BSA(4/4)** SABMF Family. 177

B.4 Parameterization of **BSA(5/5)** SABMF Family. 178

B.5 Matrix Filter Coefficients of **BSA(6/6)**, **BSA(8/6)**, and **BSA(7/9)** SABMFs. 179

List of Tables

2.1 Properties of various multiwavelet and scalar wavelet filters. (*Note that the regularity of **SA4(1)** and **SA4(3)** could not be computed accurately, as limited by the fact that their approximation orders are less than 2.) 41

2.2 Comparisons of PSNR values (in dB) of various orthonormal multiwavelet filters using different images and compression ratios. Bold entries indicate the best filters for particular image/CR combinations. 42

2.3 Comparisons of PSNR values (in dB) of various biorthogonal scalar wavelet and multiwavelet filters using different images and compression ratios. Bold entries indicate the best filters for particular image/CR combinations. . . . 44

2.4 Comparisons of PSNR values (in dB) of different images at different compression ratios using the **GHM** multiwavelet but with 3 different pre-filtering methods: (i) Hardin and Roach's pre-filter (**HRP**), (ii) Xia *et al.*'s interpolation pre-filter (**XIP**), and (iii) the proposed transformation-based pre-filter (**TP**). 45

2.5 The number of multiplications and additions required for a L-level decomposition (or reconstruction) of an $M \times N$ image using the listed scalar filters and multiwavelet filters. Here the common factor is $\mathcal{L} = MN\left(1 + \frac{1}{4} + \cdots + \frac{1}{4^{L-1}}\right)$. 47

4.1 Top to bottom: Performance comparisons (per 16×16 block) of FS, TSS, NTSS, 4SS, and UCBDS using "Miss America", "Salesman", and "Trevor White" sequences. 97

4.1 (con't) Top to bottom: Performance comparisons (per 16 × 16 block) of FS, TSS, NTSS, 4SS, and UCBDS using "Flower Garden", "Football", and "Table Tennis" sequences. 98

4.2 Performance of UCBDS versus other BMAs for the "Football" sequence. . . 99

5.1 Example of encoded bit stream of a segment using recursive direct splitting. 127

5.2 Example of encoded bit stream of a segment using recursive overlay mapping. 131

5.3 Examples of possible video decoding specifications that can be achieved from a common mulit-scalable compressed video bit stream. 152

5.4 Average PSNR results (Carphone sequence over 380 frames) for different video decoding specifications. 162

B.1 Matrix filter coefficients for the length 6/6 SABMF, **BSA(6/6)**. 180

B.2 Matrix filter coefficients for the length 8/6 SABMF, **BSA(8/6)**. 181

B.3 Matrix filter coefficients for the length 8/6 SABMF, **BSA(8/6)**. 182

List of Figures

2.1 Illustration of the concept of an equivalent system of scalar filters: (a) Multifilter framework; and (b) Equivalent scalar filter framework with a multiplexer and downsamplers. 13

2.2 One-level of decomposition and reconstruction in the multiwavelet filter bank. 33

2.3 The proposed multirate pre-analysis and post-synthesis filters that are employed for the development of a generalized and non-redundant discrete multiwavelet transform framework. 37

2.4 The proposed generalized and non-redundant discrete multiwavelet decomposition of a 2-D image, illustrating the pre-analysis multirate filtering and the multiwavelet subband structure of a 2-level transformed image. 38

2.5 Magnitude responses of the equivalent scalar filters associated with orthogonal multiwavelets **GHM**, **CL4**, **JOPT4**, and **SA4(1)**. 43

3.1 The subband structure of a 3-level multiwavelet transform. The three crosses represent the markings that will demarcate the bit stream so as to support three possible levels of spatial resolution scalability. 63

3.2 A temporal hierarchy structure that illustrates the choice of common reference frames for supporting $N = 4$ temporal scaling layers. 66

3.3 Organization of the proposed multi-scalable video bit stream hierarchy that supports simultaneous fine-granularity video scaling in terms of frame rate, bit rate, distortion, spatial resolution, color, and decoding complexity. The VH portion denotes the video header information for the entire compressed bit stream, while V_n, $n = 1, \ldots, R$ are the motion vector fields for each resolution scale (they are present only in the first LB.) 69

4.1 Motion vector distribution for "Football" sequence. 77

4.2 Diamond search pattern: (a) Original diamond search-point configuration, (b) Next step along diamond's edge, (c) Next step along diamond's face, (d) Final step with a shrunk diamond. 81

4.3 Example of unrestricted search path strategy using UCBDS. 82

4.4 Minimum possible number of search points using UCBDS, for each motion vector location. 84

4.5 Minimum possible number of search points using 4SS, for each motion vector location. 86

4.6 Performance comparisons of FS, TSS, NTSS, 4SS, and UCBDS over all 4 test criteria using "Trevor White" sequence. 100

4.7 Performance comparisons of FS, TSS, NTSS, 4SS, and UCBDS over all 4 test criteria using "Flower Garden" sequence. 101

4.8 Performance comparisons of FS, TSS, NTSS, 4SS, and UCBDS over all 4 test criteria using "Table Tennis" sequence. 102

4.9 Comparison of various block matching techniques using the Flower Garden sequence. Top to bottom and left to right: (a) original frame 10, (b) original frame 11, (c) uncompensated frame difference (average SSE per pixel: 1115.6), and motion compensated frame differences using (d) TSS (SSE: 174.9), (e) NTSS (SSE: 157.3), (f) 4SS (SSE: 166.7), (g) UCBDS (SSE: 146.4), and (h) using FS (SSE: 142.4). 103

4.10 Predicted frames from Miss America sequence: Frames (a) and (b) have poorer motion compensation due to a larger distance from the corresponding reference frames; and frames (c) and (d) have better motion prediction accuracy. 105

4.11 Reference frames from Foreman sequence: Frames (a) and (b) have poorer motion compensation due to large motions; and frames (c) and (d) have better motion prediction accuracy. Frames (a) and (c) have 10% for the base reference byte while frames (b) and (d) have 20% for the base reference byte. 107

4.12 Comparison of block motion matching in the spatial domain and wavelet domain using the Foreman video sequence: (a) An original video frame; (b) predicted frame using H.263 motion estimation in the spatial domain; and (c) predicted frame using the proposed multiresolution UCBDS in the wavelet domain. 108

5.1 Multiwavelet subband structure: (i) Luminance channel, Y; (ii) Chrominance channel, U; and (iii) Chrominance channel, V. 113

5.2 Pseudocode for encoding a color frame into resolution blocks with appropriate headers in order to generate a highly scalable compressed bit stream. . . 123

5.3 Pseudocode for encoding a given segment, $\mathbf{S}_{\ell,k}^{(J)}$, at a particular coding layer, c, using a recursive direct splitting strategy. 125

5.4 Pseudocode for encoding the multiwavelet coefficients of a given subsegment, $\mathbf{S}_{\ell,k}^{(J)}$, at a particular coding layer, c. 126

5.5 Example of a 4×4 segment that is encoded at a particular coding layer, c, using recursive direct splitting. 127

5.6 Pseudocode for encoding a given segment, $\mathbf{S}_{\ell,k}^{(J)}$, at a particular coding layer, c, using a recursive overlay mapping strategy. 129

5.7 Example of a 8×8 child segment that is encoded at a particular coding layer, c, using recursive overlay mapping. 130

5.8 Progressive refinement of quantization bin sizes of significant coefficients. . 134

5.9 Pseudocode for performing the refinement phase of a given segment, $\mathbf{S}_{\ell,k}^{(J)}$, at a coding layer, c. 136

5.10 Block diagram for encoding an inter-coded video frame using the proposed wavelet-based multiresolution UCBDS algorithm. It also shows how spatial resolution and bit rate scaling can be controlled during the encoding process. 138

5.11 Block diagram for encoding a resolution block (or a subband segment) using the proposed wavelet-based multiresolution UCBDS algorithm. It also illustrates how the problems of error propagation and loss of prediction loop can be prevented. 139

5.12 Pseudocode for decoding a scalable video bit stream with a given scalable decoding specification. Each frame is processed appropriately, if required, using either an intra-frame or inter-frame decoding mode. 142

5.13 Pseudocode for decoding a color frame from a highly scalable compressed bit stream consisting of resolution blocks with appropriate headers. Each block is either processed or discarded based on a given scalable video decoding specification. 144

5.14 Block diagram for decoding an inter-coded video frame using the proposed wavelet-based multiresolution UCBDS algorithm. It also shows how spatial resolution and bit rate scaling can be controlled during the decoding process. 149

5.15 Block diagram for decoding a resolution block (or a subband segment) using the proposed wavelet-based multiresolution UCBDS algorithm. It also illustrates how the proposed reference frame "locking" mechanism is employed to secure the prediction loop. 151

5.16 Original Carphone video sequence encoded with an encoding specification of 1 Mbps, 10 fps, full QCIF, color, and 5% for base reference byte: (a) Frame 16, (b) Frame 80, (c) Frame 176, (d) Frame 240, (e) Frame 288, and (f) Frame 368. 156

5.17 Decoded frames from Carphone video sequence with a decoding specification of 30 kbps, 5 fps, full QCIF, and grayscale: (a) Frame 16 (Y: 23.84 dB), (b) Frame 80 (Y: 24.17 dB), (c) Frame 176 (Y: 23.32 dB), (d) Frame 240 (Y: 22.25 dB), (e) Frame 288 (Y: 20.72), and (f) Frame 368 (Y: 20.03 dB). . . . 157

5.18 Decoded frames from Carphone video sequence with a decoding specification of 30 kbps, 5 fps, quarter QCIF, and color: (a) Frame 16 (Y: 27.48 dB; U: 30.15 dB; V: 28.28 dB), (b) Frame 80 (Y: 26.12 dB; U: 27.85 dB; V: 27.05 dB), (c) Frame 176 (Y: 24.11 dB; U: 25.91 dB; V: 24.33 dB), (d) Frame 240 (Y: 22.79 dB; U: 23.47 dB; V: 23.26 dB), (e) Frame 288 (Y: 21.16 dB; U: 21.95 dB; V: 22.35 dB), and (f) Frame 368 (Y: 21.47 dB; U: 22.93 dB; V: 22.48 dB). 158

5.19 Decoded frames from Carphone video sequence with a decoding specification of 15 kbps, 1.25 fps, quarter QCIF, and grayscale: (a) Frame 16 (Y: 23.38 dB), (b) Frame 80 (Y: 20.21 dB), (c) Frame 176 (Y: 20.56 dB), (d) Frame 240 (Y: 21.23 dB), (e) Frame 288 (Y: 18.17 dB), and (f) Frame 368 (Y: 21.61 dB). 159

5.20 Decoded frames from Carphone video sequence with a decoding specification of 100 kbps, 5 fps, full QCIF, and grayscale: (a) Frame 16 (Y: 24.61 dB), (b) Frame 80 (Y: 25.08 dB), (c) Frame 176 (Y: 24.10 dB), (d) Frame 240 (Y: 22.88 dB), (e) Frame 288 (Y: 21.32 dB), and (f) Frame 368 (Y: 22.05 dB). . 160

5.21 Decoded frames from Carphone video sequence with a decoding specification of 200 kbps, 10 fps, full QCIF, and color: (a) Frame 16 (Y: 33.56 dB; U: 36.48 dB; V: 34.38 dB) , (b) Frame 80 (Y: 32.52 dB; U: 36.62 dB; V: 33.65 dB), (c) Frame 176 (Y: 31.97 dB; U: 35.61 dB; V: 32.92 dB), (d) Frame 240 (Y: 29.48 dB; U: 33.30 dB; V: 30.21 dB), (e) Frame 288 (Y: 30.01 dB; U: 32.54 dB; V: 30.55 dB), and (f) Frame 368 (Y: 32.84 dB; U: 35.75 dB; V: 33.55 dB).161

5.22 Project SeeWAVE: Video server with a camera capturing the live source feed, and a wireless unicast client receiving the live stream. 165

5.23 Project SeeWAVE: Two wireless-connected PDA clients with (a) receiving a full-QCIF grayscale live video, and (b) receiving a quarter-QCIF grayscale live video. 166

5.24 Project SeeWAVE: Interfaces of PDA clients with Windows CE operating system. (a) Interface for specifying the session properties of live source feed; (b) Interface for selecting the scalable decoding specifications; (c) Client receiving a full-QCIF color live video; and (d) Client receiving a quarter-QCIF grayscale live video. 167

Symbols and Notations

bps	bits per second
fps	frames per second
kbps	kilobits per second
Mbps	megabits per second
ATM	Asynchronous Transfer Mode
BMWS	Biorthogonal Multiwavelet System
CBR	Constant Bit Rate
CPU	Central Processing Unit
CQF	Conjugate Quadrature Filter
DCT	Discrete Cosine Transform
DFD	Displaced Frame Difference
DSL	Digital Subscriber Lines
GB	Gigabytes
GHM	Geronimo-Hardin-Massopust multiwavelet
GHz	Gigahertz
GMP	Good Multifilter Properties
HDTV	High Definition Television
IP	Internet Protocol
KB	Kilobytes
LAN	Local Area Networks
LCD	Liquid Crystal Display
MB	Megabytes
MHz	Megahertz
MBONE	Internet Multicast Backbone
MEMC	Motion Estimation and Motion Compensation

MIMO	Multiple-Input Multiple-Output
MISO	Multiple-Input Single-Output
MRA	Multiresolution Analysis
MRE	Matrix Refinement Equation
MUX	Multiplexer (a MISO operator)
NTSC	National Television System Committee
PAL	Phase Alternating Line
PDA	Personal Digital Assistant
RTP	Real-Time Protocol
SAOMF	Symmetric Antisymmetric Orthonormal Multifilter
SAOMW	Symmetric Antisymmetric Orthonormal Multiwavelets
SABMF	Symmetric Antisymmetric Biorthogonal Multifilter
SABMW	Symmetric Antisymmetric Biorthogonal Multiwavelets
SECAM	Sequential Chrominance Signal and Memory
SPIHT	Set Partitioning in Hierarchical Trees
SNR	Signal to Noise Ratio
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
VBR	Variable Bit Rate

Summary

The recent explosion in digital video storage and delivery has presented strong motivations for high performance video compression solutions. Coupling this with the growing heterogeneity of user and network requirements, there is also a pressing need for a more flexible video compression framework that can support a "generate once, scale many" concept. The challenge remains as how one common scalable video bit stream can simultaneously satisfy the disparate requirements on bandwidth, display resolution, processing complexity, and viewers' preferences.

Over the past decade, the success of wavelets in solving many difficult problems has contributed to its unprecedented popularity among many research and development communities, ranging from mathematics and computer science to physics and engineering. Multiwavelets, as an extension to wavelets with only one basis, have also generated significant interest as they promise the potential to construct better multifilters with desirable properties and lower computation complexity. The challenge still remains as what constitutes good multiwavelets and how they can be constructed and applied easily.

This dissertation sets out to investigate the above open problems and propose solutions to them. On the one hand, we will formulate a definition of good multiwavelets, provide a systematic means for the construction of multiwavelets, and then present a generalized framework for their application to multiresolution signal analysis and synthesis. On the other hand, we will also look into a multi-scalable video compression architecture that will help realize a "generate once, scale many" concept. In particular, we will integrate the multiwavelet framework with multiresolution motion compensation to generate a compressed video bit stream that is sufficiently flexible to support simultaneous video scalability in terms of bit rate, distortion, frame rate, color, and spatial resolution.

Chapter 1

Introduction

"Thousands of candles can be lighted from a single candle, and the life of the candle will not be shortened. Happiness never decreases by being shared."

Siddhartha Gautama Buddha (563 - 483 B.C.)

1.1 Video Compression, Scalability, and Wavelets

One of the biggest problems that plagues the proliferation of digital technology is known as *digital obesity*, which is manifested as the voluminous amount of bits generated. Consider, for example, a typical NTSC color video frame, with 720 pixels $\times$ 480 lines, 24 bits per pixel, and 30 frames per second. This will need a transmission capacity of a whopping 237 Mbps. Without any compression, a compact disc with a storage capacity of about 650 Mbytes can store only approximately 20 seconds of NTSC video! Furthermore, full motion playback is impractical even over typical high-bandwidth connections (between 300 kbps and 1.5 Mbps). As a result, the need for good video compression algorithms is becoming more pressing as the demands for rich media applications continue to grow.

However, merely having a powerful compression scheme may not be the complete solution to some applications such as image database browsing, video-on-demand, and video communication over heterogeneous networks. In these situations, other properties such as the ability to support progressive transmission, scaling for a desired spatial resolution, frame

rate and/or bit rate, and controlling end-to-end delays in interactive applications, are fast becoming indispensible features for a good and comprehensive compression system. In general, such applications have a point-to-multipoint (multicast or broadcast) relationship in which only a few service providers cater to a wide pool of different potential users. Since different users may have disparate requirements and limitations, constraints on bit rate, display resolution, playback frame rate, and decoding complexity, cannot be anticipated in advance during compression. For example, by transmitting only the most-constrained bit rate will unfairly penalize high-bandwidth receivers as they are underutilizing their networks' capabilities. On the other hand, generating the maximum possible bit rate will cause low-bandwidth paths to become congested and receivers to get choked and stalled. This motivates the need for a highly scalable video compression system that supports a "generate-once, scale-many" concept, which can address the challenges of bandwidth heterogeneity and client diversity.

The growing demand for better and more flexible digital image and video compression algorithms continues to fuel many separate threads of research and development in both proprietary and standard-based multimedia compression frameworks. International standards such as JPEG, JPEG-2000, MPEG-1, MPEG-2, and MPEG-4 have been developed into efficient commercial software as well as dedicated hardware chipsets. In addition, some ongoing standardization efforts have been focusing on an all-encompassing multimedia framework. For example, MPEG-7 (formally known as "Multimedia Content Description Interface") aims to create a standard for describing the multimedia content data that will support a certain degree of interpretation of the information's meaning for convenient digital processing later. On the other hand, MPEG-21 is the most recent initiative to define a multimedia framework to enable transparent (interoperable) and augmented use (e.g. automation) of multimedia resources across a wide range of networks and devices used by different communities along a multimedia content delivery chain that encompasses content creation, production, delivery, and consumption.

Almost all the above standards use the *discrete cosine transform* (DCT) as the basis for frequency-based compression methods. However, newer international standards such

as MPEG-4 and JPEG-2000 may have profiles that support wavelet-based compression. In fact, wavelet-based compression has generated considerable interest among various research communities, ranging from applied mathematics to computer science and engineering. Recent breakthroughs in wavelet-based compression have demonstrated the promising future of wavelets in many real-world applications[1]. Such achievements have spawned a whole array of new commercial compression solutions such as Intel Indeo, Summus' Dynamic Wavelets, Aware's Wavelet Video, Real Networks' G2, PrimaCorp's SPIHT, Ricoh's CREW, Compression Engines' WIF, and Infinop's Lightning Strike, among others. As research in wavelets matures, extensions from scalar wavelets to multiwavelets (i.e. more than one wavelet basis function) have also been gathering momentum in both basic research and engineering applications, with the potential of improved compression efficiency and reduced implementation complexity. Multiwavelets have created new opportunities as well as many open problems that warrant more in-depth research and development; some of these issues will be the focus of this dissertation.

1.2 Organization of the Thesis

This thesis is organized into six main chapters, including this first chapter which motivates the demand for efficient and scalable digital image and video compression methods to meet the growing number of heterogeneous multiparty and disparate network environments. The success of applying wavelet-based techniques to image and video compression is also evident from its pervasive research activities worldwide, and the proliferation of wavelet-based commercial products.

Chapter 2 discusses the theory and applications of multiwavelets. Two new concepts — equivalent scalar filter bank system, and good multifilter properties are introduced — first to establish the relationship between a multiwavelet system and a set of equivalent scalar wavelets, and then to design and construct multiwavelet filters with desirable properties for image and video compression applications. This is followed by the design and con-

[1] Wavelet-based methods have been shown to be superior to many traditional approaches in areas such as signal classification, system modelling, image analysis and enhancement, signal denoising, etc.

struction of two previously unpublished classes of symmetric-antisymmetric orthonormal and biorthogonal multiwavelets that possess good multifilter properties. We then investigate the problem of multiwavelet initialization (or pre-filtering) and propose a new and generalized framework for multiresolution analysis and synthesis via discrete multiwavelet decomposition and reconstruction algorithms. Extensive image compression results are shown to verify the usefulness and efficiency of the proposed ideas and framework.

Chapter 3 focuses on video scalability, in which the importance of various types of desirable video scaling parameters are first reviewed in the context of generating a common scalable video bit stream that can simultaneously support disparate users, computing capabilities, and networking facilites. Some research and implementation issues such as error propagation in bit rate scaling, loss of prediction loop in spatial resolution scaling, and temporal hierarchy structure for supporting frame rate scaling, are investigated and solutions proposed. We also describe the details of the composition and organization of the scalable video bit stream to better understand how different combinations and degrees of video scalability can be supported and implemented.

Chapter 4 looks into an important and integral part of almost all video coding techniques: interframe motion estimation and compensation. We first review a few motion estimation methods in general, but with special focus on the popular block-based motion matching algorithms. In this dissertation, we propose a novel block matching technique called "unrestricted center-biased diamond search (UCBDS)" for fast, accurate, and robust motion estimation. We further show how to integrate the UCBDS strategy for multiresolution interframe motion compensation in the (multi)wavelet domain – a key component for supporting spatial resolution scalability. Theoretical analysis and elaborate experimental simulations are presented to verify the efficiency and effectiveness of UCBDS against other fast block matching algorithms. Results on the performance of multiscale wavelet-based UCBDS are also discussed.

Chapter 5 combines the various new ideas on multiwavelet filter design and construction, generalized discrete multiwavelet transforms, scalable video architecture, and multiscale fast block matching in the wavelet domain, to propose a novel video compression

platform for highly scalable video coding and communications. The algorithms employ a "segmented overlay mapping with divide-and-conquer" strategy to effectively exploit the various types of redundancies present in multiwavelet-transformed video frames. The details of both the scalable video encoding and decoding algorithms are explained with the help of block diagrams and psuedocodes. The fine granularity of the proposed highly scalable video compression architecture is then illustrated with the help of some examples. Experimental simulations as well as practical video communication systems employing the proposed multi-scalable video compression platform are presented and discussed.

Chapter 6 concludes the dissertation with some suggestions on future research and development directions. The appendices that follow consist of proofs of some theorems and propositions presented in earlier chapters, as well as examples of a few families of orthonormal and biorthogonal multiwavelet filters. Finally the bibliographies are listed: the first section comprises publications which we have authored, and the second section consists of a comprehensive reference of publications and Internet sites that are pertinent to the dissertation.

Chapter 2

Multiwavelets, Filter Design, and Applications

"Holding on to anger is like grasping a hot coal with the intent of throwing it at someone else; you are the one who gets burned."

Siddhartha Gautama Buddha (563 - 483 B.C.)

2.1 Introduction

The recent spate of activities in wavelets and multiwavelets provide good indications of the importance and potential impact of wavelet-based technology in solving many practical application problems. In particular, the study of multiwavelets as an extension of scalar wavelets has received considerable attention from the wavelets research communities both for theoretical development [43, 48, 49] as well as for applications such as signal compression and denoising [11, 107, 116, 121]. It was also shown in [33, 43] that multiwavelets offer simultaneous linear-phase symmetry, orthonormality, compact support, and approximation order $k > 1$, which is not possible with any real-valued scalar 2-channel wavelet systems. However, these desirable properties of multiwavelets come with a number of open problems ranging from the design and construction of good multiwavelets to the application of multiwavelets for vector-valued discrete-time signal analysis and synthesis.

This chapter focuses on some latest developments of multiwavelet filter design, construction, and applications to image and video compression. It is organized into three main sections. Section 2.2 first presents some preliminaries to the theory of multiwavelets. The basic definitions of multiresolution analysis, perfect reconstruction conditions of the associated matrix filter banks, and the requirement of the transition operator to satisfy "Condition E" are briefly reviewed. Section 2.3 introduces some new ideas in the design and construction of multiwavelet filters (or multifilters). The input-output filtering relationships between a multifilter system and a proposed concept of *equivalent scalar filter bank system* are established and explained. This subsequently leads to a new notion, which we called "good multifilter properties" (GMPs), to help us better understand the desirable filter characteristics of a useful multifilter system for image and video compression. We then introduce the construction of two novel classes of symmetric-antisymmetric multiwavelets that possess GMPs; both families of orthonormal and biorthogonal multiwavelets are presented. Section 2.4 further investigates a very important topic for the successful application of multiwavelet filters for signal decomposition and reconstruction. A fundamental problem called *multiwavelet initialization* or *pre-filtering*, which aims to generate vectorized inputs from a given (scalar) signal, is scrutinized. A new, efficient, and general multiwavelet transform framework is then proposed to address the shortcomings. Section 2.5 presents some experimental results to highlight how multiwavelets with GMPs that employ the proposed multiwavelet transform framework can achieve improved compression performance with lower computational complexity. Finally, Section 2.6 concludes this chapter with a discussion.

Notation.

Bold-faced characters are used to denote vectors and matrices. The matrices $\boldsymbol{P}^T$, $\boldsymbol{P}^*$, and $\boldsymbol{P}^{-1}$ denote respectively the transpose, conjugate transpose, and the inverse of a matrix $\boldsymbol{P}$. In addition, $\boldsymbol{P}^\sharp$ denotes the similarity transformation of $\boldsymbol{P}$ using an orthogonal matrix $\boldsymbol{U}$; that is, $\boldsymbol{P}^\sharp = \boldsymbol{U}\boldsymbol{P}\boldsymbol{U}^{-1}$. Symbols $\boldsymbol{I}$ and $\boldsymbol{0}$ denote the identity and zero matrices respectively. For a given function f, $\widehat{f}$ will denote its Fourier transform. For brevity, we will express the

eigenvector of an operator $\boldsymbol{A}$ corresponding to an eigenvalue λ as the λ-*eigenvector* of $\boldsymbol{A}$. Also, j will denote $\sqrt{-1}$. The following orthogonal matrices will also be used throughout for ease of exposition:

$$\boldsymbol{U} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \qquad \boldsymbol{S} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad \boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

2.2 Preliminaries of Multiwavelet Theory

This section presents some basic theories of multiwavelets and related multifilters (or matrix filters) that are relevant to our development. For a more complete and rigorous treatment, interested readers can refer to [48, 49]. In this section, we will review some basic properties of a biorthogonal multiwavelet with multiplicity r (where r is any positive integer), and consisting of only two subband channels: lowpass and highpass.

A biorthogonal multiwavelet system (BMWS) consists of two scaling function vectors — the *primal* scaling function vector, $\boldsymbol{\Phi} = [\phi_1, \ldots, \phi_r]^T$, and the *dual* scaling function vector, $\widetilde{\boldsymbol{\Phi}} = [\widetilde{\phi}_1, \ldots, \widetilde{\phi}_r]^T$ — that satisfy the *matrix refinement equations* (MRE):

$$\boldsymbol{\Phi}(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} \boldsymbol{H}_k \boldsymbol{\Phi}(2x - k), \tag{2.1}$$

$$\widetilde{\boldsymbol{\Phi}}(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} \widetilde{\boldsymbol{H}}_k \widetilde{\boldsymbol{\Phi}}(2x - k), \tag{2.2}$$

for some finitely supported matrix sequences $\boldsymbol{H} := \{\boldsymbol{H}_k\}_{k \in \mathbb{Z}}$ and $\widetilde{\boldsymbol{H}} := \{\widetilde{\boldsymbol{H}}_k\}_{k \in \mathbb{Z}}$. Associated with the pair of biorthogonal scaling vectors, $\boldsymbol{\Phi}$ and $\widetilde{\boldsymbol{\Phi}}$, is a pair of biorthogonal function vectors, $\boldsymbol{\Psi} = [\psi_1, \ldots, \psi_r]^T$ and $\widetilde{\boldsymbol{\Psi}} = [\widetilde{\psi}_1, \ldots, \widetilde{\psi}_r]^T$, that are related to the multi-scaling functions via the following equations:

$$\boldsymbol{\Psi}(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} \boldsymbol{G}_k \boldsymbol{\Phi}(2x - k), \tag{2.3}$$

$$\widetilde{\boldsymbol{\Psi}}(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} \widetilde{\boldsymbol{G}}_k \widetilde{\boldsymbol{\Phi}}(2x - k), \tag{2.4}$$

$\boldsymbol{G} := \{\boldsymbol{G}_k\}_{k \in \mathbb{Z}}$ and $\widetilde{\boldsymbol{G}} := \{\widetilde{\boldsymbol{G}}_k\}_{k \in \mathbb{Z}}$ are some finitely supported matrix sequences related to $\{\boldsymbol{H}_k\}$ and $\{\widetilde{\boldsymbol{H}}_k\}$. The component functions of $\boldsymbol{\Psi}$ and $\widetilde{\boldsymbol{\Psi}}$ are referred to as *multiwavelets* or *multiwavelet functions*. For a two-channel multiwavelet system, the matrix sequences

$\{\boldsymbol{H}_k\}$ and $\{\widetilde{\boldsymbol{H}}_k\}$ are termed as *matrix lowpass filters* or *lowpass multifilters*, while $\{\boldsymbol{G}_k\}$ and $\{\widetilde{\boldsymbol{G}}_k\}$ are called the *matrix highpass filters* or *highpass multifilters*.

Let V_0 and $\widetilde{V}_0$ be the closed subspaces generated by the integer shifts of ϕ_i and $\widetilde{\phi}_i, i = 1, \ldots, r$, respectively, such that

$$\begin{aligned} V_0 &= \overline{\text{span}\{\phi_i(. - k) : k \in \mathbb{Z}, i = 1, \ldots, r\}}, \\ \widetilde{V}_0 &= \overline{\text{span}\{\widetilde{\phi}_i(. - k) : k \in \mathbb{Z}, i = 1, \ldots, r\}}. \end{aligned}$$

For each fixed $\ell \in \mathbb{Z}$, we define the subspaces V_ℓ and $\widetilde{V}_\ell$ by

$$V_\ell = \{f(2^\ell .) : f \in V_0\}, \qquad \widetilde{V}_\ell = \{f(2^\ell .) : f \in \widetilde{V}_0\}.$$

A biorthogonal multiresolution analysis (MRA) is generated by a pair of sequences $\{V_\ell\}_{\ell \in \mathbb{Z}}$ and $\{\widetilde{V}_\ell\}_{\ell \in \mathbb{Z}}$ of embedded closed subspaces of $L^2(\mathbb{R})$ such that

$$\{0\} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \cdots \subset L^2(\mathbb{R}), \quad \text{and} \quad \{0\} \subset \cdots \subset \widetilde{V}_{-1} \subset \widetilde{V}_0 \subset \widetilde{V}_1 \cdots \subset L^2(\mathbb{R}),$$

with

$$\langle \phi_i(. - m), \widetilde{\phi}_k(. - n) \rangle = \delta_{i,k} \delta_{m,n}, \quad i, k = 1, \ldots, r, \quad m, n \in \mathbb{Z},$$

where $\langle f, g \rangle = \int f \overline{g}$ and $\delta_{i,k} = 1$ if $i = k$, and 0 otherwise. Also, it is well known that if (2.1) and (2.2) have solutions $\boldsymbol{\Phi}, \widetilde{\boldsymbol{\Phi}} \in L^2(\mathbb{R})$ such that $\{\phi_i(. - k) : k \in \mathbb{Z}, i = 1, \ldots, r\}$ and $\{\widetilde{\phi}_i(. - k) : k \in \mathbb{Z}, i = 1, \ldots, r\}$ form Riesz bases (or orthonormal bases) of their closed linear spans V_0 and $\widetilde{V}_0$ respectively, then both $\{V_\ell\}_{\ell \in \mathbb{Z}}$ and $\{\widetilde{V}_\ell\}_{\ell \in \mathbb{Z}}$ are multiresolutions of $L^2(\mathbb{R})$. Suppose further that W_0 is an algebraic complement of V_0 in V_1, and is orthogonal to $\widetilde{V}_0$; and $\widetilde{W}_0$ is an algebraic complement of $\widetilde{V}_0$ in $\widetilde{V}_1$, and is orthogonal to V_0. The compactly supported functions ψ_i and $\widetilde{\psi}_i, i = 1, \ldots, r$, are called *biorthogonal multiwavelets* if $\{\psi_i(. - k) : k \in \mathbb{Z}, i = 1, \ldots, r\}$ and $\{\widetilde{\psi}_i(. - k) : k \in \mathbb{Z}, i = 1, \ldots, r\}$ form Riesz bases of their closed linear spans W_0 and $\widetilde{W}_0$ respectively, and

$$\langle \psi_i(. - m), \widetilde{\psi}_k(. - n) \rangle = \delta_{i,k} \delta_{m,n}, \quad i, k = 1, \ldots, r, \quad m, n \in \mathbb{Z}.$$

It can also be shown that the biorthogonality of the multiscaling and multiwavelet functions implies the following perfect reconstruction (PR) conditions on the matrix lowpass

and highpass filters:

$$\sum_{k\in\mathbb{Z}} \boldsymbol{H}_k \widetilde{\boldsymbol{H}}^T_{k+2i} = \delta_{0,i}\boldsymbol{I}, \tag{2.5}$$

$$\sum_{k\in\mathbb{Z}} \boldsymbol{G}_k \widetilde{\boldsymbol{G}}^T_{k+2i} = \delta_{0,i}\boldsymbol{I}, \tag{2.6}$$

$$\sum_{k\in\mathbb{Z}} \boldsymbol{H}_k \widetilde{\boldsymbol{G}}^T_{k+2i} = \boldsymbol{0}, \tag{2.7}$$

$$\sum_{k\in\mathbb{Z}} \boldsymbol{G}_k \widetilde{\boldsymbol{H}}^T_{k+2i} = \boldsymbol{0}, \tag{2.8}$$

for $i \in \mathbb{Z}$, where $\widetilde{\boldsymbol{H}}^T$ is the transpose of $\widetilde{\boldsymbol{H}}$. Specifically, the sequences $\{\boldsymbol{H}_k\}_{k\in\mathbb{Z}}$ and $\{\widetilde{\boldsymbol{H}}_k\}_{k\in\mathbb{Z}}$ which satisfy (2.5) constitute a matrix conjugate quadrature filter (CQF). In the Fourier domain, we define $\widehat{\boldsymbol{H}}(\omega) := \frac{1}{\sqrt{2}}\sum_{k\in\mathbb{Z}} \boldsymbol{H}_k e^{-jk\omega}$ and $\widehat{\widetilde{\boldsymbol{H}}}(\omega) := \frac{1}{\sqrt{2}}\sum_{k\in\mathbb{Z}} \widetilde{\boldsymbol{H}}_k e^{-jk\omega}$ as the matrix lowpass frequency response functions associated with the multiscaling functions. They are $r \times r$ matrices with trigonometric polynomial entries. Similarly, we denote the matrix highpass frequency response functions as $\widehat{\boldsymbol{G}}(\omega) := \frac{1}{\sqrt{2}}\sum_{k\in\mathbb{Z}} \boldsymbol{G}_k e^{-jk\omega}$ and $\widehat{\widetilde{\boldsymbol{G}}}(\omega) := \frac{1}{\sqrt{2}}\sum_{k\in\mathbb{Z}} \widetilde{\boldsymbol{G}}_k e^{-jk\omega}$. Equivalently, the above PR conditions (2.5)–(2.7) can be also expressed as:

$$\widehat{\boldsymbol{H}}(\omega)\widehat{\widetilde{\boldsymbol{H}}}^*(\omega) + \widehat{\boldsymbol{H}}(\omega+\pi)\widehat{\widetilde{\boldsymbol{H}}}^*(\omega+\pi) = \boldsymbol{I}, \tag{2.9}$$

$$\widehat{\boldsymbol{G}}(\omega)\widehat{\widetilde{\boldsymbol{G}}}^*(\omega) + \widehat{\boldsymbol{G}}(\omega+\pi)\widehat{\widetilde{\boldsymbol{G}}}^*(\omega+\pi) = \boldsymbol{I}, \tag{2.10}$$

$$\widehat{\boldsymbol{H}}(\omega)\widehat{\widetilde{\boldsymbol{G}}}^*(\omega) + \widehat{\boldsymbol{H}}(\omega+\pi)\widehat{\widetilde{\boldsymbol{G}}}^*(\omega+\pi) = \boldsymbol{0}, \tag{2.11}$$

$$\widehat{\boldsymbol{G}}(\omega)\widehat{\widetilde{\boldsymbol{H}}}^*(\omega) + \widehat{\boldsymbol{G}}(\omega+\pi)\widehat{\widetilde{\boldsymbol{H}}}^*(\omega+\pi) = \boldsymbol{0}. \tag{2.12}$$

In this dissertation, we focus on the construction of PR matrix CQFs that satisfy the PR conditions (2.5)–(2.7) and some other desirable multifilter properties that are useful in image and video compression applications. Multifilters constructed as such, however, do not necessarily lead to multiscaling functions and multiwavelets. To verify this, one has to check the corresponding transition operators. The transition operator for $\boldsymbol{H}(\omega)$ is defined as

$$\boldsymbol{T}_{\boldsymbol{H}}\boldsymbol{M}(\omega) := \boldsymbol{H}\left(\frac{\omega}{2}\right)\boldsymbol{M}\left(\frac{\omega}{2}\right)\boldsymbol{H}^*\left(\frac{\omega}{2}\right) + \boldsymbol{H}\left(\frac{\omega}{2}+\pi\right)\boldsymbol{M}\left(\frac{\omega}{2}+\pi\right)\boldsymbol{H}^*\left(\frac{\omega}{2}+\pi\right). \tag{2.13}$$

This operator is useful for characterizing the stability and orthonormality of $\boldsymbol{\Phi}$. The scaling function vector $\boldsymbol{\Phi}$ is orthonormal if and only if $\{\boldsymbol{H}_k\}$ is a matrix CQF, and its transition

operator $\boldsymbol{T_H}$ satisfies "Condition E"[1] (see [100]). Note that for $\{\boldsymbol{H}_k\}_{k=0}^{L}$, the transition operator $\boldsymbol{T_H}$ is a linear operator on $\mathbb{M}_L$, where $\mathbb{M}_L$ is the space of all $r \times r$ matrices whose entries are trigonometric polynomials such that their Fourier coefficients are supported in $[-L, L]$.

For all the orthonormal and biorthogonal multifilters that we constructed here, we have numerically verified that their transition operators satisfy Condition E, and, as such, these multifilters are also the matrix CQFs that generate multiwavelet systems. Hence in this dissertation, the terms "multifilters" and "multiwavelets" are often used interchangably.

In most signal processing applications, an important property of a multiwavelet system is the *approximation order*[2] of the associated multiscaling functions (or the *vanishing moments* of the associated multiwavelet functions). A multiscaling function, $\boldsymbol{\Phi}$, provides an approximation order m if and only if there exist real $1 \times r$ row vectors $\boldsymbol{y}_k \in \mathbb{R}^r, k = 0, \ldots, m-1$, with $\boldsymbol{y}_0 \neq \boldsymbol{0}$, such that

$$\begin{aligned} \sum_{k=0}^{n} \binom{n}{k} (2j)^{k-n} (\boldsymbol{y}_k)^T (D^{n-k}\boldsymbol{H})(0) &= 2^{-n} (\boldsymbol{y}_n)^T, \\ \sum_{k=0}^{n} \binom{n}{k} (2j)^{k-n} (\boldsymbol{y}_k)^T (D^{n-k}\boldsymbol{H})(\pi) &= \boldsymbol{0}^T, \end{aligned} \tag{2.14}$$

where $D^{n-k}\boldsymbol{H}(\omega)$ denotes the matrix formed by the $(n-k)^{th}$ derivatives of the entries of the matrix refinement mask $\boldsymbol{H}(\omega)$. A similar result applies to $\widetilde{\boldsymbol{\Phi}}$ and its matrix refinement mask $\widetilde{\boldsymbol{H}}(\omega)$. We say that a multifilter has approximation order (m_1, m_2) if the matrix refinement masks $\boldsymbol{H}(\omega)$ and $\widetilde{\boldsymbol{H}}(\omega)$ satisfy the above conditions with $m = m_1$ and $m = m_2$ respectively.

It is finally worth noting that the above basic theories for a biorthogonal multiwavelet system can also be applied to describe an orthonormal multiwavelet system by equating the primal multiscaling functions and multiwavelet functions with the corresponding dual function vectors. In addition, when we restrict the multiplicity r to be 1, we have the scalar wavelet case.

[1] We say that a square matrix $\boldsymbol{M}$ (or a linear operator) satisfies Condition E if its spectral radius $\rho(\mathbf{M}) \leq 1$ with 1 being the only simple eigenvalue of $\boldsymbol{M}$ on the unit circle.

[2] A function vector $\boldsymbol{\Phi}$ has an approximation order of m when all polynomials of degree from 0 to $m-1$ can be exactly reproduced by a linear combination of integer translates of ϕ_k, $k = 1, \ldots, r$.

2.3 Multiwavelet Filter Design and Construction

This section is dedicated to the design and construction of some new and useful multiwavelet filters. It first formulates the concept of an *equivalent scalar filter bank system*, in which we present an equivalent and sufficient representation of a multiwavelet system with multiplicity r in terms of a set of r equivalent scalar (wavelet) filter banks. This relationship motivates a new measure called the *good multifilter properties* (GMPs), which define the desirable frequency response characteristics of the equivalent scalar filters. We then relate the notion of GMPs directly to the matrix filters as necessary eigenvector properties for the refinement masks of a given multiwavelet system. The GMPs are then exploited to construct two families of symmetric-antisymmetric orthonormal and biorthogonal multifilters. The construction steps, and parameterized examples of multifilters with different filter lengths are presented. An intrinsic relationship that allows the construction of these multifilters directly from a special class of scalar wavelets will also be expounded.

2.3.1 Multifilter System and Equivalent Scalar Filter Bank System

A multifilter system refers to a matrix filter bank system that has a multiple-input multiple-output (MIMO) relationship, as depicted in Figure 2.1 (a). In this figure, the vectorized input stream $\boldsymbol{x}$ is filtered by a matrix filter $\boldsymbol{P}$ to produce the vectorized output $\boldsymbol{y}$. In the context of a multiwavelet system with multiplicity $r > 1$, the r output streams, $\boldsymbol{y}_k, k = 1, 2, \ldots, r$, are essentially given by the convolution of the r input streams, $\boldsymbol{x}_k, k = 1, 2, \ldots, r$, with the $r \times r$ matrix filter impulse response $\boldsymbol{P}$. Mathematically, the matrix filtering process can be expressed as

$$y_k(n) = \sum_{\ell \in \mathbb{Z}} \sum_{m=1}^{r} p_{k,m}(\ell) x_m(n-\ell), \qquad n \in \mathbb{Z}, \quad k = 1, 2, \ldots, r, \tag{2.15}$$

where $\boldsymbol{P}_\ell := [p_{k,m}(\ell)]_{k,m=1}^{r}$, $\ell \in \mathbb{Z}$; $\boldsymbol{x}_k = \{x_k(n)\}_{n \in \mathbb{Z}}$; and $\boldsymbol{y}_k = \{y_k(n)\}_{n \in \mathbb{Z}}$. Equivalently, in the Fourier domain, it can be written as

$$\widehat{\boldsymbol{Y}}(\omega) = \widehat{\boldsymbol{P}}(\omega)\widehat{\boldsymbol{X}}(\omega), \tag{2.16}$$

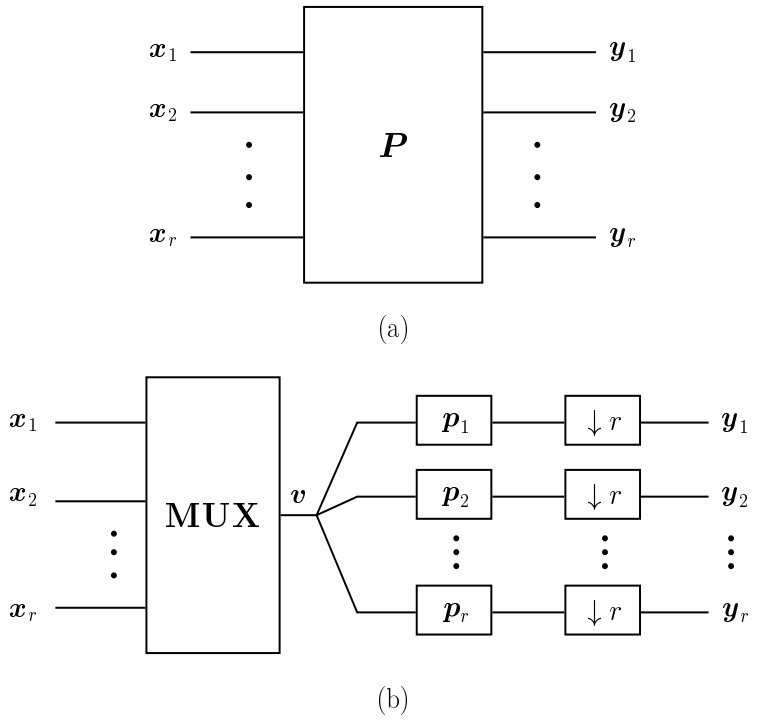


Figure 2.1: Illustration of the concept of an equivalent system of scalar filters: (a) Multifilter framework; and (b) Equivalent scalar filter framework with a multiplexer and downsamplers.

where $\widehat{\boldsymbol{P}}(\omega) := \frac{1}{\sqrt{2}} \sum_{\ell \in \mathbb{Z}} \boldsymbol{P}_\ell e^{-j\ell\omega}$ is the filter's frequency response, $\widehat{\boldsymbol{X}}(\omega) = [\widehat{\boldsymbol{x}}_1(\omega), ..., \widehat{\boldsymbol{x}}_r(\omega)]^T$, and $\widehat{\boldsymbol{Y}}(\omega) = [\widehat{\boldsymbol{y}}_1(\omega), ..., \widehat{\boldsymbol{y}}_r(\omega)]^T$. For critical sampling in the filtering process, the output streams can be further downsampled by a factor that corresponds to the number of channels (bands) of the multifilter system.

In most signal processing applications, we often desire that the filtered streams to possess certain properties; for instance, the lowpass-filtered outputs to contain only the smooth portions of the input signal, and the highpass-filtered outputs to capture only the significant signal transitions representative of the high-frequency component of the input signal. In the case of a scalar filter that accepts one input stream and generates one output stream, it is well-known that the ideal frequency response of the filter is given by the "brickwall" filter with a sharp cutoff frequency. The frequency characteristics of a matrix filter, however, are not so straightforward; this partially accounts for the lack of insights into which properties will contribute to good matrix filtering and, more challengingly, how to design matrix filters that possess certain desirable filtering properties.

In the following, we will illuminate the above problems by exploring the connection

(or relationship) between multifilters and some related scalar filters. Consider the following observation that establishes this relationship via a multiplexing operator and appropriate downsamplers:

Observation 2.1. *For any multiwavelet system with multiplicity $r > 1$ that has r input streams, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_r$, and r output streams, $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_r$, there always exists an equivalent filter bank system with a set of r scalar (wavelet) filters, $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_r$, such that the output stream $\boldsymbol{y}_k$ is a filtered and downsampled version of a multiplexed input stream, $\boldsymbol{v}$, with the scalar filter $\boldsymbol{p}_k$, for all $k = 1, 2, \ldots, r$.*

Figure 2.1 (b) illustrates an equivalent framework, from an input-output filtering viewpoint, that replaces the multifilter $\boldsymbol{P}$ with a cascade of a multiplexer, a system of equivalent scalar (wavelet) filters $\mathtt{p}_1, \mathtt{p}_2, \ldots, \mathtt{p}_r$, and downsamplers. The object now is to establish the following: (i) the relationship between the equivalent scalar filters and the associated multifilter system; and (ii) the function of the multiplexing operator. This subsection will show that the r equivalent scalar filters are, in fact, the r polyphases of the corresponding multifilter. The multiplexer operator will also be defined here, but more insights into how the multiplexer can motivate the development of the proposed multiwavelet transform framework will be given in Section 2.4.

For the meantime, assume[3] that we already have the vector input streams $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_r$, which are filtered to produce the vector output streams $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_r$. Suppose further that we multiplex the multiple input streams $\boldsymbol{x}_k, k = 1, \ldots, r$, to produce a single stream $\boldsymbol{v} = \{v(n)\}_{n\in\mathbb{Z}}$ via a multiple-input-single-output (MISO) operator, **MUX**, as depicted in Figure 2.1 (b). This subsequently allows us to filter the multiplexed stream $\boldsymbol{v} = \{v(n)\}_{n\in\mathbb{Z}}$ *independently* using each of the r scalar (wavelet) filters, $\mathtt{p}_k = \{p_k(n)\}_{n\in\mathbb{Z}}, k = 1, 2, \ldots, r$, and followed by downsampling with a decimation factor of r, such that:

$$\begin{aligned} y_k(n) &= \sum_{\ell\in\mathbb{Z}} p_k(\ell) v(rn-\ell), \\ &= \sum_{\ell\in\mathbb{Z}} \sum_{m=0}^{r-1} p_k(\ell r+m) v(rn-(\ell r+m)), \qquad n \in \mathbb{Z}. \end{aligned} \tag{2.17}$$

[3]We will postpone the discussion on an efficient technique for generating the multiple (vector) input streams to Section 2.4.

for $n \in \mathbb{Z}$. By comparing (2.15) and (2.17), we can establish that:

(i) the relationship between the equivalent scalar filters, $\mathtt{p}_k$, and the associated multifilter $\boldsymbol{P}_\ell$ is given by

$$p_k(\ell r + m - 1) = p_{k,m}(\ell), \qquad \ell \in \mathbb{Z}, \quad k, m = 1, 2, \ldots, r, \tag{2.18}$$

(ii) the operator **MUX** that multiplexes the multiple input streams $\boldsymbol{x}$ into a scalar stream $\boldsymbol{v}$ is defined as

$$\mathbf{MUX} : (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_r) \longmapsto \boldsymbol{v} : x_m(k) \longrightarrow v(rk - (m-1)). \tag{2.19}$$

Denote $\widehat{\mathtt{p}}_k(\omega) := \frac{1}{\sqrt{2}} \sum_{k \in \mathbb{Z}} p_k(n) e^{-j\omega n}$, $k = 1, 2, \ldots, r$. From (2.18), we can show that

$$\begin{aligned} \widehat{\mathtt{p}}_s(\omega) &= \sum_{k \in \mathbb{Z}} \left(\sum_{t=1}^{r} p_{s,t}(k) e^{-j(t-1)\omega} \right) e^{-jrk\omega} \\ &= \sum_{t=1}^{r} \left(\sum_{k \in \mathbb{Z}} p_{s,t}(k) e^{-jrk\omega} \right) e^{-j(t-1)\omega}, \end{aligned} \tag{2.20}$$

where the t^{th} polyphase of the s^{th} equivalent scalar filter is given by $\sum\limits_{k \in \mathbb{Z}} p_{s,t}(k) e^{-jrk\omega}$, for all $s, t = 1, 2, \ldots, r$. Hence, it is shown that any multifilter with a multiplicity $r > 1$ can be represented by a set of r equivalent scalar filters, each consisting of r polyphases. Such an observation is also related to the theory of polyphases of block or vector-valued filter banks (see e.g. [105, 114, 121]). In summary, we have formulated an equivalent scalar filter bank system that guarantees an identical MIMO relationship with that of any multifilter system. In matrix notation, (2.20) can also be expressed as

$$[\widehat{\mathtt{p}}_1(\omega), \ldots, \widehat{\mathtt{p}}_r(\omega)]^T = \widehat{\boldsymbol{P}}(r\omega)\boldsymbol{e}(\omega), \tag{2.21}$$

where $\boldsymbol{e}(\omega) = [1, e^{-j\omega}, \ldots, e^{-j(r-1)\omega}]^T$.

2.3.2 Good Multifilter Properties

Having established an equivalent MIMO relationship above, we now have a new framework to analyze and design a multifilter system by imposing desirable filter properties on the corresponding set of equivalent scalar filters. This motivates the development of a new set of multifilter design criteria called "good multifilter properties" (GMPs). We first provide

the definition of GMPs in relation to the desirable magnitude responses of the equivalent scalar filters. For convenience of notation, we will discuss in the context of orthonormal multiwavelets; for biorthogonal multiwavelets, the same definition for the primals will apply to the duals (the details can be found in Section 3 of [13]).

Definition 2.1. *A given multiwavelet system with multiplicity r is considered a good multifilter of GMP order (d_1, d_2, d_3) if its equivalent lowpass and highpass scalar filters, $\mathbf{h}_k$ and $\mathbf{g}_k$, possess the following properties:*

(i) $\widehat{\mathbf{h}}_k^{(\nu)}(0) = \delta_{\nu,0}, \quad \nu = 0, 1, ..., d_1 - 1,$

(ii) $\widehat{\mathbf{h}}_k^{(\nu)}(\pi) = 0, \qquad \nu = 0, 1, ..., d_2 - 1,$

(iii) $\widehat{\mathbf{g}}_k^{(\nu)}(0) = 0, \qquad \nu = 0, 1, ..., d_3 - 1,$

for all $k = 1, 2, \ldots, r$, where the superscript $^{(\nu)}$ denotes the ν^{th}-derivative, and $d_1, d_2, d_3 \geq 1$.

In fact, it is clear that the above criteria ensure that the equivalent scalar filters, $\mathbf{h}_k$ and $\mathbf{g}_k$, $k = 1, 2, \ldots, r$, behave as lowpass and bandpass (highpass) scalar filters, respectively. Taking the ν^{th}-derivative on both sides of (2.21), the above criteria for good filter characteristics can now be expressed explicitly in relation to a multifilter system as:

$$\text{(i)} \quad \sum_{q=0}^{\nu} \binom{\nu}{q} r^{\nu-q} \widehat{\boldsymbol{H}}^{(\nu-q)}(0) \boldsymbol{e}^{(q)}(0) = \delta_{\nu,0} \boldsymbol{e}(0), \quad \nu = 0, 1, ..., d_1 - 1,$$

$$\text{(ii)} \quad \sum_{q=0}^{\nu} \binom{\nu}{q} r^{\nu-q} \widehat{\boldsymbol{H}}^{(\nu-q)}(r\pi) \boldsymbol{e}^{(q)}(\pi) = \mathbf{0}, \quad \nu = 0, 1, ..., d_2 - 1,$$

$$\text{(iii)} \quad \sum_{q=0}^{\nu} \binom{\nu}{q} r^{\nu-q} \widehat{\boldsymbol{G}}^{(\nu-q)}(0) \boldsymbol{e}^{(q)}(0) = \mathbf{0}, \quad \nu = 0, 1, ..., d_3 - 1, \tag{2.22}$$

where $\boldsymbol{e}(\omega) = [1, e^{-j\omega}, \ldots, e^{-j(r-1)\omega}]^T$.

It will now be very interesting to understand the idea of GMPs *directly* in terms of some properties of the matrix filters. Specifically, we will investigate the eigenvector characteristics of matrix filters that correspond to multiwavelets possessing GMPs. From the above relationship, we can easily verify the following proposition:

Proposition 2.1. *Suppose that an orthogonal multiwavelet system has a GMP order* $(1,1,1)$*, then we have:*

(i) $\widehat{\boldsymbol{H}}(0)\boldsymbol{e}(0) = \boldsymbol{e}(0)$*;*

(ii) $\widehat{\boldsymbol{H}}(r\pi)\boldsymbol{e}(\pi) = \boldsymbol{0}$*; and*

(iii) $\widehat{\boldsymbol{G}}(0)\boldsymbol{e}(0) = \boldsymbol{0}$.

For any orthogonal multiwavelet system where $\widehat{\boldsymbol{H}}(0)$ satisfies Condition E and has a vanishing moment of at least order one, there exists a vector $\boldsymbol{v}$ such that

$$\boldsymbol{v}^T\widehat{\boldsymbol{H}}(\pi\nu) = \delta_{0,\nu}\boldsymbol{v}^T, \quad \nu = 0,1. \tag{2.23}$$

By setting $\omega = 0$ in the CQF relation (2.9), and multiplying it with $\boldsymbol{v}^T$ from the left side, we have

$$\boldsymbol{v}^T\widehat{\boldsymbol{H}}(0)\widehat{\boldsymbol{H}}(0)^* + \boldsymbol{v}^T\widehat{\boldsymbol{H}}(\pi)\widehat{\boldsymbol{H}}(\pi)^* = \boldsymbol{v}^T. \tag{2.24}$$

Clearly, by applying (2.23) into (2.24), we obtain

$$\widehat{\boldsymbol{H}}(0)\boldsymbol{v} = \boldsymbol{v}, \tag{2.25}$$

which implies that $\boldsymbol{v}$ is a right eigenvector of $\widehat{\boldsymbol{H}}(0)$ corresponding to an eigenvalue $\lambda = 1$. Similarly, from (2.11) and (2.23), we can derive

$$\widehat{\boldsymbol{G}}(0)\boldsymbol{v} = \boldsymbol{0}. \tag{2.26}$$

By combining (2.1) and the definition of Condition E, one can get

$$\widehat{\boldsymbol{\Phi}}(0) = \boldsymbol{v}, \tag{2.27}$$

up to a constant (i.e., $\widehat{\boldsymbol{\Phi}}(0)$ is parallel to $\boldsymbol{v}$). Hence, if a multiwavelet system has a GMP order $(1,1,1)$, then, up to a constant, we have

$$\widehat{\boldsymbol{\Phi}}(0) = \boldsymbol{v} = \boldsymbol{e}(0), \tag{2.28}$$

which proves the relation (i) of Proposition 2.1 since $\lambda = 1$ is a simple eigenvalue of $\widehat{\boldsymbol{H}}(0)$.

It is, however, noted that (2.28) imposes a rather restrictive condition on the design of multiwavelets. To alleviate this constraint, we can perform a change of basis by applying a similarity transformation to the multifilters, such that the *new* multifilter frequency responses are given by

$$\widehat{\boldsymbol{H}}^{\sharp}(\omega) = \boldsymbol{U}\widehat{\boldsymbol{H}}(\omega)\boldsymbol{U}^{-1} \quad \text{and} \quad \widehat{\boldsymbol{G}}^{\sharp}(\omega) = \boldsymbol{U}\widehat{\boldsymbol{G}}(\omega)\boldsymbol{U}^{-1}, \tag{2.29}$$

and the associated *new* multiscaling function vector and multiwavelet vector are defined as

$$\boldsymbol{\Phi}^{\sharp}(x) = \boldsymbol{U}\boldsymbol{\Phi}(x) \quad \text{and} \quad \boldsymbol{\Psi}^{\sharp}(x) = \boldsymbol{U}\boldsymbol{\Psi}(x), \tag{2.30}$$

respectively. It is worth noting that such a similarity transformation still guarantees that $\{\boldsymbol{H}_k^{\sharp}, \boldsymbol{G}_k^{\sharp}\}$ satisfies the PR criteria (2.9)—(2.11). Therefore we can say that an orthogonal multiwavelet system $\{\boldsymbol{H}_k, \boldsymbol{G}_k\}$ possesses a GMP order (d_1, d_2, d_3) if there exists an orthogonal matrix $\boldsymbol{U}$ such that $\{\boldsymbol{H}_k^{\sharp}, \boldsymbol{G}_k^{\sharp}\}$ possesses a GMP order (d_1, d_2, d_3). In fact, the orthogonal matrix $\boldsymbol{U}$ is completely determined by the (zero order) moment of multiscaling function vector $\widehat{\boldsymbol{\Phi}}(0)$ such that $\boldsymbol{U}\widehat{\boldsymbol{\Phi}}(0)$ is parallel to vector $\boldsymbol{e}(0)$.

For simplicity of the following exposition, but without loss of generality, we consider multiwavelets with multiplicity $r = 2$. In addition, as we will show later in Section 2.5, a multiwavelet system should possess a GMP order of *at least* (1,1,1) in order to produce good compression performance. As a useful multifilter design guide, let us consider the following proposition:

Proposition 2.2. *A given orthogonal multiwavelet system of multiplicity $r = 2$ has a GMP order of at least $(1, 1, 1)$ if and only if $\widehat{\boldsymbol{H}}(0)$ is singular.*

Proof. Sufficient part: Choose an orthogonal matrix $\boldsymbol{U}$ such that the vector $\boldsymbol{U}\widehat{\boldsymbol{\Phi}}(0)$ is parallel to $\boldsymbol{e}(0) = (1, 1)^T$. Clearly, from (2.25) and (2.26), we know that the GMP order components d_1 and d_3 are at least 1. In addition, suppose that $\widehat{\boldsymbol{H}}^{\sharp}(0) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then from (2.23) and (2.25), we have $a = d, b = c$. On the other hand, since $\widehat{\boldsymbol{H}}(0)$ is singular and

$$\widehat{\boldsymbol{H}}^{\sharp}(0)\boldsymbol{e}(0) = \boldsymbol{e}(0), \tag{2.31}$$

we have $a = b = c = d = 1/2$, which implies that $\widehat{\boldsymbol{H}}^{\sharp}(0)\boldsymbol{e}(\pi) = \boldsymbol{0}$. Hence, $d_2 \geq 1$. The proof for the necessary part is obvious from Proposition 2.1. □

The above discussions can now be summarized into the following step-by-step procedure to determine the GMP order of a given multiwavelet system:

Step 1 : (i) Compute the normalized 1-eigenvector, $\boldsymbol{w}$, of $\widehat{\boldsymbol{H}}(0)$.

(ii) Find an orthogonal matrix $\boldsymbol{U}$ such that $\boldsymbol{U}\boldsymbol{w} = \frac{1}{\sqrt{r}}\boldsymbol{e}(0)$.

Step 2 : Compute the transformed multifilters, $\boldsymbol{H}_k{}^{\sharp} = \boldsymbol{U}\boldsymbol{H}_k\boldsymbol{U}^T$, $\boldsymbol{G}_k{}^{\sharp} = \boldsymbol{U}\boldsymbol{G}_k\boldsymbol{U}^T$, $k \in \mathbb{Z}$.

Step 3 : Check for the highest GMP order (d_1, d_2, d_3) of $\{\boldsymbol{H}^{\sharp}, \boldsymbol{G}^{\sharp}\}$ using (2.22).

In general, a given multiwavelet system that satisfies (2.9)–(2.11) for perfect reconstruction may *not* necessarily possess the GMPs as outlined in Definition 2.1. For example, the well-known **GHM** multiwavelet [40, 43] does *not* directly satisfy most of the above good multifilter properties. From a practical signal compression viewpoint, we would consider such a multifilter system to be "ill-designed." Some simulation results in Section 2.5 later will further support our argument. An elaborate investigation on the incorporation of the above good filter characteristics directly into the construction of multiwavelet filters can be found in [10, 11, 13].

2.3.3 Symmetric-Antisymmetric Orthonormal Multifilters

This subsection introduces a new class of symmetric-antisymmetric orthonormal multifilters (SAOMFs) that possess GMPs. Here we will focus on a class of symmetric-antisymmetric orthonormal multiwavelets (SAOMWs) with multiplicity $r = 2$, and whose members have finite and real-valued matrix lowpass sequences $\{\boldsymbol{H}_k\}_{k=0}^{N}$ satisfying the following:

$$(i) \qquad \boldsymbol{H}_0 \text{ and } \boldsymbol{H}_N \text{ are non-zero matrices.} \tag{2.32}$$

$$(ii) \qquad \boldsymbol{H}_k = \boldsymbol{S}\boldsymbol{H}_{N-k}\boldsymbol{S}, \quad k = 0, 1, \ldots, N, \text{ where } \boldsymbol{S} = \mathrm{diag}(1, -1). \tag{2.33}$$

$$(iii) \qquad \widehat{\boldsymbol{H}}(0) = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix}, \quad |\lambda| < 1. \tag{2.34}$$

Collectively, we refer to the above conditions as **Condition SA** for easy referencing. The second condition (2.33) implies that the corresponding multiscaling functions form a symmetric-antisymmetric pair, as shown in [33]; i.e., $\phi_i(x) = (-1)^{i-1}\phi_i(L-x), \quad i = 1, 2$. The orthonormality of $\boldsymbol{\phi}$ also implies that $\widehat{\phi}_1(0) = 1$ and $\widehat{\phi}_2(0) = 0$. The third condition (2.34) is a necessary condition [36] for any matrix lowpass filter satisfying (2.32) and (2.33) to generate a MRA of $L^2(\mathbb{R})$.

By applying (2.29) to (2.33), we have

$$\boldsymbol{H}^{\sharp}{}_k = \boldsymbol{A}\boldsymbol{H}^{\sharp}{}_{N-k}\boldsymbol{A}, \quad k = 0, 1, \ldots, N.$$

In addition, the relationship between the equivalent scalar filters and the associated multifilter in (2.18) gives the following augmented matrix

$$\left[\boldsymbol{H}^{\sharp}{}_0 \quad \boldsymbol{H}^{\sharp}{}_1 \quad \ldots \quad \boldsymbol{H}^{\sharp}{}_N\right] = \begin{bmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \end{bmatrix},$$

where

$$h_1(\ell) = h_2(2N+1-\ell), \quad \ell = 0, 1, \ldots, 2N+1. \tag{2.35}$$

From Proposition 2.2, an orthogonal multiwavelet system with a GMP order (1,1,1) will have

$$\widehat{\boldsymbol{H}}^{\sharp}(0) = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \tag{2.36}$$

which implies that $\lambda = 0$ in (2.34). In other words, we can now construct the matrix lowpass sequences $\{\boldsymbol{H}_k\}_{k=0}^{N}$ by means of constructing $\boldsymbol{h}_1$. Since we have established in [12] the relationships between length-$2N$ and length-$(2N-1)$ multifilters that satisfy Condition SA, we will only focus on the construction of length-$2N$ multifilters below.

To further facilitate the following exposition, we first review the concept of polyphase matrix factorization of a two-channel orthonormal filter bank, as proposed by Vaidyanathan *et al.* [113, 114]. Let $H_0^{(N)}(z)$ and $H_1^{(N)}(z)$ be the z-transform of the lowpass and highpass filters associated with a two-channel orthonormal wavelet filter of length $2N$, $\{h_k\}_{k=0}^{2N-1}$, $\{g_k\}_{k=0}^{2N-1}$. It is called a power complementary CQF if $|H_0^{(N)}(z)|^2 + |H_1^{(N)}(z)|^2 = 2$ on

the unit circle $|z| = 1$, and $H_1^{(N)}(z) = -z^{-2N+1}H_0^{(N)}(-z^{-1})$. We can express $H_0^{(N)}(z) = \sum_{k=0}^{2N-1} h_k z^{-k}$ in terms of its polyphase components as $H_{00}^{(N)}(z^2) + z^{-1}H_{01}^{(N)}(z^2)$, where $H_{00}^{(N)}(z) = \sum_{k=0}^{N-1} h_{2k} z^{-k}$ and $H_{01}^{(N)}(z) = \sum_{k=0}^{N-1} h_{2k+1} z^{-k}$ are, respectively, the even- and odd-phases of $\{h_k\}_{k=0}^{2N-1}$. Similarly, the polyphase components of $H_1^{(N)}(z) = \sum_{k=0}^{2N-1} g_k z^{-k} = H_{10}^{(N)}(z^2)+z^{-1}H_{11}^{(N)}(z^2)$ are given by $H_{10}^{(N)}(z) = \sum_{k=0}^{N-1} g_{2k} z^{-k}$ and $H_{11}^{(N)}(z) = \sum_{k=0}^{N-1} g_{2k+1} z^{-k}$, where $g_k = (-1)^{k+1} h_{2N-1-k}$, $k = 0, 1, \ldots, 2N-1$. Hence the polyphase matrix of an orthonormal filter bank can be defined as

$$\boldsymbol{H}^{(N)}(z) = \begin{bmatrix} H_{00}^{(N)}(z) & H_{10}^{(N)}(z) \\ H_{01}^{(N)}(z) & H_{11}^{(N)}(z) \end{bmatrix}, \tag{2.37}$$

where the superscript $^{(N)}$ refers to filters of length $2N$.

It was further shown in [113] that all possible two-channel perfect reconstruction filter banks with impulse response length $2N$ can be parameterized by the angle parameters, α_k, as

$$\boldsymbol{H}^{(N)}(z) = \boldsymbol{R}(\alpha_0) \prod_{k=1}^{N-1} \boldsymbol{D}(z)\boldsymbol{R}(\alpha_k), \tag{2.38}$$

where

$$\boldsymbol{D}(z) = \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \quad \text{and} \quad \boldsymbol{R}(\alpha_k) = \begin{bmatrix} \cos\alpha_k & -\sin\alpha_k \\ \sin\alpha_k & \cos\alpha_k \end{bmatrix},$$

for $k = 0, \ldots, N-1$. Note that since $\boldsymbol{H}^{(N)}(z)$ is of degree $N-1$, both $H_0^{(N)}(z)$ and $H_1^{(N)}(z)$ are of degree $2N-1$. In addition, for the filter bank to be orthonormal and have at least one vanishing moment (i.e., $H_0^{(N)}(1) = \sqrt{2}$ and $H_0^{(N)}(-1) = 0$), we also need to impose the following condition (e.g., see [105, 115]):

$$\sum_{k=0}^{N-1} \alpha_k = \frac{\pi}{4} \bmod 2\pi, \tag{2.39}$$

on the angle parameters, α_k.

As shown in [12], the class of SAOMFs with a GMP order (1,1,1) is intrinsically related to a special family of length-$4N$ orthonormal scalar CQFs [15], which in turn can be derived from some length-$2N$ orthonormal scalar CQFs. Hence in order to construct the length-$2N$ multifilters, we will first investigate the construction of the special class of length-$4N$

orthonormal scalar wavelet filters, $\{h_k\}_{k=0}^{4N-1}$, which satisfy either one of the following two conditions:

$$h_{2k+1} = (-1)^k h_{2k}, \quad k = 0, 1, \ldots, 2N-1, \tag{2.40}$$

or

$$h_{2k+1} = (-1)^{k+1} h_{2k}, \quad k = 0, 1, \ldots, 2N-1. \tag{2.41}$$

More details on the requirement of these conditions can be found in [12].

By applying the parameterized representation of (2.38) in the context of a length-$4N$ orthonormal scalar filter bank, the following two theorems will provide the necessary and sufficient conditions on the α_k's so that either (2.40) or (2.41) holds:

Theorem 2.1. *$\boldsymbol{H}^{(2N)}(z)$ is the polyphase matrix of a length-$4N$ orthonormal filter bank which satisfies* (2.40) *if and only if $\boldsymbol{H}^{(2N)}(z)$ has the form*

$$\boldsymbol{H}^{(2N)}(z) = \boldsymbol{R}(\alpha_0) \prod_{k=1}^{2N-1} \boldsymbol{D}(z)\boldsymbol{R}(\alpha_k), \tag{2.42}$$

where the angle parameters, α_k, satisfy the following conditions:

$$\alpha_0 = \pi/4, \quad \alpha_{2k} = 0 \bmod 2\pi, \quad \textit{for} \quad k = 1, 2, \ldots, N-1, \tag{2.43}$$

and

$$\sum_{k=0}^{N-1} \alpha_{2k+1} = 0 \bmod 2\pi. \tag{2.44}$$

Proof: See Appendix A.1.

A closer observation shows that a filter satisfying (2.41) can, in fact, be obtained by 'flipping' or 'reversing' the order of the filter satisfying (2.40), and vice versa. Hence, the following theorem follows directly from Theorem 2.1:

Theorem 2.2. *$\boldsymbol{H}^{(2N)}(z)$ is the polyphase matrix of a length-$4N$ orthonormal filter bank which satisfies* (2.41) *if and only if $\boldsymbol{H}^{(2N)}(z)$ has the form*

$$\boldsymbol{H}^{(2N)}(z) = z^{-2N+1} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \boldsymbol{R}(\alpha_0) \prod_{k=1}^{2N-1} \boldsymbol{D}(z^{-1})\boldsymbol{R}(\alpha_k), \tag{2.45}$$

where the angle parameters α_k, $k = 0, \ldots, 2N-1$, satisfy (2.43) *and* (2.44).

Proof: The proof is similar to that in Appendix A.1, and hence can be shown easily. □

Although we can always construct the length-$4N$ scalar CQF from scratch using such methods as spectral factorization or lattice factorization, we will present here a simple method for *direct* construction starting from any existing length-$2N$ scalar CQFs, as described in the following lemma:

Lemma 2.1. *Let $\{h_k\}_{k=0}^{2N-1}$ be a length-$2N$ scalar CQF. Construct a length-$4N$ sequence $\{b_k\}_{k=0}^{4N-1}$ satisfying either (2.40) or (2.41) as follows:*

$$b_{4k} = \frac{1}{2}(h_{2k} - \tau h_{2k+1}), \quad b_{4k+2} = \frac{1}{2}(h_{2N-2-2k} + \tau h_{2N-1-2k}), \quad k = 0, 1, \ldots, N-1. \tag{2.46}$$

Then this length-$4N$ sequence is a scalar CQF.

Proof: The proof is simple as we only need to verify that the length-$4N$ sequence satisfies the following two basic conditions:

(i) $\sum_k b_k b_{k+2i} = 2\delta_{0,i}, \quad i \in \mathbb{Z}$; and

(ii) $\sum_k (-1)^k b_k = 0.$ □

By Lemma 2.1 and the relation (2.35), we can now construct the matrix lowpass filter, $\boldsymbol{H}_k$, from the special length-$4N$ orthonormal scalar filter, as follows:

$$\begin{aligned} \boldsymbol{H}_{2k} &= \frac{1}{2}\begin{bmatrix} b_{4k} & b_{4N-4k-2} \\ -b_{4k} & b_{4N-4k-2} \end{bmatrix}\begin{bmatrix} 1-\tau & -(1+\tau) \\ 1+\tau & 1-\tau \end{bmatrix}, \\ \boldsymbol{H}_{2k+1} &= \frac{1}{2}\begin{bmatrix} b_{4k+2} & b_{4N-4k-4} \\ -b_{4k+2} & b_{4N-4k-4} \end{bmatrix}\begin{bmatrix} 1+\tau & -(1-\tau) \\ 1-\tau & 1+\tau \end{bmatrix}, \quad \tau = \pm 1. \end{aligned} \tag{2.47}$$

For a multiwavelet system, we will also need to construct the matrix highpass filter once the matrix lowpass filter is known. In the case of scalar wavelets, the procedure of constructing the highpass sequence from a lowpass sequence is simple: order reverse and sign alternate the lowpass sequence. For multiwavelets, the construction of the matrix highpass sequence, $\{\boldsymbol{G}_k\}$, is more complicated; nonetheless, a general solution using a matrix extension technique has been proposed by Lawton, Lee, and Shen [72]. However, for orthonormal multiwavelets satisfying Condition SA, we have introduced in [12] two explicit

formulations for constructing the matrix highpass sequence, $\{\boldsymbol{G}_k\}$, *directly* in terms of the matrix lowpass sequence, $\{\boldsymbol{H}_k\}$. The following shows one of the two direct constructions; the other can be found in Proposition 2 of [12]:

Proposition 2.3. *Let the lowpass sequence* $\{\boldsymbol{H}_k\}_{k=0}^{2N-1}$ *be a matrix CQF satisfying Condition SA. If* $\boldsymbol{H}_k\boldsymbol{A}\boldsymbol{H}_{2N-1-k-2i}^T, k = 0, 1, \ldots, N-i-1, i = 0, 1, \ldots, N-1$, *are symmetric matrices, then the highpass sequence* $\{\boldsymbol{G}_k\}_{k=0}^{2N-1}$ *can be obtained from* $\{\boldsymbol{H}_k\}_{k=0}^{2N-1}$ *as follows:*

$$\boldsymbol{G}_k = (-1)^{k+1}\boldsymbol{H}_{2N-1-k}\boldsymbol{A}, \quad k = 0, 1, \ldots, 2N-1. \tag{2.48}$$

Proof: See Appendix A.2.

Furthermore, if the transition operator satisfies Condition E, then $\{\boldsymbol{H}_k, \boldsymbol{G}_k\}_{k=0}^{2N-1}$ generates a length-$2N$ orthonormal multiwavelet system satisfying Condition SA. In fact, the proposed direct construction technique has shown to be able to easily construct the multiwavelet functions of some earlier multiwavelet filters satisfying Condition SA, such as Chui and Lian's [33]. In [12], we showed how an even-length ($2N$) matrix CQF that satisfies Condition SA can be derived directly from an odd-length ($2N-1$) matrix CQF; and vice versa. We further proved in [15] that, for any even-length SAOMF, we can always find a corresponding odd-length SAOMF such that the implementation of discrete multiwavelet transforms using either the even- or odd-length SAOMF produces identical output for a given input signal by choosing a properly designed pre-filter.

As examples on the construction of SAOMFs, we will illustrate the process of constructing a parameterized length-4 multiwavelet system from a length-4 orthonormal scalar wavelet. Appendix B.1 presents an example starting from the popular Daubechies' length-4 scalar wavelet [37]. The parameter ν in the parameterized family of length-4 SAOMWs (which we denote as **SA4**) can be varied to incorporate some desirable multifilter design properties such as GMPs. The performances of a few members of **SA4** are analyzed in Section 2.5. A parameterized family of the length-6 SAOMF (**SA6**) is also given in Appendix B.2. Examples of other SAOMFs with longer filter lengths can be found in [15].

2.3.4 Symmetric-Antisymmetric Biorthogonal Multifilters

This subsection extends the above theory to the construction of a new class of symmetric-antisymmetric biorthogonal multifilters (SABMFs) that possess GMPs. Similarly, we will focus on a class of symmetric-antisymmetric biorthogonal multiwavelets (SABMWs) with multiplicity $r = 2$. We define the GMPs for a SABMW system as follows:

Definition 2.2. *A biorthogonal multiwavelet system is considered to have a GMP order*[4] $(d_2, \widetilde{d}_2)$ *if its refinement masks* $\widehat{\boldsymbol{H}}(0)$ *and* $\widehat{\widetilde{\boldsymbol{H}}}(0)$*, and its sets of equivalent scalar lowpass filters,* $\mathbf{h}_k$ *and* $\widetilde{\mathbf{h}}_k$*, associated with the matrix lowpass filters* $\boldsymbol{H}_k^{\sharp}$ *and* $\widetilde{\boldsymbol{H}}_k^{\sharp}$*, respectively, possess the following properties:*

(i) Both $\widehat{\boldsymbol{H}}(0)$ *and* $\widehat{\widetilde{\boldsymbol{H}}}(0)$ *have a common right 1-eigenvector,*

(ii) $\widehat{\mathbf{h}}_k^{(\nu)}(\pi) = 0, \quad \nu = 0, 1, ..., d_2 - 1,$

(iii) $\widehat{\widetilde{\mathbf{h}}}_k^{(\nu)}(\pi) = 0, \quad \nu = 0, 1, ..., \widetilde{d}_2 - 1,$

for both $k = 1, 2$*, where the superscript* $^{(\nu)}$ *denotes the* ν^{th}*-derivative, and* $d_2, \widetilde{d}_2 \geq 1$*. The transition matrix* $\boldsymbol{U}$ *is then determined by the property that the vector* $\boldsymbol{U}\widehat{\boldsymbol{\phi}}(0)$ *is parallel to the vector* $[1, 1]^T$*.*

In a biorthogonal setting, we say that the primal and dual finite-length matrix sequences $\{\boldsymbol{H}_k\}_{k=N^\ell}^{N^u}$ and $\{\widetilde{\boldsymbol{H}}_k\}_{k=\widetilde{N}^\ell}^{\widetilde{N}^u}$ satisfy Condition SA if the following conditions hold:

$$\boldsymbol{H}_k = \boldsymbol{S}\boldsymbol{H}_{N^u+N^\ell-k}\boldsymbol{S}, \quad \text{for} \quad k = N^\ell, \ldots, N^u. \tag{2.49}$$

$$\widetilde{\boldsymbol{H}}_k = \boldsymbol{S}\widetilde{\boldsymbol{H}}_{\widetilde{N}^u+\widetilde{N}^\ell-k}\boldsymbol{S}, \quad \text{for} \quad k = \widetilde{N}^\ell, \ldots, \widetilde{N}^u. \tag{2.50}$$

$$\widehat{\boldsymbol{H}}(0) = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix}, \quad \widehat{\widetilde{\boldsymbol{H}}}(0) = \begin{bmatrix} 1 & 0 \\ 0 & \widetilde{\lambda} \end{bmatrix}, \quad \text{where} \quad |\lambda|, |\widetilde{\lambda}| < 1. \tag{2.51}$$

Here (2.49) and (2.50) imply the symmetry and antisymmetry of the multiscaling functions

[4] It is important not to confuse the two-parameter GMP order for SABMWs with the three-parameter GMP order for SAOMWs. In the SABMWs case, the two parameters refer to the corresponding GMP order of the primal and dual refinement masks.

[33]; i.e.,

$$\phi_k(x) = (-1)^{k-1}\phi_k(N^u + N^\ell - x), \quad \widetilde{\phi}_k(x) = (-1)^{k-1}\widetilde{\phi}_k(\widetilde{N}^u + \widetilde{N}^\ell - x), \quad k = 1, 2. \tag{2.52}$$

Equation (2.51) entails that $\widehat{\boldsymbol{H}}(0)$ and $\widehat{\widetilde{\boldsymbol{H}}}(0)$ satisfy Condition E, which ensures the existence of the solutions of the corresponding matrix refinement equations. Further, the Condition SA implies that

$$\widehat{\boldsymbol{H}}(0)[1,0]^T = [1,0]^T; \quad \widehat{\widetilde{\boldsymbol{H}}}(0)[1,0]^T = [1,0]^T. \tag{2.53}$$

$$\boldsymbol{H}_k = \boldsymbol{S}\boldsymbol{H}_{N^u+N^\ell-k}\boldsymbol{S} \Longleftrightarrow \boldsymbol{H}_k^\sharp = \boldsymbol{A}\boldsymbol{H}_{N^u+N^\ell-k}^\sharp\boldsymbol{A}. \tag{2.54}$$

$$\widetilde{\boldsymbol{H}}_k = \boldsymbol{S}\widetilde{\boldsymbol{H}}_{\widetilde{N}^u+\widetilde{N}^\ell-k}\boldsymbol{S} \Longleftrightarrow \widetilde{\boldsymbol{H}}_k^\sharp = \boldsymbol{A}\widetilde{\boldsymbol{H}}_{\widetilde{N}^u+\widetilde{N}^\ell-k}^\sharp\boldsymbol{A}, \tag{2.55}$$

where

$$\boldsymbol{A} := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

By (2.51),

$$\widehat{\boldsymbol{H}}^\sharp(0) = \frac{1}{2}\begin{bmatrix} 1+\lambda & 1-\lambda \\ 1-\lambda & 1+\lambda \end{bmatrix}, \quad \widehat{\widetilde{\boldsymbol{H}}}^\sharp(0) = \frac{1}{2}\begin{bmatrix} 1+\widetilde{\lambda} & 1-\widetilde{\lambda} \\ 1-\widetilde{\lambda} & 1+\widetilde{\lambda} \end{bmatrix}. \tag{2.56}$$

Equations (2.53) and (2.56) lead to the following proposition:

Proposition 2.4. *A biorthogonal multifilter (BMF) satisfying Condition SA possesses a GMP order* $(1,1)$ *if and only if both* $\widehat{\boldsymbol{H}}(0)$ *and* $\widehat{\widetilde{\boldsymbol{H}}}(0)$ *are singular; i.e.,* $\lambda = \widetilde{\lambda} = 0$.

Next we address the question of constructing BMFs that satisfy Condition SA and have a GMP order of at least $(1,1)$. Equations (2.54) and (2.55) imply that the equivalent scalar filters, $\mathtt{h}_1^\sharp$ and $\mathtt{h}_2^\sharp$, are time-reversed versions of one another; so are $\widetilde{\mathtt{h}}_1^\sharp$ and $\widetilde{\mathtt{h}}_2^\sharp$. Hence the resulting matrix lowpass filters $\{\boldsymbol{H}_k^\sharp\}_{k=N^\ell}^{N^u}$ and $\{\widetilde{\boldsymbol{H}}_k^\sharp\}_{k=\widetilde{N}^\ell}^{\widetilde{N}^u}$ have the form

$$\boldsymbol{H}_k^\sharp = \begin{bmatrix} a_{2k} & a_{2k+1} \\ a_{2(N^\ell+N^u-k)+1} & a_{2(N^\ell+N^u-k)} \end{bmatrix}, \quad k = N^\ell, \ldots, N^u, \tag{2.57}$$

and

$$\widetilde{\boldsymbol{H}}_k^\sharp = \begin{bmatrix} \widetilde{a}_{2k} & \widetilde{a}_{2k+1} \\ \widetilde{a}_{2(\widetilde{N}^\ell+\widetilde{N}^u-k)+1} & \widetilde{a}_{2(\widetilde{N}^\ell+\widetilde{N}^u-k)} \end{bmatrix}, \quad k = \widetilde{N}^\ell, \ldots, \widetilde{N}^u. \tag{2.58}$$

The following conditions on scalar sequences $\{a_k\}_{k=2N^\ell}^{2N^u+1}$ and $\{\widetilde{a}_k\}_{k=2\widetilde{N}^\ell}^{2\widetilde{N}^u+1}$ are derived from the PR condition (2.9):

$$\sum a_k \widetilde{a}_{k+4i} = \delta_i, \quad i \in \mathbb{Z}, \tag{2.59}$$

$$\sum a_k \widetilde{a}_{2\widetilde{N}^\ell+(2\widetilde{N}^u+1)-k-4i} = 0, \quad i \in \mathbb{Z}, \tag{2.60}$$

$$\sum a_k \widetilde{a}_{2\widetilde{N}^\ell+(2\widetilde{N}^u+1)-k+4i} = 0, \quad i \in \mathbb{Z}. \tag{2.61}$$

Clearly if two scalar sequences satisfy Conditions (2.59)-(2.61), then the corresponding matrix CQF can be constructed via (2.57) and (2.58). Using (2.57), (2.58), and Proposition 2.4, we see that if the scalar sequences also satisfy

$$\sum_k a_{2k} = \sum_k a_{2k+1} = \frac{1}{\sqrt{2}} \quad \text{and} \quad \sum_k \widetilde{a}_{2k} = \sum_k \widetilde{a}_{2k+1} = \frac{1}{\sqrt{2}}, \tag{2.62}$$

then the corresponding BMF possesses a GMP order (1,1).

The corresponding highpass multifilters, $\{\boldsymbol{G}_k\}_{k=M^\ell}^{M^u}$ and $\{\widetilde{\boldsymbol{G}}_k\}_{k=\widetilde{M}^\ell}^{\widetilde{M}^u}$, are constructed either directly from the lowpass filters or from solving the PR conditions (2.5)–(2.7). Since symmetry/antisymmetry is also desired for the highpass multifilters, we will require them to satisfy

$$\boldsymbol{G}_k = \boldsymbol{S}\boldsymbol{G}_{M^u+M^\ell-k}\boldsymbol{S}, \quad \text{for} \quad k = M^\ell, \ldots, M^u, \tag{2.63}$$

$$\widetilde{\boldsymbol{G}}_k = \boldsymbol{S}\widetilde{\boldsymbol{G}}_{\widetilde{M}^u+\widetilde{M}^\ell-k}\boldsymbol{S}, \quad \text{for} \quad k = \widetilde{M}^\ell, \ldots, \widetilde{M}^u, \tag{2.64}$$

which are similar to those required for the lowpass multifilters. The following theorem shows that two parameters can be associated with these highpass multifilters for the purpose of filter optimization:

Theorem 2.3. *Suppose that $\{\boldsymbol{H}_k\}$ and $\{\widetilde{\boldsymbol{H}}_k\}$ are finite-length lowpass multifilters of a SABMF system, and that the corresponding finite-length highpass multifilters $\{\boldsymbol{G}_k\}$ and $\{\widetilde{\boldsymbol{G}}_k\}$ satisfy* (2.63) *and* (2.64). *Then the sequences $\{\boldsymbol{G}_k^{\triangleright}\}$ and $\{\widetilde{\boldsymbol{G}}_k^{\triangleright}\}$ where*

$$\boldsymbol{G}_k^{\triangleright} = \begin{bmatrix} \tau & 0 \\ 0 & \delta \end{bmatrix} \boldsymbol{G}_k, \quad \widetilde{\boldsymbol{G}}_k^{\triangleright} = \begin{bmatrix} 1/\tau & 0 \\ 0 & 1/\delta \end{bmatrix} \widetilde{\boldsymbol{G}}_k, \quad \tau, \delta \in \mathbb{R} \setminus \{0\},$$

also satisfy (2.63) *and* (2.64), *and that they too are highpass multifilters for the same lowpass multifilters $\{\boldsymbol{H}_k\}$ and $\{\widetilde{\boldsymbol{H}}_k\}$.*

Proof: The proof is straightforward and involves only verifying the PR conditions (2.5)–(2.7). □

In the following, we present a method for constructing SABMFs with GMPs. A step-by-step procedure to construct a SABMF with GMP order (d_1, d_2) and approximation order (p_1, p_2) is given below:

- **Step 1:** Let scalar sequences $\{a_k\}_{k=2N^\ell}^{2N^u+1}$ and $\{\widetilde{a}_k\}_{k=2\widetilde{N}^\ell}^{2\widetilde{N}^u+1}$ be related to the matrix lowpass sequences via (2.57) and (2.58). Construct the system of nonlinear equations from (2.59)-(2.61) and (2.56) with $\lambda = \widetilde{\lambda} = 0$.

- **Step 2:** Augment the system with equations for higher GMP order (d_1, d_2) using Definition 1.

- **Step 3:** Augment the system with equations for higher approximation order (p_1, p_2) using (2.14).

- **Step 4:** Seek a solution or solutions of the nonlinear equations using symbolic packages such as *Mathematica* or *Maple*.

- **Step 5:** Solve the PR equations (2.9)-(2.11) for the corresponding matrix highpass sequences. Apply Theorem 2.3 to introduce the two highpass parameters.

A good strategy for choosing d_1, d_2, p_1, p_2 for a BMF of a specific length is to pick its values such that the resulting matrix lowpass filter has one free parameter available for further optimization (such as imposing the constraint of having the magnitude responses of the equivalent scalar filters as close as possible to the ideal brick-wall filter).

Following the above procedure, a large number of SABMFs of varying lengths have been constructed; the relatively short length 4/4 and 5/5 (**BSA(4/4)** and **BSA(5/5)**) SABMFs are presented in Appendices B.3 and B.4. They are chosen not just for ease of exposition but also for their good compression performances with relatively low computational costs, as will be investigated in greater detail in Section 2.5. The method described above involves solving a system of nonlinear equations for the matrix lowpass and highpass filters of a SABMF. The difficulty level of such a task increases with the lengths of the filter sought, more so if the solution desired is to be in symbolic form (in contrast to being numeric). Symbolic algebra packages, such as *Maple V*, are able to provide symbolic solutions for up

to length 6/6 with ease and time efficiency. However beyond this length, it is more effective to adopt an alternative means of obtaining parameterized filter solutions, namely, the lifting scheme proposed by Goh *et al.* [47]. Longer SABMFs such as **BSA(8/6)** and **BSA(7/9)** were obtained from parameterized solutions generated using the lifting scheme, and then with the parameters utilized for achieving GMPs, and closest proximity to the brick-wall filters. The filter coefficients of the **BSA(6/6)**, **BSA(8/6)**, and **BSA(7/9)** SABMFs are given in Appendix B.5.

2.4 The Proposed Discrete Multiwavelet Transforms

The previous section has introduced the concept of GMPs and showed the construction of SAOMWs and SABMWs. Having good multifilters alone, however, does not directly result in good application of the multifilters. Therefore, this section focuses on the development of an efficient and effective framework for discrete multiwavelet decomposition and reconstruction of a given discrete-time signal. In particular, we will address the important issue of *multiwavelet initialization* or *pre-filtering*, which concerns the generation of multiple (vector) input streams from a given scalar source stream. In this dissertation, we propose to develop a generalized paradigm for discrete multiwavelet transforms, which works well with any given multiwavelet system, regardless of whether it possesses GMPs or not. In short, the proposed multiresolution framework for multiwavelet transforms embodies the following properties: orthogonality, low complexity, robustness, and preserving a compact (non-redundant) representation of the input signal.

2.4.1 Some Problems of Multiwavelet Initialization

Unlike scalar wavelets in which Mallat's pyramid algorithms [84] have provided a solution for good signal decomposition and reconstruction, a good framework for the application of multiwavelets, however, is still being researched. Several proposals have been made to advance the development of this area. For example, Xia *et al.* [121] have proposed an algorithm to compute the multiwavelet (vector-valued) transform coefficients by adding a

proper pre-filter (pre-processing matrix) before the vector filter banks that generate multiwavelets. They introduced an interpolation-based technique to obtain the (vector) input streams for the **GHM** multiwavelet. The method can be viewed as a discrete vector-valued wavelet transform for certain discrete-time vector-valued signals. They also presented some numerical results to indicate that the energy compaction for discrete multiwavelet transforms may be better than that for conventional discrete wavelet transforms. In addition to their work, Strela *et al.* [107] have investigated the construction of "constrained" multiwavelets for filtering two-dimensional signals, and applied them to both image denoising and image compression. The above proposals, however, suffer from a major shortcoming — they are only restricted to the **GHM** case, and hence, are likely to be *not robust* in general. As pointed out in [121], the corresponding post-processing matrix, which requires that the pre-processing matrix to be invertible, may not exist for other multiwavelets.

Liang *et al.* [79], on the other hand, have also applied multiwavelets to image coding, and proposed an inter-subband prediction scheme to exploit the correlations resulting from the direct application of the **GHM** multiwavelet. Recently Xia [122] improved upon the earlier work in [121] with a new method for pre-filtering and was able to apply it to another length-3 orthogonal multiwavelet filter [33]. Nevertheless, the proposed transform resulted in an *overcomplete or redundant representation* of the original signal, which increased the size of the input data by a factor of r (where r is the multiplicity of the multiwavelets) after pre-processing. Obviously, this does not only require more computations, but also goes against the basic idea of data compression[5]. Clearly, there is a pressing need to devise a general, non-redundant, and reversible pre-filtering technique for efficient multiwavelet transforms. A recent work in this direction can be found in [50] where an orthogonal and approximation-order preserving pre-filter was presented. The computational complexity of the pre-filter, however, increases with the approximation order of the multiwavelet filter.

[5] An overcomplete representation results in an expansion of the original signal, which will eventually complicate the coding of positional information of transform coefficients although the zero-order entropy could be lower, or it may have higher energy compaction.

2.4.2 Vector-Valued Multiresolution Analysis and Synthesis

This subsection presents the basic idea of a vector-valued multiresolution framework for discrete multiwavelet transforms (decomposition and reconstruction algorithms) of discrete-time signals. For simplicity, but without loss of generality, we consider the case of a two-channel transform using a biorthogonal multiwavelet system with multiplicity $r = 2$ in the following exposition.

Consider again the two-scale equations (2.1)–(2.4). They can be realized using a multiwavelet filter bank concept [107], much like the well-known multiresolution algorithms for scalar wavelets [84]. Denote $\boldsymbol{\phi}_{\ell,k} := [\phi_{1,\ell,k}, \phi_{2,\ell,k}]^T$ and $\widetilde{\boldsymbol{\phi}}_{\ell,k} := [\widetilde{\phi}_{1,\ell,k}, \widetilde{\phi}_{2,\ell,k}]^T$ where $\phi_{\nu,\ell,k} = 2^{\ell/2}\phi_\nu(2^\ell x - k)$ and $\widetilde{\phi}_{\nu,\ell,k} = 2^{\ell/2}\phi_\nu(2^\ell x - k)$, $\nu = 1, 2$. Likewise, we define $\boldsymbol{\psi}_{\ell,k}$, $\widetilde{\boldsymbol{\psi}}_{\ell,k}$ and their component functions $\psi_{\nu,\ell,k}$ and $\widetilde{\psi}_{\nu,\ell,k}$, $\nu = 1, 2$. Since $\widetilde{V}_\ell \subset \widetilde{V}_{\ell+1}$ for $\ell \in \mathbb{Z}$, and $\overline{\bigcup_{\ell\in\mathbb{Z}} \widetilde{V}_\ell} = L^2(\mathbb{R})$, then for a sufficiently large ℓ, say L, we can assume that a given signal $f \in \widetilde{V}_L$; this allows the signal f to be sufficiently represented as a linear combination of the vector basis functions $\widetilde{\boldsymbol{\phi}}_{L,k}$, and then be further decomposed into its constituents by projecting it onto some nested basis functions, $\widetilde{\boldsymbol{\phi}}_{\ell,k}$, for some $\ell < L$. Mathematically, we have

$$f(x) = \sum_{k\in\mathbb{Z}} \boldsymbol{p}_{L,k}^T \widetilde{\boldsymbol{\phi}}_{L,k}(x) \tag{2.65}$$

$$= \sum_{k\in\mathbb{Z}} \boldsymbol{p}_{L_0,k}^T \widetilde{\boldsymbol{\phi}}_{L_0,k}(x) + \sum_{L_0\le\ell<L} \sum_{k\in\mathbb{Z}} \boldsymbol{q}_{\ell,k}^T \widetilde{\boldsymbol{\psi}}_{\ell,k}(x), \tag{2.66}$$

for a fixed integer $L_0 < L$. Further denote $\boldsymbol{p}_{\ell,k} := [p_{1,\ell,k}, \quad p_{2,\ell,k}]^T$, and $\boldsymbol{q}_{\ell,k} := [q_{1,\ell,k}, \quad q_{2,\ell,k}]^T$. The projection (i.e., the scaling and multiwavelet) coefficients are given by

$$p_{\nu,\ell,k} = \int f(x)\phi_{\nu,\ell,k}(x)dx, \tag{2.67}$$

$$q_{\nu,\ell,k} = \int f(x)\psi_{\nu,\ell,k}(x)dx, \quad \nu = 1, 2. \tag{2.68}$$

By equations (2.1)–(2.4), one can derive the *multiwavelet decomposition algorithm* as follows:

$$\boldsymbol{p}_{\ell-1,k} = \sum_{m\in\mathbb{Z}} \boldsymbol{H}_{m-2k}\boldsymbol{p}_{\ell,m}, \tag{2.69}$$

$$\boldsymbol{q}_{\ell-1,k} = \sum_{m\in\mathbb{Z}} \boldsymbol{G}_{m-2k}\boldsymbol{p}_{\ell,m}, \quad \ell = L, L-1, \ldots, L_0+1. \tag{2.70}$$

Clearly (2.69) and (2.70) are recursive in nature; this constitute the crux of multiscale or multiresolution analysis of a signal. As the matrix filter $\boldsymbol{H}$ has lowpass properties, the application of (2.69) essentially smooths out the input signal and generates a coarser (approximation) version signal. The matrix highpass filter $\boldsymbol{G}$, on the other hand, extracts the high-frequency components and attenuates the low-frequency components of the input signal to produce a detail version of the signal. It is also worth noting from (2.69) and (2.70) that both $\boldsymbol{H}$ and $\boldsymbol{G}$ actually operate, in complement with one another, on the same input signal at a particular scale. With careful design, this ensures that the portion of the input signal that is rejected by filtering with $\boldsymbol{H}$ will then be captured by filtering with $\boldsymbol{G}$, thus making perfect reconstruction possible. Furthermore, since the filtered signals have a narrower frequency bandwidth, they can be downsampled appropriately to preserve a compact representation of the input signal. A more vigorous treatment on the topics of critical sampling, multirate filter banks, aliasing cancellation, and perfect reconstruction requirements, can be found in [114] among others.

The analysis stage practically decomposes an input composite signal into a multiresolution representation of approximation and detail components of the signal. This greatly helps in the analysis and processing of a signal; for example, one may discard or suppress some small high-frequency coefficients for noise removal, and in some signal compression applications, one can efficiently code the high-energy approximation version of the signal and ignore the low-energy detail components without much loss of information. After processing the various components of the signal, it is important to be able to reconstruct the processed signal appropriately. The synthesis stage will recombine the approximation and detail components using the following *multiwavelet reconstruction algorithm*:

$$\boldsymbol{p}_{\ell,k} = \sum_{m\in\mathbb{Z}} \widetilde{\boldsymbol{H}}_{k-2m}^{T} \boldsymbol{p}_{\ell-1,m} + \sum_{m\in\mathbb{Z}} \widetilde{\boldsymbol{G}}_{k-2m}^{T} \boldsymbol{q}_{\ell-1,m}, \quad \ell = L_0 + 1, \ldots, L. \tag{2.71}$$

The above multiwavelet decomposition and reconstruction algorithms can, in fact, be realized using a cascaded vector filter bank structure. Figure 2.2 illustrates a 1-level subband decomposition and reconstruction framework for discrete multiwavelet transforms. The left half of Figure 2.2 shows how a vector input stream is decomposed by a matrix

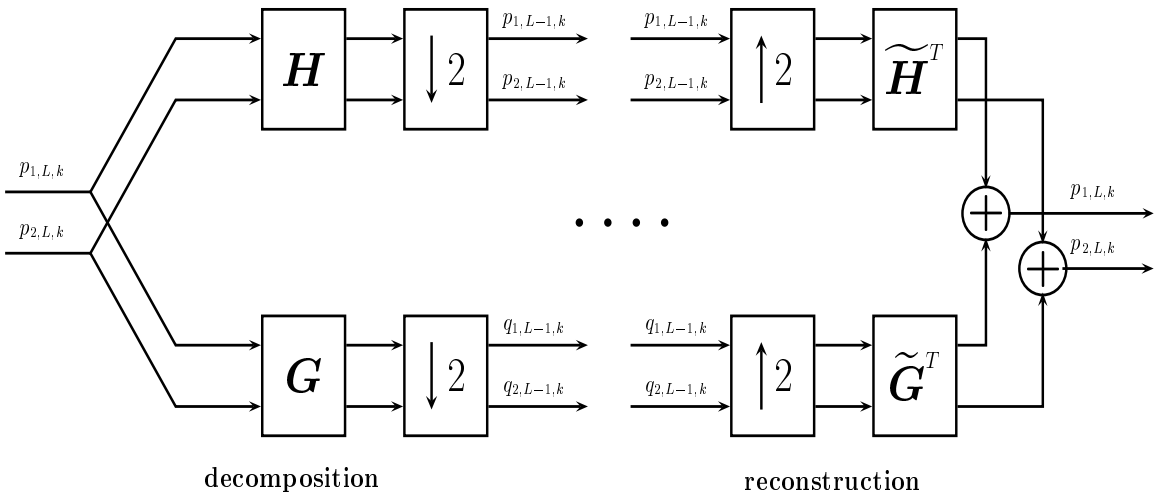


Figure 2.2: One-level of decomposition and reconstruction in the multiwavelet filter bank.

lowpass filter, $\boldsymbol{H}$, and a matrix highpass filter, $\boldsymbol{G}$, to generate the next lower resolution version of the signal. This is followed by subsampling by a factor of 2 to preserve compact representation of the input signal in a two-band filtering process. For octave-bandwidth decomposition, only the lowpass subbands will be decomposed iteratively to produce subsequent lower resolutions. An N-level decomposition will consist of a cascade of N such 1-level decompositions, each operating on the lowpass subbands of the previous resolution. The right half of Figure 2.2 depicts the corresponding 1-level multiwavelet reconstruction. The lowpass and highpass subbands are first upsampled by a factor of 2, filtered by the corresponding synthesis matrix filters, and then recombined to recover the original (but delayed) vector input signal.

2.4.3 Generalized Pre-Analysis and Post-Synthesis Multirate Filters

The previous subsection has presented both the multiresolution decomposition and reconstruction algorithms for discrete multiwavelet transforms. However, it remains an open research problem on how one can obtain the initial vector input stream from a given one-dimensional (scalar) signal such as a row or a column of an image. Referring to (2.69) and (2.70), we will note that the *initial* scaling coefficients, $\boldsymbol{p}_{L,m}$, for a fixed $L \in \mathbb{Z}$, which are representative of a discrete-time input signal $f \in \widetilde{V}_L$, are of paramount importance to begin

the analysis stage of the algorithm. From (2.67), we have

$$p_{1,L,k} = \int f(x)\phi_{1,L,k}(x)dx, \tag{2.72}$$

$$p_{2,L,k} = \int f(x)\phi_{2,L,k}(x)dx, \tag{2.73}$$

which can be approximated using some quadrature formulae. Analogously, in the context of scalar wavelet transform, the common and popular choice is using the one-point quadrature [108] where we assume that $p_{L,k} \approx f(k)$ for some sufficiently smooth signals over the local support of the scaling function[6].

Recall from Figure 2.1 (b) that the multiplexed stream $\boldsymbol{v}$ is to be fed independently into each of the equivalent scalar filters associated with a given multifilter system. On a similar note, it will be conceptually identical to consider $\boldsymbol{v}$ as the given input signal that needs to be *demultiplexed* into multiple input streams, $\boldsymbol{x}$, before they are decomposed further by the multifilters. This process of generating the multiple input streams from a single stream is known as *multiwavelet initialization* or *pre-filtering*. We denote the matrix pre-filter that precedes the recursive vector-valued decomposition process as a "pre-analysis" operator, $\mathcal{P}$. Similarly, we also need a matrix post-filter that follows the recursive vector-valued reconstruction process to multiplex the synthesized vector streams into a single output stream; we denote the filter as a "post-synthesis" operator, $\mathcal{Q}$. In our proposal, the relationship that defines the demultiplexer operation is given as a dual of the **MUX** operator defined in (2.19). For the case of a multiwavelet with multiplicity $r = 2$, we can essentially pair up the adjacent discrete-time samples of a given signal f such that $\boldsymbol{p}_{L,k} \approx [f_{2k}, f_{2k+1}]^T$. Making a similar assumption on the local smoothness of f, we have $f_{2k} \approx f_{2k+1}$; hence, $\boldsymbol{p}_{L,k} \approx \sigma[1,1]^T$ for some real constant σ. Using (2.29) and (2.31), and the assumption on the local smoothness of f, we have $\widehat{\boldsymbol{H}}(0)\boldsymbol{M}^{-1}\boldsymbol{p}_{L,k} \approx \boldsymbol{M}^{-1}\boldsymbol{p}_{L,k}$, where $\boldsymbol{M}$ is an orthogonal matrix.

Since we hope that $\boldsymbol{M}^{-1}\boldsymbol{p}_{L,k}$ is the 1-eigenvector of $\widehat{\boldsymbol{H}}(0)$, we can derive the pre-analysis and post-synthesis operators as:

[6] In this case, we have $p_{L,k} = \int f(x)\phi_{L,k}(x)dx$, where $\phi(x)$ is the scaling function associated with the scalar wavelet system

(i) Pre-analysis operator:

$$\mathcal{P} : \boldsymbol{v} \longrightarrow \boldsymbol{M}^{-1}\boldsymbol{v},$$

(ii) Post-synthesis operator:

$$\mathcal{Q} : \boldsymbol{v} \longrightarrow \boldsymbol{M}\boldsymbol{v},$$

where the orthogonal matrix $\boldsymbol{M}$ has two possible forms:

$$\boldsymbol{M} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \text{or} \quad \boldsymbol{M} = \begin{bmatrix} \sin\theta & \cos\theta \\ \cos\theta & -\sin\theta \end{bmatrix}, \tag{2.74}$$

with $\theta = -\frac{\pi}{4}$ when $\widehat{\phi}_1(0) = 0$, or otherwise $\theta = \frac{\pi}{4} - \tan^{-1}\left(\frac{\widehat{\phi}_2(0)}{\widehat{\phi}_1(0)}\right)$ for $\theta \in (-\frac{\pi}{4}, \frac{\pi}{4}]$. It is evident that the nature of the components of multiscaling functions has been taken into account in determining the initial scaling coefficients. Specifically, the pre-analysis filter is governed by the 1-eigenvector of the refinement mask $\widehat{\boldsymbol{H}}(0)$. Suppose that the input scalar signal $f(x)$ is sampled to give an even-length sequence, $x_0, \ldots, x_{n-1}$, then the initial vector-valued stream, $\boldsymbol{p}_{L,k}$, is obtained as

$$\boldsymbol{M}^{-1} \begin{bmatrix} x_{2k} \\ x_{2k+1} \end{bmatrix}. \tag{2.75}$$

It is worth noting that all SAOMFs and SABMFs have $[\widehat{\phi}_1(0), \widehat{\phi}_2(0)]^T$ in the direction of $[1, 0]^T$ since the multiscaling functions form a symmetric-antisymmetric pair. This gives $\theta = -\pi/4$ in (2.74).

From Proposition 2.2, it is apparent that if the original multiwavelet possesses GMPs, then the choice of either pre-analysis filter will still result in a GMP order of at least $(1, 1, 1)$. However the choice of a particular matrix $\boldsymbol{M}$, instead of the other matrix $\boldsymbol{M}$, can result in significantly different results in the analysis stage. We have investigated a number of desirable filter properties to help us determine the better pre-analysis filter $\boldsymbol{M}^{-1}$ for a particular multiwavelet system. For the primary application in image and video compression, we found that the following measure of deviation from the ideal "brickwall" lowpass filter is both reliable and consistent:

$$E = \sum_{\nu=1}^{2} \int_0^{\frac{\pi}{2}} (\sqrt{2} - |\widehat{\mathtt{h}_\nu^\sharp}(\omega)|)^2 d\omega + \int_{\frac{\pi}{2}}^{\pi} |\widehat{\mathtt{h}_\nu^\sharp}(\omega)|^2 d\omega,$$

where $\widehat{\mathbf{h}_1^{\natural}}(\omega)$ and $\widehat{\mathbf{h}_2^{\natural}}(\omega)$ are the equivalent scalar lowpass filters associated with the similar-transformed multifilters. The motivation for using the above objective function is the good frequency selectivity of the brickwall filter, which will eventually engender higher energy compaction for most natural images. As a result, we will select the matrix $\boldsymbol{M}$ that gives a smaller value of E as the pre-analysis filter, $\mathcal{P}$. In order to recover the original signal (without any quantization), the post-synthesis filter, $\mathcal{Q}$, must satisfy $\mathcal{Q}\mathcal{P} = \boldsymbol{I}$.

In effect, the proposed pre- and post-filters possess the following desirable properties that make possible a general and efficient framework for multiwavelet transforms. Since the orthogonal matrix $\boldsymbol{M}$ is guaranteed to have an inverse, the associated post-synthesis filter will always exist; thus the proposed multiwavelet initialization technique is *robust* to work well with any given multiwavelet systems. Furthermore, the pre- and post-filters have *low computational complexity*[7]. Specifically, multifilters that satisfy Condition SA will have a very simple matrix

$$\boldsymbol{M} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{2.76}$$

that requires minumum computations.

As the pre- and post-filters are *orthogonal*, they ensure that some desirable properties of the designed multifilters such as approximation order, regularity, and phase linearity will be preserved after pre-analysis and post-synthesis filtering. By careful analysis of (2.75), we will notice that the proposed pre- and post filters obey a multirate filtering technique that preserves a *compact or non-redundant representation* of the input signal. The vector-valued stream that is generated by the pre-analysis filter is essentially downsampled by a factor of two to maintain critical sampling of the input signal.

2.4.4 Integrated Discrete Multiwavelet Transforms

This subsection illustrates the process of integrating the proposed pre-analysis and post-synthesis multirate filters into the vector-valued multiresolution decomposition and recon-

[7] From an implementation point of view, the pre-filter can be integrated into the first level of multiresolution decomposition, and the post-filter can be combined with the last level of multiresolution reconstruction.

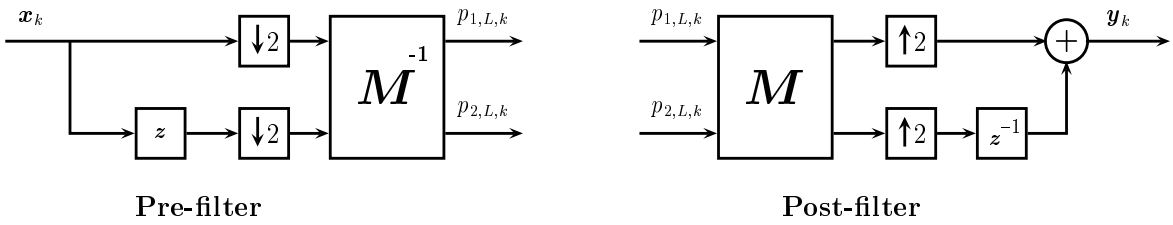


Figure 2.3: The proposed multirate pre-analysis and post-synthesis filters that are employed for the development of a generalized and non-redundant discrete multiwavelet transform framework.

struction algorithms, with special focus on the discrete multiwavelet transforms of images (2-D signals).

Figure 2.3 illustrates both the pre-filter and post-filter that are integrated into the vector-valued multiresolution decomposition and reconstruction algorithms (shown in Figure 2.2) to produce the proposed generalized framework for discrete multiwavelet transform. From the block diagram of the pre-filter, it is illustrative how the scalar input stream, $\boldsymbol{x}_k$, is appropriately downsampled and time-delayed to generate a vector stream, consisting of the even- and odd-sampled sequences, $\boldsymbol{x}_{2k}$ and $\boldsymbol{x}_{2k+1}$. This vectorized stream is then pre-filtered with the matrix filter $\boldsymbol{M}^{-1}$ to produce the desired vector input stream representative of the input signal, as expressed in (2.75). It is also noted that the downsampling is necessary for preserving a compact (non-redundant) representation of the original signal during pre-filtering. The vector input stream can now be fed to the inputs of Figure 2.2 for multiresolution decomposition. In a similar manner, the output streams of Figure 2.2 after reconstruction can be post-filtered to recover the original scalar stream. The block diagram of the post-filter in Figure 2.3 describes how this is carried out by appropriately upsampling, delaying, and combining the components of the vector stream into a scalar stream.

Using the same idea of separable decomposition along each dimension of a 2-D image, the above multirate pre-filter and post-filter can now be integrated with Mallat's pyramid algorithms [84], where tensor products of the 1-D filter banks are used to process 2-D images. Figure 2.4 portrays the proposed multiwavelet framework for image decomposition. The multirate pre-analysis filter is first applied to the rows of the image, thus generating the

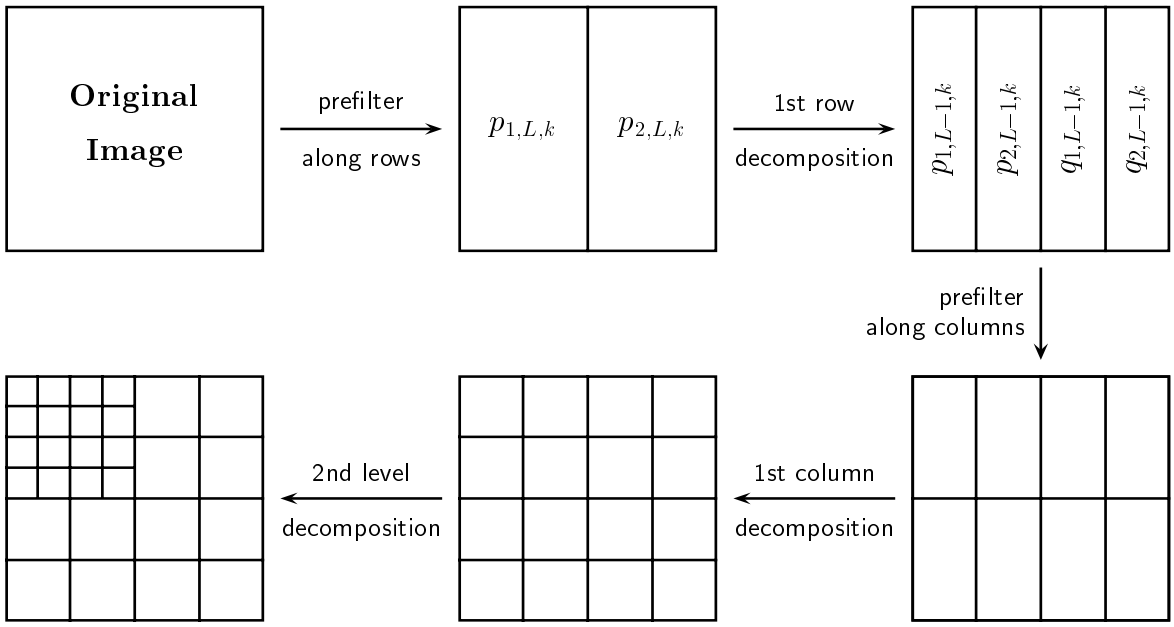


Figure 2.4: The proposed generalized and non-redundant discrete multiwavelet decomposition of a 2-D image, illustrating the pre-analysis multirate filtering and the multiwavelet subband structure of a 2-level transformed image.

vector-valued input streams (see (2.75)). This is followed by the first level of vector-valued multirate decomposition, as described in (2.69) and (2.70). The same process of multirate pre-filtering and first-level vector-valued decomposition is then performed on the resulting row-transformed image, but now along the columns. At the end of this first level of 2-D multiwavelet decomposition, we have a 16-subband intermediate image. The four subbands in the upper-left corner of the current intermediate image belong to the approximation version of the original image, whereas the other subbands constitute the detail versions of the original image in different orientations. Employing an octave-bandwidth decomposition structure, the next decomposition level (also along the rows and columns separately) is carried out only on the approximation version subimage. It is, however, important to note that there is *no* pre-filtering performed after the first level of decomposition. As a result, a L-level multiwavelet decomposition of a 2-D image will produce $4(3L+1)$ subbands. The corresponding multiwavelet reconstruction of a 2-D image can be obtained by performing all the steps described above for decomposition in the reverse order using the synthesis multifilters, and replacing the pre-analysis filter with the post-synthesis filter.

2.5 Experimental Results and Discussions

This section gives a comprehensive analysis of the proposed measure of GMPs, and the proposed generalized multiwavelet transform framework, in the context of image compression. The following three performance aspects are investigated:

1. The relative importance of GMPs as a multifilter design criterion;
2. The efficiency of the proposed pre- and post-filtering techniques; and
3. The computational complexity of the proposed multiwavelet transform framework.

For consistent and reliable comparisons, we will use a *common* image codec in all the image compression simulations presented in this chapter. The SPIHT image coding algorithm [97] is chosen[8] for its compression efficiency and popularity. Also, unless otherwise mentioned, we employed the proposed generalized transform framework for all multiwavelet filters; this also illustrates the robustness of the proposed framework to work well with any multifilters.

2.5.1 Performance Analysis of Various Multifilter Design Criteria

We will compare and contrast the relative importance of the following *six* multifilter design criteria:

(i) **Approximation Order.** A higher approximation order of the multiscaling functions corresponds to higher vanishing moments of the multiwavelets. As signals are projected onto the space spanned by the multiscaling functions, multifilters with a higher approximation order usually leads to better energy compaction (or higher coding gain) [105].

(ii) **Regularity/Smoothness.** Regularity provides a measure of the smoothness of the functions. Smoother functions (particularly for the synthesis multifilters) contribute to reduced checkerboard artifacts in the reconstructed images [23, 105, 115, 116].

[8] It is important to note that the use of another efficient image codec, such as [5] or [99], will also give similar *relative* compression results.

(iii) **Time-Frequency Localization.** Finite-length multiwavelet filters provide a flexible trade-off between time and frequency (scale) localizations. Higher localizations may contribute to more efficient coding of high-frequency wavelet coefficients [66, 105].

(iv) **Linear Phase Symmetry.** Phase linearity of a transform is determined by the symmetry of the multifilters. Symmetric multifilters help reduce phase distortions around edges and borders of the reconstructed images [116].

(v) **Stopband Attenuation.** Stopband attenuation measures the passband and stopband deviations from the ideal brick-wall filter. A sharper cutoff frequency at the transition band is useful but it usually results in longer multifilters [105].

(vi) **Good Multifilter Properties.** GMPs characterize the magnitude responses of the equivalent scalar filter bank associated with a multifilter. Multifilters possessing GMPs help prevent both DC and high-frequency leakages across bands, which can contribute to reduced smearing, blocking, and ringing artifacts [10, 11].

The following seven symmetric-antisymmetric (except for **GHM** which both scaling functions are symmetric), orthonormal 4-tap multiwavelets are chosen for the optimality in their respective multifilter properties: (i) **GHM** is one of the earliest multiwavelets constructed using fractal interpolation [40]; (ii) **REG** has the highest regularity [66]; (iii) **CL4** has the highest approximation order [33]; (iv) **JOPT4** has optimal time-frequency localization [66]; (v) **SA4(1)** has the highest GMP order of 2; (vi) **SA4(2)** has the highest approximation order while possessing GMPs; and (vii) **SA4(3)** has optimal stopband attenuation while possessing GMPs. The Daubechies' scalar wavelets **D4** and **D8** [37] are used for benchmarking purposes. It is noted that only the SA4 family[9] possesses a GMP order of at least 1. Table 2.1 summarizes some filter properties of each wavelet filter. The image compression performance comparisons of the above orthonormal multiwavelet and scalar wavelet filters are detailed in Table 2.2.

[9]For example, we obtain **SA4(1)** when $\alpha = \sqrt{15}/5$; **SA4(2)** when $\alpha = (\sqrt{19} - 2)/3$; and **SA4(3)** when $\alpha = 0.749423$.

	Multiwavelets							**Scalar Wavelets**	
	GHM	REG	CL4	JOPT4	SA4(1)	SA4(2)	SA4(3)	D4	D8
Reference	[40]	[66]	[33]	[66]	[11] & [12]			[37]	[37]
Filter Taps	4	4	4	4	4	4	4	4	8
Orthogonal	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Symmetric	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Approx. Order	2	2	3	2	1	2	1	2	4
Regularity	1.0	1.267	0.941	1.230	*	1.027	*	0.55	1.275
Stopband Error	0.601	0.370	0.357	0.381	0.346	0.357	0.324	0.448	0.317
GMP Order	None	None	None	None	2	1	1	N.A.	N.A.

Table 2.1: Properties of various multiwavelet and scalar wavelet filters. (*Note that the regularity of **SA4(1)** and **SA4(3)** could not be computed accurately, as limited by the fact that their approximation orders are less than 2.)

Among the seven length-4 symmetric-antisymmetric orthogonal multiwavelets, we noted that the **SA4** family generally outperforms **GHM**, **REG**, **CL4**, and **JOPT4**, which do not possess GMPs. Also, it is clear the **SA4(3)** multifilter, which possesses a GMP order 1 and has the smallest stopband error, produced the best compression performance. Nevertheless, it is worth noting that the performance of **CL4** is very close to those of **SA4**. A closer investigation reveals that **CL4** has "near" GMPs, where the magnitude response of the lowpass equivalent scalar filter vanishes to close zero (≈ 0.051) at $\omega = \pi$. Figure 2.5 will further illustrate the GMPs of the various multifilters. In comparison with the scalar wavelets **D4** and **D8** which employ the conventional octave-bandwidth Mallat's multiscale algorithm [84], the proposed **SA4** multifilters performed better by more than 0.54 dB, in addition to requiring only half the computational cost of that of **D8**.

While the **SA4** orthonormal family performs well against other length-4 orthogonal multiwavelets and **D8**, its compression performance is somewhat below some popular biorthogonal scalar wavelets such as the FBI's **D(7/9)** [23] and Villasenor's **V(10/18)** [116]. In the pursuit of constructing better multifilters, we have generalized the concept of GMPs to the biorthogonal setting [13]. As presented in Subsection 2.3.4, we introduced

	CR	GHM	REG	CL4	JOPT4	SA4(1)	SA4(2)	SA4(3)	D4	D8
Lena	32:1	33.58	34.14	34.37	34.01	34.42	34.33	**34.47**	33.23	34.00
	64:1	30.52	31.23	31.40	31.14	31.46	31.38	**31.50**	30.24	30.97
	128:1	27.75	28.59	28.73	28.54	28.75	28.71	**28.78**	27.72	28.33
Barbara	16:1	31.14	31.79	32.06	31.64	32.13	32.02	**32.27**	30.52	31.67
	32:1	27.44	28.15	28.23	28.07	28.30	28.23	**28.39**	27.12	27.86
	64:1	24.85	25.49	25.53	25.47	25.30	25.53	**25.59**	24.71	25.12
Boat	16:1	33.65	34.20	34.46	34.09	34.49	34.42	**34.55**	33.42	33.95
	32:1	30.04	30.68	30.85	30.59	30.87	30.82	**30.93**	29.98	30.44
	64:1	27.37	27.94	28.08	27.85	28.10	28.06	**28.12**	27.34	27.74
Goldhill	16:1	32.51	33.02	33.12	32.96	33.14	33.10	**33.18**	32.41	32.66
	32:1	29.87	30.49	30.58	30.45	30.60	30.57	**30.62**	29.84	30.09
	64:1	27.83	28.45	28.50	28.41	28.52	28.50	**28.55**	27.80	28.00

Table 2.2: Comparisons of PSNR values (in dB) of various orthonormal multiwavelet filters using different images and compression ratios. Bold entries indicate the best filters for particular image/CR combinations.

a novel class of symmetric-antisymmetric biorthogonal multifilters (SABMFs) that possess GMPs. Three SABMFs, namely, **BSA(4/4)**, **BSA(5/5)**, and **BSA(9/7)**, are compared with **D(7/9)** and **V(10/18)**. Since the subband structure of the proposed generalized multiwavelet transforms resembles that of a special case of wavelet-packet decomposition, we also included the results of **D8** that employed the same wavelet-packet subband structure (denoted as **DP8**) for a more insightful analysis. The image compression performances are displayed in Table 2.3. It is very encouraging to note that even the much shorter **BSA(4/4)** SABMF can give impressive compression performances as compared to **D(7/9)** and **V(10/18)**.

The simulation results above evidently show that the **SA4** SAOMFs could achieve very competitive compression performances even though they have a lower approximation order than both **GHM** and **CL4**, and worse time-frequency localization than **JOPT4**. Also, the **BSA** SABMFs could perform better than almost all other known multiwavelets that do not possess GMPs, as well as most popular orthonormal and biorthogonal scalar wavelets.

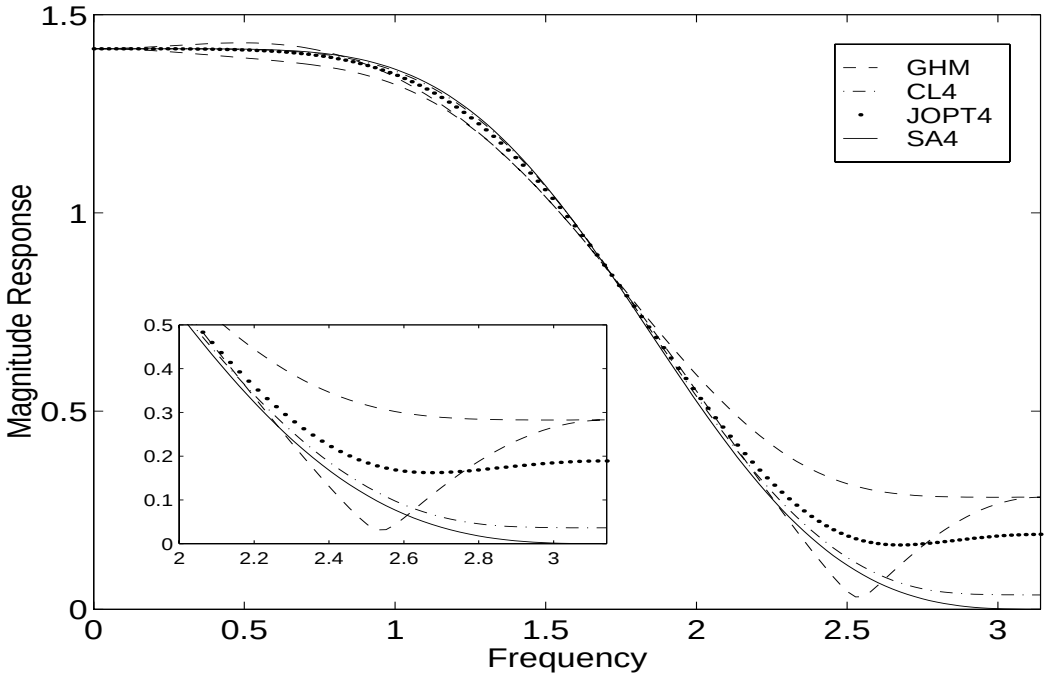


Figure 2.5: Magnitude responses of the equivalent scalar filters associated with orthogonal multiwavelets **GHM**, **CL4**, **JOPT4**, and **SA4(1)**.

In conclusion, we have shown experimentally that GMPs are definitely an important set of multifilter design criteria, though some other multifilter design properties can also be jointly incorporated in the construction of good multiwavelet filters.

2.5.2 Performance Analysis of Various Pre-Filtering Techniques

It was pointed out that the proposed pre-analysis and post-synthesis multirate filtering techniques are robust; nevertheless, it will be useful to investigate their compression performances in comparison with two other existing pre-filtering methods. The following three pre-filters are compared:

(i) Hardin and Roach's pre-filter (**HRP**) [50] that is orthogonal and approximation-order preserving;

(ii) Xia *et al.*'s pre-filter (**XIP**) [121] that is interpolatory; and

(iii) the proposed pre-filter (**TP**) [11] that uses a simple orthogonal transformation.

Image	CR	DP8	D(9/7)	V(10/18)	BSA(4/4)	BSA(5/5)	BSA(9/7)
Lena	32:1	33.64	34.74	34.83	34.77	34.88	**34.98**
	64:1	30.64	31.75	31.86	31.83	31.85	**31.96**
	128:1	28.01	29.04	29.08	29.05	29.05	**29.10**
Barbara	16:1	31.99	32.10	32.50	32.46	32.56	**32.85**
	32:1	28.18	28.13	28.32	28.55	28.53	**28.73**
	64:1	25.41	25.38	25.30	25.56	25.68	**25.90**
Boat	16:1	33.62	34.45	34.71	34.86	34.83	**34.94**
	32:1	30.15	30.97	31.05	31.15	31.15	**31.25**
	64:1	27.54	28.16	28.18	28.31	28.30	**28.33**
Goldhill	16:1	32.52	33.13	33.19	33.29	33.26	**33.42**
	32:1	29.90	30.56	30.63	**30.77**	30.73	30.75
	64:1	27.92	28.48	28.56	**28.68**	28.65	28.66

Table 2.3: Comparisons of PSNR values (in dB) of various biorthogonal scalar wavelet and multiwavelet filters using different images and compression ratios. Bold entries indicate the best filters for particular image/CR combinations.

Since the objective is to compare the significance of different pre-filters, we have adopted the popular **GHM** multiwavelet as the common[10] multifilter in our simulations. Table 2.4 illustrates the PSNR results of the three pre-filtering schemes over a wide range of compression ratios. It is observed that the **TP** and **HRP** methods have comparable performances while both methods consistently performed better than the **XIP** method. However, it should be pointed out that the **TP** method has lower computational complexity than the **HRP** method. For the case of **GHM** multiwavelet, the **TP** method requires only one matrix-vector multiplication as compared to two such multiplications for the **HRP** method. In addition, for the classes of symmetric-antisymmetric multiwavelets, it was shown in (2.76) that the **TP** method requires practically no multiplication, but only two additions for each input vector.

[10]The choice of **GHM** multiwavelet is also motivated by the existing pre-filters which designs are based on **GHM**. However, it should be emphasized that the proposed **TP** framework is robust enough to work well with any given multiwavelet.

Image	CR	HRP	XIP	TP
Lena	8:1	40.22	39.61	40.40
	16:1	36.82	36.10	36.89
	32:1	33.61	32.79	33.58
	64:1	30.50	30.11	30.52
	128:1	27.76	27.63	27.75
Barbara	8:1	36.10	34.20	36.31
	16:1	31.11	29.54	31.14
	32:1	27.45	26.47	27.44
	64:1	24.83	24.54	24.85
	128:1	23.47	23.26	23.43
Boat	8:1	37.41	37.11	38.35
	16:1	33.34	32.73	33.65
	32:1	30.00	29.53	30.04
	64:1	27.37	27.04	27.51
	128:1	25.27	25.05	25.25
Goldhill	8:1	35.80	34.99	36.02
	16:1	32.44	31.85	32.51
	32:1	29.87	29.46	29.87
	64:1	27.83	27.55	27.83
	128:1	25.96	25.76	25.93

Table 2.4: Comparisons of PSNR values (in dB) of different images at different compression ratios using the **GHM** multiwavelet but with 3 different pre-filtering methods: (i) Hardin and Roach's pre-filter (**HRP**), (ii) Xia *et al.*'s interpolation pre-filter (**XIP**), and (iii) the proposed transformation-based pre-filter (**TP**).

2.5.3 Computational Complexity Analysis of Transforms

One important consideration in most practical applications is the computational complexity of applying a given filter, which directly relates to its implementational efficiency. Such complexity is generally governed by the number of multiplications and additions involved, with the former being the dominating factor. In the context of wavelet decomposition (or reconstruction), these counts are generally directly proportional to the sum of the lengths of lowpass and highpass filters. However, it is possible to further reduce the multiplication count if we can exploit any symmetry in the filters.

Scalar Wavelet Filter System

Consider the computational cost of a L-level decomposition (or reconstruction) of an $M \times N$

image using an octave bandwidth structure. Since each level is subsampled by a factor of two along the rows and columns, the number of multiplications and additions using a scalar filter bank system with lowpass and highpass filter lengths[11] of F_ℓ and F_h, respectively, is given by the 2-tuple

$$(\#mult, \#add) = \big((F_\ell + F_h)\mathcal{L}, (F_\ell + F_h - 2)\mathcal{L}\big) \tag{2.77}$$

where $\mathcal{L} = MN\left(1 + \frac{1}{4} + \cdots + \frac{1}{4^{L-1}}\right)$.

For a linear-phase biorthogonal scalar wavelet, however, we can reduce the multiplication cost by half as the lowpass and highpass filters are symmetrical. In general, the computational cost of any symmetric/antisymmetric scalar filter can be expressed as

$$(\#mult, \#add) = \left(\left(\left\lfloor \frac{F_\ell + 1}{2} \right\rfloor + \left\lfloor \frac{F_h + 1}{2} \right\rfloor\right)\mathcal{L}, (F_\ell + F_h - 2)\mathcal{L}\right), \tag{2.78}$$

where $\lfloor x \rfloor$ denotes the largest integer that is smaller than x. The cost can be further lowered if there are embedded zeros, one's, or entries with identical magnitude in each filter sequence.

Multiwavelet Filter System

Similarly let F_ℓ and F_h denote the number of non-zero filter coefficients of the matrix lowpass and highpass filters, respectively. Also denote F_p as the number of non-zero matrix coefficients of the pre-filter (or post-filter). In general, the number of multiplications and additions required for a L-level octave-bandwidth multiresolution decomposition (or reconstruction) of an $M \times N$ image, including pre-filtering (or post-filtering), is given in (#mult,#add) form by

$$\begin{aligned} &\text{Cost of pre-filtering} + \text{Cost of } L\text{-level image decomposition} \\ = \;& \big(F_p MN, (F_p - 2)MN\big) + \left(\frac{F_\ell + F_h}{2}\mathcal{L}, \left(\frac{F_\ell + F_h}{2} - 2\right)\mathcal{L}\right). \end{aligned} \tag{2.79}$$

Note that the factor of 1/2 in the second term is introduced by the non-redundant pre-filtering framework which allows a subsampling by 2, and halving of the input signal length

[11] From a computational viewpoint, the *length* represents the number of non-zero filter coefficients. It is also worth noting that filter coefficients that are integer powers of two will require no multiplications but mere shift operations by the processor.

Scalar filter	(#mult,#add) ($\times\mathcal{L}$)	Multiwavelet filter	(#mult,#add) ($\times\mathcal{L}$)
D(2/6)	(3, 6)	BSA(4/4)	(6, 14)
D(6/10)	(8, 14)	BSA(5/5)	(8, 20)
V(10/18)	(14, 26)	BSA(6/6)	(11, 22)
D(9/7)	(9, 14)	BSA(8/6)	(12, 26)
		BSA(7/9)	(16, 28)

Table 2.5: The number of multiplications and additions required for a L-level decomposition (or reconstruction) of an $M \times N$ image using the listed scalar filters and multiwavelet filters. Here the common factor is $\mathcal{L} = MN\left(1 + \frac{1}{4} + \cdots + \frac{1}{4^{L-1}}\right)$.

[11]. In a similar manner, we will show below that the cost of pre- and post-filtering can in fact become negligible for the proposed classes of SAOMFs and SABMFs.

Comparison between scalar and multiwavelet systems

At first glance, the requirements for pre- and post-filtering may introduce additional computations. However, as shown in (2.76), it is clear that we will always have, up to a normalization constant, coefficients of the pre- and post-filters which are ± 1; hence, they involve no multiplications, but only addition/subtraction operations. Furthermore, the normalization constant (in this case, $\frac{1}{\sqrt{2}}$) can be absorbed into the *first* level and the *last* level of the decomposition and reconstruction algorithms, respectively. By exploiting symmetry, the numbers of multiplications and additions required by using some of the SABMFs are given in Table 2.5.

From Table 2.5, we observe that the application of **BSA(4/4)** will demand a computational cost involving a multiplication count of only 2/3 of that for **D(9/7)**, albeit the addition counts are the same. Both **BSA(5/5)** and **BSA(6/6)** have lower addition and multiplication counts (with 43% and 21% savings, respectively) when compared against **V(10/18)**. The possible speedup obtained using the proposed multifilters will become more critical for real-time implementation in video compression. The application of **V(10/18)** is also more costly than the **BSA(8/6)** multifilter which has better compression performance.

2.6 Conclusion

This chapter investigated a number of open research and application problems related to multiwavelets. We introduced new classes of previously unpublished orthonormal and biorthogonal multiwavelets that possess symmetric-antisymmetric pairs of multiscaling functions and multiwavelet functions, and a generalized pre-analysis and post-synthesis framework for multiwavelet initialization.

In doing so, we introduced the idea of an *equivalent scalar (wavelet) filter bank system*, which provides an equivalent and sufficient representation of the multiple-input multiple-output relationship of a given multiwavelet system. We showed that the r equivalent scalar filters are, in fact, the r polyphases of the corresponding multifilter, and the mulitplexer operation actually motivated the development of the proposed pre-filtering framework. The notion of *good multifilter properties* (GMPs) was then proposed as a new tool for analysis, construction, and application of good multiwavelets for discrete-time signal processing, particularly in image compression. We also presented the necessary and sufficient conditions for determining the existence of GMPs, explained the eigenvector properties of the matrix refinement mask associated with multiwavelets possessing GMPs, and defined the GMP order of a given multiwavelet system. In addition, we applied these ideas to construct new symmetric-antisymmetric orthornormal and biorthogonal multiwavelets. New methods for direct construction of the matrix highpass filters were presented, and various multiwavelet systems of different filter lengths were constructed. Next, we introduced a generalized framework for pre-filtering which is robust enough to integrate well with any discrete multiwavelets. The framework is also orthogonal, low in computational complexity, and provides a compact (non-redundant) representation of the input signal. Finally, extensive simulations in image compression verified the significance of GMPs for designing good multiwavelets, and the efficiency of the proposed pre-filtering framework for discrete multiwavelet transforms.

Chapter 3

Video Scalability and Scalable Video Architecture

"Let yourself be open and life will be easier. A spoon of salt in a glass of water makes the water undrinkable. A spoon of salt in a lake is almost unnoticed."

Siddhartha Gautama Buddha (563 - 483 B.C.)

3.1 Introduction

The demand for high performance and highly scalable video compression has become more and more challenging ever since the proliferation of digital video delivery to mass audiences having disparate viewing requirements over diverse networks. A number of earlier research works have focused on the generation of such scalable video bit streams in an attempt to address those needs. Among them, Taubman and Zakhor [110, 111] have pioneered some interesting work for three-dimensional wavelet compression with video scalability. In a separate research, we [3, 5] have also proposed a new highly scalable video compression framework for very low bit rate applications. Recent compression standards such as MPEG-4 [61] are also geared toward supporting video scalability over a wide range of bit rates. Not surprisingly, some commercial compression applications are also providing some (limited) video scalability features, such as Microsoft's Windows Media Technology and

RealNetwork's RealSystem, among others. In this dissertation, however, we focus on the development and implementation of a novel and truly multi-scalable video compression architecture that exploits new research results in multiwavelets and multiscale wavelet-based motion compensation algorithms.

This chapter will first illustrate, in Section 3.2, a few possible scenarios where video scalability can be useful. This is followed by a discussion of eight desirable video scaling properties in Section 3.3. A multi-scalable video compression architecture is then introduced in Section 3.4, where some inherent problems for supporting bit rate, spatial resolution, and frame rate scalabilities are highlighted. Solutions are then proposed for each of the problems and we showed how different video scaling properties can be supported simultaneously within the same compressed bit stream. An insight into how the multi-scalable bit stream is generated, organized, and parsed will also be presented.

3.2 Scenarios for Video Scalability and Communication

The principal idea of video scalability essentially refers to the fact that the video bit stream may be flexibly manipulated after the compresed bit stream has been generated. Such a capability is obviously very important and appealing for many multimedia applications, especially in scenarios where detailed knowledge of the potential disparate clients may not be available in advance at the time of generation of the compressed video source. The heterogeneity that is inherent in the diverse network infrastructure and end systems' processing and display capabilities has continually challenged the need for scalable algorithms. In a more general setting, however, video scalability need not only be constrained to scaling after the generation of the compressed source, but the video bit stream scaling can actually be carried out in either one or a combination of the following three stages: (i) during the generation of the compressed video at the source producer end; (ii) during the dissemination of the (packetized) compressed video at the transmission network layer; or (iii) during the decoding and reconstruction of the received video at the source consumer end.

In the scenario of stage (i), the source producer may need to have access to some prior

knowledge of the specifications (requirements or constraints) of a particular source consumer *before* compressing the video source. The video encoder can then tailor, say, the target bit rate of the compressed video to meet the required specifications. Such an approach is ideal in a unicast (point-to-point) situation where the source producer encodes and serves each client independently. However, in most situations such as in a multiparty or on-demand environment, detailed specifications of each client may be substantially different; the details may also be not known in advance at the time of encoding, and they may change with time as the network traffic fluctuates or when the client's system load varies during the session. From a networking viewpoint, some remedial strategies such as network traffic monitoring, feedback, and policing can be employed to inform the source producer to reshape the instantaneous bit rate of the compressed video. Such an adaptive rate control mechanism may work very well in a unicast situation or when the various clients are requesting for the same bit rate. However, in the event of heterogeneous client specifications, a simple feedback control will not suffice. The source producer will now need to generate different compressed versions of the same video, with each version having a different bit rate, for the various clients. This is obviously counter-productive from the viewpoint of data compression or effective network bandwidth utilization. A very promising solution to this problem is to generate one highly scalable video bit stream that meets the bit rate and resolution requirements of the most demanding client. Different subsets can then be selected from the *same* scalable bit stream to generate several lower-specification versions that will satisfy individual client specifications independently. This scalable capability could possibly be the best and most efficient solution for on-demand environment where heterogeneous clients request for the same video source stored in a video server. The main advantage here is obvious since only one compressed video is actually needed, and scalability will take care of the client and network heterogeneity.

In the second video scaling scenario, we illuminate the point that a highly scalable video codec may still need the support of a scalable network architecture and protocol in order to achieve truly scalable video distribution. This is especially true in a multipoint connection environment such as multiparty video conferencing or collaboration, and live

video broadcasting. In a unicast or point-to-point connection, the source producer can select only the pertinent video packets from the generated scalable video *before* transmission to the source consumer. However in a point-to-multipoint or multipoint-to-multipoint connection, a network that affords scalable video distribution, such as the Internet Multicast Backbone [38] (MBONE[1]), will be an integral component of a scalable video communication system. Using MBONE, the source producer can assign the various subsets of a scalable compressed video, which correspond to different video scalings, to different multicast nodes (groups). Clients with disparate decoding specifications can selectively subscribe to and unsubscribe from various multicast groups to dynamically reshape their respective video bit rates and video resolutions according to what they want, independently of the other clients. This scenario is significantly different from the previous video scaling scenario since the source producer does *not* need to know or anticipate the differing clients' requirements during compression. Another viable solution is to prioritize the video packets in such a manner that the network will automatically discard low-priority packets in the face of network congestion. A highly scalable video codec will generate better refinement layers for improved packet prioritization. However, this approach also requires that the network routers and switches to be intelligent and conform to certain established packet prioritization protocols (e.g. ATM network allows the specification of priority of cells).

In the third video scaling scenario, video scalability is performed at the client side. Once the client has received the compressed video packets, he or she can still choose to decode and playback only a portion of the received packets. This down-scaling of the video bit rate and display resolution may be motivated by a number of factors at the client side such as constraints on CPU processing power and display resolution. However from the viewpoint of network bandwidth utilization, it is noted that this does not effectively result in true bit rate scaling since considerable network bandwidth would have had been consumed to first receive all the video packets. The same bandwidth usage also applies to video resolution scaling if the scalable decoding is to be carried out only after receiving the

[1]The MBONE is a virtual network that is layered on top of sections of the physical Internet. An informative online guide can be accessed via "http://www.3com.com/nsc/501303.html."

full-resolution video bit stream. Nonetheless, such wastages in network bandwidth will not occur if the actual bit rate scaling performed by a client at any given time is defined as the total number of video packets that the client has received so far. Progressive browsing is a very useful application exploiting this scalability feature. For example, consider a client who is browsing a large digital image from a huge medical image database or a digital art gallery. As more packets are progressively received, the client can decode and reconstruct a higher quality version of the image. Once an intelligible image can be discerned, the client can now make a decision to either continue receiving and further improving the image quality, or to terminate browsing the current image and select another image. If the client ultimately chooses to stop browsing the image, the actual total bandwidth consumed is only as much as the total number of packets received up to the instance of termination. In this scenario, the bit rate scaling is tightly coupled to the actual network bandwidth scaling.

3.3 Desirable Video Scalability Features

Video scalability is primarily concerned with the flexibility of manipulating the digital video sources for efficient storage and customizable delivery of the video to a wide pool of heterogeneous clients over diverse networks. Digital video, as a multidimensional signal, allows many possible specifications such as the picture quality, picture size, picture playback rate, and picture color depth. The ability to scale and choose different combinations of these video specifications is crucial for simultaneous distribution to disparate clients. We denote the different scalable video features as *video scaling parameters*, which characterize the flexibility of the compressed video sources in supporting a "generate-once, scale-many" concept. In the following subsections, we will describe eight desirable video scaling parameters — bit rate, distortion, spatial resolution, temporal resolution, alphabet, hardware, complexity, and object scalability.

3.3.1 Bit Rate Scalability

Bit rate scalability allows a party (either a source producer or consumer) to gracefully scale for a wide range of different data rates from the *same* scalable video source. Such heterogeneity in bit rate can easily span a dynamic range from less than 9.6 kbps for wireless connections to over 100 Mbps in Ethernet and ATM network environments. From the viewpoint of network communications, it is evident that video bit rate scaling is closely related to network bandwidth utilization scaling. To realize this, the client must have a means to specify and receive only the relevant subset of the compressed video bit stream that meets a specific target bit rate. In fact, having bit rate scalability also allows precise bit rate control at both the source producer and consumer. Due to constraints in effective network bandwidth, it is very desirable to have constant bit rate (CBR) videos; however, this usually results in intermittent fluctuation of video quality around frames with high motion content. Although precise bit rate control can be obtained at both the encoder and decoder, it may not be very useful in practice when the compressed bit stream is packetized for transmission over a network. In this case, it would be sufficiently flexible if the codec can provide granulated bit rate scaling in terms of multiples of the average packet size[2]. On the other hand, precise bit rate control can be useful for scalable image compression where a particular compressed image size is needed for storage in a diskette, for example.

3.3.2 Distortion Scalability

Distortion scalability, also referred to as signal-to-noise ratio (SNR) scalability, allows a party to select different levels of video quality (or fidelity) from a common compressed video bit stream. Generally, the video quality (both objectively and perceptually) improves as more video data (packets) are used to reconstruct the video. As a result, there is an intrinsic one-to-one relationship between video bit rate scalability and video distortion scalability, if all other scaling parameters remain unchanged. However unlike bit rate scaling,

[2]Some latest commercial codecs such as Microsoft's Window Media Encoder and Real Networks' RealProducer also allow the specification of multiple bit rate settings in one compressed bit stream in order to target both dialup modem and high-bandwidth users. However, once the bit stream is generated, the consumers are restricted to choose only from the few preset bandwidth settings.

distortion scaling aims to provide flexibility in controlling the average quality of the playback video. Constant distortion rate, or commonly known as variable bit rate (VBR), video is a very desirable feature in which the instantaneous video bit rate is allowed to fluctuate (within certain limits) so as to guarantee a smooth overall quality of the entire video. For example, the average bit rate will increase when encoding intra-coded frames or inter-coded frames that contain considerable amount of motion. In fact, VBR encoding helps to reduce the overall stream data rate without perceptible quality loss by dynamically dropping bandwidth during static scenes that are easier to compress[3].

3.3.3 Spatial Resolution Scalability

Video spatial resolution scalability refers to the flexibility to support different display resolutions (or picture sizes) by means of selecting different pertinent subsets of a common compressed video bit stream. This video scaling parameter essentially allows a consumer to play back the same video on various display devices with disparate display resolutions. Higher spatial resolution obviously displays crisp clear pictures; on the other hand, lower spatial resolution inevitably destroys fine details and compromises clarity. For example, on the lower end of the display resolution spectrum, a personal digital assistant (PDA) such as a Palm handheld has only 160 $\times$ 160 pixel resolution, while a high-end monitor can support a display resolution of more than 1600 $\times$ 1200 pixels per inch. The capability of scalable video to simultaneously support a broad range of display resolutions is key in a heterogeneous multiparty environment, which can span the range from very high-resolution devices such as high-definition television (HDTV) [51] to very low-resolution gadgets such as PDAs and mobile phones.

3.3.4 Temporal Resolution Scalability

Video temporal resolution scalability empowers a consumer with the flexibility to choose different video frame rates for playback from a common compressed video source. A higher

[3]Real Networks' latest RealSystem 8 employs a two-pass encoding strategy that helps identify these easy-to-compress sequences, thus enabling the encoder to steal bandwidth from these scenes and use it for hard-to-compress scenes to improve the overall quality.

frame rate will allow smooth motion rendition, while a lower frame rate causes perception of jerkiness. For example, the standard temporal resolution for HDTV is 60 frames per second (fps); the corresponding standards for PAL[4], SECAM[5], and NTSC[6] systems are 25, 25, and 30 fps, respectively; and some low-end devices can play back only as low as 1-10 fps. Scaling of video playback frame rate is in fact one of the best choices for bit rate scalability while preserving the average video quality level. Alternatively, we can also reduce the frame rate to exchange for improved video frame quality at the same video bit rate.

3.3.5 Alphabet Scalability

Alphabet scalability refers to the capability of the same compressed video bit stream to support decoding with a different alphabet size (e.g. color depth) in all the pixels of a video frame. This can range from just 1-bit (black and white) to 8-bit grayscale, and from 16-bit color to 24-bit color. Some lower-end devices such as the Palm PDA handheld can only support a 2-bit or 4-bit grayscale LCD display. The flexibility to simultaneously support a multitude of devices with varying display color depths using only one compressed video bit stream is also an object of video scalability. Generally, it will be very useful if a compressed color video source can be scaled for either an 8-bit monochrome or a 24-bit true color video to suit different display requirements. For example, alphabet scalability can be achieved by encoding the luminance and chrominance channels separately, and subsequently the decoder may discard the chrominance channels if only an 8-bit grayscale video is required.

3.3.6 Hardware Scalability

Hardware scalability provides the ability to gracefully trade-off different video specifications for some physical hardware constraints at both the source producer and consumer ends.

[4]PAL stands for Phase Alternating Line is a color television standard that is used in West Germany, The United Kingdom, parts of Europe, South America, parts of Asia, and Africa. The PAL system transmits 625 lines at 25 fps with 2:1 line interlacing.

[5]SECAM stands for Sequential Couleur à Mémoire (or sequential chrominance signal and memory) is used in France, Eastern Europe, and Russia. The system uses 625 lines at 25 fps with 2:1 line interlacing.

[6]The NTSC system is widely used in North America and Japan, and uses 525 scan lines per frame, 30 fps, and two interlaced fields per frame.

The total main memory capacity at an encoder or decoder, for example, can impose serious limitations on the maximum number of temporary video buffers during processing. To a large extent, memory constraint also limits the maximum supported video spatial resolution and frame rate. The heterogeneity in memory capacity can range from only 2 MB of physical memory in a typical handheld PDA to more than 256 MB of physical memory in a high-end PC or workstation. The amount of on-board graphics memory can also be another constraining factor on the various combinations of display resolution and color depth, which subsequently influence the preferred choice of video spatial resolution and color depth for playback. Obviously, the maximum supported display resolution of the monitor or display panel will also affect the choice of video spatial resolution. Similarly, a 56 kbps dialup modem or a 9.6 kbps wireless connection will need to scale down (this could be a combination of spatial resolution, frame rate, and bit rate) the effective data rate received as compared to DSL, cable, and ATM connections. Other important hardware scalings may include CPU processing power, battery life span[7], network connection bandwidth, and network latency[8]. Hence, a scalable video bit stream can provide the flexibility to select different combinations of video scaling parameters to meet disparate hardware constraints.

3.3.7 Complexity Scalability

Complexity scalability refers to the graceful trade-off between the computational complexity and rate-distortion performance of the video encoder and/or decoder by means of scaling for different subsets of a common compressed video. Computational complexity naturally corresponds to the CPU processing speed during encoding and/or decoding. In fact, it is rather intuitive that computational scaling can be achieved indirectly by choosing different sets of video scaling parameters. A lower video spatial resolution and video frame rate, for example, will directly contribute to fewer computations that meet the processing limitations of low-speed devices, such as the Palm PDA that operates at 16-33 MHz. At the other end

[7]This hardware constraint can become a serious consideration factor in mobile devices such as laptops, PDAs, and mobile phones. Scaling for lower video specifications can reduce the processing load and cut down on battery energy comsumption.

[8]Network latency can vary from about 200 - 400 ms in wireless medium to only about 1 ms in Ethernet.

of the processing speed spectrum, a powerful workstation with a 500 MHz to 1.8 GHz CPU, or specialized hardware with parallel processing capability, can process many times faster. Since real-time encoding and decoding is critical in many video applications, a scalable video coding algorithm should provide the flexibility to the lower-end processors to scale down the video frame rate, for instance, in order to maintain real-time encoding and/or decoding.

3.3.8 Object-Based Scalability

Object-based scalability is often associated with the idea of content-based scaling, where independent objects or sprites can be added, manipulated, and removed easily, either during the generation of the compressed video source, or after the compressed video bit stream has been generated. Object scaling provides the key to interactivity. Nevertheless, this mode of video scalability is rather different from the other video scaling parameters described above. In order to support object scalability, the compression algorithm cannot just treat a video frame as a two-dimensional signal, but in fact it must perform accurate foreground-background segmentation of the video, and efficient object-oriented or region-based coding of arbitrarily-shaped objects.

3.4 A Multi-scalable Video Compression Framework

In Section 3.3 we have discussed eight desirable video scaling parameters and understood how each type of video scalability can be useful in a heterogeneous environment. The purpose of this section is to introduce a novel multi-scalable video compression framework using a multiwavelet decomposition structure. In particular, we will investigate and propose solutions to address the issues of bit rate, spatial resolution, and frame rate video scalability. More challengingly, the proposed framework will support the different video scaling parameters simultaneously with fine granularity in the same compressed bit stream. The following subsections 3.4.1 – 3.4.4 will provide insights into the various video scaling parameters that are supported by the proposed scalable framework. Some inherent problems that

impede the development of a highly scalable video compression framework are illuminated, and solutions proposed to overcome the shortcomings. In subsection 3.4.5 we proceed to explain the generation of a highly scalable video bit stream, and show how the subsets of the scalable bit stream are judiciously organized to support simultaneous scalability of the various video scaling parameters with fine granularity. Combining the solutions in this section, detailed algorithm development of the multi-scalable video compression framework will further be discussed in Chapter 5.

3.4.1 Bit Rate Scaling and Some Related Problems

As described earlier, bit rate scalability offers the decoder the option to selectively decode and reconstruct a smaller subset of the encoded bit stream. In order to achieve this, the encoder usually has to generate the bit stream in multiple coding layers. Typically, a base layer is first generated, and followed by a few refinement layers where each additional layer will further improve the fidelity of the reconstructed signal.

For the case of video coding, consider an input sequence of frames, f_n, where $n = 0, 1, \ldots$, which are to be encoded using a conventional hybrid motion estimation and motion compensation scheme. In this scheme, the first frame, f_0, is usually intra-frame coded with no dependency on any other frames. Each of the following frames, $f_1, f_2, \ldots$, will then be inter-frame coded[9] by first estimating a prediction of the frame of interest and then encoding the associated prediction error frame. Motion estimation and compensation is commonly used to account for the motion changes between the current frame and a previously encoded (reference) frame. The motion vectors, $\boldsymbol{v}$, as well as the displaced frame difference (DFD), E, are encoded and transmitted to the decoder. Let the motion field for frame f_n be

$$\boldsymbol{v}_n = \mathcal{ME}(f_n, \tilde{f}_{n-1}) \tag{3.1}$$

such that the corresponding prediction error frame

$$E_n = f_n - \mathcal{MC}(\tilde{f}_{n-1}, \boldsymbol{v}_n), \tag{3.2}$$

[9]MPEG, for example, employs a group of pictures concept where every other 16^{th} frame is encoded using an intra-frame coding mode.

where $\mathcal{ME}$ and $\mathcal{MC}$ are, respectively, the motion estimation and motion compensation operators, and $\tilde{f}_{n-1}$ is the previously reconstructed frame representing the reference frame. In order to support bit rate scalability, E_n is encoded in, say, L embedded coding layers such that

$$E_n^{(L)} = \left\{ e_n^{(0)}, e_n^{(1)}, \ldots, e_n^{(L-1)} \right\}.$$

In the conventional hybrid coding scheme, the reconstructed frames at the encoder are given by

$$\tilde{f}_{n+1}^{(L)} = \mathcal{MC}(\tilde{f}_n^{(L)}, \boldsymbol{v}_{n+1}) + E_{n+1}^{(L)}. \tag{3.3}$$

Let the sequence, $\tilde{g}_n$, denote the reconstructed frames at the decoder. Assume now that the decoder scales down the bit rate by decoding only a subset of the L layers of E_n to recover a coarse approximation, $E_n^{(k)}$, by means of a reduce operator $\mathcal{R}(.)$ such that

$$E_n^{(k)} = \left\{ \left(e_n^{(0)}, \ldots, e_n^{(L-1)} \right) | e_n^{(k)} = \mathcal{R}\left(e_n^{(k-1)} \right) \right\},$$

where $k = 1, 2, \ldots, L$. Hence the reconstructed frame, $\tilde{g}_{n+1}$, is given by

$$\tilde{g}_{n+1}^{(k)} = \mathcal{MC}(\tilde{g}_n^{(k)}, \boldsymbol{v}_{n+1}) + E_{n+1}^{(k)}$$

if the decoder selects only the first k coding layers. Since the decoder would *not* have a copy of $\tilde{g}_n^{(L)}$, which was used as the reference frame by the encoder, the coarser version of the reconstructed frame, $\tilde{g}_n^{(k)}$, is used to reconstruct the next frame. Also the same motion vector field, $\boldsymbol{v}_{n+1}$, which was estimated at the encoder, is applied at the decoder. It is noted that even if the decoder were to recover all the L coding layers of the current frame now, the reconstructed frame will be

$$\tilde{g}_{n+1}^{(L)} = \mathcal{MC}(\tilde{g}_n^{(k)}, \boldsymbol{v}_{n+1}) + E_{n+1}^{(L)}. \tag{3.4}$$

This also means that the decoder can never recover the originally encoded video frames once it has started to scale down the bit rate. An error, D, between (3.3) and (3.4) can be written as

$$\begin{aligned} D_{n+1}^{(k)} &= \mathcal{E}\left\{ \tilde{f}_{n+1}^{(L)} - \tilde{g}_{n+1}^{(L)} \right\} \\ &= \mathcal{E}\left\{ \mathcal{MC}(\tilde{f}_n^{(L)}, \boldsymbol{v}_{n+1}) - \mathcal{MC}(\tilde{g}_n^{(k)}, \boldsymbol{v}_{n+1}) \right\}, \end{aligned}$$

where $\mathcal{E}$ is some error function such as the mean squared error. This is known as the *prediction error drift* that will propagate and grow larger in magnitude as the decoder recursively reconstructs subsequent frames with only k coding layers.

The key to avoiding the prediction error drift problem is to ensure that the predicted frame, $\hat{g}_n^{(k)} = \mathcal{MC}(\tilde{g}_{n-1}^{(k)}, \boldsymbol{v}_n)$, at the decoder is always identical to the corresponding predicted frame, $\hat{f}_n^{(k)} = \mathcal{MC}(\tilde{f}_{n-1}^{(k)}, \boldsymbol{v}_n)$, at the encoder, for all n. Some solutions such as the drift correction schemes have been presented [86]. In this dissertation, we propose a simple *prediction frame "locking"* mechanism that fixes the coding of the prediction error frame to a certain predetermined coding level[10], $\mathcal{R}\left(e_n^{(K-1)}\right)$, for some fixed value K. To do so, the encoder keeps a copy of the reference frame, $\tilde{f}_n^{(K)}$, when it reaches the coding layer K while encoding the displaced frame difference, $E_n^{(L)}$, in a successive approximation manner, where $K \leq L$. Although the encoder may continue to encode the succeeding coding layers, $e_n^{(K)}, e_n^{(K+1)}, \ldots, e_n^{(L-1)}$, it uses the predetermined (fixed) reference frame, $\tilde{f}_n^{(K)}$, for motion estimation and compensation. The decoder locks onto the same reference frame, $\tilde{g}_n^{(K)}$, by also keeping a copy of it for subsequent motion compensation. In other words, we will ensure that the reference frames used for motion compensation are always maintained in synchrony (i.e. $\tilde{f}_n^{(K)} = \tilde{g}_n^{(K)}$ for all n) at both the encoder and decoder. In this manner, the decoder may also continue to decode succeeding coding layers of E_n to further improve the fidelity of the reconstructed video frames without breaking the "lock", but only at the expense of a higher decoding bit rate. Nonetheless, in order to avoid any prediction error drift, it is worth noting that the proposed "locking" mechanism does introduce a minimum decodeable bit rate constraint (i.e., the bit rate for representing the first K coding layers) on the scalable video bit stream that is generated by the encoder.

As we will introduce later in subsection 4.4.2, the proposed motion estimation and compensation is performed in the (multi)wavelet transform domain instead of in the image domain. In this case, the "locking" mechanism is enforced at a subband resolution scale level while performing motion compensation at either the encoder or the decoder. Some

[10] Alternatively, we can also specify a certain predetermined base data rate such that the "locking" mechanism is activated when the encoding (or decoding) process reaches the base reference byte.

experimental results on the effect of choosing a different reference rate control to "lock" the reference frames are shown in subsection 4.5.2. More detailed explanations, including block diagrams, will be presented in subsections 5.3.3 and 5.4.2 when we delve into the proposed scalable video encoding and decoding algorithms.

3.4.2 Spatial Resolution Scaling and Some Related Problems

A compressed video bit stream that supports spatial resolution scalability will allow a scalable decoder to separately select only subsets of the bit stream that are pertinent to the reconstruction of a lower spatial resolution video. Ideally, this will also mean that the decoder will not receive and process all portions of the compressed bit stream and motion vectors that correspond to the unwanted higher spatial resolution scales. Since a scalable encoder does not have apriori knowledge, at the time of encoding, about the possible scaling configurations that will be selected by different decoders, the encoder must generate both the motion vectors and prediction error frames in a certain top-down manner such that the portion of the bit stream that corresponds to a particular resolution scale is not dependent on any information contained in other portions of the bit stream that correspond to all higher resolution scales. Otherwise, a loss of prediction loop will occur when the decoder scales down the spatial resolution and performs motion compensation using a different version of the reference frame than that was used by the encoder.

Consider again the estimated motion vector field, $\boldsymbol{v}$, and the prediction error frame or, E, as defined in (3.1) and (3.2). Here the encoder has computed these quantities with respect to the full frame spatial resolution. The problem arises when the decoder has to scale down both $\boldsymbol{v}$ and E while reconstructing the video at a reduced spatial resolution. Suppose we denote the frame sequence at the decoder as

$$g_n^{[s]} = \mathcal{T}(g_n, s), \quad n = 0, 1, \ldots,$$

where the spatial scaling operator, $\mathcal{T}(g, s)$, essentially reduces each dimension of g by a factor of $s = 1, 2, \ldots$ via some fixed operations. Now consider a scenario in which the decoder chooses to reconstruct a spatially reduced video sequence (with a spatial scaling

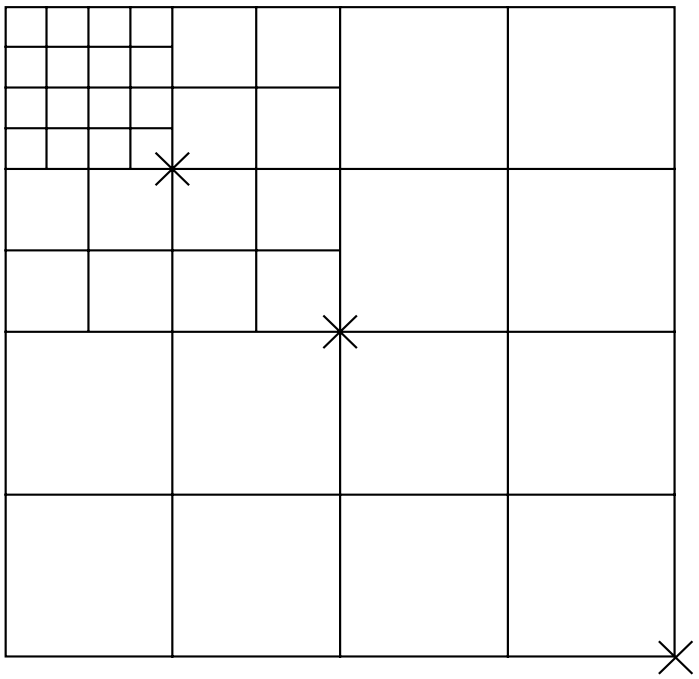


Figure 3.1: The subband structure of a 3-level multiwavelet transform. The three crosses represent the markings that will demarcate the bit stream so as to support three possible levels of spatial resolution scalability.

factor, S), $\tilde{g}_n^{[S]}, n = 0, 1, \ldots$, from an encoded full-resolution video. To do so, the decoder must also scale both $\boldsymbol{v}_n$ and E_n accordingly; this will generate

$$\tilde{g}_n^{[S]} = \mathcal{MC}(\tilde{g}_{n-1}^{[S]}, \Gamma(\boldsymbol{v}_n, S)) + \mathcal{T}(E_n, S),$$

where Γ is also a spatial scaling operator, which may or may not be the same as $\mathcal{T}$. Since $\mathcal{MC}$ is *not* a linear operator, the spatially-reduced predicted frame, $\hat{g}_n^{[S]} = \mathcal{MC}(\tilde{g}_{n-1}^{[S]}, \Gamma(\boldsymbol{v}_n, S))$, will gradually drift out of synchronism with the full-resolution prediction error frame, $E_n = f_n - \mathcal{MC}(\tilde{f}_{n-1}, \boldsymbol{v}_n)$, which was generated at the encoder. This is because

$$\mathcal{T}(E_n, S) \;\neq\; \mathcal{T}(f_n, S) - \mathcal{MC}(\mathcal{T}(\tilde{f}_{n-1}, S), \Gamma(\boldsymbol{v}_n, S)).$$

This loss of prediction loop will propagate as the spatially-reduced frame decoding process continues.

Several proposals have been made to achieve spatial resolution scalability. Among these, the approach of using resolution pyramids that decomposes each frame into multiple resolution layers for motion compensation and coding is commonly employed (e.g. in MPEG-2 [60], MPEG-4 [61], and [63]). A method for securing the prediction loop was also

proposed by Cheng and Kuo in [29]. There, motion prediction of high-frequency subbands at a particular resolution scale is achieved by first performing motion estimation using the next higher resolution low-frequency subbands. A single-scale forward wavelet transform is then applied on the motion predicted lowpass subband, and the corresponding highpass subbands of the decomposition will constitute the motion predicted high-frequency subbands of interest. Clearly, this approach introduces additional computation complexity with multiple forward and inverse wavelet transforms on different resolution scales during the encoding and decoding processes. In this dissertation, we employ a more efficient spatially scalable motion compensation framework with top-down multiresolution motion estimation in the multiwavelet domain. Subsection 4.4.2 will further explain how the spatially scalable motion vector field and prediction subband error for each resolution scale can be generated without significant processing overhead. Also, we will not only secure the prediction loop but also be able to simultaneously support bit rate scalability, as described earlier.

3.4.3 Temporal Scaling and Some Related Problems

The earlier subsections have addressed the issues of supporting bit rate and spatial resolution video scalability. The remaining question is how we can also simultaneously support frame rate scalability in the same compressed bit stream. As can be appreciated from the above exposition, the key to securing the prediction loop depends heavily on the ability to maintain synchonization of the reference frame in both the encoder and decoder when different video scaling parameters are chosen at the decoder. Recall that the encoder would not have apriori knowledge during the encoding process about the different scaling parameters that disparate decoders may select after the compressed bit stream has been generated. This motivates the need for the encoder to determine judiciously a common reference frame that the decoder can later use for motion compensation, as long as the decoder selects a frame rate that is within the allowable temporal resolution scaling as determined by the encoder. The following explains how the common reference frame can be determined by means of defining a temporal hierarchy structure:

Let $\mathcal{D} := \{d_1, d_2, \ldots, d_N \in \mathbb{Z}^+ : d_k = 2^{k-1}\}$ be a small set of *frame rate divisors* that describes the granularity of allowable frame rate scaling, where N is some fixed positive integer that defines the *dimensionality* of supported video temporal scalability. For example,

$$\mathcal{D} = \{1, 2, 4, 8, 16\}$$

represents $N = 5$ different supported temporal scaling layers — full, half, quarter, one-eighth, and one-sixteenth of the encoded frame rate — that can be chosen from a particular scalable video bit stream. Denote $\mathcal{F}_d$ as the collection of indexed frames, $f_n, n = 0, 1, \ldots,$ that is represented by a frame rate divisor, $d \in \mathbb{Z}^+$, such that

$$\mathcal{F}_d := \{f_n : n \; mod \; d = 0\}.$$

Clearly, $\mathcal{F}_d$ comprises all the indexed frames that support video temporal scalability by a scaling factor d. Also, let $\mathcal{D}_n \subset \mathcal{D}$ denote the subset of frame rate divisors whose respective collection of indexed frames contains a particular indexed frame f_n; i.e.,

$$\mathcal{D}_n := \{d \in \mathcal{D} : f_n \in \mathcal{F}_d\}.$$

In the temporal hierarchy structure, we define the *reference frame*, $\mathcal{R}_{f_n}$, as the previously encoded frame which is used to perform motion compensation for the current frame of interest, f_n. Ideally, $\mathcal{R}_{f_n}$ should also be a previous frame that is as close as possible to f_n; otherwise, the accuracy of motion estimation may be further impaired. Using the above notation, we define the frame index, r, for the reference frame $\mathcal{R}_{f_n}$ as

$$\max \{r < n : r \triangle \mathcal{D}_n = 0\}, \tag{3.5}$$

where $\{m \triangle \mathcal{D}_n = k\}$ is defined as $\{m \in \mathcal{W} : \forall b \in \mathcal{D}_n, \; m \; mod \; b = k\}$.

For a certain fixed dimesionality, N, of supported temporal scaling layers, the choice of $\mathcal{R}_{f_n}$ for a particular frame of interest, f_n, could be different. Figure 3.2 illustrates the temporal hierarchy structure for $N = 4$. The vertical axis represents the temporal scaling layers, $\mathcal{L} = 1, 2, \ldots, N$, where $\mathcal{L} = 1$ denotes the encoded full frame rate. It is noted that these temporal layers correspond to the frame rate divisors of $\mathcal{D}$. The horizontal axis represents the index number of the frames, f_n, $n = 0, 1, \ldots,$ which are denoted by the

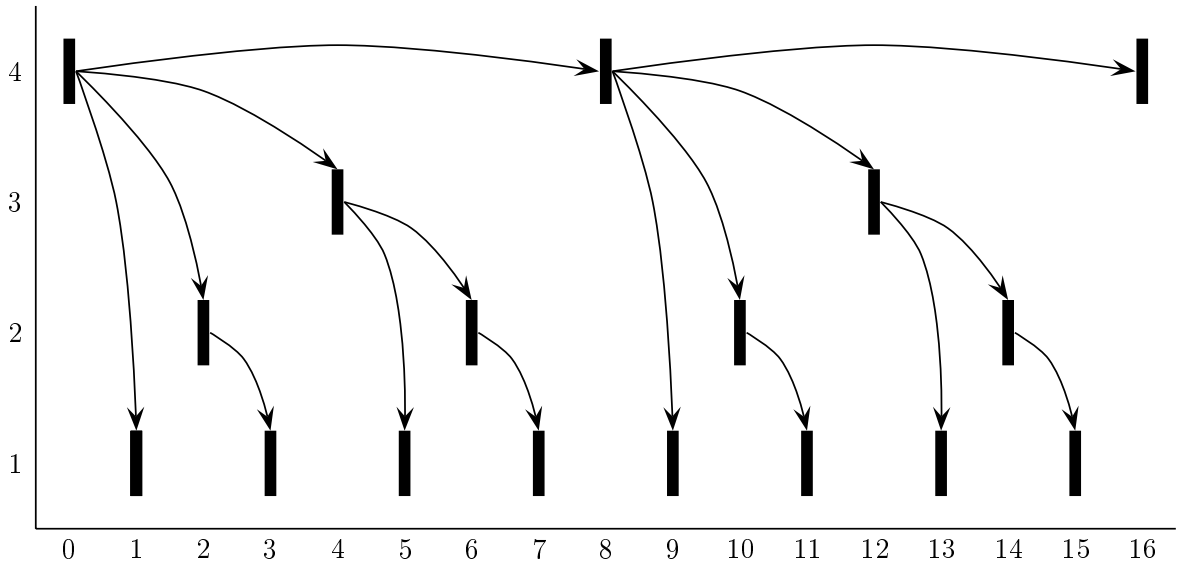


Figure 3.2: A temporal hierarchy structure that illustrates the choice of common reference frames for supporting $N = 4$ temporal scaling layers.

vertical bars. Each frame is assigned to a temporal scaling layer in such a manner that the lowest resolution temporal layer (i.e. $N = 4$) is populated first with the frames $k2^{N-1}$ for $k = 0, 1, \ldots,$ and then followed by the next higher resolution temporal layer. Using the definition of common reference frame in (3.5), we can determine the $\mathcal{R}_{f_n}$ for each frame f_n, which is shown by the arrow that points to it. The definition in (3.5) has also ensured the choice of the closest possible reference frame for a given N. Hence, the maximum distance between the current frame and the reference frame is bounded by 2^{N-1}.

It can be seen from the figure that the choice of a reduced temporal resolution by the decoder will not compromise the temporal prediction loop that is used by the encoder. For example, suppose that the decoder chooses to reconstruct at only half of the encoded frame rate (i.e. all the frames in temporal scaling layers 2, 3, and 4 only.) In this case, we have the reference frames $\mathcal{R}_{f_2} = f_0,\ \mathcal{R}_{f_4} = f_0,\ \mathcal{R}_{f_6} = f_4,\ \mathcal{R}_{f_8} = f_0$, and so on. It is evident that the decoded frames, f_{2n}, at half the encoded temporal resolution are still employing the same reference frames for motion compensation as those reference frames that were used by the encoder while encoding at full temporal resolution. Hence, the temporal prediction loop is still preserved even when the decoder were to scale down its frame rate after the bit stream has been generated. It is, however, noted that the price to pay for temporal

scalability with this scheme is that performance at full frame rate could be less accurate than the non-scalable scheme which uses adjacent reference frames for motion prediction at full frame rate. A similar temporal layering structure was also proposed by Lee *et al.* ([75], [76], [77]), where temporal video scalability is made possible. However, the main contribution in the dissertation is to carefully integrate the temporal hierarchy structure for multiresolution MEMC in the wavelet domain (instead of the spatial domain) with a recursive embedded coding framework. The details are given in subsections 4.4.2 and 5.3.2. As a result, the proposed multi-scalable video compression architecture does not only support temporal video scalability but also simultaneously support bit rate, spatial resolution, and color scalabilities with the same compressed bit stream.

3.4.4 Alphabet and Complexity Scaling

As described earlier, alphabet scalability refers to the capability of the same compressed bit stream to support decoding with different pixel depths. In particular, we are interested in reconstructing either an 8-bit grayscale or a 24-bit color video. In this dissertation, the proposed multi-scalable bit stream will also support alphabet scalability in addition to bit rate, spatial resolution, and frame rate scalings. To do so, we will encode the luminance component (i.e. Y channel) of a color frame separately from the corresponding chrominance components (i.e. U and V channels). As it will become more evident in subsection 5.3.2, the encoder will process, in each coding layer, all the subbands in the luminance component prior to processing the subbands in the U and V channels. As a result, the compressed bit stream will be partitioned into distinct luminance and chrominance segments. This subsequently allows the decoder to decode only the luminance segments in order to reconstruct a grayscale video, or decode all segments to reconstruct a color video.

Complexity scalability is a direct consequence of a combination of other video scaling parameters. This may refer to both encoding as well as decoding complexity scaling. Complexity scaling at the encoder side can become critical in situations where real-time encoding is required, such as in video conferencing. The processing requirements can be scaled down significantly by encoding at a reduced spatial resolution, frame rate, and bit

rate. This, however, constrains the allowable scaling selection at the decoder end. In a similar manner, complexity scaling at the decoder can be achieved by selecting a combination of lower decoding specification from the allowable video scaling selection of a given scalable compressed bit stream. In this dissertation, we will be able to provide implicit complexity scalability in the same video bit stream through the flexible selection of different combinations of bit rate, spatial resolution, frame rate, and color scalings.

3.4.5 Generation and Organization of Multi-Scalable Video Bit Stream

In earlier subsections, we have provided insights into the possible problems that could impede the generation of a multi-scalable compressed video bit stream, and proposed solutions to support the various video scaling parameters. This subsection will further illuminate how the multi-scalable bit stream is generated and how the various scalable segments are organized within the bit stream in order to support the following video scaling parameters simultaneously: frame rate, bit rate, distortion, spatial resolution, color, and decoding complexity.

Figure 3.3 shows a cross-sectional view of the video bit stream that is generated by the proposed multi-scalable video compression architecture. At a coarse level, the bit stream is composed of *four* embedded resolution layers, as described below:

- **Frame block (FB):** The portion of the bit stream in each FB comprises all the encoded information of one video frame, which is using either an intra-coding or inter-coding mode. Usually, FB_1 is encoded as an intra-coded frame. The shaded segment of the bit stream that precedes each FB represents the header information for the FB. It contains sufficient information about the FB in order to support frame rate scalability. While decoding at a reduced temporal resolution, some FB's can be discarded accordingly without breaking the temporal prediction loop by means of choosing the appropriate temporal scaling layers from the temporal hierarchy structure, as explained in subsection 3.4.3. For example, all FB_{2n}, $n = 1, 2, \ldots$, can be discarded when decoding at half the encoded frame rate.

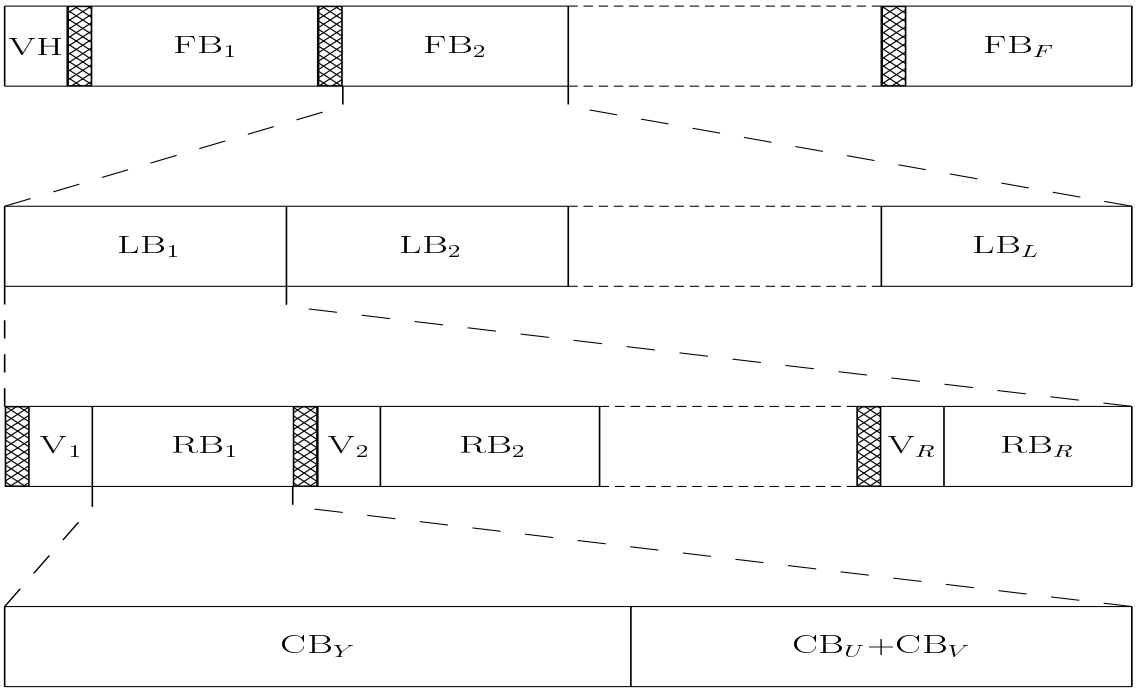


Figure 3.3: Organization of the proposed multi-scalable video bit stream hierarchy that supports simultaneous fine-granularity video scaling in terms of frame rate, bit rate, distortion, spatial resolution, color, and decoding complexity. The VH portion denotes the video header information for the entire compressed bit stream, while $V_n,\ n = 1, \ldots, R$ are the motion vector fields for each resolution scale (they are present only in the first LB.)

- **Layer block (LB):** As explained in subsection 3.4.1, each frame is encoded in multiple coding layers consisting of a base layer and several refinement layers. This coding strategy is manifested as a cascade of LB's within each FB. Supposed that frame f_1 was encoded with L coding layers. During the decoding process, each LB is processed in a sequential manner starting from LB_1 until a certain target byte that is allocated for the particular frame is reached. Clearly, by decoding up to LB_ℓ will reproduce a frame with higher fidelity than just decoding until LB_m, where $\ell > m$. Therefore, bit rate scalability can easily be achieved from the same compressed bit stream by terminating the decoding at any time when the target byte is met. Similarly, the decoder can also scale for different distortion levels during the decoding process. By employing the *prediction frame "locking"* mechanism proposed in subsection 3.4.1, we require that a certain number of minimum coding layers be processed, say, K layers. Hence, by choosing $\ell, m \geq K$, we ensure that the prediction frame that is used for

motion compensation during decoding is identical to that used at the encoder.

- **Resolution block (RB):** This layer is responsible for supporting spatial resolution scalability. Assume that the encoder generates a scalable bit stream that supports R allowable spatial resolution scales, thus producing R RB's within each LB. For an inter-coded FB, the *first* LB will also contain the motion vector fields, as denoted by the $\mathrm{V}_1, \ldots, \mathrm{V}_R$ segments preceding the corresponding LB's. The shaded segments represent the header information for the respective RB's. It is worth pointing out again that the multiresolution motion vector fields are organized in such a manner that all the motion vectors belonging to a particular supported resolution scale are encoded just before the corresponding bit stream segment representing the encoded prediction frame (subband) error. As a result, the decoder can easily ignore the motion fields of any (higher) resolution scales that are not required during processing. For example, by discarding all the highest resolution blocks, RB_R, in each LB, the decoder can reconstruct a scaled version of the video at only half the encoded spatial resolution in each dimension.

- **Color block (CB):** Each RB is further comprised of two CB's, namely, the luminance block, CB_Y, and the chrominance blocks, CB_U and CB_V, as explained in subsection 3.4.4. With this organization of the bit stream, the decoder can reconstruct a grayscale video by discarding all the CB_U and CB_V bit stream segments in each RB. It is worth noting that there are no motion vectors for the chrominance segments since they use a scaled version of the corresponding motion vector fields of the luminance segments for motion compensation.

For all the exposition in the following chapters, we may often refer to a *resolution block* as the basic scaling entity that can either be chosen or discarded accordingly so as to satisfy a certain combination of video scaling parameters. By ignoring the generation of certain resolution blocks during the encoding process, or discarding the processing of certain resolution blocks during the decoding process, both the encoder and decoder can achieve a certain degree of complexity scaling. With the above organization of the compressed video

bit stream, it is evident that the proposed multi-scalable video compression architecture can support different desirable video scaling parameters simultaneously. In fact, the newly generated scaled-down version of the encoded bit stream can subsequently be further re-scaled to fulfil some other combinations of reduced video scaling specifications. It is also worth noting that precise bit rate control can be achieved at both the encoder and decoder, regardless of the chosen frame rate, spatial resolution, and color scaling. Chapter 5 will later delve into the algorithmic level of the encoding and decoding processes, which further illuminate the generation and parsing of the above multi-scalable video bit stream.

3.5 Conclusion

This chapter first presented the need for highly scalable video compression algorithms which can simultaneously cater to disparate receivers in heterogeneous network environments. Three plausible video scalability scenarios were discussed: (i) video scaling of the original input source by the encoder during the generation of the scalable compressed bit stream; (ii) layered transmission of different segments of the bit stream while delivering the scalable video over multiple network channels; and (iii) progressive reconstruction of the scalable video by the decoder.

We then explained eight desirable video scaling parameters, and showed how each scaling parameter can be useful in different situations. In particular, an insight into each of the following video scaling parameters were given: bit rate, distortion, spatial resolution, temporal resolution, alphabet resolution, hardware, complexity, and object scalability. The proposed multi-scalable video compression architecture explicitly supports the first five scaling parameters, while both hardware and complexity scalabilities are made possible indirectly by means of choosing different combinations of the other video scaling parameters.

Several inherent problems that would impede the development of bit rate, spatial resolution, and frame rate scalings were expounded and solutions proposed. More importantly, we also presented a multi-scalable compression framework that can support simultaneous video scalability in terms of bit rate, distortion, spatial resolution, frame rate, and color

from the same compressed bit stream. Finally, we analyzed how the scalable segments of the video bit stream are organized judiciously in four embedded resolution layers: frame block, layer block, resolution block, and color block. The distinct arrangement of the various scaling entities will then allow the decoder to selectively process or discard a particular coding block depending on the chosen combination of video scaling parameters.

Chapter 4

Fast Block Motion Estimation and Compensation

"Health is the greatest gift, contentment the greatest wealth,
faithfulness the best relationship."
Siddhartha Gautama Buddha (563 - 483 B.C.)

4.1 Introduction

One of the major issues in video sequence coding is the exploitation of temporal redundancies. Each frame in a typical video sequence is made up of some changed regions of the previous (reference) frame, except at scene cuts where the current frame is unrelated to the previous frame. Frame motion can generally be classified as either *global/camera motion* or *local/object motion*. Global motion refers to the movement of the entire scene of a frame due to camera motions such as panning, zooming, rotation, translation, and vibration. Local motion, on the other hand, is due to movements of objects in a scene. As different objects may exhibit different types of movement, local motion estimation is usually more difficult to compensate. Furthermore, the problem of dealing with covered and uncovered areas due to motion is still an elusive one. Nevertheless, it is observed that local motions are also usually small and restricted, especially for most low motion content video conferencing and

visual telephony sequences.

The main objective of any motion estimation algorithm is thus to exploit the strong interframe correlation along the temporal dimension. If we can estimate the set of motion vectors that map the previous frame to the current frame, then we only need to code and transmit the motion vectors and possibly the error frame associated with the difference between the motion-compensated and the current frames. Since the error frame has a much lower zero-order entropy than the current frame, fewer bits are needed to convey the same amount of information. Hence compression of the video source can be achieved even after coding the motion vectors. In view of the advantages of interframe coding as compared to intraframe coding of video sequences, a wide range of motion estimation and motion compensation (MEMC) approaches have been investigated. Some of the more popular methods include block-matching algorithms (BMAs), parametric or motion models, optical flow, and pel-recursive [27],[117] techniques. In BMA, the motion vector of a block is defined as a two-dimensional vector, which models the block motion as translational motion in the horizontal and vertical directions. Such a translational motion model may not sufficiently describe the complex local motion information, but it is a very good compromise between computational efficiency and accuracy of the motion estimates.

Higher dimensional motion (parametric) models have been proposed to better estimate the interframe motion. With a three-parameter motion model [56], Höetter has shown that motion consisting of both the change of scale (zooming) and translation (panning) can be compensated. A four-parameter motion model [120] will be capable of modelling rotational motion as well. Others such as the six-parameter [69], seven-parameter [98], eight-parameter [55], and the 12-parameter [93] motion models have also been reported in the literature. Generally better motion prediction is achieved by increasing the number of motion parameters in the motion model [93]. For example, the 12-parameter motion model can estimate second-order geometric transformations, which include warping and shearing. However, such improvement is achieved at the cost of extra motion overhead and computational complexity. As a compromise of computational cost, motion overheads, and accuracy of motion estimation, we [2],[3] found that using a 3-parameter motion model

similar to [56] can be sufficient for the purpose of interframe motion compensation.

Another approach for interframe motion estimation is the optical flow technique. In addition to just finding the motion vectors, optical flow methods are usually employed for the measurement of accurate and dense optic flow (or image velocity) in image sequences. These dense motion fields can be used for a wide variety of tasks such as the inference of egomotion and surface structure, and the estimation of 3-D motion. Different variants of optic flow computation comprising different prefiltering or smoothing techniques, and different means for measuring the spatio-temporal derivatives can be found (e.g. [22], [52], [57], [83], [89], [102]). A comprehensive review that performs a quantitative evaluation of nine regularly cited optical flow techniques was presented by Barron *et al.* in [24]. Recently, Bernard [25] proposed to measure optic flow based on the projection of the optic flow gradient constraint on discrete analytic wavelets.

Among the many approaches for MEMC, BMA seems to be the most popular method due to its effectiveness and simplicity for both software and hardware implementations. This is evident as BMA is used extensively in all current international video compression standards which include MPEG-1 [59], MPEG-2 [60], H.261 [42], and H.263 [62].

4.2 A Review of Block Matching Algorithms

The main idea of any block-based methods is the assumption of congruent motion information for the entire block of an image; such an assumption, however, has its limitations, especially for blocks that straddle two or more objects with distinctly different motion contents. Nevertheless, block-based methods are practically the most efficient approach to use when the chosen block size is relatively small with respect to the image dimensions, thus accounting for a good approximation of local object motions.

In BMA the current frame is first partitioned into disjoint or overlapping blocks. Each block is then matched against the reference frame to find the best motion vector that maps a block from the reference frame to that block of interest in the current frame. The best (optimum) motion vector is usually defined as the one that mininises some predetermined

block distortion measure (BDM) such as the mean square error. The challenge in BMA is to find the optimum motion vector, or to estimate a suboptimal motion vector, with as few computations as possible.

For BMA the most accurate strategy is the full-search (FS) method which exhaustively evaluates all possible candidate motion vectors over a pre-determined neighborhood search window to find the optimum. The candidate that gives the best match for a given BDM is chosen as the estimated true[1] motion vector. Nevertheless the FS method has not been a popular choice because of its high computational cost. For example, a search window with a maximum motion of $\pm W$ pixels in both the horizontal and vertical directions will require $(2W+1)^2$ candidate search points for each block of interest. As a result, this motivates a host of other computationally efficient, but suboptimal, search variants such as the three-step search (TSS) [71], 2-D logarithmic search [64], orthogonal search [96], cross search [44], conjugate directional search [103] and its simplified version called one-at-a-time search (OTS) [68], simple and efficient search (SES) [82], block-based gradient descent search (BBGDS) [81], and dynamic search-window adjustment and interlaced search [73]. They are suboptimal because they choose to evaluate only a suitable subset of all the candidate motion vectors in each stage of the block matching procedure. Unlike FS, the principle that underlies suboptimal search algorithms is based on the *unimodal error surface assumption*, which assumes that the BDM increases *monotonically* as the search deviates from the position of true motion vector (or global minumum error). This assumption, however, may not always hold due to reasons such as the aperture problem and the inconsistent block segmentation of foreground and background motions. As a result, the suboptimal block search algorithms are susceptible to being trapped in local optima. These faster variations, nevertheless, are usually employed in practice, especially in real-time video coding applications, as experimental results have shown their near-optimal performances even when the assumption is not exactly true [103], [80].

Among these suboptimal BMAs, TSS [71] is one of the earliest and most widely used

[1] In the following, we connote "true" as the optimum motion vector that is found using the FS method; using another search strategy may or may not result in the true motion vector for the same block of interest.

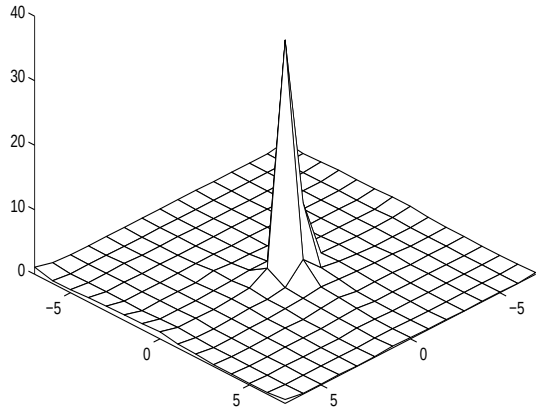


Figure 4.1: Motion vector distribution for "Football" sequence.

techniques for fast block estimation. It consists of three evaluation steps — each step contains nine uniformly-spaced search points which get closer after every step. The best candidate search point in the previous step becomes the center of the current step. Hence TSS requires a fixed $(9 + 8 + 8) = 25$ search points per block, which leads to a speedup ratio of 9 over the FS when $W = 7$. The main drawback of TSS is the relatively large search pattern in the first step which renders it inefficient for finding blocks with small motions. Moreover, its uniformly-spaced search pattern is not well-matched to most real-world video sequences in which the motion vector distribution is non-uniformly biased towards the zero vector. This is depicted in Figure 4.1, in which more than 80% of the blocks are stationary and quasi-stationary (within a central 3 x 3 pixel area) even in the fast-motion "Football" sequence. Smaller motion sequences such as "Miss America" and "Salesman" contain an average of 90% and 99% of the blocks having motion vectors within ± 3 pixels, respectively.

In order to exploit the characteristics of the center-biased motion vector distribution, a new three-step search (NTSS) algorithm [78] was introduced. It employs a center-biased search pattern in the first step by adding a smaller central eight-point pattern to that of the TSS. As a result, the worst-case scenario of the NTSS will require $(25 + 8) = 33$ block

matches. Moreover the NTSS also allows termination of the search after the first or second step. Using this technique, only 17 search points are needed for stationary blocks, and either 20 or 22 search points for quasi-stationary blocks (within ± 2 pixels). According to the results in [78], the speed of NTSS is within $\pm 18\%$ of TSS, but it gives almost consistently better motion estimates. It is, nevertheless, noted that the computational requirement of NTSS can be higher than the TSS for sequences that have a lot of large motion vectors, for example, due to fast camera panning or accelerating objects in the scene.

Recently a new four-step search (4SS) algorithm [94] was proposed to speed up both the worst-case and average-case computational requirements of NTSS. It also exploits the center-biased motion vector distribution characteristic by utilizing a nine-point search pattern on a 5 x 5 grid in the first step instead of a 9 x 9 grid as in the TSS. As a result of starting with a smaller search grid pattern, 4SS requires four search steps as compared to only three steps in both the TSS and NTSS, for the same search window of $W = 7$. In spite of this, simulation results in [94] show that the total number of candidate search points in 4SS actually ranges from the best-case of 17 to the worst-case of 27 points. According to [94], 4SS gives a speedup of 6 block matches for the worst-case, and an average of 2 block matches less than the NTSS. More importantly, 4SS still manages to maintain motion estimation performance comparable to the NTSS, which in turn is better than the TSS.

All the BMAs described above operate at a single spatial resolution; hence, they are called *monogrid* block matching algorithms. A number of papers (see e.g. [26], [28], [39], [74]) have proposed to extend the concept to a multiresolution (multigrid) framework — pyramidal or hierarchical — with the objectives of speeding up the matching process, obtaining more accurate motion vectors, and generating a multiresolution representation of the motion vector field. In fact, the monogrid FS algorithm has also been implemented using a hierarchical block matching framework to speed up its processing while achieving optimal motion vectors (e.g. hierarchical mean calculation search, HMCS, [112]). There are many other different approaches proposed to further improve the performance of BMAs. Among these improvements include:

- reducing the computational complexity of BDM evaluation, such as using the decimated MAD measure [80], the projection-based technique [70], and the matched pixel counting (or pel difference classification) method [45];

- enhancing the rate-distortion performance of motion estimation, such as employing rate-distortion-constrained statistical algorithms to produce smoother motion field [34], [46];

- increasing the reliability and accuracy of motion matching, such as using a two-stage global and local motion compensation technique [54] for finding more reliable foreground and background motions. Others include a generic motion search (GMS) algorithm [30] that first chooses a random selection of initial candidate motion vectors and then using the similar genetic processes of mutation and evolution to find the optimum motion vector;

- speeding up the search process, such as employing motion field subsampling (with the 2:1 block and/or pixel subsampling) [80], applying a thresholding method that terminates the search process once a certain predefined accuracy is satisfied [101], and exploiting the multiresolution-spatio-temporal correlations of motion vectors in neighboring blocks to select a good initial candidate search motion vector [28], [125]; and

- enhancing the subjective quality of block-based matching, such as using overlapped block matching to greatly suppress the blocking artifacts [62], [92].

In this dissertation, we introduce a novel *Unrestricted Center-Biased Diamond Search* (UCBDS) algorithm [4] for both monogrid and multigrid suboptimal block motion estimation, which can be more efficient, effective, and robust than previous suboptimal BMAs. Section 4.3 will describe in detail the new features in the UCBDS algorithm, and investigate the theoretical improvement of UCBDS against other suboptimal BMAs. We first focus on monogrid block matching in the spatial (or image) domain. The extension of UCBDS to a multigrid framework that operates in the wavelet domain will be investigated in Section 4.4.

Section 4.5 then presents and discusses extensive experimental simulations to verify the efficiency and effectiveness of UCBDS, before the conclusion is drawn in Section 4.6.

4.3 Unrestricted Center-Biased Diamond Search Algorithm

The proposed suboptimal BMA called *Unrestricted Center-Biased Diamond Search* (UCBDS) [4] is a fast and robust algorithm that exploits the center-biased motion vector distribution and adopts a halfway-stop search strategy. UCBDS has a best-case scenario of only 13 search points and an average of only 15.5 block matches. This makes UCBDS consistently faster than other suboptimal block matching techniques, and yet achieve comparable, if not better, block motion prediction accuracy. The following subsections will explain the search-point configuration, the search-path strategy, the step-by-step algorithm, and the theoretical improvement of UCBDS.

4.3.1 Algorithm Development of UCBDS

In any BMA, each frame is first divided into blocks of size N x N pixels; $N = 16$ is used in MPEG-1 [59], MPEG-2 [60], H.261 [42] and H.263 [62][2]. Furthermore in low and very low bit-rate video applications, the search for each block match is usually performed over a 15 x 15 search area[3], requiring 225 possible candidate search points per block when the FS is used. As this can be too computationally expensive, our main objective is concerned with choosing a suitable subset of these 225 points for a suboptimal version of the search algorithm. Obviously the choice of the subset of search points will have a significant influence on the speed, accuracy, and robustness of the BMA. The average number of block matches, the robustness to noise in finding the optimum motion vector, and the effectiveness in exploiting the center-biased motion vector distribution for higher statistical gain in block matching speed, are some key factors in the design of the search-point configuration and search-step strategy.

Figure 4.2 (a) depicts a basic diamond search-point configuration used in UCBDS. It

[2]Although H.263+ has an advanced option to switch to four 8 x 8 blocks.

[3]However, as will be explained later, UCBDS is flexible enough to operate on any sized search window.

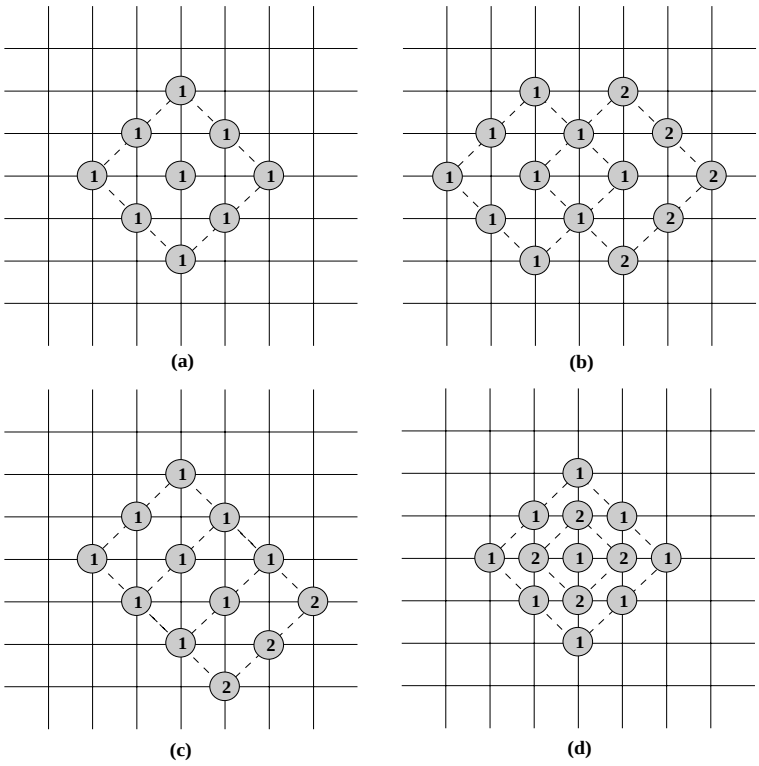


Figure 4.2: Diamond search pattern: (a) Original diamond search-point configuration, (b) Next step along diamond's edge, (c) Next step along diamond's face, (d) Final step with a shrunk diamond.

consists of nine candidate search points. This pattern is inspired by its compact[4] structure that is designed to exploit the center-biased characteristic of motion vector distribution. Figures 4.2 (b) and (c) show the positions of the diamond, with respect to the previous position, for the next search step along the diamond's vertex and face, respectively. Note that a maximum overlapping region is chosen so that there are only five or three new candidate search points, respectively, to be evaluated in every next step. Maximum overlapping is required to minimize the number of search points at each step. However, UCBDS also attempts to reach out as far as possible in each step to search for larger motions. Such a search-step strategy is critical to reducing its susceptibility of being trapped in local optima. Figure 4.2 (d) illustrates the final search step where the diamond is shrunk to only four

[4] A more compact structure is either the smaller 5-point diamond or the 9-point square within ± 1 units. However, simulations show that a larger sized search pattern is necessary for achieving greater accuracy in motion estimation. This is especially true for larger-motion blocks whereby the central block motion field surface can be relatively smooth and mislead to trapping in local optima.

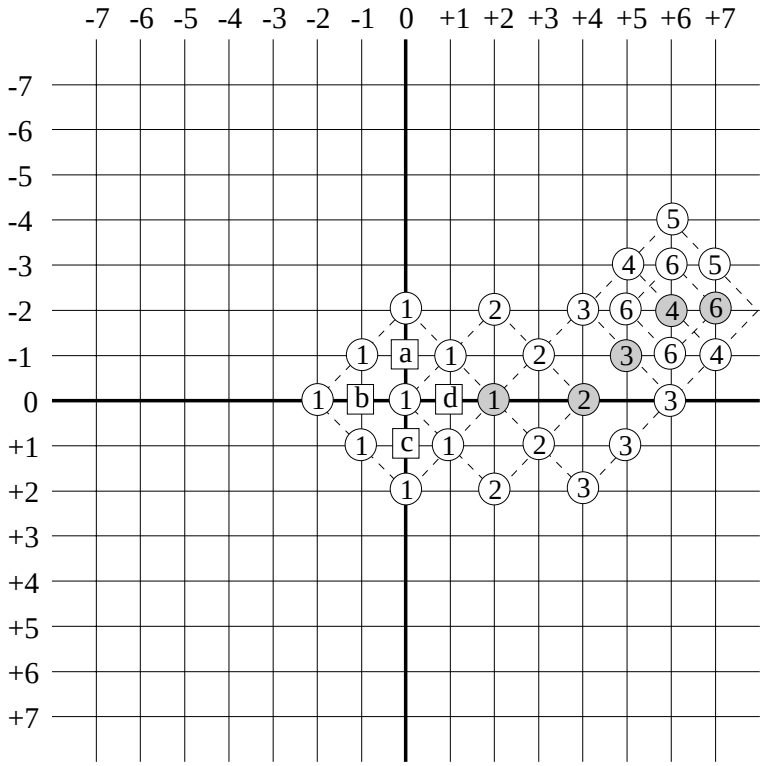


Figure 4.3: Example of unrestricted search path strategy using UCBDS.

new candidates for internal-point checking.

As a good compromise between the computational cost and the accuracy of using a particular BDM as the objective function for each block's evaluation, we have chosen to use the sum of absolute difference (SAD) given by

$$\mathrm{SAD}_{(p,q)}(m_x, m_y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f_t(p+i, q+j) - f_{t-\triangle t}(p+i+m_x, q+j+m_y)|, \quad (4.1)$$

where $-W \leq m_x, m_y \leq +W$. Here each N x N block with its upper left corner at a position (p, q) in the current frame f_t is matched with a block of the same size in the reference (reconstructed) frame $f_{t-\triangle t}$, displaced by a motion vector (m_x, m_y). Other BDM's such as sum of square error (SSE) and maximum pixel count (MPC) can also be used. Simulation results later will show that the choice of the simpler SAD measure also gives comparably accurate motion estimates as those obtained by using the SSE measure.

Figure 4.3 illustrates an example of the unrestricted search path strategy using UCBDS. It is unrestricted in the sense that the algorithm imposes no restriction on the number of

search steps and the search window can be arbitrarily large, if needed. Assume that the true motion vector of the block is (m_x, m_y) = (+7, -2). We begin at (0,0) with an original diamond pattern marked as 1. The minumum BDM of the nine candidate search points is found. If this occurs at (0, 0), the four search points at a, b, c and d are evaluated. Suppose that the lowest BDM is found at point (+2, 0); this motion vector now becomes the center of the new diamond (marked as 2) in the next search step. Five new candidate search points are compared for this vertex search. In this example, we require six search steps, where the shaded candidate points have the lowest BDM so far along the search path. Notice that the best point in step 5 coincides with that of step 4. This signals us to shrink the diamond pattern and perform the last search step via internal-point checking. Altogether we have performed 28 block matches in this example.

The following is the pseudo-code of unrestricted search path strategy using the UCBDS algorithm:

- **Starting** : The original diamond pattern (Figure 4.2 (a)) is placed at (0, 0), the center of the search window. The BDM is evaluated for each of the *nine* candidate search points. If the minimum BDM point is found to be at the center (c, c) of the diamond, proceed to **Ending**; otherwise proceed to **Searching**.

- **Searching** : If the minimum BDM point in the previous search step is located at one of the four vertices (i.e., either (c-2, c), (c+2, c), (c, c-2), or (c, c+2)), then proceed to **Vertex Search**. Else, if it is located at one of the four possible faces of the previous diamond (i.e., either (c-1, c+1), (c-1, c-1), (c+1, c-1), or (c+1, c+1)), then proceed to **Face Search**.

  - **Vertex Search** : The diamond pattern of Figure 4.2 (b) is used with the center of the new diamond coinciding with the lowest BDM point (i.e., updating the center (c, c)). Five new candidate search points are evaluated.

  - **Face Search** : The diamond pattern of Figure 4.2 (c) is used with the center of the new diamond coinciding with the lowest BDM point (i.e., updating the center (c, c)). Three new candidate search points are evaluated.

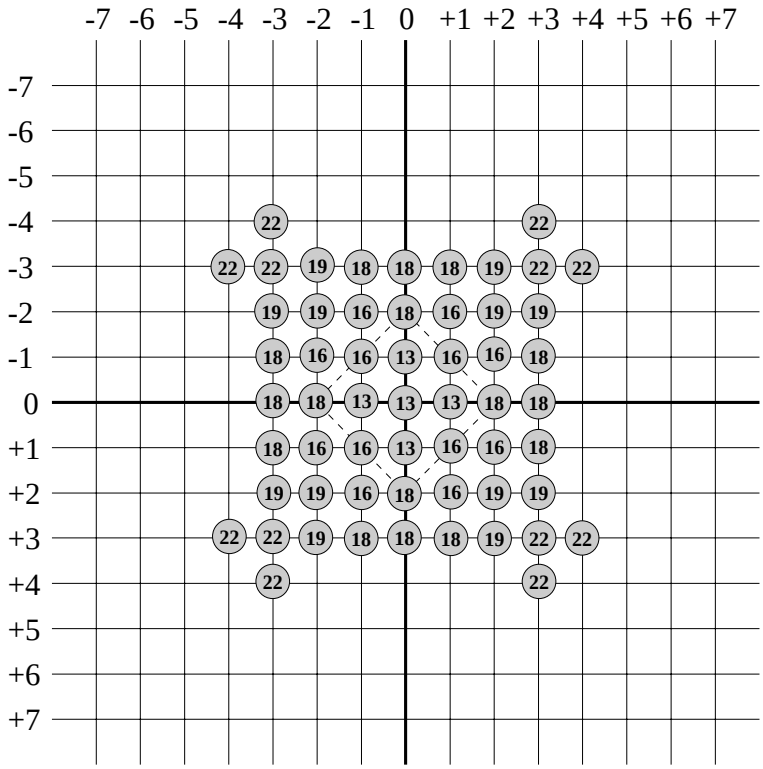


Figure 4.4: Minimum possible number of search points using UCBDS, for each motion vector location.

Note that any candidate point that extends beyond the search window is ignored. The minimum BDM is again identified. If the minimum BDM is found at (c, c), then proceed to **Ending**; otherwise proceed to **Searching** to continue the next search step.

- **Ending** : The shrunk diamond pattern of Figure 4.2 (d) is used with the same center (c, c). Now the final *four* internal points of the previous diamond are evaluated. Similarly, any internal candidate point that extends beyond the search window is also ignored. The candidate point that gives the lowest BDM is chosen as the estimated motion vector, $(\hat{m}_x, \hat{m}_y)$. The current block's search process is completed. Proceed to **Starting** for the next block, if any.

4.3.2 Theoretical Analysis of Fast UCBDS

This subsection aims to give insight into why UCBDS is truly center-biased and how speed improvement can be obtained over other suboptimum BMAs. In order to derive some lower

bounds on potential speedup, we compare UCBDS against the fast 4SS; comparisons with the TSS, NTSS, and FS are similar. Our main argument in this analysis is based heavily on the observed center-biased motion vector distributions. To begin, we first compute the minimum[5] number of search points, N_s, within a region of ± 3 pixels about the stationary motion vector (0, 0). Figures 4.4 and 4.5 illustrate the minimum N_s for UCBDS and 4SS, respectively. When compared with 4SS for the search space where $N_s \leq 22$, it is clear that UCBDS covers a slightly larger area and has lower values of N_s for the corresponding search points. It is, however, noted that 4SS can become more efficient beyond the ± 3 region. To get a better insight of the gain in N_s, we subtract the corresponding search points of UCBDS from 4SS over this ± 3 region; this gives a saving of up to 4 block matches per block.

We can further quantify this gain in N_s for block estimation by defining the following probabilities of occurrence:

- P_0 – probability of stationary blocks (i.e., the motion vector is (0, 0));
- P_1 – probability of quasi-stationary blocks within ± 1, but *excluding* (0, 0);
- P_2 – probability of quasi-stationary blocks within ± 2, but *excluding* the ± 1 region at the center;
- P_3 – probability of quasi-stationary blocks within ± 3, but *excluding* the ± 2 region at the center; and
- P' – probability of blocks in the region where $4 \leq |m_x|, |m_y| \leq W$.

Taking the average of the differences in N_s between 4SS and UCBDS over each of the above stationary and quasi-stationary regions, the statistical average gain of UCBDS over 4SS can

[5]This is the shortest path with the minumum possible number of search points needed to conclude that a candidate point (m_x, m_y) is the estimated true motion vector. In practice, the same motion vector may need more than this minimum value. Actually it depends heavily on the gradient of the block motion field surface. For example, the motion vector at (0, +1) will need 13 block matches if (0, 0) was chosen, but it will need 18 block matches if (0, +2) was chosen instead in the first search step. The same problem also plagues all other block-based search algorithms.

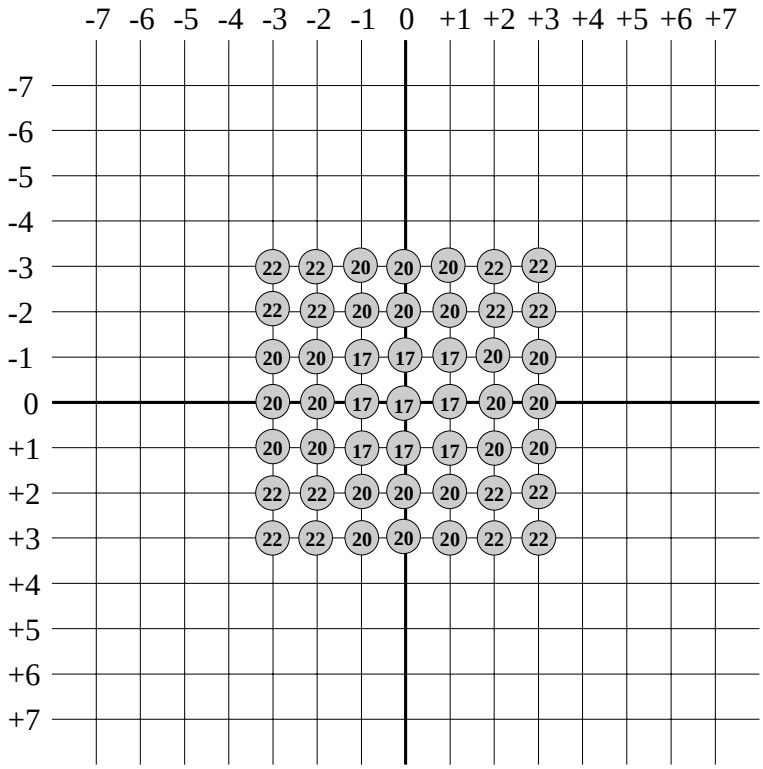


Figure 4.5: Minimum possible number of search points using 4SS, for each motion vector location.

be represented as

$$\text{Gain in } N_s = 4P_0 + (\frac{20}{8})P_1 + (\frac{52}{16})P_2 + (\frac{48}{24})P_3 + nP', \tag{4.2}$$

where $P' = 1 - (P_0 + P_1 + P_2 + P_3)$, and n is some negative number. Suppose further that we assume a uniform probability distribution over the ± 3 pixel region at the center, and that no motion vectors lie outside of this region. Then from (4.2), we will have an uniformly-distributed average gain of

$$\begin{aligned} \text{Uniform gain in } N_s &= 0.25 \times (4 + 2.5 + 3.25 + 2) \\ &= 2.94 \text{ search points per block.} \end{aligned} \tag{4.3}$$

However, observations from most real-world sequences show very peaked probabilities around P_0 and P_1, as depicted in Figure 4.1. This means that an average gain of more than 2.94 search points per block can be expected. More simulation results later will justify this statement. At the other extreme, if all blocks are stationary or have motion vectors of

either (0, -1), (0, +1), (+1, 0) or (-1, 0), then we can have the maximum possible gain of 4 search points, or a 31% speed improvement, per block over the fast 4SS. Similarly, we can also compute the theoretical speedup factors of UCBDS over the TSS, NTSS, and FS.

4.4 Fast Block Matching in Wavelet Domain

This section investigates the application of block-based motion estimation and motion compensation in the wavelet domain, instead of in the spatial (or image) domain. As wavelet decomposition provides a multiresolution structure, we can extend the monogrid UCBDS algorithm to a multigrid block matching framework. Such a wavelet-based multiresolution motion estimation and motion compensation framework constitutes an integral part of a fully scalable video coding architecture. In addition to a multiresolution representation of the motion vector field, the fact that the block matching is operating in the wavelet domain provides seamless integration with the scalable compression algorithms that will be described in Section 5.3. A key issue of block matching in the wavelet domain is the problem of shift variance of critically-sampled discrete wavelet transform, and this will be illuminated next. We then propose a simple wavelet-based multiresolution block matching scheme using the fast UCBDS algorithm. Some simulation results will also be presented.

4.4.1 Non-Translational Invariance of Wavelet Transform

The conventional discrete wavelet transform, as introduced by Mallat [84], employs a critically-sampled (non-redundant) octave-bandwidth structure, where the dimension of the wavelet-decomposed signal is the same as that of the input signal. For a two-channel (lowpass and highpass) wavelet filterbank system, both the lowpass and highpass filtered signals are decimated by a factor of two at each decomposition level. Such a non-redundant decomposition framework is attractive in many applications such as data compression, mainly because it preserves a compact representation of the input signal. Unfortunately, critical sampling engenders a translation variance problem, where a linear shift in the original (time or spatial domain) signal does *not necessarily* result in a corresponding linear

shift in the wavelet domain. This is because the translation operator does not commute with the decimation operator.

Consider the following simple illustration: Let a one-dimensional length-N signal, $f[n], n = 0, 1, \ldots, N-1$, be translated by some fixed k *positive units* to generate a new shifted signal, $f_{t=k}[n]$, where $f_{t=k}[n] = f[n-k]$ for $n = 0, 1, \ldots, N-1$, and with some boundary extension being applied at the borders of $f[n]$. To simplify the exposition, we choose the two-tap Haar wavelet for filtering. It is noted, however, that the following illustration can be easily extended to higher-dimensional signals and other longer wavelet filters. Further denote $s^{\ell}[n], d^{\ell}[n]$ as the ℓ^{th}-level lowpass and highpass wavelet coefficients (in a two-band octave decomposition structure) of the original unshifted signal, and $s^{\ell}_{t=1}[n], d^{\ell}_{t=1}[n]$ as the corresponding ℓ^{th}-level wavelet coefficients of the signal translated by one positive unit. Hence, we have:

$$\begin{aligned}
s^1[n] &= \frac{1}{\sqrt{2}}(f[2n] + f[2n-1]), \\
d^1[n] &= \frac{1}{\sqrt{2}}(f[2n] - f[2n-1]), \\
s^1_{t=1}[n] &= \frac{1}{\sqrt{2}}(f_{t=1}[2n] + f_{t=1}[2n-1]), \\
&= \frac{1}{\sqrt{2}}(f[2n-1] + f[2n-2]), \\
d^1_{t=1}[n] &= \frac{1}{\sqrt{2}}(f_{t=1}[2n] - f_{t=1}[2n-1]), \\
&= \frac{1}{\sqrt{2}}(f[2n-1] - f[2n-2]),
\end{aligned} \tag{4.4}$$

for all $n = 0, 1, \ldots, \lfloor \frac{N}{2} \rfloor$, where either the symmetric or periodic boundary extension method can be applied at the borders of $f[n]$. Clearly, the lowpass and highpass filtered signals have been subsampled by a factor of two in the above Haar decomposition process.

In order to demonstrate the issue of shift variance of critically-sampled wavelet decomposition, consider the following two cases:

Case 1: Smooth input signal

Consider a smooth signal where $f[n] \approx f[n-1] = m, n \in \mathbb{Z}$ for a certain fixed value m. At

the extreme, we may imagine a constant input signal where $f[n] = m, n \in \mathbb{Z}$. This gives

$$s^1[n] = s^1_{t=1}[n] = \sqrt{2}m,$$
$$d^1[n] = d^1_{t=1}[n] = 0, \quad n \in \mathbb{Z}.$$

Case 2: High-frequency input signal

Consider a high-frequency signal with many abrupt transitions such as edges and sharp patterns, where $|f[n] - f[n-1]|$ is very large at the positions of transition. At the extreme, we may imagine an oscillating signal with sharp transitions with a period of two units such that $f[2n] = 0$ and $f[2n+1] = m \gg 0, n \in \mathbb{Z}$. This produces

$$s^1[n] = s^1_{t=1}[n] = \frac{m}{\sqrt{2}},$$
$$d^1[n] = -\frac{m}{\sqrt{2}}, \quad \text{and} \quad d^1_{t=1}[n] = +\frac{m}{\sqrt{2}}, \quad n \in \mathbb{Z}.$$

Clearly, a shift of one positive unit around an edge in the time domain does not result in a corresponding shift in the wavelet domain, especially for the highpass wavelet coefficients. For such sharp edges or oscillating signals, the signs of the corresponding wavelet coefficients can become the opposite. Hence, the conventional block matching algorithm may not work accurately in high-frequency subbands so as to capture such motion shifts around sharp edges in a scene.

The same phenomenon also holds true for ramp-like boundaries (or a step edge) of the scene, where $f[k] = \varepsilon \approx 0$, for all $k < 2K$; and $f[k] = m \gg \varepsilon$, for $k \geq 2K$, where $k \in \mathbb{Z}$ and K is some fixed positive integer. In this case, we have

$$s^1[K] \approx \frac{m}{\sqrt{2}}, \quad \text{and} \quad s^1_{t=1}[K] = 0,$$
$$d^1[K] \approx \frac{m}{\sqrt{2}}, \quad \text{and} \quad d^1_{t=1}[K] \approx 0.$$

It is noted that a unit translation around a ramp-like edge can result in unmatched magnitudes of the corresponding lowpass and highpass coefficients.

As pointed out earlier, the root cause of this is attributed to the *shift variant* property of the downsampling (or upsampling) operation. However, there are exceptions to the above phenomenon whereby certain linear translations in the spatial (or time) domain will have

the corresponding linear shifts in the wavelet domain. Let us further analyze (4.4) for a signal that is shifted by an *even* number of pixels. We now have

$$\begin{aligned} s^1_{t=2k}[n] &= \frac{1}{\sqrt{2}}(f_{t=2k}[2n] + f_{t=2k}[2n-1]), \\ &= \frac{1}{\sqrt{2}}(f[2n-2k] + f[2n-2k-1]), \\ &= s^1[n-k], \\ d^1_{t=2k}[n] &= \frac{1}{\sqrt{2}}(f_{t=2k}[2n] - f_{t=2k}[2n-1]), \\ &= \frac{1}{\sqrt{2}}(f[2n-2k] - f[2n-2k-1]), \\ &= d^1[n-k], \end{aligned}$$

for some fixed integer k. This shows that accurate block motion estimation can, in fact, be performed in the wavelet domain if the actual spatial shift is some even number of pixels. By extending the result to a higher wavelet decomposition level, we obtain

$$s^\ell_{t=2k\ell}[n] = s^\ell[n-k], \quad \text{and} \quad d^\ell_{t=2k\ell}[n] = d^\ell[n-k],$$

where $\ell = 1, 2, ...,$ and for some fixed integer k. It is, therefore, possible to have accurate motion estimates for higher resolution wavelet subbands as well. However, we may not obtain such accuracies for other spatial shifts of the original signal. In such a case, we may still be able to approximate the true motion vector, and then encode the prediction residue between the displaced (predicted) subband block and the corresponding subband block of wavelet coefficients in the current frame of interest. In other words, we make the best reasonable effort to search for the block of wavelet coefficients that will minimize a certain block distortion metric. Some simulations described later will illustrate the performance of the proposed block motion compensation in the wavelet domain.

4.4.2 Wavelet-Based Multiresolution UCBDS Algorithm

One of the main motivations for wavelet-based motion compensation is that wavelet decomposition offers a natural multiscale representation of the video frames; this presents an attractive framework for multiresolution representation of the motion vectors. A quick literature review will show several earlier works on motion estimation and compensation

in the wavelet domain, for example, [29, 67, 87, 88, 91, 123, 124]. Each of these works proposed a method to address certain issues, ranging from securing the motion prediction loop to improving the motion estimates in the highpass subbands.

For example, Cheng *et al.* [29] proposed to prevent a loss of motion prediction loop for each wavelet scale and ensure that the prediction generated at a certain scale at the decoder is exactly the same as the prediction generated by the full scale decoder. They employed a top-down multiresolution motion estimation approach that first motion compensates the low-low frequency subband of the previous reference frame, performs a single-scale forward wavelet transform on this motion-compensated subband, and then uses the resulting highpass subbands of this decomposition as predictions of the corresponding highpass subbands of the current frame of interest. The prediction residue is coded along with a multiresolution representation of the motion field. In the work reported by Meyer *et al.* [87, 88], motion compensation for a particular resolution was carried out on the lowpass subband at the next finer resolution (instead of predicting the highpass coefficients directly at the current resolution). They also proposed a different decomposition structure where, at each resolution scale, the decimated lowpass subband is filtered twice and the three highpass subbands are undecimated. During motion compensation, the twice lowpass filtered subband is reconstructed into an undecimated lowpass subband. Together with the other three undecimated highpass subbands, they reconstruct four shifted subbands from the undecimated coefficients and then take an average of these subbands to obtain the final predicted subband.

It is observed that both of the above methods require high computations and working memory during the multiscale motion compensation process. In this dissertation, we will employ a new and fast multiresolution motion estimation and compensation approach in the multiwavelet domain by further exploiting the efficiency and flexibility of the UCBDS block matching algorithm. Also, the proposed wavelet-based motion compensation will enable the generation of a truly multi-scalable video compression platform, which will be considered further in Chapter 5. In essence, the proposed wavelet-based multiresolution UCBDS framework benefits from the following useful features:

- **Low computational complexity** — all motion estimation and compensation is carried out in the wavelet domain without the need to perform multiple forward and inverse multiwavelet transforms while computing the motion fields.

- **Low memory requirement** — unlike 3-D wavelet transform methods which require the storage of many transformed video frames, the proposed framework only keeps a few reference frames so as to support temporal scalability.

- **Low motion vector overhead** — the top-down hierarchical motion compensation exploits the motion vector relationships among different resolution scales. Since only the differential motion vectors are coded and transmitted, fewer bits are needed to represent the motion information. Also, since a significant number of motion vectors in the high frequency subbands can be zero (due to sparse wavelet coefficients in these subbands), no motion vectors are generated for these blocks since they are encoded using an intra-coding mode.

- **No frame coding latency** — the current frame of interest is motion compensated and encoded (or decoded) as soon as it is available, without the need to wait for a future reference video frame (such as using the bi-directional motion compensation mode in MPEG standards) nor delay processing until a 3-D group of frames becomes available. Such a zero frame delay is critical for real-time interactive video applications.

- **Non-redundant coding in wavelet domain** — the motion compensation process operates in a compact representation of multiwavelet decomposition subband structure. Unlike conventional multiresolution motion compensation using an overcomplete pyramid tree, there are no extra wavelet coefficients to be coded here.

- **No blocking artifacts** — since block matching is performed in the wavelet domain across the multiple subband scales, the reconstructed video frame is virtually free from annoying blocking artifacts because the block boundaries would have been smoothened out by the (interpolative) filtering process during inverse multiwavelet

transform. This eliminates the need for sophisticated overlapped block matching techniques or CPU-intensive post-processing/deblocking operations.

- **Fast motion search** — the center-biased UCBDS algorithm is fully exploited here for finding small local motion vectors. This is because most motion vectors will now have only small magnitudes as a direct result of the application of block matching with a lower resolution grid of the wavelet subbands and the use of motion vectors found in a lower resolution subband as motion vector seeds that initiate the new motion search in the current subband.

- **Support multi-scalability** — the motion vectors are generated and transmitted in a multiresolution manner. Hence, a decoder with a spatially scaled-down version of the video will only need to receive and decode the motion vectors that correspond to the displayed resolution. Also, the proposed method integrates seamlessly with the multi-scalable video compression framework to also support simultaneous bit rate, frame rate, and color video scalabilities.

The following outlines the steps involved in the proposed wavelet-based multiresolution UCBDS algorithm. The block diagrams in Figure 5.10 and Figure 5.11 will further illustrate the data flow and processing units in the above algorithm. More explanation will also be given in subsections 5.3.3 and 5.4.2.

1. For a current input video frame that is to be encoded using an inter-frame coding mode, determine the corresponding reference frame for motion compensation as described by the temporal hierarchy structure in subsection 3.4.3.

2. With both the current and reference frames in the multiwavelet transform domain, motion compensation is performed for each subband of the luminance channel in a top-down manner, starting from the coarsest resolution scale to the finest scale, and then followed by the subbands of the two chrominance channels.

3. For each subband in the current frame, the reference subband used for motion compensation is the corresponding subband in the reference frame with the same orientation,

resolution scale, and channel.

4. Since UCBDS is a block-based algorithm, the subband is first subdivided into an array of blocks with a certain fixed block size (say, 8×8). In order to maintain the same number of blocks in each resolution subband, the block size is doubled in each subsequent finer resolution scale.

5. For each block in a subband, a monogrid UCBDS is performed within a certain search window to find the optimum motion vector. Either full-pixel or half-pixel motion vector accuracy can be used. The motion vector is then encoded using an adaptive model arithmetic coder[6].

6. For a block in a subband which has a parent (as determined by the parent-child relationship defined in Section 5.2), the motion vector of the corresponding parent block which has been found earlier is used to initialize the motion search for the current block. The differential motion vector (with respect to that of the parent block) is then entropy encoded.

7. In recognition of the fact that a significant number of multiwavelet coefficients in the higher frequency subbands can be zero (due to coarse quantization while encoding the reference subbands) and the shift variance problem of motion compensating the high frequency coefficients, we employ the following two mechanisms while performing the motion search:

   - Intra/Inter block motion switching — each block in the higher frequency scale is first analyzed to determine if its coefficients are sparse. If so, intra-coding mode is used for the block (i.e., no motion compensation is performed); else, the above inter-coding mode is applied to find the optimum motion vector.
   - Zero motion vector bias — for each inter-coded block, the miminum error distortion value found using the optimum motion vector is compared with the error distortion value using the zero motion vector. If the former value is smaller than

[6] A Huffman entropy coder with optimized probability tables can also be used.

the latter by less than a certain predefined value, the zero motion vector is still preferred.

8. The motion information of all the blocks within a particular resolution scale will be encoded and packetized together with the (prediction error) portion of the bit stream corresponding to that resolution block. With a top-down approach, the motion information of the finer resolution subbands can later be discarded in order to support spatial scalability, without incurring any loss in the prediction loop (as explained out in subsection 3.4.2.) The block diagrams in Figures 5.10 and 5.14 will further illustrate how motion estimation and compensation is performed at each resolution scale.

4.5 Experimental Results and Discussions

The theoretical analysis in subsection 4.3.2 has made use of the *minimum* number of search points per block, which may not always be the case in practice. While searching for the optimum motion vector, the search path can be misled to one that is not the shortest path, and the final estimated motion vector may or may not be the true motion vector. This subsection, therefore, reports and analyzes the actual experimental performance of UCBDS. The first part of the experiment will focus on the application of monogrid UCBDS in the image domain, with the main objective of verifying the efficiency and robustness of the proposed algorithm. The second part will then discuss some results of the wavelet-based multiresolution UCBDS algorithm.

4.5.1 Performance Analysis of Monogrid UCBDS

In the first part of the experiment, we use the SAD block distortion measure, block size $N = 16$, and search window size $W = 7$. For vigorous testing, a total of six standard sequences were used, namely, "Football", "Flower Garden", "Table Tennis", "Miss America", "Salesman", and "Trevor White". These sequences were selected for a diverse range of camera and object motions. For example, the latter three video conferencing sequences have a stationary background and limited foreground motion. On the other hand, "Flower Garden"

consists mainly of stationary objects but with a fast camera panning motion; "Football" contains large local object motion; and "Table Tennis" involves camera zoom-in motion.

We compared the UCBDS against four other BMAs – FS, TSS, NTSS, and 4SS – using the following four test criteria:

1. **Average SSE per pixel** — this shows the magnitude of distortion per pixel; using SAD for the BDM gives similar plots;

2. **Probability of finding true motion vector per block** — this gives the likelihood of the estimated motion vectors to be the same as those found using the FS method; this indirectly provides an indication on the susceptibility of each suboptimal search method to being trapped in local optima;

3. **Average distance from true motion vector per block** — this measures the Euclidean distance between the estimated and the true motion vectors; and

4. **Average number of search points per block** — this provides an equivalent measure of the actual CPU runtime, as justified below.

Using SAD in (4.1) as the BDM, we need $X = N^2 Abs + (N^2 - 1)Add$ operations per search point for a block of size $N \times N$ pixels. If each frame is partitioned into B such blocks, then the total number of operations per frame is given by $\sum_{b=1}^{B} N_s^{(b)} X$, where $N_s^{(b)}$ is the total number of search points of the b^{th} block. Since we employed the same SAD measure and the same number of blocks, B, for each of the block matching schemes, the average number of search points per block, therefore, provides an equivalent measure of the actual CPU search time. Furthermore, measurements of CPU runtimes are highly dependent on system loads, and algorithm implementation.

Table 4.1 summarizes the experimental performance of each search technique over the 4 test criteria, for both the SSE and SAD block distortion measures, using the six video sequences. The first column tabulates the search speed criterion[7], in which the minimum, maximum, average, and speedup factor (S.F.) with respect to FS are reported. It

[7]For conciseness, only the results obtained using the SAD measure are shown as similar positively correlated results were obtained using the SSE measure; SAD is preferred because of its computational simplicity.

	Search points				Avg. BDM		Avg. dist. ($\times 10^0$)		Avg. prob.	
	Min.	**Max.**	**Avg.**	**S.F.**	**SSE**	**SAD**	**SSE**	**SAD**	**SSE**	**SAD**
FS	225	225	225	1.00	3208	617	-	-	-	-
TSS	25	25	25.0	9.00	3443	642	1.95	2.05	0.648	0.627
NTSS	17	33	20.9	10.8	3236	621	1.32	1.33	0.770	0.765
4SS	17	27	19.2	11.7	3450	645	2.00	2.06	0.629	0.612
UCBDS	13	37	16.8	13.4	3390	640	1.92	1.98	0.656	0.637

	Search points				Avg. BDM		Avg. dist.($\times 10^{-2}$)		Avg. prob.	
	Min.	**Max.**	**Avg.**	**S.F.**	**SSE**	**SAD**	**SSE**	**SAD**	**SSE**	**SAD**
FS	225	225	225	1.00	2753	473	-	-	-	-
TSS	25	25	25.0	9.00	2790	474	4.72	4.02	0.985	0.981
NTSS	17	33	17.5	12.8	2775	474	2.98	1.68	0.990	0.994
4SS	17	27	17.2	13.1	2777	474	3.84	3.21	0.983	0.987
UCBDS	13	33	13.3	16.9	2770	474	3.11	2.81	0.989	0.990

	Search points				Avg. BDM		Avg. dist.($\times 10^{-2}$)		Avg. prob.	
	Min.	**Max.**	**Avg.**	**S.F.**	**SSE**	**SAD**	**SSE**	**SAD**	**SSE**	**SAD**
FS	225	225	225	1.00	6089	788	–	–	–	–
TSS	25	25	25.0	9.00	6535	802	12.8	11.5	0.958	0.961
NTSS	17	33	18.2	12.3	6178	791	5.91	6.41	0.981	0.979
4SS	17	27	17.5	12.9	6389	800	9.95	10.1	0.962	0.964
UCBDS	13	36	13.8	16.3	6212	792	6.91	7.21	0.978	0.978

Table 4.1: Top to bottom: Performance comparisons (per 16×16 block) of FS, TSS, NTSS, 4SS, and UCBDS using "Miss America", "Salesman", and "Trevor White" sequences.

	Search points				Avg. BDM		Avg. dist.(x10^{-1})		Avg. prob.	
	Min.	**Max.**	**Avg.**	**S.F.**	**SSE**	**SAD**	**SSE**	**SAD**	**SSE**	**SAD**
FS	225	225	225	1.00	32553	1585	–	–	–	–
TSS	25	25	25.0	9.00	40821	1790	6.14	4.96	0.804	0.823
NTSS	17	33	22.6	9.96	34205	1620	1.83	1.62	0.925	0.931
4SS	17	27	20.1	11.2	38708	1748	4.17	3.49	0.840	0.852
UCBDS	13	33	17.8	12.6	33781	1619	1.69	1.47	0.947	0.950

	Search points				Avg. BDM		Avg. dist.(x10^{-1})		Avg. prob.	
	Min.	**Max.**	**Avg.**	**S.F.**	**SSE**	**SAD**	**SSE**	**SAD**	**SSE**	**SAD**
FS	225	225	225	1.00	70035	2395	–	–	–	–
TSS	25	25	25.0	9.00	74115	2442	6.75	5.47	0.864	0.890
NTSS	17	33	23.2	9.69	73393	2440	6.18	5.48	0.871	0.886
4SS	17	27	20.1	11.2	74195	2459	7.30	6.99	0.867	0.877
UCBDS	13	40	18.3	12.3	74475	2461	7.53	7.05	0.888	0.896

	Search points				Avg. BDM		Avg. dist.(x10^{-1})		Avg. prob.	
	Min.	**Max.**	**Avg.**	**S.F.**	**SSE**	**SAD**	**SSE**	**SAD**	**SSE**	**SAD**
FS	225	225	225	1.00	25975	1586	-	-	-	-
TSS	25	25	25.0	9.00	30899	1695	10.5	9.07	0.790	0.814
NTSS	17	33	19.6	11.5	27049	1609	3.51	2.83	0.908	0.922
4SS	17	27	18.7	12.0	29012	1652	6.51	5.38	0.856	0.880
UCBDS	13	37	15.2	14.9	27437	1613	3.54	2.90	0.928	0.940

Table 4.1: (con't) Top to bottom: Performance comparisons (per 16×16 block) of FS, TSS, NTSS, 4SS, and UCBDS using "Flower Garden", "Football", and "Table Tennis" sequences.

	TSS	NTSS	4SS	FS
Percentage Increase in SAD Distortion	0.78	0.85	0.08	2.68
Percentage Improvement in Search Speed	36.7	26.8	9.84	1130

Table 4.2: Performance of UCBDS versus other BMAs for the "Football" sequence.

is worthwhile to note that UCBDS has both the minimum and maximum numbers of search points per block due to its center-biased and unrestricted search strategies, respectively. However, the average N_s per block with UCBDS < 4SS < NTSS < TSS < FS; such observations were true for all the six test sequences we used. This shows that UCBDS is generally more efficient (i.e. it has a faster search) than the other schemes, regardless of the presence of panning, zooming, small or large motions in the sequence.

An interesting question now is how much does UCBDS trade-off block distortion for higher search speed. From column two of Table 4.1, it can be observed that UCBDS actually performs very competitively in terms of low block distortion even though it has the lowest average number of search points. Table 4.2 provides an insight to the question by giving the percentage[8] differences between UCBDS and other BMAs, using the "Football" sequence as an example. It can be seen that UCBDS has marginally worse BDM performance but the speed improvements are quite substantial when compared to the other search techniques. In addition, UCBDS may also achieve both lower BDM and higher search speed such as in the "Flower Garden" sequence. A possible explanation for the good performance of UCBDS is that it has a very compact search configuration that speeds up the search of most blocks, while the unrestricted search strategy minimizes the risk of being trapped in local minima. From the third and fourth columns of Table 4.1, we can also conclude that UCBDS generally gives a lower average Euclidean distance error from the true motion vectors, and a higher average probability of finding the true motion vectors, when compared with the other suboptimal search techniques.

Figures 4.6, 4.7, and 4.8 plot the actual performance of each search scheme on a frame-

[8] A 100% improvement means it has twice better performance.

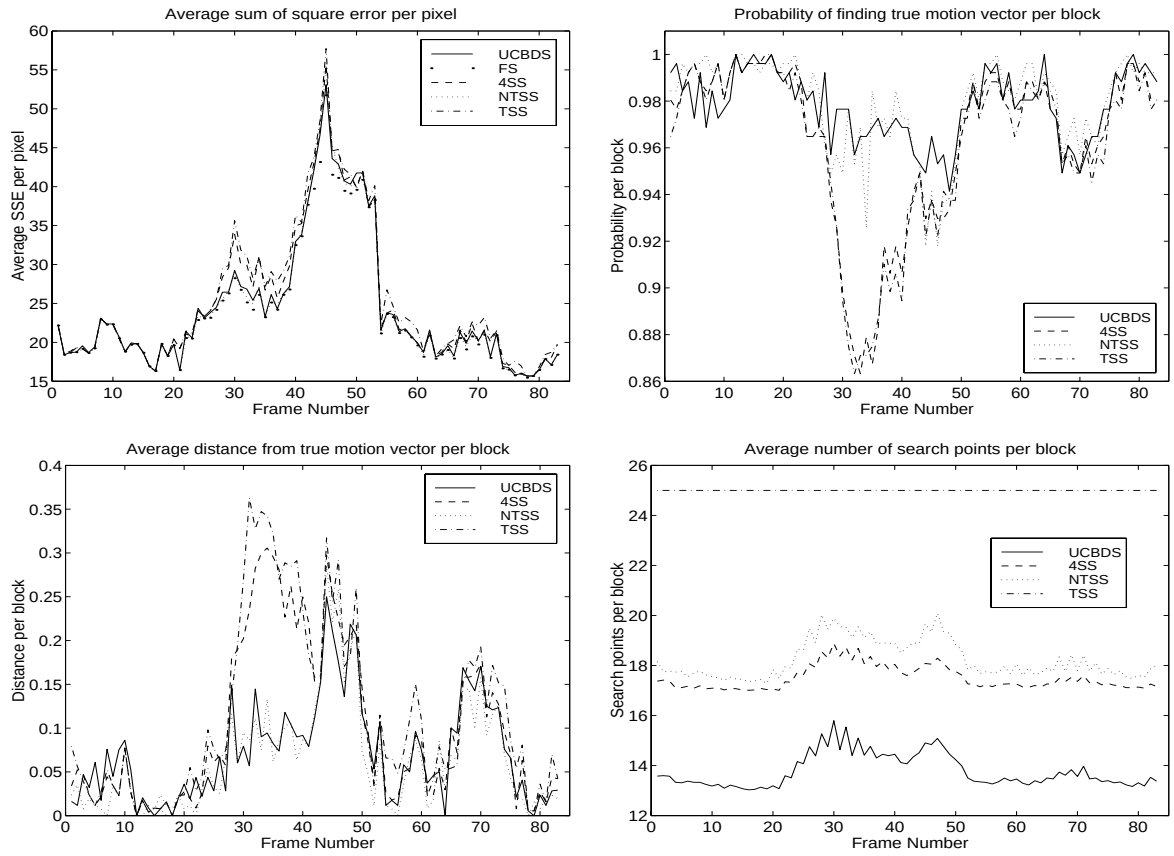


Figure 4.6: Performance comparisons of FS, TSS, NTSS, 4SS, and UCBDS over all 4 test criteria using "Trevor White" sequence.

by-frame basis. It is clear that UCBDS performs very well in terms of block distortion, while it consistently outperforms the other methods in terms of search speed. Finally, we present some subjective results on the effectiveness of motion compensation using various BMAs. Figures 4.9 (a) and (b) show two consecutive frames of the "Flower Garden" sequence. Figure 4.9 (c) depicts the difference frame (shifted by +128, but without scaling) between the two frames *without* any motion compensation, and Figures 4.9 (d) - (h) illustrate the corresponding motion-compensated difference frames using the TSS, NTSS, 4SS, UCBDS, and FS methods, respectively. It can be seen that UCBDS gives very good motion estimates (with an average SSE per pixel of 142.4) when compared to the FS (SSE per pixel is 146.4), but UCBDS is about 13 times faster than the FS method.

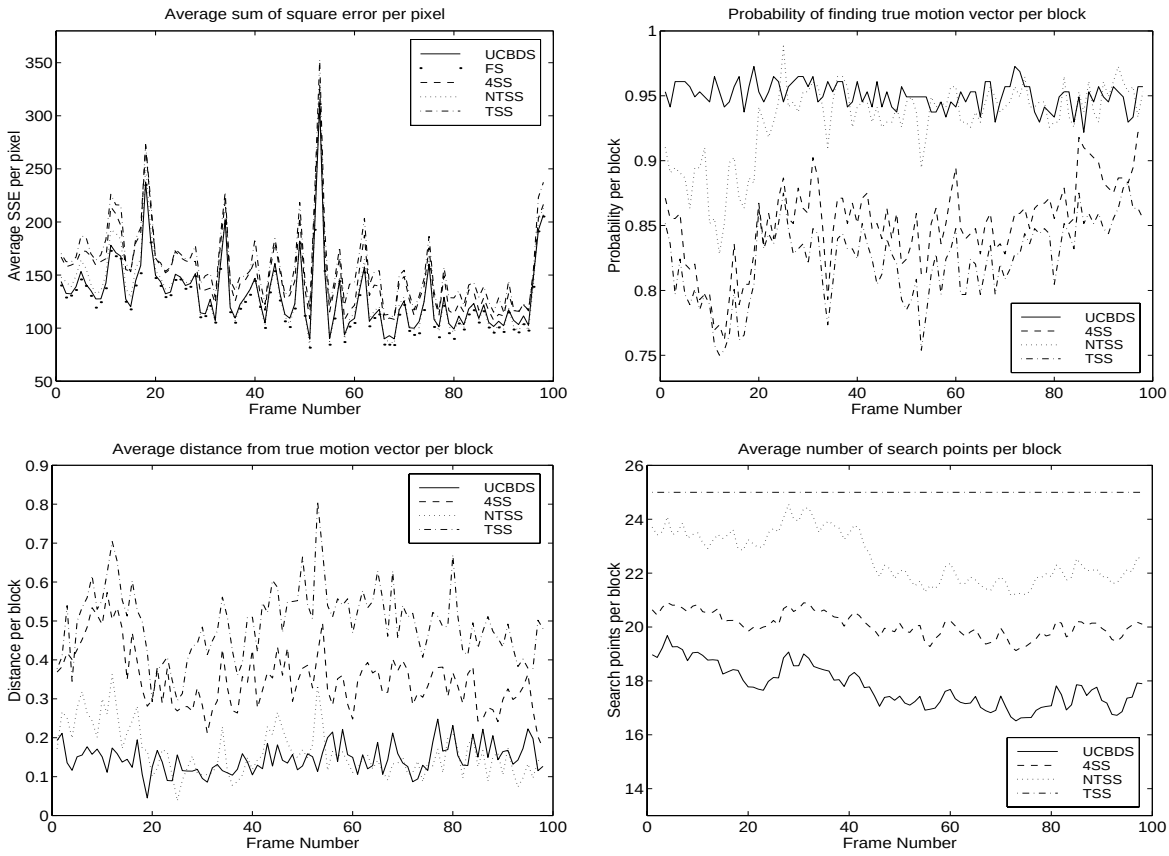


Figure 4.7: Performance comparisons of FS, TSS, NTSS, 4SS, and UCBDS over all 4 test criteria using "Flower Garden" sequence.

4.5.2 Performance Analysis of Multiresolution Wavelet-based UCBDS

For the second part of the experiment, we investigate the performance of the proposed multiresolution UCBDS in the wavelet domain. The simulations in this part of the experiment mainly focus on presenting a visual perspective of the results of performing motion compensation on the wavelet subbands. As pointed out in subsection 4.4.2, all MEMC is carried out in the wavelet domain; hence, it has low computational complexity and the predicted frames are free from blocking artifacts. Also, a top-down multiscale representation of the motion vector fields enables support for spatial resolution scalability. In the simulations, the following two video sequences were used:

- **Miss America:** This is a typical video conferencing sequence with a talking head-and-shoulder person and static background. There is only small local motion and no

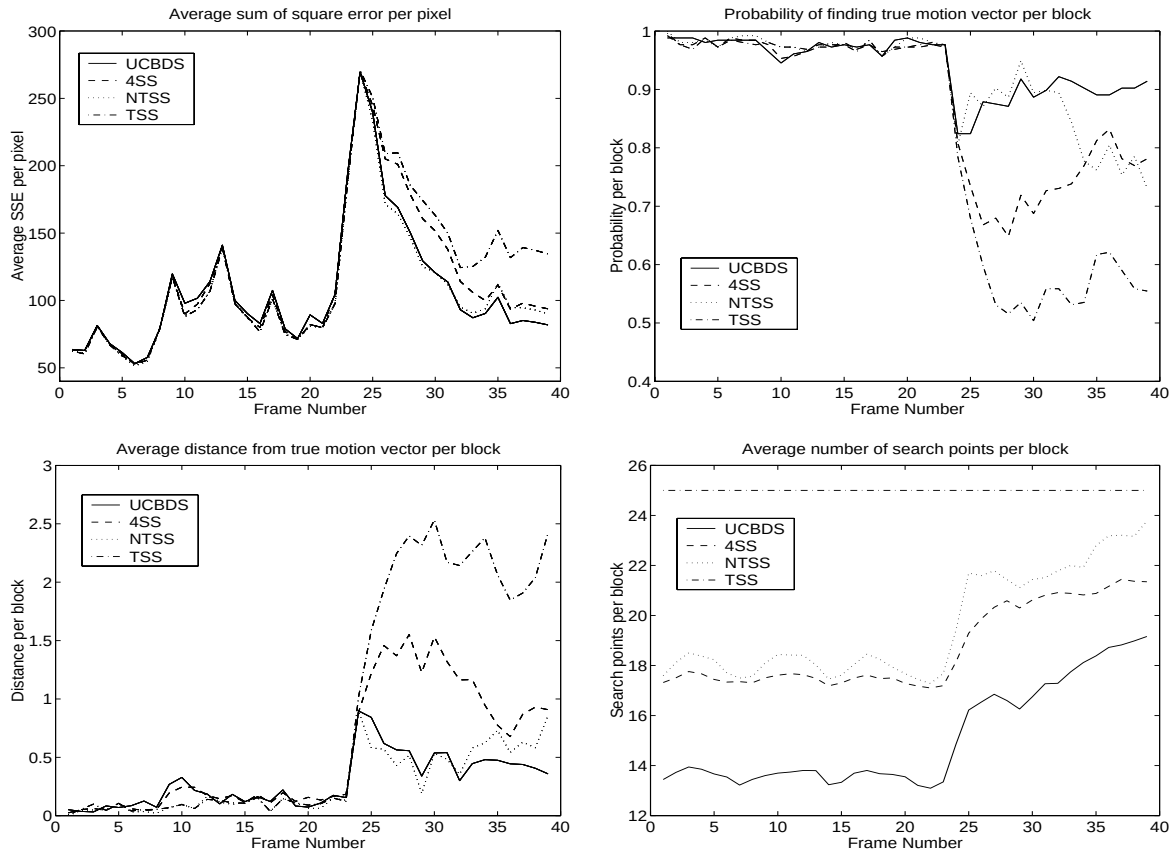


Figure 4.8: Performance comparisons of FS, TSS, NTSS, 4SS, and UCBDS over all 4 test criteria using "Table Tennis" sequence.

global motion.

- **Foreman:** This is an outdoor video sequence with a foreground object (the foreman) and a background with details (the building). The sequence also contains a video shot with camera panning that constitutes fast background motion.

Figure 4.10 portrays four different predicted frames in the Miss America sequence after the application of motion compensation. Since MEMC is performed in the wavelet domain, all the predicted frames, reference frames, and prediction error frames are represented in the wavelet domain, without the need to perform inverse wavelet transform. However, the predicted frames shown here are reconstructed back to the spatial domain merely to illustrate the visual quality of the decoded video when *no* further prediction error information is added back to the predicted frames. It is noted that during an inter-frame coding mode,

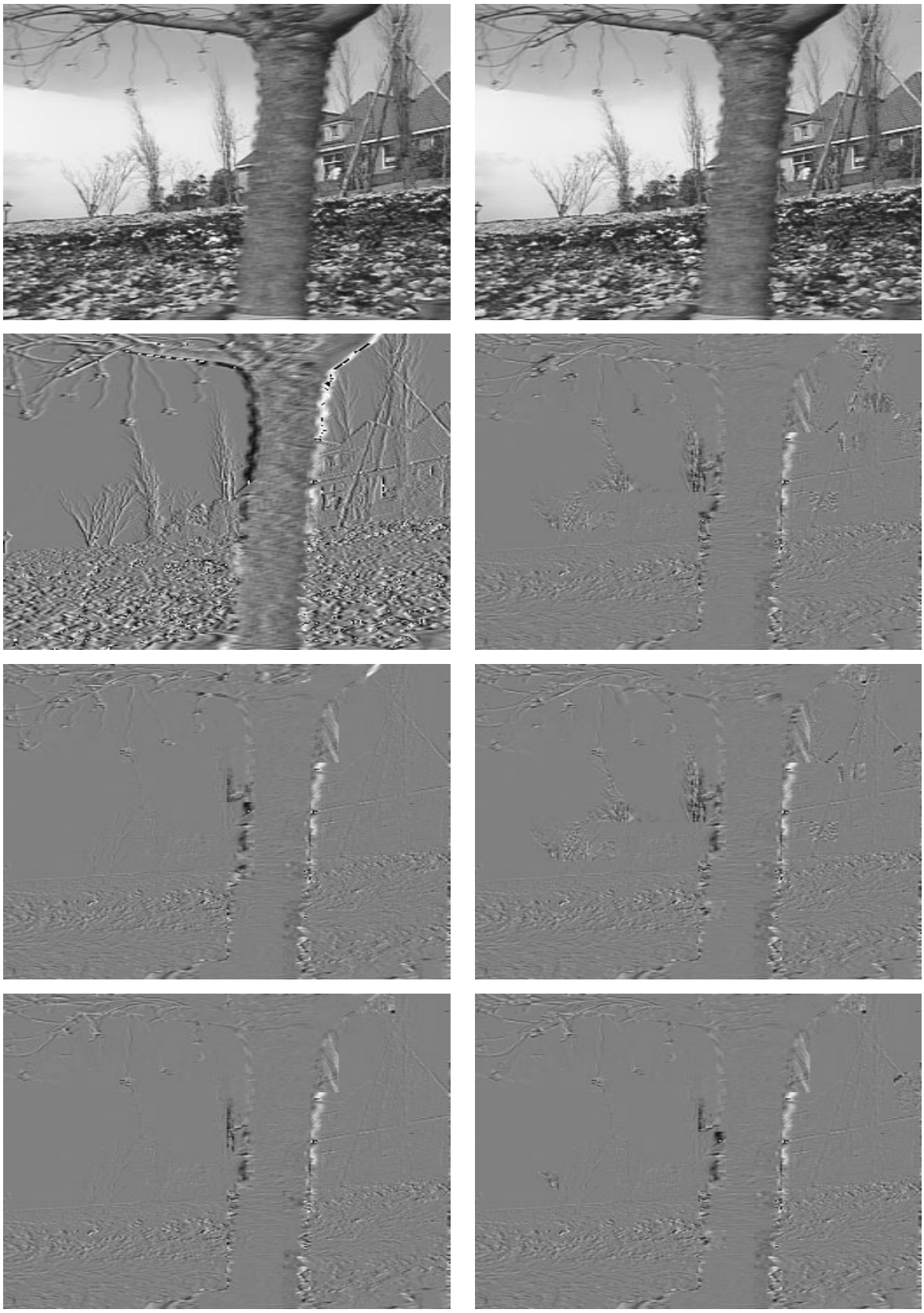


Figure 4.9: Comparison of various block matching techniques using the Flower Garden sequence. Top to bottom and left to right: (a) original frame 10, (b) original frame 11, (c) uncompensated frame difference (average SSE per pixel: 1115.6), and motion compensated frame differences using (d) TSS (SSE: 174.9), (e) NTSS (SSE: 157.3), (f) 4SS (SSE: 166.7), (g) UCBDS (SSE: 146.4), and (h) using FS (SSE: 142.4).

a pre-determined portion of the prediction error frame will be added back to the predicted frame to generate the reference frame for encoding (or decoding) a future frame. Also, an even more significant portion of the prediction error frame is usually added back to the predicted frame when generating the final decoded frame for display. After all, it is worth noting that there is generally no requirement that the motion predicted or reference frames to be legible images, as long as sufficient prediction error information is added back to reconstruct good decoded frames for display. It can be seen that frames (a) and (c) have poorer prediction accuracy as compared to frames (b) and (d). This observation is, in fact, a direct manifestation of the temporal hierarchy structure, as presented in subsection 3.4.3. Employing a four-layer temporal hierarchy, the distance between both frames (a) and (c) and their corresponding reference frames is larger (8 frames apart) than that that was used for frames (b) and (d), which is only two frames apart. As the distance (and motion) gets larger, we can generally expect the motion prediction to become less accurate.

Figure 4.11 depicts four reference frames from the Foreman video sequence. In this case, however, frame pair (a) and (b) are actually the same frame instance, as well as for the frame pair (c) and (d). The difference between the frames within each pair is a direct consequence of enforcing a different base reference value during the *reference frame "locking"* mechanism for MEMC, as explicated in subsection 3.4.1. Both frames (a) and (c) employed a "lock" with only 10% of the target bit rate, while frames (b) and (d) have twice (i.e. 20%) the amount for the base reference bytes. In other words, the reference frames in the latter case will have higher fidelity since twice as much information from the prediction error frames are added back to the previous predicted frames. As a result, the predicted frames (b) and (d) have better quality since their corresponding reference frames (from which motion compensation is applied) are of higher fidelity, albeit the fact that the distances between the current and reference frames are the same in both cases. The first frame pair has been carefully chosen to highlight the effect of wavelet-based MEMC in the event of very high local and global motions. It is clear that there are no blocking artifacts despite the poor motion prediction accuracy, but they do exhibit severe ringing effects[9].

[9]The target encoding byte budget was set to only 100 kbps with the aim to highlight the possible adverse effect of poor motion prediction accuracy in high motion frame segments as well as the trade-off in the choice

(a) (b)

(c) (d)

Figure 4.10: Predicted frames from Miss America sequence: Frames (a) and (b) have poorer motion compensation due to a larger distance from the corresponding reference frames; and frames (c) and (d) have better motion prediction accuracy.

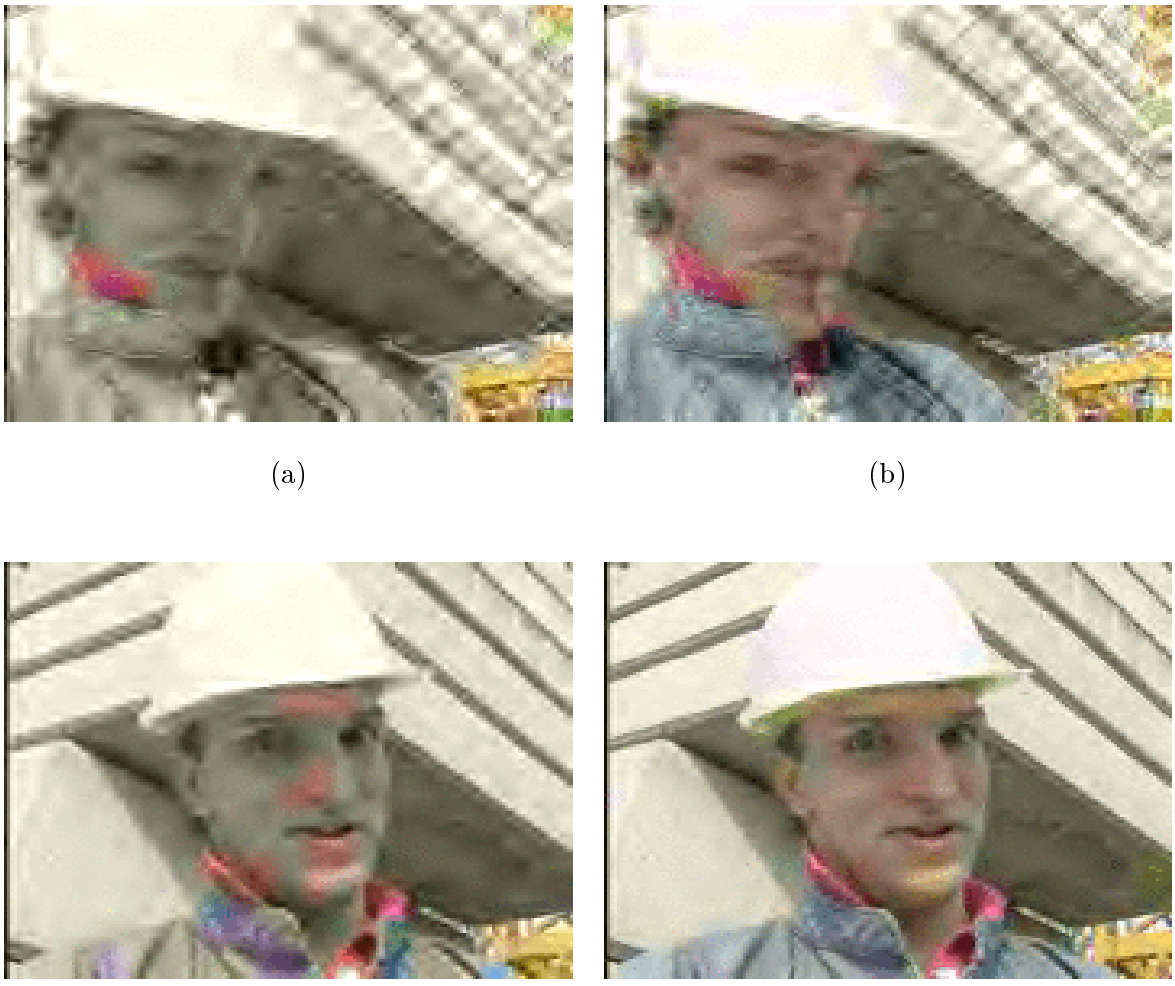
As explained earlier, employing a higher base reference byte will increase the minimum decodable bit rate of a given compressed bit stream.

Subsection 4.4.2 explained the various trade-offs between MEMC in the spatial domain and wavelet domain, and highlighted the advantages of the proposed wavelet-based multiresolution MEMC in comparison with other wavelet-based methods. Figure 4.12 further provides some visual insights between block-based MEMC in the spatial domain (using H.263 compression [62]) and the proposed wavelet-based MEMC. Frame (a) shows an original video frame of the Foreman sequence. It is evident from frame (b) that H.263 may suffer from severe blocking artifact in areas of high local motions. In fact, the poor spatial block mismatches are most pronounced around the nose, mouth, and helmet regions in the motion predicted frame. Frame (c) displays the corresponding motion predicted frame using wavelet-based MEMC. Clearly there is no blocking artifact although some ringing artifact is visible. Nevertheless, motion mismatches are also manifested around the mouth and helmet areas. In these blocks where no good block matching is possible, the proposed wavelet-based UCBDS algorithm may have switched from an inter-block to intra-block matching mode, or have chosen the preference zero motion vector bias strategy. It is also worth noting that H.263 performs MEMC in the original image resolution and thus it does not generate a scalable motion vector field. On the other hand, the proposed wavelet-based multiresolution MEMC is performed using a top-down hierarchical UCBDS motion search and it supports a true multiresolution scalable motion vector field that is needed for spatial resolution video scalability.

4.6 Conclusion

This chapter first motivated the need for motion estimation and motion compensation (MEMC) in video compression, and then presented a brief survey of the various MEMC approaches with particular focus on block-based matching algorithms. Based on the observation about the center-biasness of motion vector distribution in most video sequences, we

of different base reference values for the reference frame "locking" mechanism.

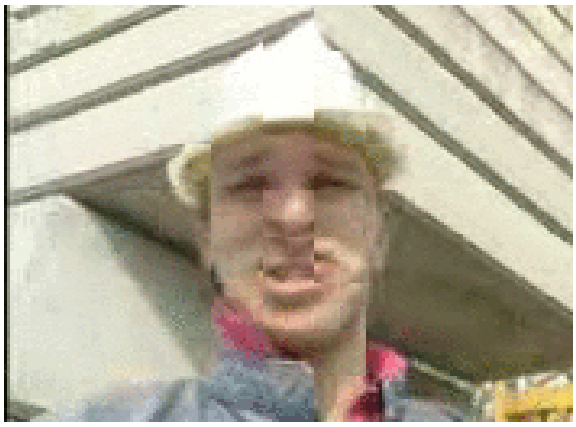


(a) (b)

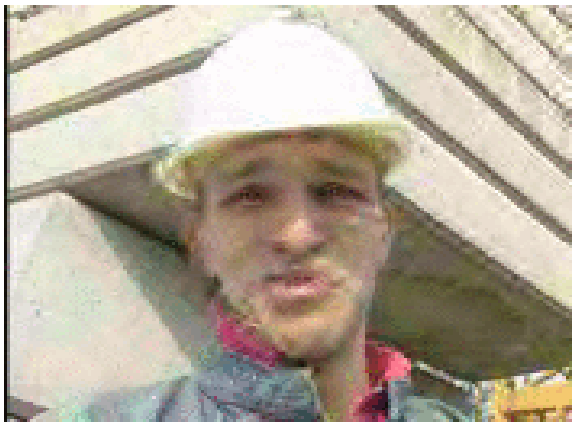
(c) (d)

Figure 4.11: Reference frames from Foreman sequence: Frames (a) and (b) have poorer motion compensation due to large motions; and frames (c) and (d) have better motion prediction accuracy. Frames (a) and (c) have 10% for the base reference byte while frames (b) and (d) have 20% for the base reference byte.

(a)



(b)



(c)

Figure 4.12: Comparison of block motion matching in the spatial domain and wavelet domain using the Foreman video sequence: (a) An original video frame; (b) predicted frame using H.263 motion estimation in the spatial domain; and (c) predicted frame using the proposed multiresolution UCBDS in the wavelet domain.

introduced a novel Unrestricted Center-Biased Diamond Search (UCBDS) block matching algorithm that capitalizes on this fact. UCBDS also adopts a halfway-stop search strategy, and has a best-case and average scenarios of only 13 and 15.5 block matches, respectively. Extensive simulations further validated the theoretical search speed gain against other block matching algorithms. For example, UCBDS could achieve up to 31% speed improvement over the fast four-step search algorithm, and was over 13 times faster than the full-search method using various test video sequences. In addition, UCBDS has also demonstrated consistently better performance with various metrics such as the average prediction error per pixel, probability of finding the true motion vector (with respect to the full-search), average distance from the true motion vector, and average number of search points per block.

In an effort to support spatial resolution video scalability, as well as other video scaling parameters, we extended the proven monogrid UCBDS algorithm to a top-down multiscale UCBDS framework where MEMC is carried out in the multiresolution scales of multiwavelet subbands. This approach benefits from a multitude of good features, ranging from lower computational complexity and no blocking artifacts to a multiresolution representation of the motion vector fields. However, performing MEMC in the wavelet domain is plagued by the inherent non-translational invariance property of critically-sampled wavelet transform. In this respect, we discussed the possible difficulty of motion compensating high-frequency wavelet signals, as well as the effect of motions with even and odd pixel shifts. Finally, we presented some simulation results of predicted frames using the proposed multiresolution wavelet-based UCBDS. It was shown that the prediction accuracy would deteriorate as the distance between the current and reference frames becomes larger. Also, the choice of a higher base reference byte while employing the reference frame locking mechanism can result in better quality predicted frames at the expense of increasing the minimum decodable bit rate in a given scalable bit stream.

Chapter 5

Multi-scalable Video Compression Platform

"The secret of health for both mind and body is not to mourn for the past, worry about the future, or anticipate troubles, but to live in the present moment wisely and earnestly."

Siddhartha Gautama Buddha (563 - 483 B.C.)

5.1 Introduction

In earlier chapters, we introduced and explained some new research results in the following areas: (i) a framework to design good orthogonal and biorthogonal multiwavelet filters and to apply them efficiently for multiresolution signal decomposition and reconstruction; (ii) a multi-scalable video compression framework with a simple prediction frame locking mechanism to achieve bit rate scaling; an efficient secured motion prediction loop to achieve spatial resolution scaling; a temporal hierarchy structure to achieve frame rate scaling; and an embedded bit stream hierarchy of frame blocks, layer blocks, resolution blocks, and color blocks to achieve simultaneous multi-scalable video compression; and (iii) a wavelet-based multiresolution motion estimation and compensation framework using the fast UCBDS algorithm that secures the motion prediction loop and generates a spatially scalable motion

vector field. This chapter carefully unifies these ideas to develop a novel multi-scalable video compression platform that supports simultaneous bit rate, frame rate, spatial resolution, and color video scalabilities at both the encoder and decoder. The primary goal here is on the development of the proposed multi-scalable framework and algorithm exposition of the scalable video encoder and decoder, rather than focusing on code optimization and compression performance efficiency of the codec (which can be further optimized in separate research).

A quick literature survey reveals the many research interests in subband or wavelet coding of video as well as to achieve different types of video scalability. Among them, video coding based on a three-dimensional (3-D) spatio-temporal subband decomposition framework has been widely investigated. For example, Podilchuk *et al.* [95] proposed a 3-D subband coding of video, with the application of a new adaptive differential pulse code modulation scheme for the lowest frequency band and a new geometric vector quantization technique for the higher frequency bands. However, no explicit video scalability was addressed. With the popularization of embedded or layered image coding strategies that support bit rate scaling (such as the zerotree compression by Shapiro [99] and the SPIHT compression by Said and Pearlman [97]), a new genre of scalable video compression frameworks based on 3-D wavelet coding were introduced. Tham *et al.* [5] have extended the spirit of zerotree image coding to a new 3-D zerotree data structure for video coding. In that work, motion-compensated temporal decomposition of a group of video frames and a new prioritization protocol for embedded video coding were employed to achieve simultaneous video scalability in terms of bit rate, spatial resolution, frame rate, color, and decoding complexity. However, it was shown that the unresolved open issue of motion-compensated temporal decomposition has limited its useful application to very low bit rate environments.

In a separate research, Taubman and Zakhor [110, 111] have also introduced a common framework for rate and distortion based scaling using a 3-D subband coding approach; however, its rate-distortion performance is not much better than that of MPEG since no motion estimation and compensation was employed. Other interesting research works include a zerotree wavelet video coder [85], a scalable video codec using virtual zerotree [118], and an

object-based scalable video codec [109]. In spite of the many research directions, none of the above frameworks is capable of supporting multi-scalable video compression with efficient and low-complexity encoding and decoding algorithms. All 3-D subband video coding approaches inherit a number of intrinsic problems: large memory requirement to store the entire set of frames for forward and inverse 3-D wavelet transforms; high frame coding latency that renders it unsuitable for real-time interactive video applications; and blurred reconstructed frames for the lower scale temporal resolution as a direct consequence of lowpass filtering (averaging) along the motion-shifted video frames. This clearly motivates the need for the proposed low-complexity multi-scalable video compression framework that supports fine-granularity video scaling for a wide range of live and on-demand applications.

Section 5.2 begins with an overview of the multiwavelet-based scalable video framework and defines terminology and notation that are useful for subsequent exposition. Section 5.3 then describes the algorithm development of the scalable video encoder with the help of block diagrams and pseudocode. Two new algorithms are introduced in subsection 5.3.2: Recursive direct splitting strategy in segmentation phase, and recursive overlay mapping strategy in segmentation phase. Detailed working examples are also provided to explain the two proposed recursive coding algorithms. Subsection 5.3.3 provides further insights into how the proposed wavelet-based motion estimation and compensation can be integrated for inter-frame coding. The corresponding scalable decoder counterpart with detailed examples is then expounded in Section 5.4 for both intra-frame and inter-frame coding modes. This is followed by subsection 5.4.3, which illuminates the degree and granularity of various supported video scaling parameters. Finally, Section 5.5 presents some simulation results and highlights two promising scalable video applications, followed by the conclusions.

5.2 Overview, Terminology, and Notation

This section presents the relevant terminology and notation that will aid the exposition of the proposed multi-scalable video encoder and decoder. In the course of doing so, we will be indirectly introducing a general overview of the video codec framework.

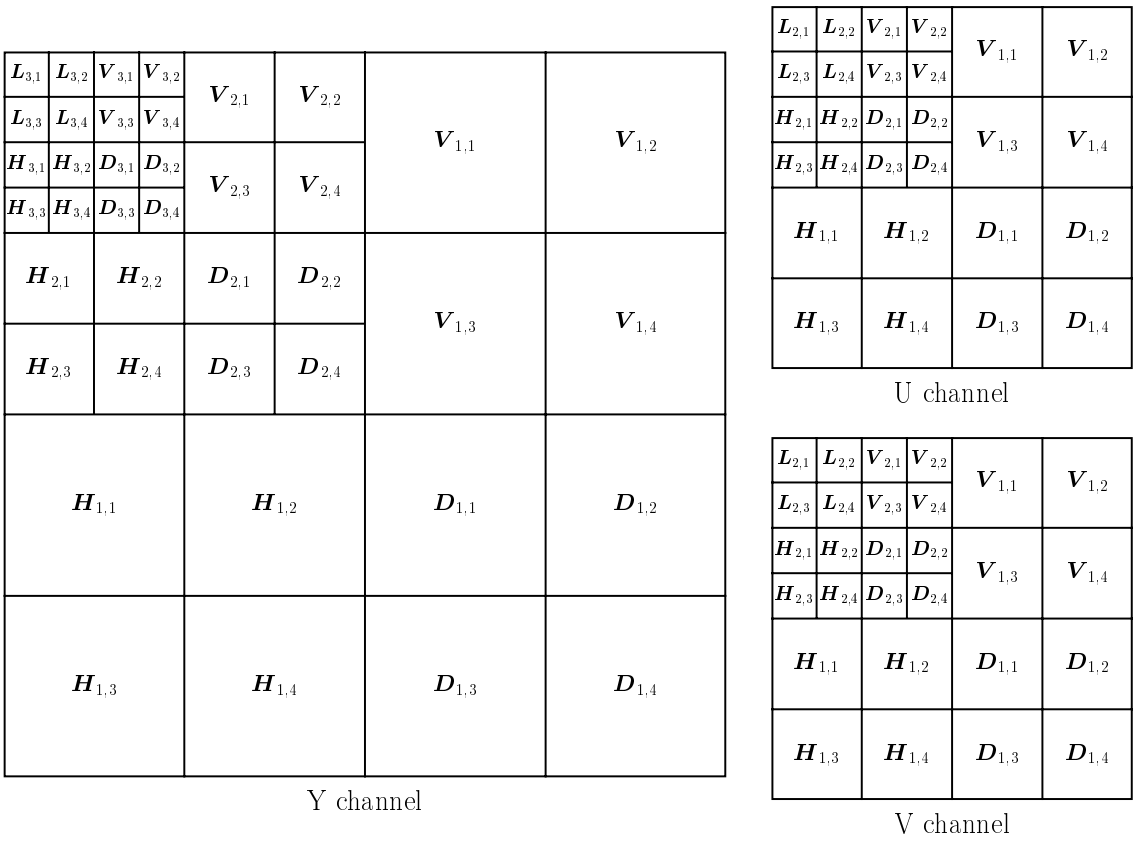


Figure 5.1: Multiwavelet subband structure: (i) Luminance channel, Y; (ii) Chrominance channel, U; and (iii) Chrominance channel, V.

We begin with a recapitulation of the subband structures of discrete multiwavelet transform, as introduced in Section 2.4. It is noted that each color video frame or image, $\mathcal{I}$, with the color triplets [R, G, B] is first transformed into the YUV 4:1:1 color domain, and all subsequent processing of the proposed video codec is performed in this transformed color domain. The subsampling ratio 4:1:1 denotes subsampling of the chrominance channels, U and V, by a factor of 2 in both horizontal and vertical directions. The luminance channel, Y, represents the intensity (brightness) of a video frame, while the chrominance channels, U and V, constitute the color information. In fact, a grayscale video frame can be obtained by discarding the U and V color channels and setting their values to 128 during reconstruction. The YUV color space and subsampling ratio are widely used in current compression standards such as MPEG [59], [60], [61], and H.263 [62].

Applying the generalized discrete multiwavelet transforms, as proposed in Section 2.4, we transform a video frame or image, $\mathcal{I}$, in the YUV domain to a multiwavelet-transformed

frame, $\mathcal{I}_{MW}$. Figure 5.1 illustrates the organization and labelling of the multiwavelet subbands of Y, U, and V channels. The Y channel is decomposed into $\mathcal{L}$ multiwavelet octave scales (in this case, $\mathcal{L} = 3$); the U and V channels are decomposed into $\mathcal{L} - 1$ scales as a result of the corresponding subsampling. The symbols $\mathbf{V}, \mathbf{H}, \mathbf{D}$, respectively, denote the vertical, horizontal, and diagonal orientation subbands of the decomposition at each resolution scale, while the symbol $\mathbf{L}$ represents the lowpass-filtered subimages. In general, we denote $\mathbf{S} \in \{\mathbf{L}, \mathbf{V}, \mathbf{H}, \mathbf{D}\}$ as a subband segment, or simply, a *segment*. The subscripts, ℓ and k, of a segment $\mathbf{S}_{\ell,k}$ represent a subband k at a particular resolution scale, ℓ, where $k \in \{1, 2, 3, 4\}$ and $\ell \in \{1, 2, \ldots, \mathcal{L}\}$. Also, let $J \in \{Y, U, V\}$ denote one of the three channels.

In the following, we will introduce the terminologies with respect to the Y channel; the corresponding notation for U and V channels is similar. As it is clear from the proposed multiwavelet transform, a segment consists of a set of multiwavelet coefficients, $\mathtt{s}(m, n)$; each coefficient comprises of a *sign* (denoted as `SIGN`) and *magnitude* (denoted as `MAG`), defined as:

$$\mathtt{SIGN}(\mathtt{s}(m,n)) = \begin{cases} + & \text{if } \mathtt{s}(m,n) \geq 0 \\ - & \text{if } \mathtt{s}(m,n) < 0, \end{cases}$$

and

$$\mathtt{MAG}(\mathtt{s}(m,n)) = |\mathtt{s}(m,n)|.$$

In an effort to improve the encoding efficiency, we will exploit the intra-subband (or intra-segment) relationship among the multiwavelet coefficients within a particular segment by encoding them in groups of adjacent coefficients. A group consisting of a set (or a rectangular block, in particular) of coefficients is denoted by the parameters $[m, n, M, N]$, where the block starts with a top-left reference coordinates (m, n) (with respect to the particular segment, $\mathbf{S}_{\ell,k}$, that the block lies in) and has dimensions of M and N in the horizontal and vertical directions, respectively. Clearly, a block is actually a *subsegment* that constitutes a portion of a segment. Hence, a subsegment or a block can be expressed as follows:

$$\mathbf{S}_{\ell,k}[m, n, M, N] = \bigcup_{a=0}^{M-1} \bigcup_{b=0}^{N-1} \mathtt{s}_{\ell,k}(m + a, n + b),$$

and a subband segment is represented as:

$$\mathbf{S}_{\ell,k}[0, 0, M_\ell, N_\ell],$$

where M_ℓ and N_ℓ are the dimensions of the subband segment at resolution scale ℓ. The dimension of a segment or subsegment is hence denoted as

$$\texttt{DIM}(\mathbf{S}[m, n, M, N]) := \{M, N\}.$$

For concise notation, let us also define $\mathbf{S}_{\ell,\mathbf{K}}$ as the union of the four segments with the same orientation and at the same resolution scale, ℓ, such that:

$$\mathbf{S}_{\ell,\mathbf{K}} := \bigcup_{k=1}^{4} \mathbf{S}_{\ell,k}.$$

In the process of encoding a segment of coefficients, we need to split the segment into a small set of subsegments if the segment of interest is not homogeneous (a more detailed definition will be given later). We have experimented with a number of splitting configurations and decided to employ a strategy similar to the popular quadtree splitting structure in our algorithms based on its simplicity and efficiency. Therefore each segment can be split further when necessary; each split level of a segment (or subsegment) is thus denoted by a superscript q. The following split operator, Ξ, will further illuminate the splitting process of an inhomogeneous segment:

$$\begin{aligned}\Xi: \quad \mathbf{S}^q[m, n, M, N] \quad \longrightarrow \quad & \mathbf{S}^{q+1}[m, n, \frac{M}{2}, \frac{N}{2}] \ \cup \ \mathbf{S}^{q+1}[m + \frac{M}{2}, n, \frac{M}{2}, \frac{N}{2}] \ \cup \\ & \mathbf{S}^{q+1}[m, n + \frac{N}{2}, \frac{M}{2}, \frac{N}{2}] \ \cup \ \mathbf{S}^{q+1}[m + \frac{M}{2}, n + \frac{N}{2}, \frac{M}{2}, \frac{N}{2}],\end{aligned}$$

where the four subsegments are denoted as $\Xi_{(1)}(\mathbf{S}^q), \Xi_{(2)}(\mathbf{S}^q), \Xi_{(3)}(\mathbf{S}^q)$, and $\Xi_{(4)}(\mathbf{S}^q)$, respectively, in that above order of appearance. Each subsegment can subsequently be viewed as a new segment that may be further split into four other subsegments — recursive splitting. Clearly, $M, N \geq 2$ for the splitting to be possible. Conversely, we can also define the inverse split operator, Ξ^{-1}, for any one of the four split subsegments to be the previous (just before splitting) segment from which that subsegment was derived from. Hence, we have

$$\Xi^{-1}(\mathbf{S}^q) = \mathbf{S}^{q-1}, \quad \forall q \geq 1, \quad \text{and} \quad \Xi^{-1}(\mathbf{S}^0) = \mathbf{S}^0.$$

As the proposed codec provides support for bit rate scalability, the algorithm encodes in multiple embedded *coding layers*, c, where each subsequent coding layer further improves the fidelity of a video frame that has been reconstructed using all previous coding layers. Therefore, to be more specific, a subsegment at a particular coding layer, c, at a particular split level, q, and of a particular channel, $J \in \{Y, U, V\}$, can be denoted as

$$\mathbf{S}_{\ell,k}^{c,q}[m, n, M, N], \quad c, q = 0, 1, 2, \ldots .$$

For concise notation, however, all or part of the subscripts and superscripts are omitted whenever it is clear from the context.

In addition to intra-subband relationship, we also exploit inter-subband relationships among subband segments at adjacent resolution scales. To do so, we represent the *parent* (denoted as PARENT) of a particular segment (or subsegment) as the corresponding segment (or subsegment) with the same orientation and subimage index at a higher (coarser) resolution scale; the inverse relationship is denoted as the *child* (denoted as CHILD). More specifically, we have the following parent-child relationships:

$$\texttt{PARENT}(\mathbf{S}_{\ell,k}) = \mathbf{S}_{\ell+1,k}; \quad \text{except} \quad \begin{cases} \texttt{PARENT}(\mathbf{L}_{\mathcal{L},k}) = \text{NULL} \\ \texttt{PARENT}(\mathbf{V}_{\mathcal{L},k}) = \mathbf{L}_{\mathcal{L},2} \\ \texttt{PARENT}(\mathbf{H}_{\mathcal{L},k}) = \mathbf{L}_{\mathcal{L},3} \\ \texttt{PARENT}(\mathbf{D}_{\mathcal{L},k}) = \mathbf{L}_{\mathcal{L},4}, \quad k = 1, 2, 3, 4, \end{cases}$$

and

$$\texttt{CHILD}(\mathbf{S}_{\ell,k}) = \mathbf{S}_{\ell-1,k}; \quad \text{except} \quad \begin{cases} \texttt{CHILD}(\mathbf{L}_{\mathcal{L},1}) = \texttt{CHILD}(\mathbf{S}_{1,k}) = \text{NULL} \\ \texttt{CHILD}(\mathbf{L}_{\mathcal{L},2}) = \bigcup_{k=1}^{4} \mathbf{V}_{\mathcal{L},k} \\ \texttt{CHILD}(\mathbf{L}_{\mathcal{L},3}) = \bigcup_{k=1}^{4} \mathbf{H}_{\mathcal{L},k} \\ \texttt{CHILD}(\mathbf{L}_{\mathcal{L},4}) = \bigcup_{k=1}^{4} \mathbf{D}_{\mathcal{L},k}, \quad k = 1, 2, 3, 4, \end{cases}$$

for $\mathbf{S} \in \{\mathbf{V}, \mathbf{H}, \mathbf{D}\}$. This further leads to the following *overlay mapping* operator, Υ:

$$\Upsilon: \quad \mathbf{S}_{\ell,k}[m, n, M, N] \quad \longrightarrow \quad \mathbf{S}_{\ell-1,k}[2m, 2n, 2M, 2N]$$

for $\mathbf{S} \in \{\mathbf{V}, \mathbf{H}, \mathbf{D}\}$ and $k = 1, 2, 3, 4$. The map operator for $\mathbf{S} \in \{\mathbf{L}_{\mathcal{L},2}, \mathbf{L}_{\mathcal{L},3}, \mathbf{L}_{\mathcal{L},4}\}$, however,

is defined as follows:

$$\begin{aligned}
\Upsilon: \quad & \mathbf{L}_{\mathcal{L},2}[m,n,M,N] \longrightarrow \bigcup_{k=1}^{4} \mathbf{V}_{\mathcal{L},k}[m,n,M,N], \\
\Upsilon: \quad & \mathbf{L}_{\mathcal{L},3}[m,n,M,N] \longrightarrow \bigcup_{k=1}^{4} \mathbf{H}_{\mathcal{L},k}[m,n,M,N], \\
\Upsilon: \quad & \mathbf{L}_{\mathcal{L},4}[m,n,M,N] \longrightarrow \bigcup_{k=1}^{4} \mathbf{D}_{\mathcal{L},k}[m,n,M,N],
\end{aligned}$$

and $\Upsilon(\mathbf{L}_{\mathcal{L},1})$ is undefined.

Associated with each coding layer, c, is a *threshold*, T_c, which differentiates one coding layer from another (previous or future) coding layer. In order to generate an embedded compressed bit stream, where a succeeding coding layer, $c = C + 1$, adds more refinement information to a frame that has been reconstructed from all previous base coding layers, $c \leq C$, we require that the set of thresholds, $\{\, T_c, c = 0, 1, \ldots \}$, should satisfy

$$T_0 > T_1 > \ldots > T_c > T_{c+1} > \ldots ,$$

for some predetermined initial threshold, T_0. As shown in [1], the choice of T_0 can be determined adaptively on a frame-by-frame basis so that an "optimum" value of T_0 will result in improved compression performance. However, for the purpose of real-time application such as video coding, we choose

$$T_0 = \frac{1}{2}\max\{\mathbf{s}(m,n) \in \mathcal{I}_{MW}\}.$$

In order to emulate an embedded bit-plane coding strategy (similar in spirit to that employed by Shapiro [99], and Said and Pearlman [97]), we set

$$T_c = \frac{1}{2}\, T_{c-1}, \quad c = 1, 2, \ldots .$$

A multiwavelet coefficient, $\mathbf{s}(m,n)$, is considered *significant* at a coding layer, c, if $|\mathbf{s}(m,n)| \geq T_c$; hence, we denote as $\mathtt{SIG}_c(\mathbf{s}(m,n)) = 1$. Otherwise, we denote $\mathtt{SIG}_c(\mathbf{s}(m,n)) = 0$ to show that a coefficient, $|\mathbf{s}(m,n)| < T_c$, is *insignificant* at the coding layer, c. Obviously, if $\mathtt{SIG}_{c=C}(\mathbf{s}(m,n)) = 1$, then we also have $\mathtt{SIG}_c(\mathbf{s}(m,n)) = 1$ for all $c > C$ in view of the monotonously decreasing thresholds. The definition can be extended to represent the *significance* of a segment, as follows:

$$\mathtt{SIG}_c(\mathbf{S}) = \begin{cases} 1 & \text{if } \exists \mathbf{s} \in \mathbf{S} \text{ s.t. } |\mathbf{s}| \geq T_c \\ 0 & \text{otherwise.} \end{cases}$$

In other words, a segment is considered significant if there is at least one coefficient within that segment is significant at the current coding layer; otherwise, the segment is insignificant if all its coefficients are still insignificant.

Focusing *only* on those coefficients within a segment that are still insignificant at all previous coding layers, we may further restrict the definition of significance of a segment to be exclusive to a particular coding layer (say, $c = C$) only. This *exclusive significance* of a segment can be expressed as follows:

$$\texttt{EXCLSIG}_C(\mathbf{S}) = \begin{cases} 1 & \text{if } \exists \mathbf{s} \in \mathbf{S} \text{ s.t. } T_C \leq |\mathbf{s}| < T_{C-1} \\ 0 & \text{otherwise.} \end{cases}$$

With these definitions, it is possible that a segment is significant at the current coding layer but, at the same time, it is still exclusively insignificant. Also, an insignificant segment will always imply exclusive insignificance but the converse may not always be true. As a result, we say that a given segment is *homogeneous* if all those previously insignificant coefficients within the segment satisfy either one of the following conditions at the current coding layer: (i) those coefficients are all exclusive significant, or (ii) those coefficients are all exclusive insignificant. Clearly, if all coefficients within a segment have already been significant at a previous coding layer, the segment is considered homogeneous at the current and all future coding layers. Otherwise, the segment is considered *inhomogeneous* or *heterogeneous* if it contains a mixture of exclusive significant and exclusive insignificant coefficients at the current coding layer.

We can now classify a segment into the following two complementary *significance regions*, $\mathbf{R}_c$ and $\mathbf{R'}_c$, which are representative of the collections of significant and insignificant multiwavelet coefficients, respectively, at a coding layer c:

$$\left.\begin{aligned} \mathbf{R}_c(\mathbf{S}) &= \{\mathbf{s} \in \mathbf{S} : \texttt{SIG}_c(\mathbf{s}) = 1\} \\ \mathbf{R'}_c(\mathbf{S}) &= \{\mathbf{s} \in \mathbf{S} : \texttt{SIG}_c(\mathbf{s}) = 0\} \end{aligned}\right\} \quad \text{s.t.} \quad \mathbf{S} = \mathbf{R}_c(\mathbf{S}) \ \cup \ \mathbf{R'}_c(\mathbf{S}).$$

These significance regions can then be overlay-mapped from a parent segment onto a child segment to provide a statistically reliable guide for contextual coding of multiwavelet coefficients in the child segment. For notational simplicity, let us also denote the following

mapped significance regions:

$$\begin{aligned} \mathbf{M}_c(\mathbf{S}_\ell) &:= \Upsilon(\mathbf{R}_c(\textsc{parent}(\mathbf{S}_\ell))), \\ \mathbf{M'}_c(\mathbf{S}_\ell) &:= \Upsilon(\mathbf{R'}_c(\textsc{parent}(\mathbf{S}_\ell))), \end{aligned}$$

for a resolution scale, ℓ. Equivalently, we can view a mapped significant (or insignificant) region as the set of multiwavelet coefficients in the child segment at resolution scale ℓ whose corresponding parent coefficients at resolution scale $\ell + 1$ are significant (or insignificant) at the current coding layer, c. Clearly, the mapped significance regions are only defined for those segments that have valid child segments.

5.3 Multi-scalable Video Encoder

This section details the proposed algorithms for compressing a video source into a binary bit stream consisting of multiple resolution blocks that supports flexible video scalability. The following subsection first describes the various types of redundancies present in video coding, and shows how they can be exploited for improved compression performance. This is followed by a meticulous discussion of the scalable video encoder. Pseudocode and examples of main encoding procedures are given to help illuminate the steps involved in generating a scalable compressed bit stream.

5.3.1 Exploitation of Redundancies

For efficient video coding, the encoder algorithm has to exploit the various redundancies that are present in the video frames. The following describe five different types of redundancies, and briefly point out how they can be exploited in the proposed video encoding algorithms:

- **Intra-subband:** Although the multiwavelet coefficients are decorrelated after the transform, they are not independent even within a subband segment. There is a high probability that adjacent coefficients within a small neighborhood exhibit similar values, especially in homogeneous (smooth) regions of an image. This interdependence

within a neighborhood can be exploited by encoding a group of multiwavelet coefficients together in blocks or subsegments.

- **Inter-subband:** On a similar note, there exists interdependency between coefficients (or subsegments) across different resolution scales; this is primarily due to the self-similar structure of octave-bandwidth multiwavelet transform. A parent segment will have a one-to-four dependency with its child segments. Conversely, the significance characteristics of coefficients in a child segment will be closely related to that of the parent segment. This interdependency can be exploited by mapping the significance regions of the parent segment onto the child segment, and using these overlay maps to guide the encoding of the coefficients of the child segment.

- **Inter-channel:** This is due to the presence of luminance and chrominance channels in a color image or video frame. Although the channels are decorrelated via transformation from RGB to YUV domain, there still exist some dependencies among segments across the channels, both within the same resolution scale and across scales. Such dependencies can be exploited by using a similar overlay mapping strategy from one segment onto another segment across the channels.

- **Inter-coding layer:** This is due to the fact that the magnitudes of multiwavelet coefficients are being encoded in multiple embedded coding layers. Since the series of thresholds are monotonically decreasing, coefficients that are already significant in a previous coding layer will definitely be significant with respect to the threshold at a future coding layer. This also means that only insignificant coefficients at all previous coding layers need to be evaluated and encoded in the current coding layer. By excluding all significant coefficients when encoding a segment at a particular segmentation phase[1], the algorithm can focus on effective coding of those coefficients that are still insignificant within the segment. Also, the overlay map of the significant region (as a union of all the significant regions across all previous coding layers up to the current layer) from its parent segment will provide some contextual hints in

[1] These significant coefficients will then be refined during the refinement phase.

addition to its own significance regions.

- **Inter-frame:** There exist high temporal correlations between adjacent video frames, especially in regions of frames without large motions (e.g. object occlusions, scene cuts, etc.). Motion estimation can be a very effective mechanism to exploit such correlations. In doing so, a set of motion vectors that map blocks from a reference segment (in a reference frame) onto blocks of the current segment (in the current frame) are estimated using a fast block matching algorithm. The difference between the current segment and the motion-compensated segment can then be encoded in a more effective manner with fewer bits (even after encoding the motion vectors).

5.3.2 Intra-Frame Scalable Encoding Algorithms

As explained in Section 3.4 and in the general overview earlier, we have presented a brief introduction to the proposed scalable video codec and how the multi-scalable compressed bit stream is organized into embedded frame, layer, resolution and color blocks. This subsection will further explain the scalable encoder algorithms with appropriate pseudocode and examples.

The proposed scalable video encoder consists of two main components: (i) switching between the intra-frame coding mode and the inter-frame with motion compensation coding mode; and (ii) generating a highly scalable compressed bit stream that consists of resolution blocks for both coding modes. The first component has been clearly explained in Subsection 3.4.3 with a procedure to select the appropriate reference frame for inter-frame motion estimation so as to support temporal scalability of a certain granularity. In contrast to the inter-frame coding mode, intra-frame coding does not depend on a previously encoded frame, and hence it requires no motion compensation. Nevertheless, the scalable algorithms that are employed for encoding the subband segments are, in fact, similar in both coding modes, except that the content (i.e., the values of the multiwavelet coefficients) of the segments will be different after motion compensation. In view of this, we will focus the following exposition of the encoder algorithms in the context of intra-frame coding of a particular multiwavelet-decomposed color frame. Clearly, this is directly applicable to still

image coding as well.

As pointed out earlier, the scalable encoder encodes in multiple embedded coding layers, where each layer comprises of two coding phases: segmentation phase and refinement phase. In each coding phase, all subband segments in the luminance and chrominance channels are encoded to generate the resolution blocks. In order to simultaneously support the various video scaling parameters, a *prioritization protocol* (see also [5]) is needed to encode the segments in a specific order. Employing this protocol, all segments in the luminance channels are encoded first before encoding those in the chrominance channels. Within each coding layer, segments within the same resolution scale are grouped and encoded into *one* resolution block, starting from the lowpass subimage (with the lowest frequency content) to the finest resolution scale. This also means that parent segments are always encoded prior to encoding the child segments.

During a segmentation phase at a particular coding layer, each coefficient in a segment is classified (segmented) as either significant or insignificant with respect to the threshold associated with the coding layer. In each coding layer, the position and magnitude of each significant coefficient are (indirectly) encoded simply by the above thresholding mechanism (which implies that the magnitude lies between the current threshold and the larger threshold of the previous coding layer) using a specific coefficient scanning order. The corresponding sign (either a positive or negative) is encoded explicitly. After encoding all segments with this segmentation phase, a refinement phase is performed for the same coding layer. It is noted that the magnitudes of significant coefficients have been coarsely quantized with relatively large quantization bin sizes. Clearly, this results in large quantization errors and hence poor quality of the reconstructed frame. To remedy this weakness, the refinement phase will add another bit of precision by halving the quantization bin sizes. In essence, each subsequent coding layer will identify more new significant coefficients and also refine the precision of all previously significant coefficients. Such a multi-layer coding strategy with successive refinement of significant coefficients has been successfully employed in popular image codecs, such as those proposed by Shapiro [99], and Said and Pearlman [97].

```
EncodeOneColorFrame( ) {
   c = 0;  /* initialize current coding layer */
   /* encode until the target bit budget for current frame is exhausted */
   while(current_bits_used < allocated_frame_bits) {
      /* encode each of both coding phases alternately */
      for(phase ∈ {Segmentation, Refinement}) {
         /* encode luminance component */
         EncodePhase(L^{(Y),c,0}_{L,1}, L^{(Y),c,0}_{L,2}, L^{(Y),c,0}_{L,3}, L^{(Y),c,0}_{L,4}, phase);
         EncodePhase(V^{(Y),c,0}_{L,K}, H^{(Y),c,0}_{L,K}, D^{(Y),c,0}_{L,K}, phase);
         InsertHeaderAndPacketize( );
         for(each remaining resolution scale, ℓ = L − 1, ... , 1) {
            EncodePhase(V^{(Y),c,0}_{ℓ,K}, H^{(Y),c,0}_{ℓ,K}, D^{(Y),c,0}_{ℓ,K}, phase);
            InsertHeaderAndPacketize( );
         }
         /* encode chrominance components, if necessary */
         if(it is a color frame) {
           EncodePhase(L^{(U),c,0}_{L−1,1}, L^{(V),c,0}_{L−1,1}, phase);
           EncodePhase(L^{(U),c,0}_{L−1,2}, L^{(V),c,0}_{L−1,2}, L^{(U),c,0}_{L−1,3}, L^{(V),c,0}_{L−1,3}, L^{(U),c,0}_{L−1,4}, L^{(V),c,0}_{L−1,4}, phase);
           InsertHeaderAndPacketize( );
           for(each remaining resolution scale, ℓ = L − 2, ... , 1) {
              EncodePhase(V^{(U),c,0}_{ℓ,K}, V^{(V),c,0}_{ℓ,K}, H^{(U),c,0}_{ℓ,K}, H^{(V),c,0}_{ℓ,K}, D^{(U),c,0}_{ℓ,K}, D^{(V),c,0}_{ℓ,K}, phase);
              InsertHeaderAndPacketize( );
           }
         }
      }  /* next coding phase */
      c++;
   }  /* next coding layer */
}
```

Figure 5.2: Pseudocode for encoding a color frame into resolution blocks with appropriate headers in order to generate a highly scalable compressed bit stream.

Figure 5.2 presents the pseudocode for encoding a color frame into appropriate resolution blocks. It also clearly outlines the order of encoding the segments according to the prioritization protocol. The function `EncodePhase()` is responsible for encoding one segment (by choosing a segment at a time from the input list of segments) using either the segmentation or refinement phase. For the segmentation phase, a segment can be encoded using either one of two different coding strategies: (i) *recursive direct splitting*, which encodes a segment without a valid parent; or (ii) *recursive overlay mapping*, which exploits encoded information of a parent segment to guide the encoding process of a (child) segment. Detailed explanation on these two coding strategies will be the emphasis of the next two subsections. Also, the algorithm for performing the refinement coding phase will be explained.

5.3.2.1 Recursive Direct Splitting Strategy in Segmentation Phase

Figure 5.3 illustrates the pseudocode for encoding a given segment at a particular coding layer using the recursive direct splitting strategy (via the `DirectSplitEncodeOneSegment()` function). As mentioned, the segment that is to be encoded does not have a valid parent segment. Nonetheless, this coding strategy exploits intra-segment relationship by encoding blocks of neighboring coefficients. A symbol is entropy encoded to represent the homogeneity of the segment. If the segment is heterogeneous, it will be split via the split operator Ξ; each of the four subsegments will further be split (if necessary) and encoded in a depth-first recursive manner until its dimensions become smaller than a predetermined value `min_dim`, or the subsegment at a particular split level, q, is homogeneous. If the dimensions reach a minimum size, splitting is terminated and the function `EncodeSubsegment()` is called. Figure 5.4 presents the pseudocode of this function, which essentially encodes the signs and positions of all exclusive significant coefficients within the segment in a raster-scan (left-to-right and top-to-bottom) order.

Example 5.1.

Consider an example 4×4 segment, $\mathbf{S}_\ell$, as portrayed in Figure 5.5. Assume that the example segment does not have a valid parent segment, and hence it is encoded using a recursive

```
DirectSplitEncodeOneSegment(S_{ℓ,k}^{(J),c,q}) {
   /* terminate recursive splitting if the dimension is small enough */
   if(max{DIM(S_{ℓ,k}^{(J),c,q})} <= min_dim) {
      EncodeSubsegment(S_{ℓ,k}^{(J),c,q});
      return;
   }
   /* adaptive entropy encode the exclusive significance of current segment */
   AdaptArithEncSym(EXCLSIG_c(S_{ℓ,k}^{(J),c,q}), amodelC);
   /* recursive split the current segment into four subsegments */
   if(S_{ℓ,k}^{(J),c,q} is heterogenous) {
      DirectSplitEncodeOneSegment(Ξ_(1)(S_{ℓ,k}^{(J),c,q}));
      DirectSplitEncodeOneSegment(Ξ_(2)(S_{ℓ,k}^{(J),c,q}));
      DirectSplitEncodeOneSegment(Ξ_(3)(S_{ℓ,k}^{(J),c,q}));
      DirectSplitEncodeOneSegment(Ξ_(4)(S_{ℓ,k}^{(J),c,q}));
   }
}
```

Figure 5.3: Pseudocode for encoding a given segment, $\mathbf{S}_{\ell,k}^{(J)}$, at a particular coding layer, c, using a recursive direct splitting strategy.

direct splitting strategy. In fact, this segment may equally well be a split subsegment at a particular split level q. Let the small squares within this segment represent each of the 16 multiwavelet coefficients. Those marked with "+" and "–" are exclusive significant coefficients at the current coding layer; those marked with "$\oplus$" and "$\ominus$" are coefficients that have already been significant prior to the current coding layer; and those unmarked denote insignificant coefficients at the current coding layer c.

Table 5.1 illustrates the steps for generating the bit stream of encoding the example segment using the recursive direct splitting algorithm, as presented in Figure 5.3. For this example, we set the constant `min_dim` $= 2$ in Figure 5.3 so that the recursive splitting terminates when a subsegment reaches the dimensions of $M = N = 2$. We begin with arithmetic encoding the exclusive significance of the 4×4 segment (i.e., the symbol $\textsc{exclsig}_c(\mathbf{S}_{\ell,k}^{c,0}[m, n, 4, 4])$, which clearly is a **1** as there are three "+" and one "–" exclu-

```
EncodeSubsegment(S_{ℓ,k}^{(J),c,q}) {
   /* encode a bit representative of the significance of subsegment */
   AdaptArithEncSym(EXCLSIG_c(S_{ℓ,k}^{(J),c,q}), amodelD);
   /* encode information of significant coefficients within subsegment */
   /* in a raster-scan manner from left-to-right and from top-to-bottom */
   for(each coefficient s.t. SIG_{c-1}(s ∈ S_{ℓ,k}^{(J),c,q}) == 0) {
      if(SIG_c(s ∈ S_{ℓ,k}^{(J),c,q}) == 1) {  /* significant coefficient */
         AdaptArithEncSym(SIGN(s), amodelE);
         s = MAG(s) − T_c;
      }
      else  AdaptArithEncSym(0, amodelE);  /* insignificant coefficient */
   }
}
```

Figure 5.4: Pseudocode for encoding the multiwavelet coefficients of a given subsegment, $\mathbf{S}_{\ell,k}^{(J)}$, at a particular coding layer, c.

sive significant coefficients within this segment. Consequently, the heterogeneous segment is split via Ξ into four 2×2 subsegments: $\mathbf{S}_{\ell,k}^{c,1}[m,n,2,2]$, $\mathbf{S}_{\ell,k}^{c,1}[m+2,n,2,2]$, $\mathbf{S}_{\ell,k}^{c,1}[m,n+2,2,2]$, and $\mathbf{S}_{\ell,k}^{c,1}[m+2,n+2,2,2]$. The subsegment $\Xi_{(1)}$ is first evaluated and is encoded with a **1** as it contains one "+" exclusive significant coefficient. As it is evident from the pseudocode in Figure 5.3, the recursive splitting process is *depth-first* instead of *breath-first*. Hence the subsegment $\Xi_{(1)}$ will be further split and encoded first before considering subsegments $\Xi_{(2)}, \Xi_{(3)}$, and $\Xi_{(4)}$. Since the dimensions of $\Xi_{(1)}$ have reached a minimum of 2×2, the four multiwavelet coefficients, $\mathbf{s}$, will now be encoded by calling the function `EncodeSubsegment(`$\mathbf{S}_{\ell,k}^{c,1}[m,n,2,2]$`)`. This produces a symbol **0** for the exclusive insignificant coefficient, $\mathbf{s}(0,0)$, and a symbol + for the exclusive significant coefficient, $\mathbf{s}(1,0)$, which has a positive sign. The other two coefficients, $\mathbf{s}(0,1)$ and $\mathbf{s}(1,1)$, need not be encoded (hence, denoted by the symbol "?" in Table 5.1) as they would have been encoded in a previous coding layer; this indirectly exploits the inter-coding layer dependency of coefficients.

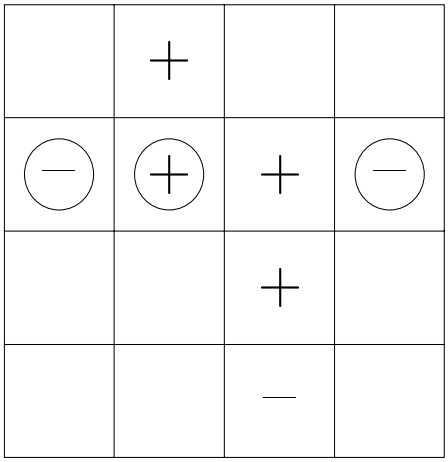


Figure 5.5: Example of a 4×4 segment that is encoded at a particular coding layer, c, using recursive direct splitting.

$M \times N$	4×4	2×2	1×1
Bit stream	(1)	(1)	(0 + ? ?)
		(1)	(0 0 + ?)
		(0)	(?)
		(1)	(+ 0 − 0)

Table 5.1: Example of encoded bit stream of a segment using recursive direct splitting.

When the function `EncodeSubsegment(`$\mathbf{S}_{\ell,k}^{c,1}[m, n, 2, 2]$`)` returns, the exclusive significance of the subsegment $\Xi_{(2)}$ is encoded; in this case, a symbol **1** is generated because of the presence of one "+" coefficient. This leads to calling the function `EncodeSubsegment(`$\mathbf{S}_{\ell,k}^{c,1}[m+2, n, 2, 2]$`)`, which produces three symbols: **0**, **0**, and +. Next, the subsegment $\Xi_{(3)}$ is encoded with a **0** as it has no exclusive significant coefficients at all. Finally, the subsegment $\Xi_{(4)}$ is encoded with a **1** for having two exclusive significant coefficients, and the call to the function `EncodeSubsegment(`$\mathbf{S}_{\ell,k}^{c,1}[m+2, n+2, 2, 2]$`)` yields the four symbols: +, **0**, −, and **0**. As a result, the symbol stream representing the encoding of the 4×4 segment, $\mathbf{S}_{\ell,k}^{c,0}[m, n, 4, 4]$, via a recursive direct splitting strategy is given by

$$(\mathbf{1}, \mathbf{1}, \mathbf{0},\ +,\ \mathbf{1}, \mathbf{0}, \mathbf{0},\ +,\ \mathbf{0}, \mathbf{1},\ +,\ \mathbf{0},\ -,\ \mathbf{0}),$$

which is essentially a two-symbol stream (i.e., **1** or **0**). However, the symbols generated while encoding the 1×1 subsegments/coefficients form a three-symbol stream (i.e., **0**, +, or −).

We can effectively entropy encode the above symbol stream by switching adaptively between a two-symbol and a three-symbol arithmetic model (see e.g. [119]) according to the subsegment that is being encoded. For simplicity, a fixed two- or three-alphabet model with predefined probabilities can be used. However, improved compression performance can be expected by employing an adaptive model where the probability distributions of the symbols are updated only after a new symbol has been encoded. As expounded in [119], an adaptive arithmetic model can better keep track of the actual probability distributions of various symbols and hence better approach the entropy limit of the source. In addition, we can also exploit some other information (or clues) during encoding by adopting a *contextual modelling/switching* approach that uses a finite set of adaptive models, each being tailored to a particular context (see e.g. [32, 90]). The contexts provide some form of prediction for switching to the appropriate adaptive model when encoding a symbol. If the contexts are reliable, the probability distribution of each adaptive model will become skewed (biased) to a certain symbol, although the overall probability distibution of the symbols can be equally likely. The arithmetic encoder, therefore, can encode the same symbol stream with fewer bits. Context switching is employed in our next coding strategy.

5.3.2.2 Recursive Overlay Mapping Strategy in Segmentation Phase

Unlike the recursive direct splitting technique, the recursive overlay mapping strategy also exploits inter-subband relationships across resolution scales in addition to intra-subband relationships. As explained in the overall algorithm in Figure 5.2, a parent subband is encoded prior to its child subband in every coding layer. This means that the encoded information of a parent segment at a particular coding layer can be used to provide some contextual guide for encoding the child segment at the current and future coding layers. In the proposed algorithm, we will map the significant and insignificant regions of a parent segment, and then overlay the map as a mask onto the child segment, with the sole object of

```
OverlayMapEncodeOneSegment(S_{ℓ,k}^{(J),c,q}) {
   /* terminate recursive splitting if the dimension is small enough */
   if(max{DIM(S_{ℓ,k}^{(J),c,q})} <= min_dim) {
      EncodeSubsegment(S_{ℓ,k}^{(J),c,q});  return;
   }
   /* adaptive entropy encode both significance regions of current segment */
   if(R_c(PARENT(S_{ℓ,k}^{(J),c,q})) != NULL) {  /* both significant overlay and */
      if(R'_c(PARENT(S_{ℓ,k}^{(J),c,q})) != NULL) {  /* insig. overlay are present */
         if(EXCLSIG_c(R_c(Ξ^{-1}(S_{ℓ,k}^{(J),c,q}))) == 1)
            AdaptArithEncSym(EXCLSIG_c(M_c(S_{ℓ,k}^{(J),c,q})), amodelA);
         if(EXCLSIG_c(R'_c(Ξ^{-1}(S_{ℓ,k}^{(J),c,q}))) == 1)
            AdaptArithEncSym(EXCLSIG_c(M'_c(S_{ℓ,k}^{(J),c,q})), amodelB);
      } else {  /* only significant overlay is present */
         if(EXCLSIG_c(R_c(Ξ^{-1}(S_{ℓ,k}^{(J),c,q}))) == 1)
            AdaptArithEncSym(EXCLSIG_c(M_c(S_{ℓ,k}^{(J),c,q})), amodelA);
         else  return;
      }
   } else {  /* only insignificant overlay is present */
      if(EXCLSIG_c(R'_c(Ξ^{-1}(S_{ℓ,k}^{(J),c,q}))) == 1)
         AdaptArithEncSym(EXCLSIG_c(M'_c(S_{ℓ,k}^{(J),c,q})), amodelB);
      else  return;
   }
   /* recursive split the current segment into four subsegments */
   if(S_{ℓ,k}^{(J),c,q} is heterogeneous) {
      OverlayMapEncodeOneSegment(Ξ_{(1)}(S_{ℓ,k}^{(J),c,q}));
      OverlayMapEncodeOneSegment(Ξ_{(2)}(S_{ℓ,k}^{(J),c,q}));
      OverlayMapEncodeOneSegment(Ξ_{(3)}(S_{ℓ,k}^{(J),c,q}));
      OverlayMapEncodeOneSegment(Ξ_{(4)}(S_{ℓ,k}^{(J),c,q}));
   }
}
```

Figure 5.6: Pseudocode for encoding a given segment, $\mathbf{S}_{\ell,k}^{(J)}$, at a particular coding layer, c, using a recursive overlay mapping strategy.

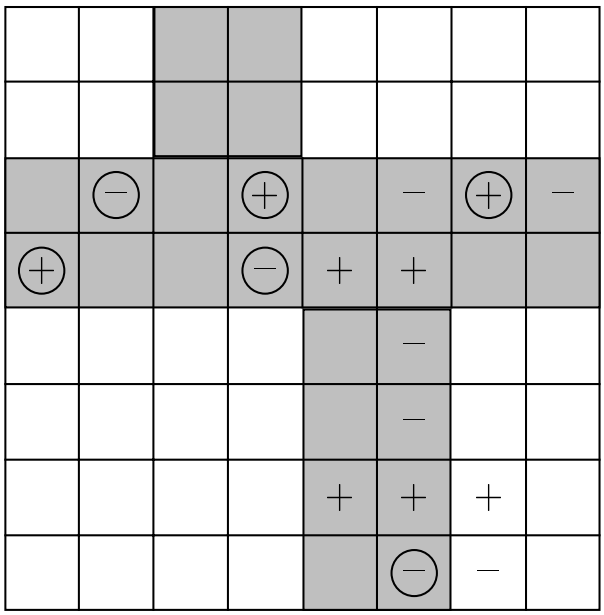


Figure 5.7: Example of a 8×8 child segment that is encoded at a particular coding layer, c, using recursive overlay mapping.

further improving the overall compression efficiency. The motivation for doing this is based on the observation that, if a parent coefficient is insignificant, there is a high probability that the coefficients in the corresponding child segment will be exclusive insignificant. Also, if a child coefficient is exclusive significant at a particular coding layer, then the corresponding parent coefficient is very likely to be significant. Hence, it is envisaged that there will be significant savings in the number of bits used if the overlay maps have provided reliable contexts when encoding a child segment. Figure 5.6 presents the pseudocode for the proposed recursive overlay mapping algorithm (via function `OverlayMapEncodeOneSegment()`).

Example 5.2.

In this example, we assume the segment that is to be encoded has a valid 4×4 parent segment (as shown in Figure 5.5) which we had encoded earlier in the previous example for recursive direct splitting. Figure 5.7 portrays the 8×8 child segment that is to be encoded at the same current coding layer, c, as that for the parent segment. The signs of the significant and exclusive significant coefficients, as well as the insignificant coefficients, are shown using the same notation. Consider the current segment $\mathbf{S}_{\ell,k}^{c,q}[m, n, 8, 8]$. To begin the encoding process in Figure 5.6, we set $\texttt{EXCLSIG}_c(\mathbf{R'}_c(\Xi^{-1}(\mathbf{S}_{\ell,k}^{c,0}))) = 1$, and

$M \times N$	8×8	4×4	2×2	1×1
	(1 1)	(0 0)		
		(1 0)	(?)	(?)
			(?)	(?)
			(1)	(0 – + +)
Bit stream			(1)	(? – 0 0)
		(? 0)		
		(1 1)	(1)	(0 – 0 –)
			(0)	
			(1)	(+ + 0 ?)
			(?)	(+ 0 – 0)

Table 5.2: Example of encoded bit stream of a segment using recursive overlay mapping.

the constant `min_dim` = 2. We also overlay map the significant and insignificant regions, $\mathbf{R}_c(\texttt{PARENT}(\mathbf{S}_{\ell,k}^{c,q}))$ and $\mathbf{R'}_c(\texttt{PARENT}(\mathbf{S}_{\ell,k}^{c,q}))$, of the parent segment onto the current segment; these maps are distinguished by the shaded and unshaded regions, respectively, in Figure 5.7. It is worth emphasizing that the overlay significant map is due to all the *significant* coefficients in the parent segment so far, and *not* only due to the exclusive significant coefficients there at the current coding layer.

Table 5.2 helps explain the algorithm presented in Figure 5.6 by showing the steps involved in generating the bit stream of a child segment using a recursive overlay mapping technique. We start with the 8×8 segment, $\mathbf{S}_{\ell,k}^{c,0}[m, n, 8, 8]$. As it contains both the significant and insignificant overlay maps, a symbol *may*[2] be generated for each of these maps. The set of coefficients in the current segment that is masked with a significant overlay map is always encoded before the complementary set that is masked with an insignificant overlay map. In this example, a symbol $\texttt{EXCLSIG}_c(\mathbf{M}_c(\mathbf{S}_{\ell,k}^{c,0})) = \mathbf{1}$ is produced for the overlaid

[2]Note that a symbol need not be produced for an overlaid region that is either already completely significant in a previous coding layer, or its similar region in the previous (larger) segment before the splitting has no exclusive significant coefficients.

significant region (as it contains four "+" and four "–" exclusive significant coefficients), and a symbol $\texttt{EXCLSIG}_c(\mathbf{M'}_c(\mathbf{S}_{\ell,k}^{c,0})) = \mathbf{1}$ is then generated for the overlaid insignificant region (as it contains one "+" and one "–" exclusive significant coefficient). It is noted, however, that the symbols are arithmetic entropy encoded using different adaptive models (namely, `amodelA` and `amodelB`, as shown in Figure 5.6) that are tailored to the contexts provided by the corresponding significance regions of the parent segment.

The algorithm splits the current segment into four subsegments using the operator Ξ, and then calls the recursive function $\texttt{OverlayMapEncodeOneSegment}(\mathbf{S}_{\ell,k}^{c,1}[m,n,4,4])$. Clearly, the current 4×4 segment contains both overlaid significant and insignificant regions, each of which generates a symbol **0** as they have no exclusive significant coefficients. In other words, the segment $\mathbf{S}_{\ell,k}^{c,1}[m,n,4,4]$ is homogeneous, and this terminates the splitting process of the current segment at split level $q = 1$. It then proceeds to recursive encode the 4×4 segment, $\mathbf{S}_{\ell,k}^{c,1}[m+4,n,4,4]$, which also contains both overlaid significant and insignificant regions. The significant mapped region of the current segment produces a symbol $\texttt{EXCLSIG}_c(\mathbf{M}_c(\mathbf{S}_{\ell,k}^{c,1}[m+4,n,4,4])) = \mathbf{1}$ as it comprises two "+" and two "–" exclusive significant coefficients. The insignificant mapped region, however, generates a symbol $\texttt{EXCLSIG}_c(\mathbf{M'}_c(\mathbf{S}_{\ell,k}^{c,1}[m+4,n,4,4])) = \mathbf{0}$ because all the eight coefficients are insignificant at the current coding layer.

As there is at least one exclusive significant coefficient, the heterogeneous 4×4 segment is further split into four 2×2 subsegments with $q = 2$. This is followed by calling the recursive function $\texttt{OverlayMapEncodeOneSegment}(\mathbf{S}_{\ell,k}^{c,2}[m+4,n,2,2])$. Clearly, this 2×2 subsegment contains only an overlaid insignificant region. Since $\texttt{EXCLSIG}_c(\mathbf{R'}_c(\Xi^{-1}(\mathbf{S}_{\ell,k}^{c,2}[m+4,n,2,2]))) = 0$, it has already implicitly implied the absence of any exclusive significant coefficients in the subsegment; hence, no symbol is generated. By the same token, no symbol is produced for the next 2×2 subsegment $\mathbf{S}_{\ell,k}^{c,2}[m+6,n,2,2]$. However, the third 2×2 subsegment, $\mathbf{S}_{\ell,k}^{c,2}[m+4,n+2,2,2]$, which comprises only the overlaid significant region, is encoded with a symbol **1** as it contains three exclusive significant coefficients. It is also noted that no additional symbol is needed to represent the absence of an overlaid insignificant region in the subsegment since this piece of information has had been implicitly

conveyed by exploiting the overlay maps of its parent subsegment. As the subsegment's dimension now reaches `min_dim`, the function `EncodeSubsegment(`$\mathbf{S}^{c,2}_{\ell,k}[m+4, n+2, 2, 2]$`)` is called to generate the following four symbols: **0**, −, +, and +. Lastly, the fourth subsegment, $\mathbf{S}^{c,2}_{\ell,k}[m+6, n+2, 2, 2]$, is encoded with a symbol **1** to represent the presence of at least one exclusive significant coefficient. The positional and sign information of the exclusive significant coefficients is further encoded by the function `EncodeSubsegment()` as three symbols: −, **0**, and **0**. Note that the first positive coefficient, which was already significant in a previous coding layer, is not encoded (an example of exploiting inter-coding layer redundancy).

The algorithm continues to recursively encode the next 4×4 segment, $\mathbf{S}^{c,1}_{\ell,k}[m, n+4, 4, 4]$. Since it does not contain an overlaid significant region, no symbol is generated. The overlaid insignificant region, however, is homogeneous as being exclusive insignificant, and hence it is encoded with only one symbol **0**. The segment need not be recursively split and no other symbols are generated.

Lastly, the last 4×4 segment, $\mathbf{S}^{c,1}_{\ell,k}[m+4, n+4, 4, 4]$, is recursively encoded. Two symbols, **1** and **1**, are generated to convey the presence of at least one exclusive significant coefficient in the overlaid significant and insignificant regions, respectively. The segment is further split into four 2×2 subsegments. The first subsegment, $\mathbf{S}^{c,2}_{\ell,k}[m+4, n+4, 2, 2]$, is encoded with a symbol **1** to denote the presence of two exclusive significant coefficients, which are subsequently encoded with a symbol stream of **0**, −, **0**, and −. The second subsegment, which is homogeneous with no exclusive significant coefficient, can be completely encoded with only a symbol **0**. Similar to the first subsegment, the third subsegment, $\mathbf{S}^{c,2}_{\ell,k}[m+4, n+6, 2, 2]$, is first encoded with a symbol **1** and then followed with a symbol stream of +, +, and **0**. Finally, we encode the fourth 2×2 subsegment $\mathbf{S}^{c,2}_{\ell,k}[m+6, n+6, 2, 2]$. It is worth noting, however, that the expected symbol **1** that represents the presence of two exclusive significant coefficients in this subsegment is *not* needed. Recall that the symbol of $\textsc{exclsig}_c(\mathbf{M'}_c(\Xi^{-1}(\mathbf{S}^{c,2}_{\ell,k}[m+6, n+6, 2, 2]))) = \mathbf{1}$ has had encoded the presence of exclusive significant coefficients within the overlaid insignificant region of the previous (larger) segment, $\mathbf{S}^{c,1}_{\ell,k}[m+4, n+4, 4, 4]$. Since the homogeneous subsegment $\mathbf{S}^{c,2}_{\ell,k}[m+6, n+4, 2, 2]$

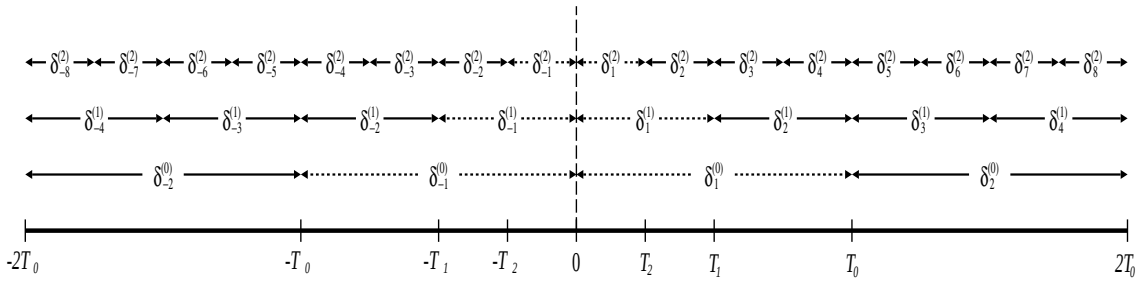


Figure 5.8: Progressive refinement of quantization bin sizes of significant coefficients.

has no exclusive significant coefficients, it is implicitly known that the fourth subsegment $\mathbf{S}^{c,2}_{\ell,k}[m+6, n+6, 2, 2]$ must have at least one exclusive significant coefficient. It is subsequently encoded with four symbols: $+, \mathbf{0}, -$, and $\mathbf{0}$.

As a result, the 8×8 segment $\mathbf{S}^{c,0}_{\ell,k}[m, n, 8, 8]$ is completely encoded using a recursive overlay mapping strategy with the following symbol stream:

$$(\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0}, -, +, +, \mathbf{1}, -, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{0}, -, \mathbf{0}, -, \mathbf{0}, \mathbf{1}, +, +, \mathbf{0}, +, \mathbf{0}, -, \mathbf{0}),$$

which is also essentially a two-symbol stream, except for those symbols produced by the function `EncodeSubsegment()`, which constitutes a three-symbol stream. As it has been pointed out earlier, a contextual switching approach that employs multiple two- and three-alphabet adaptive models is used to encode the symbol stream according to some contexts. Obviously, it is important that the switching from one adaptive model to another must be *implicitly* known by the decoder without any overhead. This can be guaranteed if the contexts at the encoder are derived only from causal information such that the decoder can later reproduce the same contexts while decoding the stream.

5.3.2.3 Progressive Precision Enhancement in Refinement Phase

The main object of performing the refinement phase in each coding layer is to progressively enhance the precision of all previously significant coefficients. Recall that the threshold, T_c, that is associated with each coding layer, c, is halved for the next coding layer. To help us better understand the relationship between threshold values and quantization bin sizes, consider the illustration in Figure 5.8. Let the bold horizontal line denote the value of coefficients within a segment. The markings on the line partition the dynamic range

into a finite number of octave-width significance layers according to the set of thresholds, T_0, $T_1, \ldots$. Note also that the partitions are symmetrical about the value 0, which separates positive coefficients on the right-half from negative coefficients on the left-half. For each coding layer, c, we have a set of mutually exclusive (non-overlapping) quantization bins, $\delta_{\pm k}^{(c)}, k = 1, 2, \ldots, 2^{c+1}$, where each coefficient must belong to only one of these bins at any time.

For simplified notation in the following exposition, we will use only the right-half of the quantization bins by considering the magnitude of coefficients (their signs are encoded separately). After the segmentation phase in the first coding layer, $c = 0$, the threshold T_0 will discriminate all coefficients within a segment as either insignificant (inside bin $\delta_1^{(0)}$) or significant (inside bin $\delta_2^{(0)}$). Without any additional information about the frequency distribution of coefficients within each bin, the assumption of a *uniform distribution* is found to be reasonable. Consequently, it is easy to prove that the optimum reconstruction value (in the least mean square error sense) for each bin will be given by the *middle-point* of the bin. Hence, the reconstruction values of insignificant and significant coefficients after the first segmentation phase are $\frac{1}{2}T_0$ and $\frac{3}{2}T_0$, respectively. As pointed out earlier, the mean reconstruction error now is clearly very high because of the large quantization bin sizes, which correspond to coarse quantization. To enhance the precision of reconstruction values, the bin sizes have to be reduced (in this case, they are halved). As is evident in the next coding layer, $c = 1$, the bin $\delta_2^{(0)}$ is partitioned into two smaller non-overlapping bins: $\delta_3^{(1)}$ (*lower bin*), and $\delta_4^{(1)}$ (*upper bin*). Hence, in general, we can improve the reconstruction values by another bit of precision in the refinement phase via the *refinement operator*, Λ, such that:

$$\Lambda: \quad \{\delta_k^{(c)}\} \quad \longrightarrow \quad \{\delta_{2k-1}^{(c+1)}\} \ \cup \ \{\delta_{2k}^{(c+1)}\},$$

where $\{\delta_{2k-1}^{(c+1)}\} \cap \{\delta_{2k}^{(c+1)}\} = \emptyset$, for $k = 1, 2, \ldots, 2^{c+1}$.

Figure 5.9 shows the pseudocode for performing the refinement phase of a segment in one coding layer. For a particular coding layer $c = C$, only coefficients that have already been significant in previous coding layers (i.e., $c < C$) will be refined. In other words, exclusive significant coefficients at the current coding layer, $c = C$, will only be refined

```
RefinementEncodeOneSegment(S_{ℓ,k}^{(J),c,q}) {
   /* encode sig. coefficients within segment in a raster-scan manner */
   /* exclude exclusive sig. coefficients in the current coding layer */
   for(each coefficient s.t. SIG_{c-1}(s ∈ S_{ℓ,k}^{(J),c,q}) == 1) {
      if(bit = (s > T_c))
         s -= T_c;
      AdaptArithEncSym(bit, amodel);
   }
}
```

Figure 5.9: Pseudocode for performing the refinement phase of a given segment, $\mathbf{S}_{\ell,k}^{(J)}$, at a coding layer, c.

from the next coding layer onwards. This procedure is clearly depicted in Figure 5.8 by employing the refinement operator, Λ, in each coding layer. It also means that there is no refinement phase for the first coding layer, $c = 0$. For each significant coefficient that is to be refined, the index of the smaller bin (i.e., either the lower or upper bin) which the coefficient lies in will be entropy encoded using an adaptive arithmetic model. Multiple adaptive models can also be used for contextual switching, but this often contributes, if any, only marginal improvements because the probabilities of coefficients lying in the lower and upper bins are almost equal. Hence, the modelling (prediction) process is usually not very accurate and no significant coding gain can be expected.

5.3.3 Inter-Frame Scalable Encoding Algorithms

Recall from subsections 3.4.1, 3.4.2 and 3.4.3 about the various problems that could impede the development of a truly multi-scalable video coding platform. We have proposed a solution to each of the problems and explicated how the various coding segments are organized in multiple embedded layers to generate a multi-scalable bit stream that simultaneously supports bit rate, spatial resolution, frame rate, and color video scalabilities. In this subsection, we will further analyze the encoding algorithms for inter-frame coding of a video frame using the wavelet-based multiresolution motion compensation framework, as described in subsection 4.4.2. Appropriate block diagrams are also used to help illuminate

the proposed "prediction frame locking" mechanism and the generation of multiresolution motion vector fields.

Referring to the subband structure in Figure 3.1, the proposed multiresolution motion compensation process will perform motion estimation on each subband to determine the motion vector field. The motion fields corresponding to all the subbands within a given resolution scale (as marked by a cross in Figure 3.1) are then aggregated for entropy-encoding before being output to the bit stream. For ease of exposition, we may logically group the motion estimation process to each resolution scale (i.e., all the subbands corresponding to a particular resolution block), as depicted by the wavelet-based motion estimation and motion compensation (MEMC) block in Figure 5.10. In fact, the MEMC module will first perform motion estimation and then encode the motion predicted residues of the subbands in the corresponding resolution scale (as illustrated in Figure 5.11).

To better understand the inter-frame scalable encoding process, let us refer to Figure 5.10. In order to initiate the multiresolution block matching process, both the discrete multiwavelet transformed frame and the corresponding multiwavelet transformed reference frame (which is determined using the temporal hierarchy structure shown in Figure 3.2) are provided as inputs to the algorithms. The **DEMUX** units essentially split the input and reference subband structure into multiple spatial resolution scales for the MEMC module. The **SW** switching units that precede the inputs to the MEMC modules serve to control spatial resolution scalability at the encoder. In effect, each **SW** unit is either turned on or off depending on whether the corresponding resolution scale is required for encoding. When only a reduced spatial resolution video is required during encoding, the **SW** corresponding to the finer resolution scale is turned off. MEMC is performed for all the pertinent subbands for which **SW** is turned on, and the resulting motion vector fields are then embedded into the compressed bit stream at the position shown in Figure 3.3, thus achieving simultaneous video scalability. These various scalable segments of the bit stream for the current frame are then transmitted via the communication channels, or they are aggregated and stored in a compressed file.

Figure 5.11 provides greater insights into the MEMC module, which accepts the res-

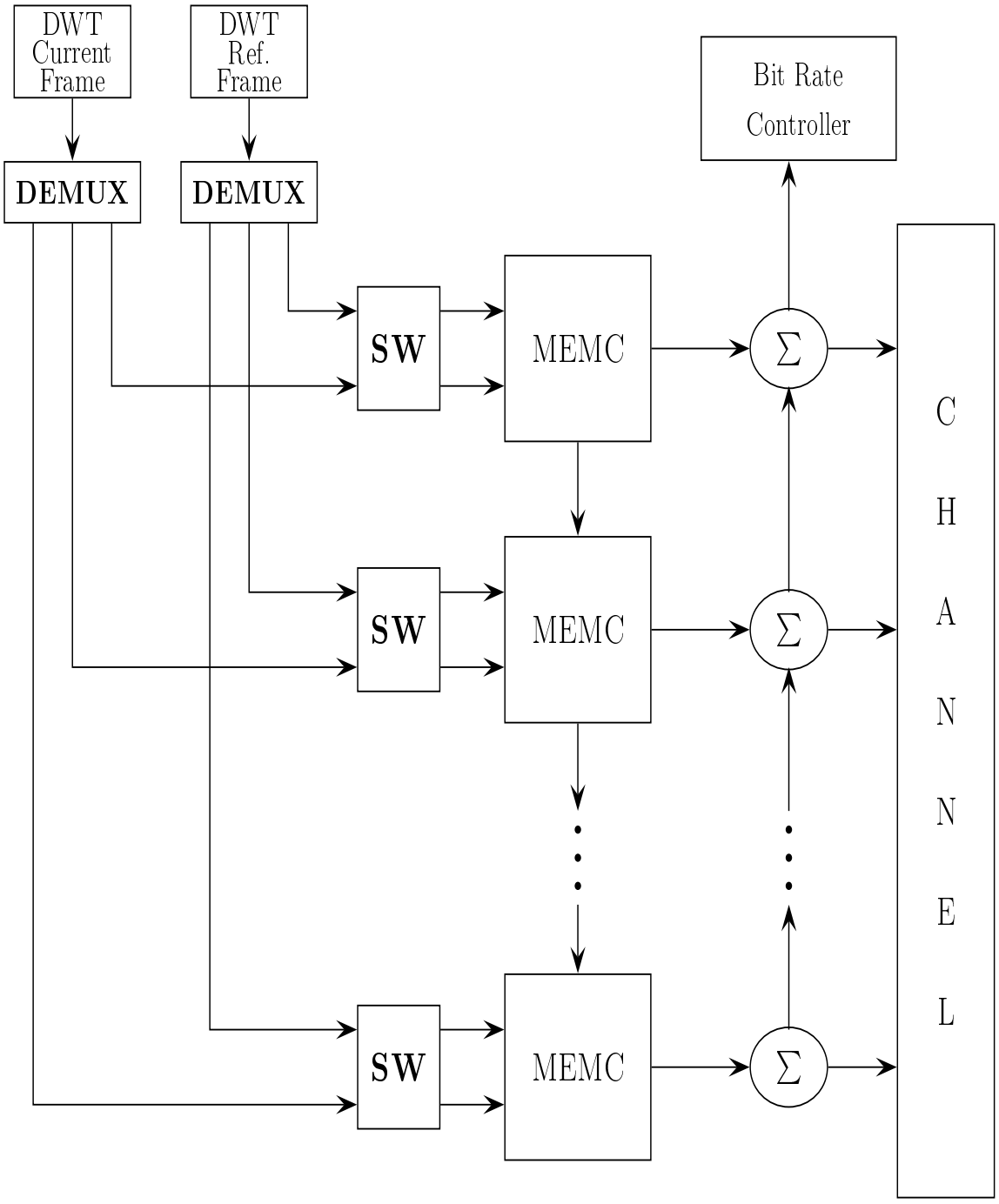


Figure 5.10: Block diagram for encoding an inter-coded video frame using the proposed wavelet-based multiresolution UCBDS algorithm. It also shows how spatial resolution and bit rate scaling can be controlled during the encoding process.

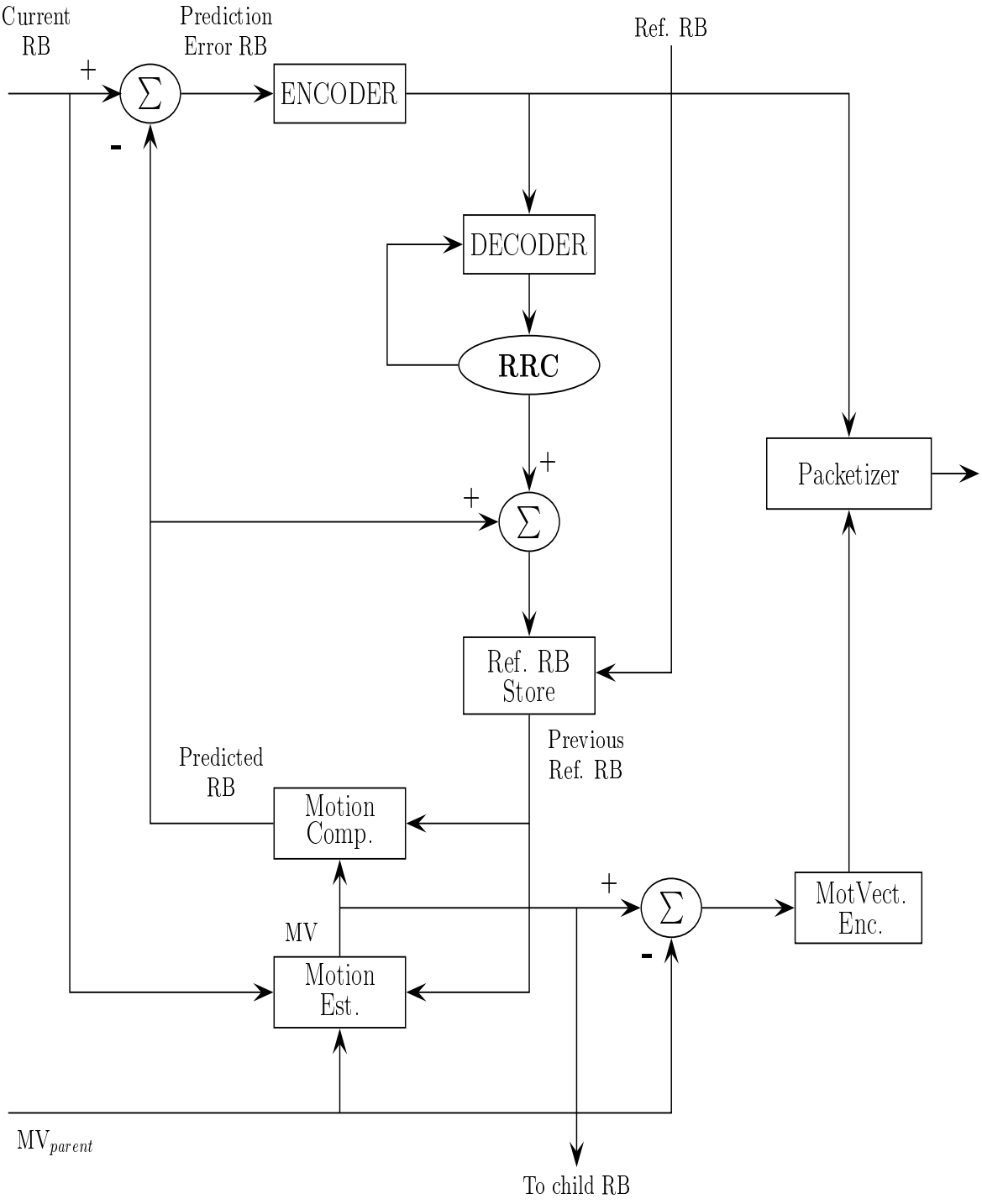


Figure 5.11: Block diagram for encoding a resolution block (or a subband segment) using the proposed wavelet-based multiresolution UCBDS algorithm. It also illustrates how the problems of error propagation and loss of prediction loop can be prevented.

olution blocks (RB's) of the current and reference frames for motion estimation using the proposed wavelet-based multiresolution UCBDS algorithm. The motion vector field of the corresponding parent subband is used to initialize the motion search of the current subband, and the difference between these motion fields is eventually entropy-encoded. Using the estimated motion field, motion compensation is then performed to determine the predicted RB, which is subtracted from the current input RB to produce the prediction residue RB. The prediction residue RB is then encoded using either the proposed recursive direct splitting strategy or the recursive overlay mapping strategy, as expounded in subsection 5.3.2.

As explained, multiple coding layers will be performed to generate the layer blocks that enable bit rate scalability. At a higher level, all layer blocks for the current frame are generated until a certain target frame budget is reached, and these blocks are grouped into a frame block before the next frame is processed. The distinct frame blocks clearly show how the decoder can easily scale for different frame rates by simply discarding later frame blocks. The resulting compressed prediction residue is finally combined with the corresponding encoded motion vector field, which will collectively constitute the resolution block in the first layer block, as illustrated in Figure 3.3. Subsequent processing of the next resolution scale will generate the other resolution blocks that will enable spatial resolution scaling. For coding efficiency, motion compensation of the chrominance channels is carried out using the scaled version of the corresponding motion fields of the luminance channel. Since the chrominance bit stream segments are embedded into distinct bit stream segments, color video scalability can also be easily supported. In particular, it is noted that the reference rate control (**RRC**) unit will "lock" the reference RB that is stored in a buffer for MEMC of a future inter-coded frame according to the temporal hierarchy structure depicted in Figure 3.2. As will be seen later, the same "locking" mechanism is also employed at the decoder to ensure synchronization of the reference RB with the encoder and hence secure the prediction loop critical for supporting bit rate scalability.

5.4 Multi-scalable Video Decoder

This section presents the corresponding scalable video decoder, which inputs appropriate resolution blocks from a compressed bit stream, decodes the selected subsets of the bit stream, and then outputs the reconstructed video frames. Both intra-frame and inter-frame scalable decoding algorithms are explained in the next two subsections with the help of some pseudocode. This is followed by some examples to illustrate the various possible combinations of supported video scaling parameters, and also to illuminate the degree and granularity of supported video scalability.

The decoding algorithms are essentially the reverse operation of the encoding counterpart which was described in subsection 5.3.2. Based on a given scalable video decoding specifications, the decoder will either process or discard (parse[3]) each coding block in order to reconstruct the required frame rate, spatial resolution, bit rate, and color depth of the scalable video. The pseudocode in Figure 5.12 illustrate the scalable video decoding process, given the scalable compressed bit stream and the decoding specifications. Each video frame is processed according to the chosen decoding frame rate, which in turn determines the required temporal layers from the temporal hierarchy structure in Figure 3.2. The conditional switch, `if(current frame is required)`, ensures that only the required frame blocks (according to the decoding specifications) are decoded. Otherwise, the blocks are discarded via `ParseFrameBlock()`. As a result, frame rate scalability can be achieved by selective decoding of the frame blocks in the same compressed bit stream. For each required frame block in the input bit stream, `DetermineCodingMode()` decides if the current frame is to be decoded using intra-frame or inter-frame coding mode, which will be explained in the following two subsections. For each coding mode, the reference frame is generated while adhering to the same reference rate locking value that was used at the encoder. The reference frame is then used for motion compensation of a future inter-frame decoded video frame. The decoded current video frame is finally reconstructed by applying the inverse

[3]In the case of scalable video distribution over layered communication channels (such as via the MBONE multicast networks), the decoder would not have subscribed to the channels that deliver those resolution blocks irrelevant to the given decoding specifications. Hence, the decoder merely processes all the received resolution blocks to recover the scalable video.

multiwavelet transform before it is being stored to a file or rendered for display.

```
DecodeScalableVideo( ) {
   while(current_frame <= target_last_frame) {
      if(current frame is required) {
        DetermineCodingMode( );
        if(CodingMode == INTRA) {
          DecodeIntraFrame( );
          StoreReferenceFrame( );
          ReconstructCurrentFrame( );
        }
        else if(CodingMode == INTER) {
           DetermineReferenceFrame( );
           DecodeInterFrame( );
           MotionCompensateFrame( );
           StoreReferenceFrame( );
           ReconstructCurrentFrame( );
        }
      } else ParseFrameBlock( );
   } /* decode next frame */
}
```

Figure 5.12: Pseudocode for decoding a scalable video bit stream with a given scalable decoding specification. Each frame is processed appropriately, if required, using either an intra-frame or inter-frame decoding mode.

5.4.1 Intra-Frame Scalable Decoding Algorithms

Figure 5.13 presents the pseudocode for decoding an intra-coded frame. In a manner similar to the encoder, the decoding algorithms iterate through one coding layer at a time until a certain target bit rate is reached. This allows bit rate scalability at the decoder, where each additional coding layer further improves the quality of the reconstructed video regardless of the other video scaling parameters. Within each coding layer, the segmentation phase (either via recursive direct splitting strategy or recursive overlap mapping strategy) is first

performed and then followed by the refinement phase. In each phase, all required resolution blocks are decoded from the lowest resolution scale to the highest scale, and the luminance channel is processed before the chrominance channels.

It is also worth noting how both spatial resolution scalability and color scalability are achieved at the decoder by means of the `if(current resolution block is required)` and `if(color video is required)` conditional switches, respectively. All the resolution blocks and color blocks that are irrelevant to the given scalable decoding specifications are discarded by calling `ParseResolutionBlock()` and `ParseAllColorBlocks()` in both the segmentation and refinement phases of each coding layer. For each block that is discarded, the corresponding decoding bit budget for that block is used to reconstruct other required blocks in subsequent coding layers. As a result, it is possible to achieve better video quality at the same decoding bit rate when the spatial resolution and/or color depth is scaled down. Otherwise, the effective decoding bit rate can be reduced in exchange for a lower spatial resolution and/or color depth of the video.

The following two examples will revisit the *recursive direct splitting* and *recursive overlap mapping* techniques but doing so from the decoder viewpoint. Here, an input bit stream corresponding to a particular segment is entropy decoded, and the recover symbols are used to reconstruct the coefficients of the segment during the segmentation phase of the current coding layer.

Example 5.3. *Decoding via Recursive Direct Splitting Technique*

In this example, the goal is to explain how the 4×4 segment in Figure 5.5 can be recovered from the decoded symbol stream

$$(\mathbf{1}, \mathbf{1}, \mathbf{0},\ +,\ \mathbf{1}, \mathbf{0}, \mathbf{0},\ +,\ \mathbf{0}, \mathbf{1},\ +,\ \mathbf{0},\ -,\ \mathbf{0}).$$

It is noted that the three coefficients with the circles have already been significant prior to decoding at the current coding layer. The objective of the current segmentation phase now is to recover the positions and signs of all exclusive coefficients at the current coding layer. To begin, the decoder receives the first symbol $\mathbf{1}$, which indicates that the 4×4 segment comprises at least one exclusive significant coefficient. Hence, the segment $\mathbf{S}_{\ell,k}^{c,0}[m, n, 4, 4]$ is

```
DecodeOneColorFrame( ) {
   c = 0;  /* initialize current decoding layer */
   while(current_decoded_bits < target_frame_bits) {
      for(phase ∈ {Segmentation, Refinement}) {
         if(current resolution block is required) {
           DepacketizeAndDecodeHeader( );
           DecodePhase(L^{(Y),c,0}_{L,1}, L^{(Y),c,0}_{L,2}, L^{(Y),c,0}_{L,3}, L^{(Y),c,0}_{L,4}, phase);
           DecodePhase(V^{(Y),c,0}_{L,K}, H^{(Y),c,0}_{L,K}, D^{(Y),c,0}_{L,K}, phase);
         } else ParseResolutionBlock( );
         for(each remaining resolution scale, ℓ = L − 1, ... , 1) {
            if(current resolution block is required) {
              DepacketizeAndDecodeHeader( );
              DecodePhase(V^{(Y),c,0}_{ℓ,K}, H^{(Y),c,0}_{ℓ,K}, D^{(Y),c,0}_{ℓ,K}, phase);
            } else ParseResolutionBlock( );  }
         if(color video is required) {
           if(current resolution block is required) {
             DepacketizeAndDecodeHeader( );
             DecodePhase(L^{(U),c,0}_{L−1,1}, L^{(V),c,0}_{L−1,1}, phase);
             DecodePhase(L^{(U),c,0}_{L−1,2}, L^{(V),c,0}_{L−1,2}, L^{(U),c,0}_{L−1,3}, L^{(V),c,0}_{L−1,3}, L^{(U),c,0}_{L−1,4}, L^{(V),c,0}_{L−1,4}, phase);
           } else ParseResolutionBlock( );
           for(each remaining resolution scale, ℓ = L − 2, ... , 1) {
              if(current resolution is required) {
                DepacketizeAndDecodeHeader( );
                DecodePhase(V^{(U),c,0}_{ℓ,K}, V^{(V),c,0}_{ℓ,K}, H^{(U),c,0}_{ℓ,K}, H^{(V),c,0}_{ℓ,K}, D^{(U),c,0}_{ℓ,K}, D^{(V),c,0}_{ℓ,K}, phase);
              } else ParseResolutionBlock( );  }
         } else ParseAllColorBlocks( );
      }  /* next decoding phase */  c++;
}  }  /* next decoding layer */
```

Figure 5.13: Pseudocode for decoding a color frame from a highly scalable compressed bit stream consisting of resolution blocks with appropriate headers. Each block is either processed or discarded based on a given scalable video decoding specification.

split via Ξ into four 2×2 subsegments: $\mathbf{S}_{\ell,k}^{c,1}[m,n,2,2]$, $\mathbf{S}_{\ell,k}^{c,1}[m+2,n,2,2]$, $\mathbf{S}_{\ell,k}^{c,1}[m,n+2,2,2]$, and $\mathbf{S}_{\ell,k}^{c,1}[m+2,n+2,2,2]$. The subsegment $\Xi_{(1)}$ is first processed in a depth-first manner. The decoder receives the second symbol **1**, which indicates the presence of at least one exclusive significant coefficient in subsegment $\Xi_{(1)}$. Since the dimensions of subsegment $\Xi_{(1)}$ have reached `min_dim` $= 2$, the signs of all exclusive significant coefficients will be decoded. The decoder knows, at this point, that there are only two previously insignificant coefficients in subsegment $\Xi_{(1)}$ (as the third and fourth coefficients are already significant in previous coding layers). This guides the decoder to read in only the next two symbols, **0** and +, which will assign the first coefficient as insignificant and the second coefficient as positive significant. In other words, the reconstruction value of the first coefficient is still zero, while the second coefficient has a reconstruction value of $+\frac{1}{2}(T_c + T_{c+1})$.

The decoder continues to recover the second subsegment $\mathbf{S}_{\ell,k}^{c,1}[m+2,n,2,2]$, which fourth coefficient is already significant in a previous coding layer. The symbol **1** also indicates the presence of at least one exclusive significant coefficient. Since the dimensions of the subsegment have reached `min_dim` $= 2$, it is not further split. Instead, the decoder reads in the next three symbols, **0**, **0**, and +, to assign the first and second coefficients of the subsegment as insignificant, while the third coefficient as positive significant with a reconstruction value of $+\frac{1}{2}(T_c + T_{c+1})$. Now, the third subsegment $\mathbf{S}_{\ell,k}^{c,1}[m,n+2,2,2]$ is processed. The next symbol **0** indicates exclusive insignificance of the entire subsegment; hence no further processing is required. The fourth subsegment $\mathbf{S}_{\ell,k}^{c,1}[m+2,n+2,2,2]$ has a decoded symbol **1**, which prompts the decoder to read in the next four symbols for all the four coefficients that are still insignificant prior to the current coding layer. The symbols, +, **0**, −, and **0**, assign the the reconstruction values of $+\frac{1}{2}(T_c + T_{c+1})$ and $-\frac{1}{2}(T_c + T_{c+1})$ to the first and third coefficients, respectively, while the second and fourth coefficients will still be zero.

Example 5.4. *Decoding via Recursive Overlay Mapping Technique*

In this example, the goal is to explain how the 8×8 child segment in Figure 5.7 can be

recovered from the decoded symbol stream

$$(\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0}, -, +, +, \mathbf{1}, -, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{0}, -, \mathbf{0}, -, \mathbf{0}, \mathbf{1}, +, +, \mathbf{0}, +, \mathbf{0}, -, \mathbf{0})$$

based on the contextual information from its parent segment in Figure 5.5. At the beginning of decoding the 8×8 segment in the current coding layer, the decoder already has the knowledge about the positions and significance of the six coefficients which were recovered in previous coding layers. It also knows the overlaid significance maps since the corresponding parent segment for the current coding layer has already been processed.

To begin, the decoder will first read in two symbols for both the significant and insignificant overlay maps. The symbols, **1** and **1**, imply that there is at least one exclusive significant coefficient in each of these maps. Hence, both overlay maps will need to be processed further by splitting the 8×8 segment $\mathbf{S}_{\ell,k}^{c,0}[m, n, 8, 8]$ into four subsegments using the operator Ξ. The first subsegment, $\mathbf{S}_{\ell,k}^{c,1}[m, n, 4, 4]$, contains both significant and insignificant overlay maps. This prompts the decoder to read in the next two symbols, **0** and **0**, which indicate that the entire subsegment, $\mathbf{S}_{\ell,k}^{c,1}[m, n, 4, 4]$, is homogeneously exclusive insignificant. A homogeneous segment needs not to be processed further in the current coding layer.

The decoder moves on to decode the second subsegment, $\mathbf{S}_{\ell,k}^{c,1}[m+4, n, 4, 4]$. Since it also has both significant and insignificant overlay maps, two symbols are read in. The first symbol **1** indicates the presence of at least one exclusive significant coefficient in the significant overlay mapped region. The second symbol **0** implies that the insignificant overlay mapped region is homogeneously exclusive insignificant and this region requires no further processing in the current coding layer. The subsegment is further split into four smaller subsegments since it is heterogeneous and contains at least one exclusive significant coefficient. The first and second subsegments, $\mathbf{S}_{\ell,k}^{c,2}[m+4, n, 2, 2]$ and $\mathbf{S}_{\ell,k}^{c,2}[m+6, n, 2, 2]$, are ignored but the third and fourth subsegments, $\mathbf{S}_{\ell,k}^{c,2}[m+4, n+2, 2, 2]$ and $\mathbf{S}_{\ell,k}^{c,2}[m+6, n+2, 2, 2]$, will require further processing. Adopting the same depth-first decoding, all coefficients in the third subsegment are processed before those in the fourth subsegment. Since the third subsegment comprises only the significant overlay map, the decoder will read in one new

symbol. The symbol **1** here prompts the decoder to read in four more symbols, **0**, −, +, and +, because the subsegment has reached `min_dim` = 2. This means that the first coefficient of the subsegment is still insignificant, while the other three coefficients have a reconstruction value of $\frac{1}{2}(T_c + T_{c+1})$ but with a −, +, and + sign, respectively. The fourth subsegment, $\mathbf{S}_{\ell,k}^{c,2}[m+6, n+2, 2, 2]$, also has only the significant overlay map, which guides the decoder to read in only one next symbol **1**. As the first coefficient was already significant, the decoder will read in only three other symbols, −, **0**, and **0**.

The third subsegment, $\mathbf{S}_{\ell,k}^{c,1}[m, n+4, 4, 4]$, is processed next. As it consists of only the insignificant overlay map, the decoder reads in only one new symbol **0**. This implies a homogeneously exclusive insignificant subsegment that requires no further processing in the current coding layer.

Finally, the decoder processes the fourth subsegment, $\mathbf{S}_{\ell,k}^{c,1}[m+4, n+4, 4, 4]$ by reading in two symbols, **1** and **1**. Hence both the significant and insignificant overlay mapped regions will require further processing. The subsegment is now split into four smaller subsegments. The first subsegment, $\mathbf{S}_{\ell,k}^{c,2}[m+4, n+4, 2, 2]$, has a symbol **1**, which prompts the decoder to read in four new symbols, **0**, −, **0**, and −. The second subsegment, $\mathbf{S}_{\ell,k}^{c,2}[m+6, n+4, 2, 2]$ has a symbol **0**, which implies that it is homogeneously exclusive insignificant (hence, no more symbols are read in for this subsegment). The third subsegment, $\mathbf{S}_{\ell,k}^{c,2}[m+4, n+6, 2, 2]$, also has a symbol **1**, which guides the decoder to read in three other symbols, +, +, and **0** as the fourth coefficient has already been significant in a previous coding layer. Finally the decoder implicitly knows that the fourth subsegment, $\mathbf{S}_{\ell,k}^{c,2}[m+6, n+6, 2, 2]$, must have at least one exclusive significant coefficient because the second subsegment is homogeneously exclusive insignificant. The decoder will then read in four new symbols, +, **0**, −, and **0** for the four previously insignificant coefficients, respectively.

5.4.2 Inter-Frame Scalable Decoding Algorithms

Referring to the INTER coding mode portion of Figure 5.12, the first step in inter-frame video decoding is to determine the reference frame for motion compensation of the current frame of interest. The subroutine `DetermineReferenceFrame()` employs the tempo-

ral hierarchy structure in Figure 3.2 to determine and retrieve the reference frame from the buffer; this reference frame is then used to perform `DecodeInterFrame()`. The inter-frame decoding algorithms are almost identical to the pseudocode in Figure 5.13 except that the motion vector field corresponding to each required resolution scale is decoded as well during the first coding layer. The current predicted frame is then recovered via `MotionCompensateFrame()` by motion compensating the previous reference frame with the decoded motion vector fields in the multiwavelet domain. A predetermined portion of the current prediction error frame that was reconstructed via `DecodeInterFrame()` is then added to the current predicted frame. The application of the same "reference frame locking" mechanism generates a new reference frame, which is stored in the buffer via the subroutine `StoreReferenceFrame()`. Depending on the target bit budget for the current frame, `DecodeInterFrame()` will continue to reconstruct more coding layers of the current prediction error frame so that `ReconstructCurrentFrame()` can recover the current video frame with higher fidelity.

The block diagram in Figure 5.14 illustrates the processing units and data flows while performing scalable video decoding of one video frame using an inter-frame coding mode. As the scalable bit stream is separated into distinct resolution blocks (RB), the switching units **SW** can selectively process or discard certain RB's depending on the chosen scalable decoding specifications. When **SW** is turned on, the corresponding RB will be motion compensated via the MEMC module, given the reference RB that is retrieved from the buffer. All the required RB's are decoded and multiplexed (via the **MUX** operator) in order to reconstruct the current video frame.

The detailed processing of each MEMC module is depicted in Figure 5.15. The bit stream of a particular resolution block is first depacketized to retrieve the motion vector field and the compressed prediction frame error of the current frame. For each RB that has a parent, the corresponding motion vector field of the parent is also used to motion compensate the current reference RB in order to reproduce the current predicted RB. In a separate processing thread, the decoder continues to reconstruct the prediction frame error until a given bit budget for the frame is reached. The current RB is recovered by

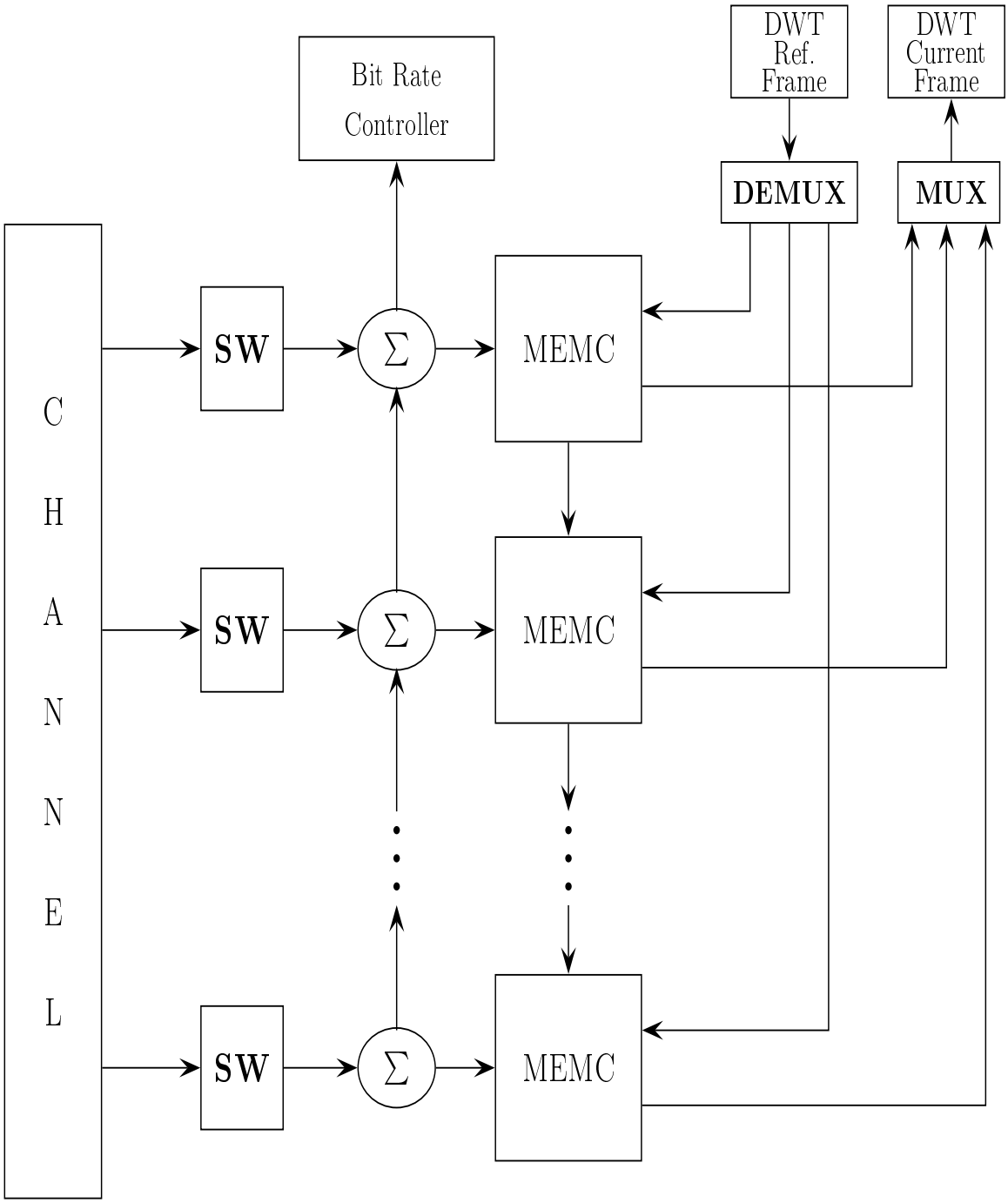


Figure 5.14: Block diagram for decoding an inter-coded video frame using the proposed wavelet-based multiresolution UCBDS algorithm. It also shows how spatial resolution and bit rate scaling can be controlled during the decoding process.

combining the current predicted RB with the decoded prediction frame error. It is again emphasized that the reference rate control (**RRC**) unit will "lock" a predetermined portion of the prediction frame error, which is added to the current predicted RB to generate a new reference RB that is stored in the buffer for MEMC of a future inter-coded RB.

5.4.3 Examples of Scalable Decoding Specifications

This subsection aims to demonstrate the extent and granularity of the supported multi-scalable compressed video bit stream. In particular, we show how different video scaling parameters can be chosen simultaneously from the same scalable bit stream. Consider an input video sequence:

- Frame rate: 30 fps
- Color depth: 24 bits per pixel
- Spatial resolution: 640 $\times$ 480 pixels

which is encoded using the proposed multi-scalable video coding framework with the following encoding specifications:

- Temporal scalability: 4 temporal layers supported
- Color scalability: Enabled
- Encoded bit budget: 1 Mbps
- Base reference frame "lock": 10% of encoded bit budget
- Number of multiwavelet decomposition: 4 octave levels
- Spatial resolution scalability: 3 resolution scales supported.

The total number of supported temporal layers is solely dependent on the choice of the proposed temporal hierarchy structure in Figure 3.2 (with 4 temporal layers). A different choice of the temporal hierarchy structure will determine the maximum allowable frame rate scalability available to the decoder. Color scalability is enabled by encoding

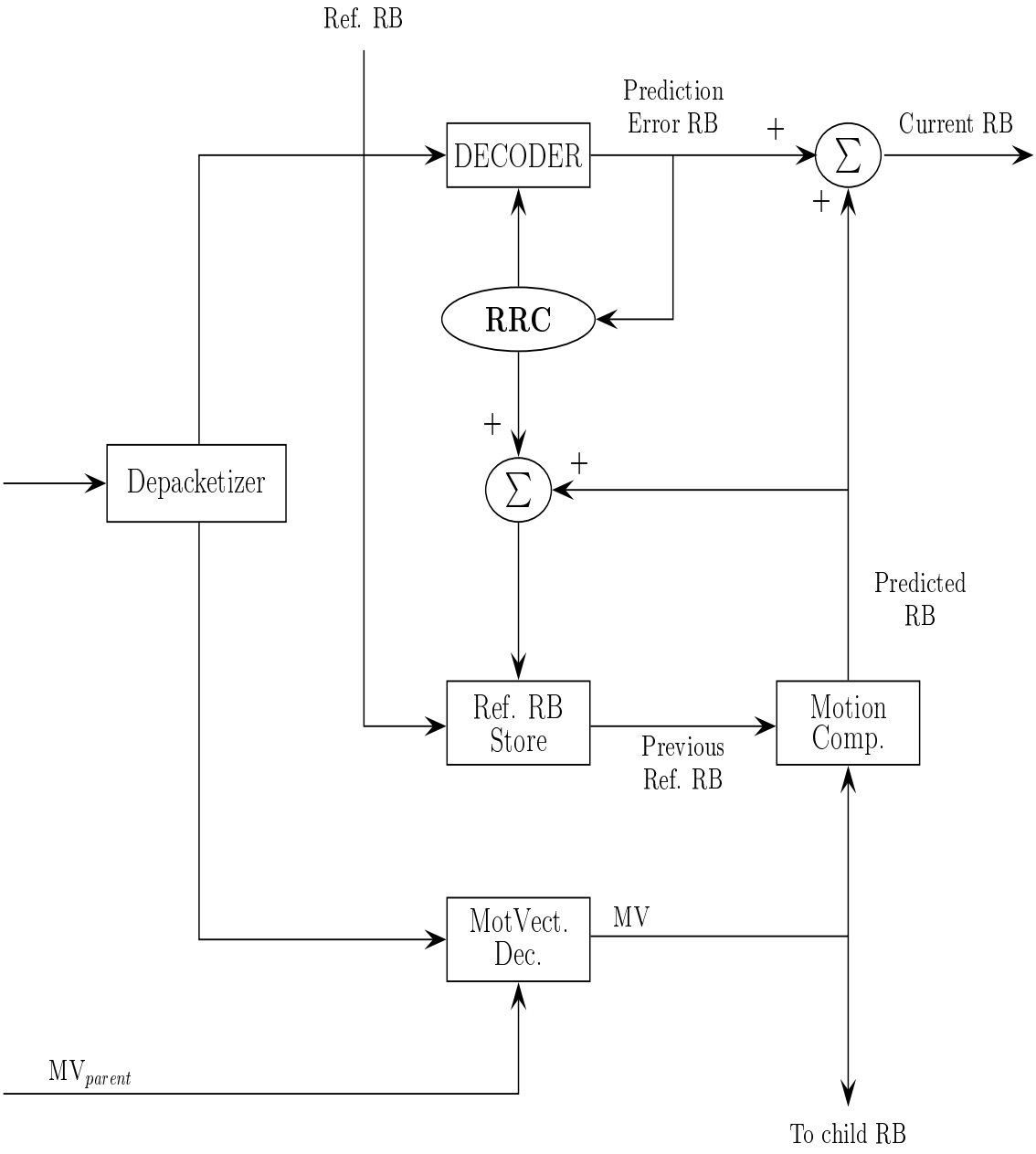


Figure 5.15: Block diagram for decoding a resolution block (or a subband segment) using the proposed wavelet-based multiresolution UCBDS algorithm. It also illustrates how the proposed reference frame "locking" mechanism is employed to secure the prediction loop.

Specs.	Bit Rate (kbps)	Frame Rate (fps)	Color (Y/N)	Spatial size (pixels)
(a)	650	30	Y	640 × 480
(b)	300	30	Y	320 × 240
(c)	300	15	N	320 × 240
(d)	120	15	Y	160 × 120
(e)	80	7.5	Y	320 × 240
(f)	150	7.5	Y	320 × 240
(g)	34	3.75	N	160 × 120

Table 5.3: Examples of possible video decoding specifications that can be achieved from a common mulit-scalable compressed video bit stream.

the chrominance components of the input video into distinct color blocks with appropriate headers. The decoder can subsequently discard all chrominance blocks in order to reconstruct a grayscale video. The target encoded bit rate is set to 1 Mbps; in fact, the target bit budget can be any arbitrary value since it is independent of the other scalable encoding specifications. In order to enforce the "reference frame locking" mechanism, a fixed predetermined target is set (in this case, it is at 10% or 100 kbps). The same "locking" mechanism is then employed at the decoder to secure the motion prediction loop. Also, we have chosen to perform four levels of octave multiwavelet transform using the proposed initialization framework. As a consequence, a maximum of three possible spatial resolution scalability can be supported. In this example, we have chosen to provide all the three spatial resolution scaling for the decoder.

Using the above scalable encoding specifications, Table 5.3 shows some examples of a wide range of possible video decoding specifications that can be achieved from the *same* compressed bit stream. As explained in subsection 3.4.1, the "locking" mechanism imposes a minimum decoding bit rate since the base reference layer must be satisfied to guarantee synchronization of the reference frames between the encoder and decoder. In this example, the lowest possible decoded bit rate is at 100 kbps if full frame rate, color, and spatial

resolution are required. However, when one or more of these video scaling parameters are scaled down, lower decoding bit rates are possible such as those with decoding specifications (e) and (g). In addition to the above scalability in terms of bit rate, frame rate, color, and spatial resolution, decoding complexity was also clearly achieved with a lower decoding specification. This was evident from the fact that more frames could be decoded per second when the bit rate, color and/or spatial resolution was reduced.

It is also worth noting that any of the above decoded bit streams is a multi-scalable bit stream itself; however, the maximum degree of scalability for the various video scaling parameters has been reduced accordingly. For example, the decoded video with specification (b) can be further re-scaled to obtain another version of the same video, such as that with a decoding specification (d), (e), (f) or (g). However, the decoding specification (c) cannot be achieved from (b) since there is simply insufficient information to attain the same maximum bit rate (i.e. 300 kbps) by scaling down the frame rate and color.

5.5 Simulation Results and Promising Video Applications

The section presents some simulation results of the proposed multi-scalable video compression framework. Although the current emphasis of the dissertation has not been on optimizing the encoding and decoding algorithms in terms of processing speed and compression efficiency, the simulations here will demonstrate the various supported combination and granularity of video scalability and present an appreciation of the possible trade-offs among the disparate video scaling parameters in order to meet specific decoding requirements or limitations. This is followed by a preview of some interesting applications of scalable video such as multi-party video communication in heterogenous environments. In particular, we will illuminate with some snapshots of a layered live multicasting project that exploits the multi-scalable features of the proposed video compression platform.

5.5.1 Simulation Results of Multi-Scalable Video Codec

In this investigation, the standard *Carphone* video sequence is used to illustrate the flexibility and granularity of the proposed multi-scalable bit stream. This video sequence consists of a foreground object (a head-and-shoulder person who moves his upper body while speaking) and a constantly moving background (the exterior street view from a moving car). The following encoding specifications were used:

- Frame rate: 10 fps
- Color depth: 24 bits per pixel
- Spatial resolution: Full QCIF (176 $\times$ 144 pixels)
- Temporal scalability: 4 temporal layers supported
- Color scalability: Enabled
- Encoded bit budget: 1 Mbps
- Base reference frame "lock": 5% of encoded bit budget
- Number of multiwavelet decomposition: 3 octave levels
- Spatial resolution scalability: 2 resolution scales supported.

Figure 5.16 displays six selected original frames (namely, the $16^{th}, 80^{th}, 176^{th}, 240^{th}, 288^{th}$, and 368^{th} frames) from the Carphone sequence.

The same multi-scalable compressed video bit stream generated was then used to reconstruct the following five different versions of the video:

(a) 30 kbps, 5 fps, full QCIF, and grayscale video;

(b) 30 kbps, 5 fps, quarter QCIF, and color video;

(c) 15 kbps, 1.25 fps, quarter QCIF, and grayscale video;

(d) 100 kbps, 5 fps, full QCIF, and grayscale video;

(e) 200 kbps, 10 fps, full QCIF, and color video.

Figures 5.17, 5.18, 5.19, 5.20, and 5.21, respectively, portray the same set of six video frames in Figure 5.16 but they were reconstructed using each of the above five decoding specifications. Clearly, many other combinations of the video scaling parameters can be achieved with a target bit rate of up to 1 Mbps.

From Figure 5.17, we observed some ringing effects when decoding at 30 kbps with only a grayscale video. Using the same target bit rate, we could alternatively scale down the spatial resolution but support color video at the same frame rate (as shown in Figure 5.18). In order to illustrate the flexibility of the multi-scalable bit stream in supporting very low bit rate applications simultaneously, Figure 5.19 depicts the reconstructed sub-QCIF video frames at only 15 kbps but with a reduced frame rate of 1.25 fps. This decoding specification can become useful for video playback on a mobile PDA over wireless connection. On the other hand, we could also decode at a higher target bit rate of 100 kbps or 200 kbps at the expense of higher computation complexity. As expected, the video quality in Figure 5.20 is clearly more superior than that in Figure 5.17 when the decoded bit rate was increased with all the other video scaling parameters unchanged. Figure 5.21 further increased the bit rate to recover a high-quality video with full color, full spatial resolution, and full frame rate.

In addition to subjective observations, an objective measure of the PSNR values of the reconstructed video frames are also provided in Figures 5.17 – 5.21. Table 5.4 presents a summary of the average PSNR values of the Carphone video sequence over 380 frames. For the sub-QCIF decoded videos, their corresponding PSNR values are measured against a downsampled version (using bilinear interpolation) of the original QCIF video. It is observed that the PSNR values correspond very well with the subjective video quality. For example, there is an average improvement of 1.29 dB by increasing the bit rate from 30 kbps to 100 kbps while leaving the other decoding specifications unchanged. At 200 kbps, the average PSNR value of the luminance component has further increased considerably; this clearly supports the observed improvements in the decoded video quality.

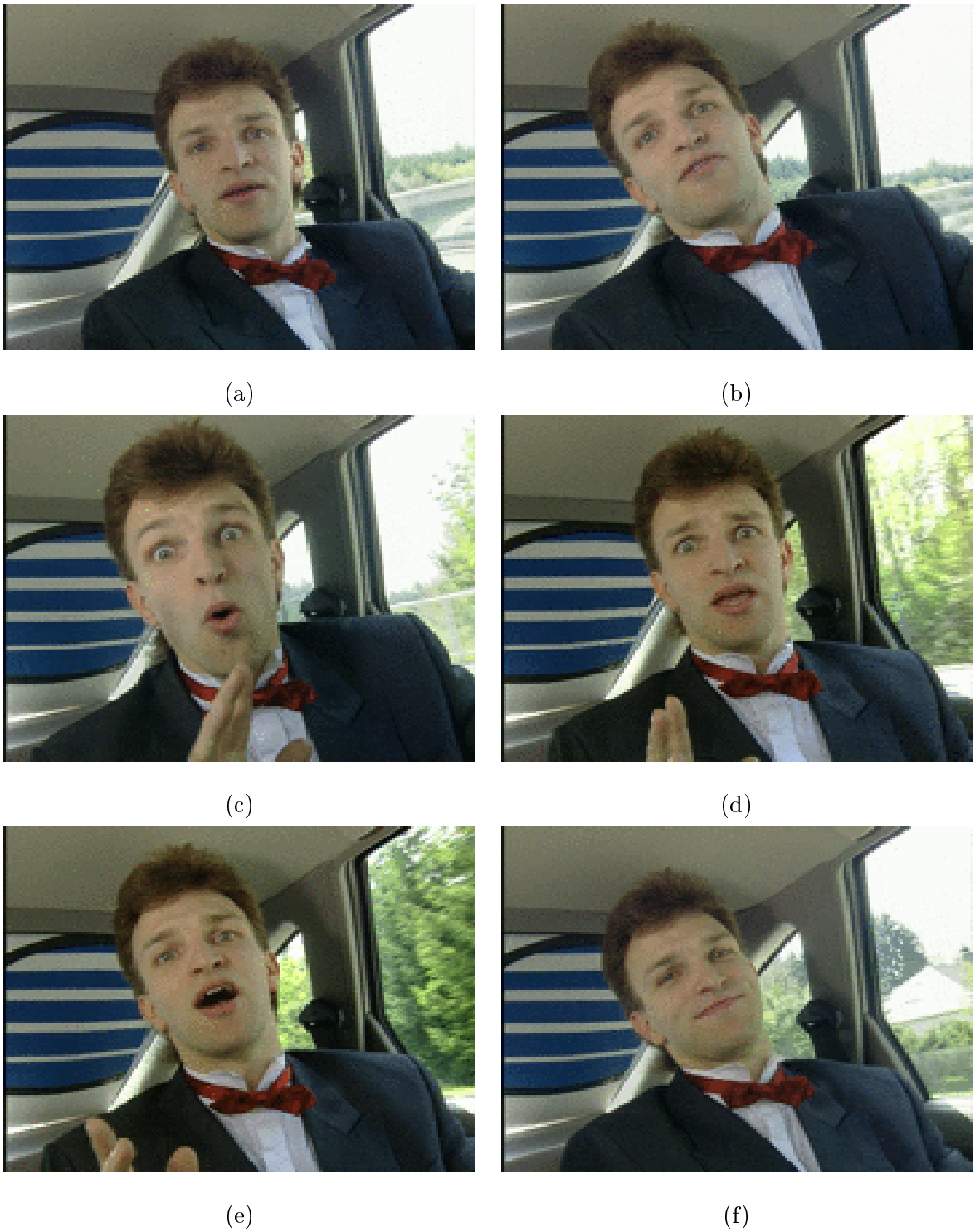
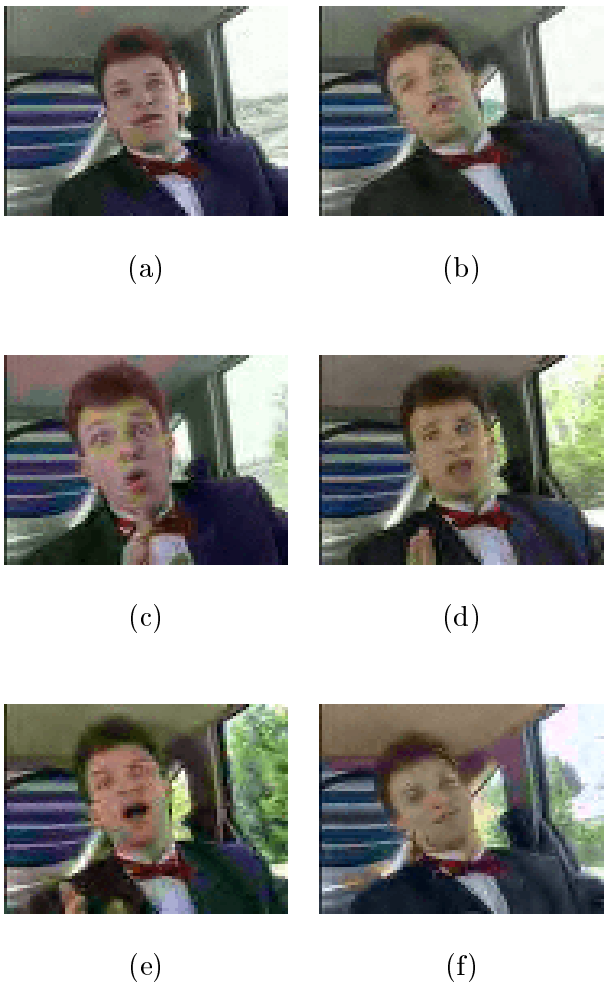


Figure 5.16: Original Carphone video sequence encoded with an encoding specification of 1 Mbps, 10 fps, full QCIF, color, and 5% for base reference byte: (a) Frame 16, (b) Frame 80, (c) Frame 176, (d) Frame 240, (e) Frame 288, and (f) Frame 368.

Figure 5.17: Decoded frames from Carphone video sequence with a decoding specification of 30 kbps, 5 fps, full QCIF, and grayscale: (a) Frame 16 (Y: 23.84 dB), (b) Frame 80 (Y: 24.17 dB), (c) Frame 176 (Y: 23.32 dB), (d) Frame 240 (Y: 22.25 dB), (e) Frame 288 (Y: 20.72), and (f) Frame 368 (Y: 20.03 dB).



(a) (b)

(c) (d)

(e) (f)

Figure 5.18: Decoded frames from Carphone video sequence with a decoding specification of 30 kbps, 5 fps, quarter QCIF, and color: (a) Frame 16 (Y: 27.48 dB; U: 30.15 dB; V: 28.28 dB), (b) Frame 80 (Y: 26.12 dB; U: 27.85 dB; V: 27.05 dB), (c) Frame 176 (Y: 24.11 dB; U: 25.91 dB; V: 24.33 dB), (d) Frame 240 (Y: 22.79 dB; U: 23.47 dB; V: 23.26 dB), (e) Frame 288 (Y: 21.16 dB; U: 21.95 dB; V: 22.35 dB), and (f) Frame 368 (Y: 21.47 dB; U: 22.93 dB; V: 22.48 dB).

Figure 5.19: Decoded frames from Carphone video sequence with a decoding specification of 15 kbps, 1.25 fps, quarter QCIF, and grayscale: (a) Frame 16 (Y: 23.38 dB), (b) Frame 80 (Y: 20.21 dB), (c) Frame 176 (Y: 20.56 dB), (d) Frame 240 (Y: 21.23 dB), (e) Frame 288 (Y: 18.17 dB), and (f) Frame 368 (Y: 21.61 dB).

Figure 5.20: Decoded frames from Carphone video sequence with a decoding specification of 100 kbps, 5 fps, full QCIF, and grayscale: (a) Frame 16 (Y: 24.61 dB), (b) Frame 80 (Y: 25.08 dB), (c) Frame 176 (Y: 24.10 dB), (d) Frame 240 (Y: 22.88 dB), (e) Frame 288 (Y: 21.32 dB), and (f) Frame 368 (Y: 22.05 dB).

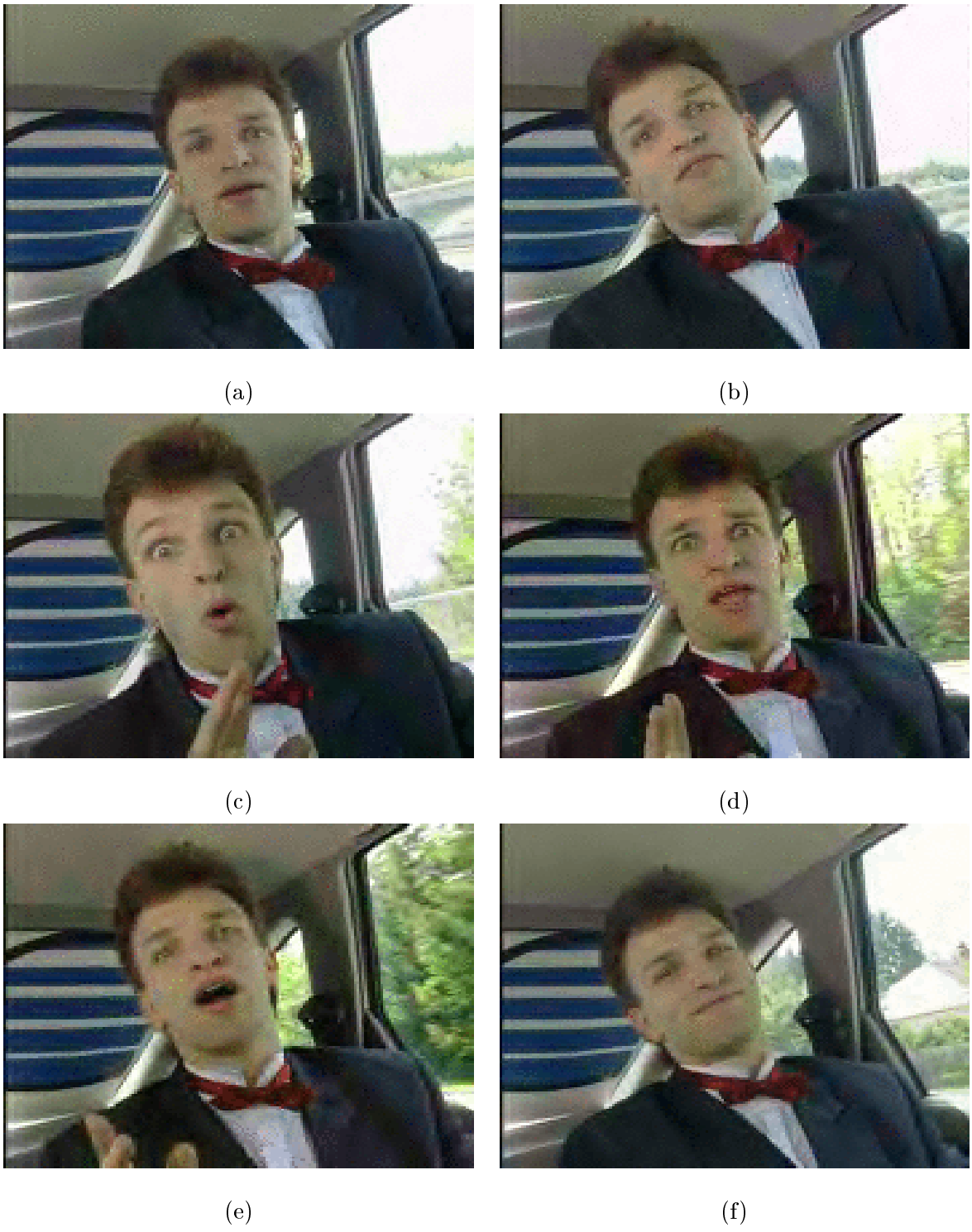


Figure 5.21: Decoded frames from Carphone video sequence with a decoding specification of 200 kbps, 10 fps, full QCIF, and color: (a) Frame 16 (Y: 33.56 dB; U: 36.48 dB; V: 34.38 dB) , (b) Frame 80 (Y: 32.52 dB; U: 36.62 dB; V: 33.65 dB), (c) Frame 176 (Y: 31.97 dB; U: 35.61 dB; V: 32.92 dB), (d) Frame 240 (Y: 29.48 dB; U: 33.30 dB; V: 30.21 dB), (e) Frame 288 (Y: 30.01 dB; U: 32.54 dB; V: 30.55 dB), and (f) Frame 368 (Y: 32.84 dB; U: 35.75 dB; V: 33.55 dB).

Bit Rate (kbps)	Frame Rate (fps)	Color (Y/N)	Spatial size (pixels)	Average PSNR (dB)	
30	5	N	176 × 144	Y U V	23.77 – –
30	5	Y	88 × 72	Y U V	22.30 25.41 23.92
15	1.25	N	88 × 72	Y U V	20.91 – –
100	5	N	176 × 144	Y U V	25.06 – –
200	10	Y	176 × 144	Y U V	31.78 35.03 32.55

Table 5.4: Average PSNR results (Carphone sequence over 380 frames) for different video decoding specifications.

5.5.2 Some Promising Applications of Multi-scalable Video Codec

Many useful video applications that cater to heterogeneous multiparty and diverse streaming network environments can be realized using highly scalable and efficient video compression. As expounded in Section 3.2, video scalability can be achieved in three different scenarios: at the encoder end, during video delivery over the networks, or at the decoder end. In this subsection, we present a brief introduction to two real-world video communication systems that have been developed using an earlier version of the proposed multi-scalable video compression architecture:

(a) **Scalable video multicasting to heterogeneous receivers over diverse networks:**[4] This project aims to exploit the highly scalable video compression framework

[4]This project, codenamed *SeeWAVE*, is a joint-project between the Department of Electrical Engineering and the Center for Wavelets, Approximation, and Information Processing (CWAIP) under the Academic

for scalable video delivery to disparate receivers over the MBONE multicast network. In doing so, the scalable encoder captures a live feed and compresses it into different resolution layers of the video bit stream; each layer is then assigned to a separate multicast channel and streamed via RTP over UDP. A receiver at the other end will select and subscribe to only those multicast channels that are required according to the preferred scalable decoding specifications. A session manager maintains a list of all available video servers and the list of multicast channels carrying the live or pre-encoded video streams. For clients or networks that are not supporting multicasting, a proxy gateway is employed to receive the multicast streams, buffer and re-package them before relaying the required version of the video in a unicast manner. Both high-performance desktop computers as well as wireless-connected PDAs have been used as clients in this project. More information about this project can be found at http://ccn.ece.nus.edu.sg/seewave/ (and also in [31]).

(b) **Client-driven video on-demand (or retrieval) system:** This project aims to develop a flexible video-on-demand system that archives the video assets in a scalable compressed video format. An indexing methodology is used to reference the various resolution layers of the scalable bit stream so that each layer can be quickly accessed for streaming from the video server. When a client requests for a particular video, information about the degree of scalability supported by that video is first retrieved from the server. Based on this information, the client can select a preferred set of scalable decoding specifications and then make the appropriate request. The video server will retrieve only the pertinent layers of that video and stream them to the requesting client in a unicast manner via TCP/IP or UDP/IP. In this manner, efficient bandwidth utilization can be achieved; moreover, the video server needs to store only one scalable version of each video in the archive in order to cater to disparate clients.

The pictures below illustrates some snapshots of the SeeWAVE project. Figure 5.22 shows a high-performance video server with a camera that captures the live source video.

Research Fund, NUS. The author is grateful to Dr. Tham Chen Khong and team for the permission to include some snapshots of the project into the dissertation.

The session information and the preview video capture window are displayed on the monitor, while the handheld unicast client is receiving the live video via a wireless LAN card. Figure 5.23 depicts two unicast PDA clients which receive the video stream using a wireless mobile phone modem and infra-red connection. Figure 5.24 provides a closer look into the scalable video client interfaces for a handheld PDA using the Windows CE platform. It shows the interfaces for the session manager and the scalable video selection menu. Depending on the selected decoding specifications, the clients can receive different unicast video streams with disparate spatial, frame rate, bit rate, and color resolutions.

5.6 Conclusion

The chapter introduced a new framework for multi-scalable video compression using the discrete multiwavelet transform and multiresolution motion estimation and compensation in the wavelet domain. Detailed algorithms development for both the scalable encoder and decoder was expounded with appropriate pseudocode and examples. In particular, we have also introduced two new algorithms for efficient encoding of the multiwavelet coefficients: (i) Recursive direct splitting strategy, which exploits the spatial relationship of the coefficients within a local neighborhood; and (ii) Recursive overlay mapping strategy, which further exploits the inter-subband relationship of the parent and children coefficients. In the case of intra-frame coding, the subband coefficients are encoded using either one of the above recursive coding strategies. For inter-frame coding, however, the two strategies are used to encode the prediction frame (subband) error coefficients that was obtained after performing wavelet-based motion compensation.

We also explained how multiresolution motion estimation and compensation were performed on each resolution scale so that the resulting bit stream supports spatial resolution scalability without compromising the prediction loop. The encoding and decoding algorithms clearly explained how different resolution blocks can be generated, organized in the bit stream, and then decoded (or discarded) accordingly in order to achieve a given set of scalable decoding specifications. Examples on various combinations of supported granu-

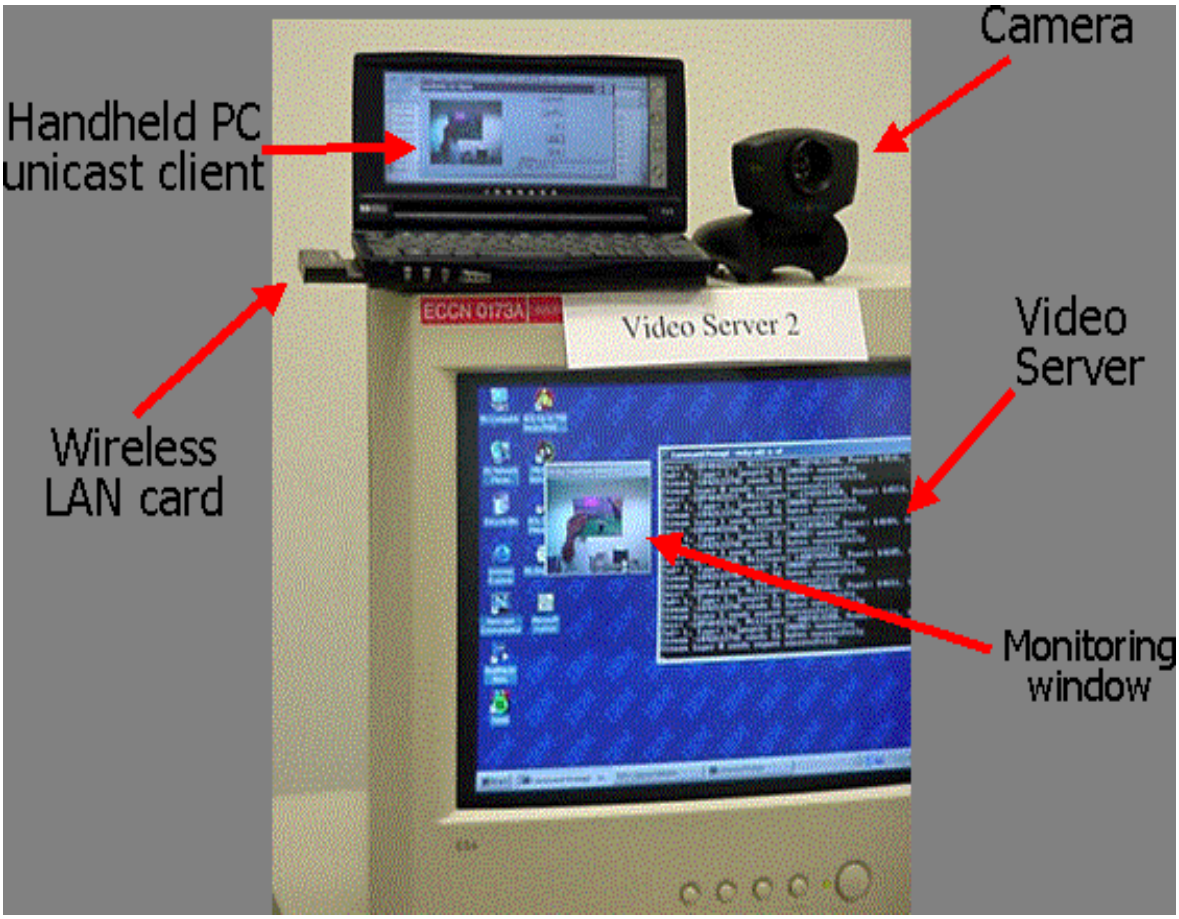
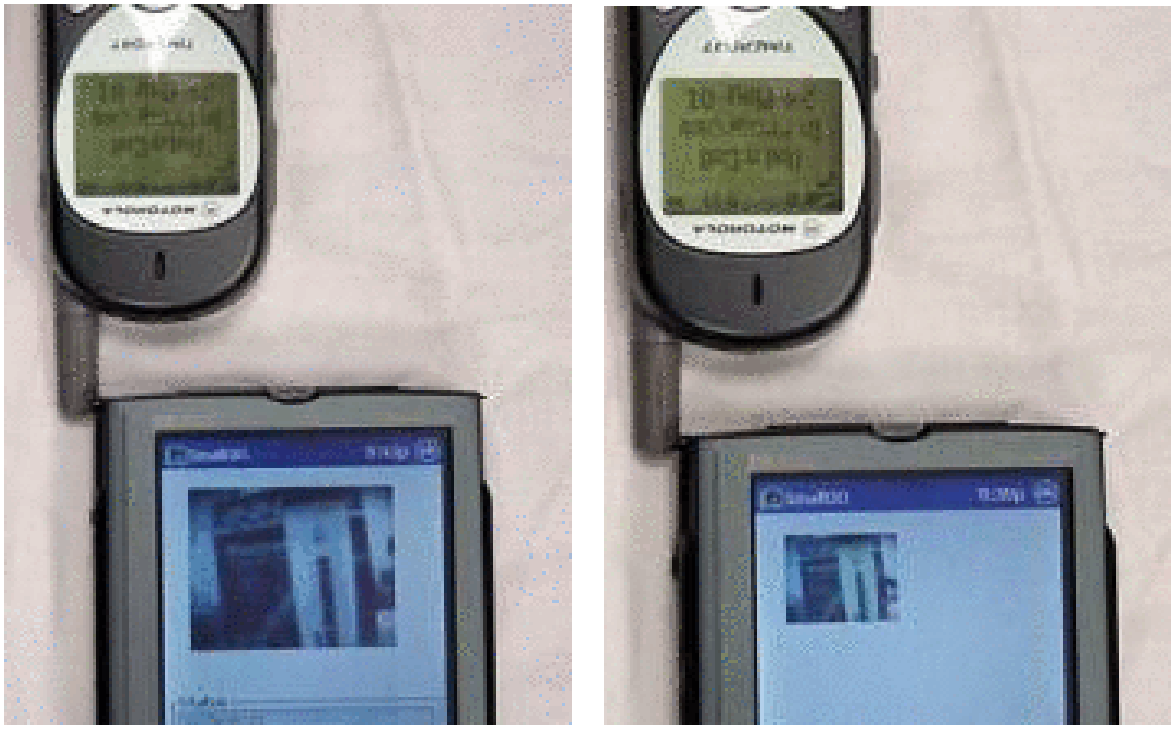


Figure 5.22: Project SeeWAVE: Video server with a camera capturing the live source feed, and a wireless unicast client receiving the live stream.



(a) (b)

Figure 5.23: Project SeeWAVE: Two wireless-connected PDA clients with (a) receiving a full-QCIF grayscale live video, and (b) receiving a quarter-QCIF grayscale live video.

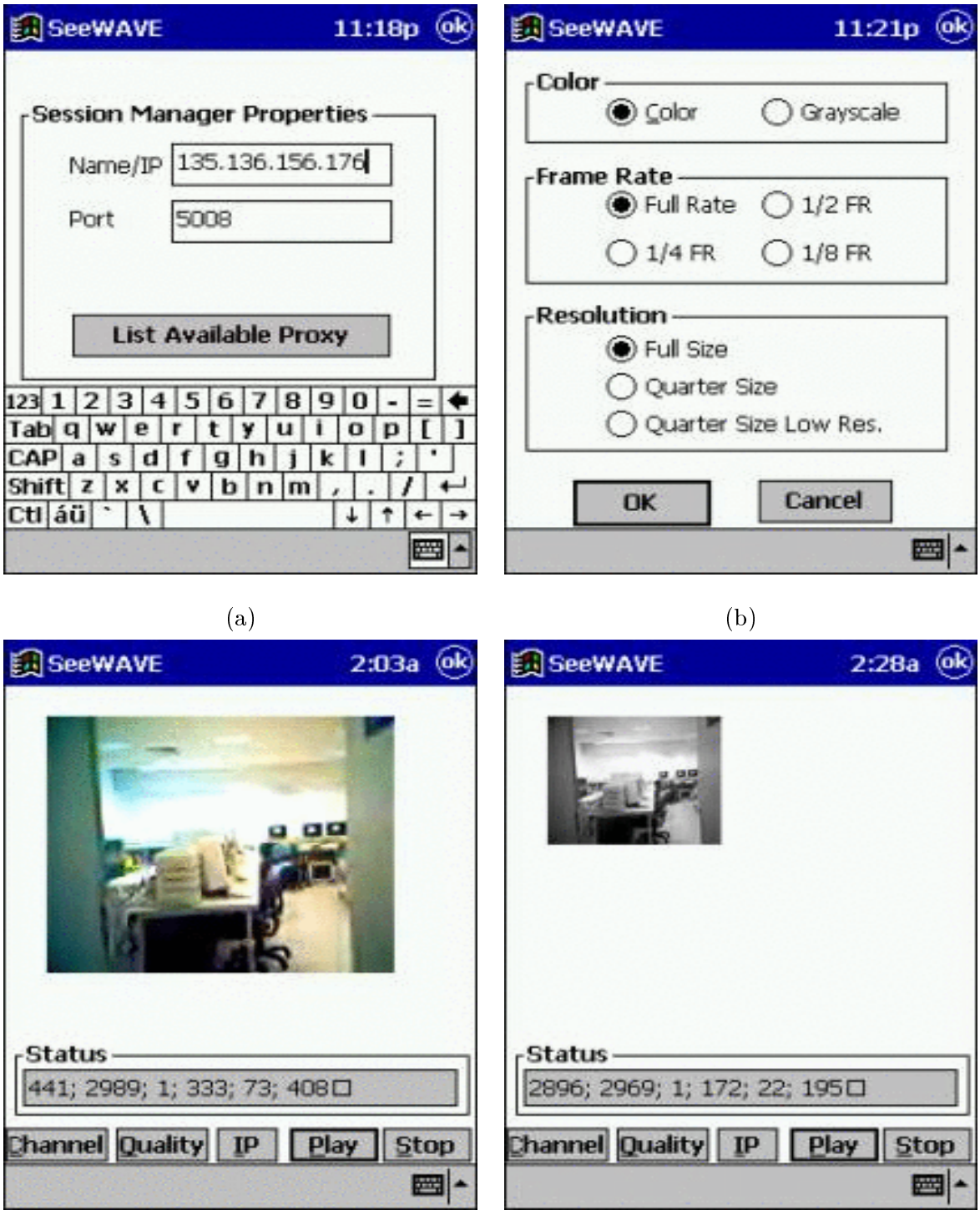


Figure 5.24: Project SeeWAVE: Interfaces of PDA clients with Windows CE operating system. (a) Interface for specifying the session properties of live source feed; (b) Interface for selecting the scalable decoding specifications; (c) Client receiving a full-QCIF color live video; and (d) Client receiving a quarter-QCIF grayscale live video.

larity of video scaling parameters were presented, together with a visual representation of the reconstructed video frames. Finally, we demonstrated two actual video communication systems that were developed using the proposed multi-scalable video compression platform. These projects validated the flexibility and usefulness of scalable video compression to simultaneously cater for client diversity and network heterogeneity using only one scalable video bit stream.

Chapter 6

Conclusions and Future Research Directions

"There is nothing more dreadful than the habit of doubt. Doubt separates people. It is a poison that disintegrates friendships and breaks up pleasant relations. It is a thorn that irritates and hurts; it is a sword that kills."

Siddhartha Gautama Buddha (563 - 483 B.C.)

6.1 Conclusions

In this dissertation, we have focused on the research and application development of the following three core areas:

- **Multiwavelets:** We first established the concept of an *equivalent scalar filter bank system* for a given multiwavelet system. This allowed us to gain new insights into the input-output relationship of discrete-time multiwavelet transform from the perspective of the input-output relationship a set of equivalent scaler wavelet system. This subsequently led to the notion of "good multifilter property (GMP)" which essentially characterizes the property of the corresponding frequency responses of the equivalent scalar wavelets. Using GMP as a new tool, we showed step-by-step procedures to construct multiwavelet filters with desirable properties useful for image and

video compression applications. We further introduced two previously unpublished classes of orthogonal and biorthogonal multiwavelets having one symmetric and one antisymmetric wavelet basis; these multiwavelets also possess a GMP order of at least one. We then proposed a generalized framework for multiwavelet initialization (or pre-filtering), which is key for performing effective discrete multiwavelet decomposition and reconstruction algorithms. The pre-filtering framework does not only require very low processing overhead but it is also robust enough to work well with any multiwavelets. Experimental results in image and video compression provided conclusive evidence that the proposed classes of multiwavelets with GMPs could consistently outperform other multiwavelets and popular scalar wavelets with lower computation complexity.

- **Motion Estimation and Compensation (MEMC):** In this area, we introduced a novel block matching algorithm called "unrestricted center-biased diamond search" (UCBDS), which capitalizes on the observation that most motion vector distributions are centrally biased toward the zero motion vector. Extensive theoretical and experimental simulations have demonstrated the robustness, accuracy, and efficiency of UCBDS. A speed performance improvement of up to 31% could be achieved over some other fast block matching methods such as the four-step search, and over 13 times faster than the full-search method. The monogrid UCBDS was then extended to a multiresolution UCBDS framework with the goal of supporting spatial resolution video scalability. Here, MEMC was performed on the multiwavelet subbands instead of in the image domain. This approach enjoys many benefits such as lower computation complexity, no blocking artifacts, and multiscale representation of the motion vector field. We also discussed the inherent problem of non-translational invariance of critically-sampled wavelet transform, and showed how the proposed framework could secure the prediction loop and prevent a prediction drift, which would otherwise render multi-scalable video compression impossible.

- **Multi-scalable video compression:** Eight primary video scaling parameters (bit rate, distortion, spatial resolution, temporal resolution, alphabet, hardware, complexity, and object scalability) were discussed and related to scenarios on how such video scalability features can be useful to simultaneously support heterogeneous client and network requirements. In particular, we illumminated the potential barriers to supporting bit rate, frame rate, and spatial resolution video scalability. A "reference frame locking" mechanism, a secured prediction loop for multiresolution motion compensation, and a temporal hierarchy structure were proposed, and later showed how these solutions can be combined to provide *simultaneous* video scalability. Two new coding algorithms (namely, the recursive direct splitting, and the recursive overlay mapping strategies) for scalable intra- and inter-frame encoding and decoding were explicated with complete pseudocode, block diagrams, and examples. We also presented an insight into how the multi-scalable compressed video bit stream is comprised of an embedded hierarchy of frame blocks, layer blocks, resolution blocks, and color blocks, which subsequently allows the scalable decoder to select from a wide range of decoding specifications with fine granularity. Finally, we discussed two real-world video communication applications: Scalable multicast video delivery over the Internet, and a scalable client-driven video-on-demand system.

6.2 Suggested Future Research Directions

The proposed multi-scalable video compression architecture opens up a new paradigm for many other advanced areas that warrant further research and application development. The following list highlights a few suggestions for future extension to the current work:

- **Optimum bit allocation:** This is a challenging problem on how a given target bit budget can be optimally allocated for intra and inter-coded frames based on the semantics and motion content of a video sequence. Also, it remains an open issue as to how a scalable decoder should distribute a given decoding bit budget when performing spatial resolution and/or frame rate scaling. Optimum bit allocation in

both constant bit rate (CBR) and variable bit rate (VBR) modes for a highly scalable video compression platform remains an elusive problem. This area can contribute directly to improving the compression efficiency by a sizeable margin.

- **Video post-processing:** Unlike blocking artifacts that are found in typical block-based compression methods, wavelet compression suffers from ringing or mosquito artifacts at high compression ratios. Research on how to suppress or minimize ringing artifacts can contribute to significant improvement in perceptual video quality. We have presented a simple and non-iterative approach for ringing suppression in [8]; however, more advanced techniques can be researched, either as a post-processing step or an integrated mechanism during the decoding process.

- **Video object scalability:** Although the current multi-scalable video compression framework supports various video scaling parameters, object-based scalability is still not possible unless the objects within a video frame are segmented and encoded separately. Research on multiwavelet transform and scalable encoding of arbitrarily shaped objects and a framework that describes the composition of the objects can be promising areas to support interactive video.

- **Shift-invariant wavelet filters and wavelet-based MEMC:** The area is warranted due to the shift variance property of critically-sampled discrete (multi)wavelet transform, which will negatively impact the performance of motion compensation in the wavelet domain. More research can be focused on the construction of new multiwavelet filters with improved shift invariance properties. Research for more efficient MEMC algorithms in the wavelet domain can also be promising.

- **M-band multiwavelets** The two classes of orthogonal and biorthogonal multiwavelets introduced in this dissertation consist of only two channels: lowpass and highpass filtering. Extension to more than two bands may contribute to new results in signal representation and energy compaction since there will be a lowpass and multiple bandpass filtering. Also, an M-band multiwavelet could possess a higher degree of desirable filter properties for a given length of a compactly supported multifilter.

Appendix A

Proofs

A.1 Proof of Theorem 2.1

Sufficient part: Suppose that the α_k's satisfy (2.43), then we have

$$\boldsymbol{H}^{(2N)}(z) \;=\; \boldsymbol{R}(\alpha_0)\boldsymbol{D}(z)\boldsymbol{R}(\alpha_1)\prod_{k=1}^{N-1}\boldsymbol{D}(z^2)\boldsymbol{R}(\alpha_{2k+1}). \tag{A.1}$$

Denote $\boldsymbol{B}(z) = \begin{bmatrix} b_{11}(z) & b_{12}(z) \\ b_{21}(z) & b_{22}(z) \end{bmatrix} := \prod_{k=1}^{N-1}\boldsymbol{D}(z)\boldsymbol{R}(\alpha_{2k+1})$. Expanding the right-hand side of (A.1), we have

$$\begin{aligned}
\sqrt{2}H_{00}^{(2N)}(z) &= (\cos\alpha_1 - z^{-1}\sin\alpha_1)b_{11}(z^2) + (-\sin\alpha_1 - z^{-1}\cos\alpha_1)b_{21}(z^2), \\
\sqrt{2}H_{01}^{(2N)}(z) &= (\cos\alpha_1 + z^{-1}\sin\alpha_1)b_{11}(z^2) + (-\sin\alpha_1 + z^{-1}\cos\alpha_1)b_{21}(z^2).
\end{aligned}$$

Thus, we have $H_{00}^{(2N)}(z) = H_{01}^{(2N)}(-z)$. This implies that the filter $\{h_k\}_{k=0}^{4N-1}$ satisfies (2.40). Obviously, $\boldsymbol{H}^{(2N)}(z)$ is unitary on the circle $|z| = 1$. On the other hand, since the determinant of $\boldsymbol{B}(z)$ equals $\Delta(z) := \det(\boldsymbol{B}(z)) = z^{-N+1}$, and $(\boldsymbol{B}^T(z^2))^{-1} = \prod_{k=1}^{N-1}\boldsymbol{D}(z^{-2})\boldsymbol{R}(\alpha_{2k+1}) = \boldsymbol{B}(z^{-2})$, we have

$$b_{22}(z^{-2}) = \frac{b_{11}(z^2)}{\Delta(z^2)} = z^{2(N-1)}b_{11}(z^2) \quad \text{and} \quad b_{12}(z^{-2}) = -\frac{b_{21}(z^2)}{\Delta(z^2)} = -z^{2(N-1)}b_{21}(z^2).$$

Thus, $H_1^{(2N)}(z) = -z^{-4N+1}H_0^{(2N)}(-z^{-1})$. Therefore, $\boldsymbol{H}^{(2N)}(z)$ is the polyphase matrix of a length-$4N$ orthonormal filter bank which satisfies (2.40).

Necessary part: If an orthonormal filter $\{h_k\}_{k=0}^{4N-1}$ satisfies (2.40), then from (2.37), its polyphase matrix $\boldsymbol{H}^{(2N)}(z)$ can be written as

$$\boldsymbol{H}^{(2N)}(z) = \begin{bmatrix} H_{00}^{(2N)}(z) & H_{10}^{(2N)}(z) \\ H_{00}^{(2N)}(-z) & H_{10}^{(2N)}(-z) \end{bmatrix}.$$

In [41], Evangelista proved that such a matrix can be factorized as

$$\boldsymbol{H}^{(2N)}(z) = \boldsymbol{R}(\alpha_0)\boldsymbol{D}(z)\boldsymbol{R}(\alpha_1)\prod_{k=1}^{N-1}\boldsymbol{D}(z^2)\boldsymbol{R}(\alpha_{2k+1}),$$

where $\alpha_0 = \pi/4$. Using the fact $\boldsymbol{D}(z^2) = \boldsymbol{D}(z)\boldsymbol{I}\boldsymbol{D}(z)$, we have

$$\boldsymbol{H}^{(2N)}(z) = \boldsymbol{R}(\alpha_0)\boldsymbol{D}(z)\boldsymbol{R}(\alpha_1)\prod_{k=1}^{N-1}\boldsymbol{D}(z)\boldsymbol{I}\boldsymbol{D}(z)\boldsymbol{R}(\alpha_{2k+1}).$$

By comparing it with (2.38), we obtain $\boldsymbol{R}(\alpha_{2k}) = \boldsymbol{I}$. Thus we have $\alpha_{2k} = 0 \bmod 2\pi$, $k = 1, \ldots, N-1$. □

A.2 Proof of Proposition 2.3

We only need to prove that $\{\boldsymbol{H}_k\}_{k=0}^{2N-1}$ and $\{\boldsymbol{G}_k\}_{k=0}^{2N-1}$ satisfy the PR conditions (2.10) and (2.11). For $i \in \mathbb{Z}$,

$$\begin{aligned}
\sum_{k=0}^{2N-1-2i}\boldsymbol{H}_k\boldsymbol{G}_{k+2i}^T &= \sum_{k=0}^{2N-1-2i}\boldsymbol{H}_k((-1)^{k+2i+1}\boldsymbol{H}_{2N-1-k-2i}\boldsymbol{A})^T \\
&= \left(\sum_{k=0}^{N-1-i} + \sum_{k=N-i}^{2N-1-2i}\right)\boldsymbol{H}_k(-1)^{k+2i+1}\boldsymbol{A}\boldsymbol{H}_{2N-1-k-2i}^T \\
&= \sum_{k=0}^{N-1-i}\boldsymbol{H}_k\ (-1)^{k+1}\boldsymbol{A}\boldsymbol{H}_{2N-1-k-2i}^T + \sum_{k=0}^{N-1-i}\boldsymbol{H}_{2N-1-2i-k}(-1)^k\boldsymbol{A}\boldsymbol{H}_k^T.
\end{aligned}$$

Using the assumption that $\boldsymbol{H}_k\boldsymbol{A}\boldsymbol{H}_{2N-1-2i-k}^T$ are symmetric matrices for all $k = 0, 1, \ldots, N-i-1$, $i = 0, 1, \ldots, N-1$, we have

$$\sum_{k=0}^{2N-1-2i}\boldsymbol{H}_k\boldsymbol{G}_{k+2i}^T = \boldsymbol{0}_{2\times 2}, \quad i \in \mathbb{Z}. \tag{A.2}$$

It is easy to show that $\{\boldsymbol{G}_k\}_{k=0}^{2N-1}$ given by (2.48) satisfies the PR condition (2.10). □

Appendix B

Examples of Multifilters

For concise presentation, all the filter coefficients shown in the following examples should be rescaled by a division with $\sqrt{2}$.

B.1 Parameterization of Length-4 SAOMF Family.

We will construct a length-4 orthonormal multiwavelet system starting from the Daubechies' one-parameter length-4 orthonormal scalar wavelet with the lowpass sequence $\{h_k\}_{k=0}^3$ given by

$$h_0 = \frac{\nu(\nu-1)}{\nu^2+1},\ h_1 = \frac{1-\nu}{\nu^2+1},\ h_2 = \frac{1+\nu}{\nu^2+1},\ h_3 = \frac{\nu(\nu+1)}{\nu^2+1}.$$

This sequence is a scalar CQF, and the parameter ν can be used for various filter design purposes.

Applying Lemma 2.1 to $\{h_k\}_{k=0}^3$ results in the following length-8 scalar CQF $\{b_k\}_{k=0}^7$:

$$\begin{aligned}
b_0 &= \frac{(\nu-1)(\nu+\tau)}{2(1+\nu^2)}, & b_1 &= \frac{-\tau(\nu-1)(\nu+\tau)}{2(1+\nu^2)}, & b_2 &= \frac{(1+\nu)(1+\nu\tau)}{2(1+\nu^2)},\\
b_3 &= \frac{\tau(1+\nu)(1+\nu\tau)}{2(1+\nu^2)}, & b_4 &= \frac{(1+\nu)(1-\nu\tau)}{2(1+\nu^2)}, & b_5 &= \frac{-\tau(1+\nu)(1-\nu\tau)}{2(1+\nu^2)},\\
b_6 &= \frac{(\nu-1)(\nu-\tau)}{2(1+\nu^2)}, & b_7 &= \frac{\tau(\nu-1)(\nu-\tau)}{2(1+\nu^2)}.
\end{aligned}$$

We then apply the procedure in (2.47) to obtain the parameterized length-4 matrix lowpass

sequence $\{\boldsymbol{H}_k\}_{k=-1}^{2}$ as

$$\begin{aligned}
\boldsymbol{H}_{-1} &= \tfrac{1}{2(1+\nu^2)} \begin{bmatrix} (\nu-1)^2 & \tau(1-\nu^2) \\ \tau(\nu-1)^2 & \nu^2-1 \end{bmatrix}, \\
\boldsymbol{H}_0 &= \tfrac{1}{2(1+\nu^2)} \begin{bmatrix} (\nu+1)^2 & \tau(1-\nu^2) \\ -\tau(\nu+1)^2 & 1-\nu^2 \end{bmatrix}, \\
\boldsymbol{H}_1 &= \boldsymbol{S}\boldsymbol{P}_1\boldsymbol{S} \quad \text{and} \quad \boldsymbol{H}_2 = \boldsymbol{S}\boldsymbol{P}_0\boldsymbol{S}.
\end{aligned}$$

The associated length-4 matrix highpass sequence $\{\boldsymbol{G}_k\}_{k=-1}^{2}$ can be obtained directly using Proposition 2.3.

B.2 Parameterization of Length-6 SAOMF Family.

Using the polyphase matrix factorization in (2.38) with the constraints on the angle parameters, $\alpha_j, j = 0, 1, \ldots, N-1$, as shown in (2.39), we can construct a parameterized SAOMF of length-6 when $N = 3$. Taking $\alpha_0 = 2n\pi + \frac{\pi}{4} - \alpha_1 - \alpha_2$, $\beta_1 = \tan\alpha_1$, $\beta_2 = \tan\alpha_2$, and $\gamma = (1+\beta_1^2)(1+\beta_2^2)$, we obtain the following length-6 scalar CQF:

$$\begin{aligned}
h_0 &= (\beta_1+\beta_2-\beta_1\beta_2+1)/\gamma, & h_1 &= -(\beta_1+\beta_2+\beta_1\beta_2-1)/\gamma, \\
h_2 &= \beta_1(\beta_1-1)/(1+\beta_1^2), & h_3 &= \beta_1(\beta_1+1)/(1+\beta_1^2), \\
h_4 &= \beta_2(\beta_1+\beta_2+\beta_1\beta_2-1)/\gamma, & h_5 &= \beta_2(\beta_1+\beta_2-\beta_1\beta_2+1)/\gamma.
\end{aligned}$$

Applying Lemma 2.1, we obtain the length-12 scalar CQF with even-indexed coefficients:

$$\begin{aligned}
b_0 &= \big((\beta_1+\beta_2)(\tau+1) + (\beta_1\beta_2-1)(\tau-1)\big)/(2\gamma), \\
b_2 &= \beta_2\big((\beta_1+\beta_2)(\tau+1) - (\beta_1\beta_2-1)(\tau-1)\big)/(2\gamma), \\
b_4 &= \beta_1\big(\beta_1-1-\tau(\beta_1+1)\big)/(2(1+\beta_1^2)), \\
b_6 &= \beta_1\big(\beta_1-1+\tau(\beta_1+1)\big)/(2(1+\beta_1^2)), \\
b_8 &= \beta_2\big((\beta_1+\beta_2)(-\tau+1) + (\beta_1\beta_2-1)(\tau+1)\big)/(2\gamma), \\
b_{10} &= \big((\beta_1+\beta_2)(-\tau+1) - (\beta_1\beta_2-1)(\tau+1)\big)/(2\gamma),
\end{aligned}$$

and odd-indexed coefficients $b_{2k+1} = \tau(-1)^{k+1}b_{2k}$, $k = 0, 1, \ldots, 5$. By means of (2.47) with $\tau = -1$, the parameterized matrix lowpass filter is given by

$$\boldsymbol{H}_{-2} = \frac{1}{(\beta_1^2+1)(\beta_2^2+1)} \begin{bmatrix} -\beta_1\beta_2 + 1 & -\tau(\beta_1 + \beta_2) \\ -\tau(\beta_1\beta_2 - 1) & \beta_1 + \beta_2 \end{bmatrix},$$

$$\boldsymbol{H}_{-1} = \frac{\beta_2}{(\beta_1^2+1)(\beta_2^2+1)} \begin{bmatrix} \beta_1 + \beta_2 & -\tau(-\beta_1\beta_2 + 1) \\ -\tau(\beta_1 + \beta_2) & \beta_1\beta_2 - 1 \end{bmatrix}, \quad \boldsymbol{H}_0 = \frac{\beta_1}{\beta_1^2+1} \begin{bmatrix} \beta_1 & \tau \\ \beta_1\tau & -1 \end{bmatrix},$$

and $\boldsymbol{H}_k = \boldsymbol{S}\boldsymbol{H}_{1-k}\boldsymbol{S}$, $k = 1, 2, 3$. The matrix highpass sequence $\{\boldsymbol{G}_k\}_{k=-2}^{3}$ can be constructed via Proposition 2.3. The **SA6** family of SAOMFs have two free parameters that allow the incorporation of different desirable multifilter design properties.

B.3 Parameterization of BSA(4/4) SABMF Family.

A family of length 4/4 SABMFs with a GMP order of at least (3,1) can be constructed with the following matrix lowpass sequences:

$$\boldsymbol{H}_{-1} = \begin{bmatrix} \frac{2\gamma}{8\gamma-3} & \frac{1}{8} \\ 2\alpha\gamma & \frac{\alpha(8\gamma-1)}{8} \end{bmatrix}, \quad \boldsymbol{H}_0 = \begin{bmatrix} \frac{3(2\gamma-1)}{8\gamma-3} & \frac{1}{8} \\ \alpha(2\gamma-1) & -\frac{\alpha(8\gamma-1)}{8} \end{bmatrix},$$

$\boldsymbol{H}_1 = \boldsymbol{S}\boldsymbol{H}_0\boldsymbol{S}$, $\boldsymbol{H}_2 = \boldsymbol{S}\boldsymbol{H}_{-1}\boldsymbol{S}$, and

$$\widetilde{\boldsymbol{H}}_{-1} = \begin{bmatrix} 2\gamma & \frac{16\gamma(2\gamma-1)}{8\gamma-3} \\ \frac{2}{3}\alpha\gamma(8\gamma-1) & \frac{16}{3}\alpha\gamma(2\gamma-1) \end{bmatrix}, \quad \widetilde{\boldsymbol{H}}_0 = \begin{bmatrix} 1-2\gamma & \frac{16\gamma(2\gamma-1)}{8\gamma-3} \\ -\alpha(8\gamma-1)(2\gamma-1) & -\frac{16}{3}\alpha\gamma(2\gamma-1) \end{bmatrix},$$

$\widetilde{\boldsymbol{H}}_1 = \boldsymbol{S}\widetilde{\boldsymbol{H}}_0\boldsymbol{S}$, $\widetilde{\boldsymbol{H}}_2 = \boldsymbol{S}\widetilde{\boldsymbol{H}}_{-1}\boldsymbol{S}$, with γ being the parameter, and $\alpha = \sqrt{3/(8\gamma-1)/(8\gamma-3)}$. Note that when $\gamma = (17 - \sqrt{241})/64$, this SABMF has a GMP order $(3, 2)$.

The corresponding matrix highpass sequences are obtained by solving the PR conditions, and they are given by

$$\boldsymbol{G}_{-1} = \begin{bmatrix} \frac{16\rho\gamma(2\gamma-1)}{8\gamma-3} & 2\rho\gamma \\ 2 & \frac{(8\gamma-5)}{4(2\gamma-1)} \end{bmatrix}, \quad \boldsymbol{G}_0 = \begin{bmatrix} \frac{-16\rho\gamma(2\gamma-1)}{8\gamma-3} & -\rho(6\gamma-1) \\ 2 & \frac{-(16\gamma^2-18\gamma+3)}{8\gamma(2\gamma-1)} \end{bmatrix},$$

$\boldsymbol{G}_1 = \boldsymbol{S}\boldsymbol{G}_0\boldsymbol{S}$, $\boldsymbol{G}_2 = \boldsymbol{S}\boldsymbol{G}_{-1}\boldsymbol{S}$, and

$$\widetilde{\boldsymbol{G}}_{-1} = \begin{bmatrix} \dfrac{1}{8\rho} & 2 \\ \dfrac{\gamma(2\gamma-1)}{8\gamma-3} & \dfrac{16\rho\gamma(2\gamma-1)}{8\gamma-5} \end{bmatrix}, \quad \widetilde{\boldsymbol{G}}_0 = \begin{bmatrix} \dfrac{-1}{8\rho} & \dfrac{16\gamma^2-18\gamma+3}{\gamma(8\gamma-5)} \\ \dfrac{\gamma(2\gamma-1)}{8\gamma-3} & \dfrac{8\rho(2\gamma-1)(6\gamma-1)}{8\gamma-5} \end{bmatrix},$$

$\widetilde{\boldsymbol{G}}_1 = \boldsymbol{S}\widetilde{\boldsymbol{G}}_0\boldsymbol{S}$, $\widetilde{\boldsymbol{G}}_2 = \boldsymbol{S}\widetilde{\boldsymbol{G}}_{-1}\boldsymbol{S}$, and $\rho = \alpha^2\gamma(8\gamma-5)/3$. Two additional parameters δ and τ for the highpass filters can be introduced as described in Theorem 2.3. Thus the length 4/4 family of SABMFs has three parameters, namely, γ, δ and τ. Through minimizing the deviation of the frequency responses for both matrix lowpass and highpass filters from those of the ideal brickwall filters, we have found a solution with $(\gamma, \delta, \tau) = (0.02491, 0.066063, -0.044016)$, which we denote as the **BSA(4/4)** SABMF.

B.4 Parameterization of BSA(5/5) SABMF Family.

Using Method 2, we obtain a family of length 5/5 SABMFs with GMP order at least (3,3) and approximation order (1,2) with the matrix lowpass sequences given by

$$\boldsymbol{H}_{-1} = \begin{bmatrix} \beta_1 & 2\beta_1 \\ \beta_2 & 2\beta_2 \end{bmatrix}, \quad \boldsymbol{H}_0 = \begin{bmatrix} \dfrac{1}{2} & \dfrac{1-3\gamma}{8\gamma} \\ \dfrac{112\gamma-3}{16(1-16\gamma)} & \dfrac{1}{1-16\gamma} \end{bmatrix}, \quad \boldsymbol{H}_1 = \begin{bmatrix} \dfrac{11\gamma+1}{16\gamma} & 0 \\ 0 & \dfrac{16\gamma+11}{8(16\gamma-1)} \end{bmatrix},$$

$\boldsymbol{H}_2 = \boldsymbol{S}\boldsymbol{H}_0\boldsymbol{S}$, $\boldsymbol{H}_3 = \boldsymbol{S}\boldsymbol{H}_{-1}\boldsymbol{S}$, and

$$\widetilde{\boldsymbol{H}}_{-1} = \begin{bmatrix} 2\beta_3 & \beta_3 \\ 2\beta_4 & \beta_4 \end{bmatrix}, \quad \widetilde{\boldsymbol{H}}_0 = \begin{bmatrix} \dfrac{1}{2} & 2\gamma \\ \dfrac{1-16\gamma}{16\gamma} & \dfrac{1-16\gamma}{4} \end{bmatrix}, \quad \widetilde{\boldsymbol{H}}_1 = \begin{bmatrix} \dfrac{1}{2}+4\gamma & 0 \\ 0 & \dfrac{128\gamma^2+8\gamma-1}{32\gamma} \end{bmatrix},$$

$\widetilde{\boldsymbol{H}}_2 = \boldsymbol{S}\widetilde{\boldsymbol{H}}_0\boldsymbol{S}$, $\widetilde{\boldsymbol{H}}_3 = \boldsymbol{S}\widetilde{\boldsymbol{H}}_{-1}\boldsymbol{S}$, with γ being the parameter, and $\beta_1 = \dfrac{5\gamma-1}{32\gamma}$, $\beta_2 = \dfrac{16\gamma-5}{32(1-16\gamma)}$, $\beta_3 = \dfrac{1}{8} - \gamma$ and $\beta_4 = \dfrac{128\gamma^2-24\gamma+1}{64\gamma}$. Note that when $\gamma = 3/14$, this SABMF has a GMP order $(4,3)$.

The corresponding highpass sequences are obtained by solving the PR conditions, and

they are given by

$$\boldsymbol{G}_{-1} = \begin{bmatrix} \frac{1}{\rho} & \frac{2}{\rho} \\ 1 & 2 \end{bmatrix}, \quad \boldsymbol{G}_0 = \begin{bmatrix} \frac{\rho-2}{\rho} & 2 \\ \rho-2 & 2\rho \end{bmatrix}, \quad \boldsymbol{G}_1 = \begin{bmatrix} \frac{2(1-\rho)}{\rho} & 0 \\ 0 & 4(\rho-1) \end{bmatrix},$$

$\boldsymbol{G}_2 = \boldsymbol{S}\boldsymbol{G}_0\boldsymbol{S}$, $\boldsymbol{G}_3 = \boldsymbol{S}\boldsymbol{G}_{-1}\boldsymbol{S}$, and

$$\widetilde{\boldsymbol{G}}_{-1} = \begin{bmatrix} 2\beta_5 & \beta_5 \\ 2\beta_6 & \beta_6 \end{bmatrix}, \quad \widetilde{\boldsymbol{G}}_0 = \begin{bmatrix} \frac{\rho}{4(\rho-2)} & \frac{16+11\rho}{64(\rho+2)} \\ \frac{6+5\rho}{32\rho(\rho-2)} & \frac{1}{8\rho} \end{bmatrix}, \quad \widetilde{\boldsymbol{G}}_1 = \begin{bmatrix} \frac{13\rho^2+22\rho-32}{32(4-\rho^2)} & 0 \\ 0 & \frac{19\rho-22}{64\rho(\rho-2)} \end{bmatrix},$$

$\widetilde{\boldsymbol{G}}_2 = \boldsymbol{S}\widetilde{\boldsymbol{G}}_0\boldsymbol{S}$, $\widetilde{\boldsymbol{G}}_3 = \boldsymbol{S}\widetilde{\boldsymbol{G}}_{-1}\boldsymbol{S}$, where $\rho = \frac{-16\gamma}{8\gamma-1}$, $\beta_5 = \frac{3\rho^2+10\rho+32}{128(4-\rho^2)}$ and $\beta_6 = \frac{3\rho+10}{128\rho(2-\rho)}$. Introducing the highpass parameters as described in Theorem 2.3 gives us a set of parameters (γ, τ, δ) for filter design. By imposing the design constraint to minimize the deviation from the ideal brick-wall filter, we obtained $(\gamma, \tau, \delta) = (0.166, 0.02904, 0.3823)$, which gives the **BSA(5/5)** SABMF.

B.5 Matrix Filter Coefficients of BSA(6/6), BSA(8/6), and BSA(7/9) SABMFs.

For concise presentation, only the left halves of the symmetric and antisymmetric matrix sequences are given; the other half of the matrix sequences can be obtained directly via (2.49), (2.50), (2.63), and (2.64). The four elements of each 2×2 matrix are shown in the tables below.

	matrix elements			
	$(1,1)$	$(1,2)$	$(2,1)$	$(2,2)$
$\boldsymbol{H}_{-2}$	$3.7500000000\times10^{-3}$	$1.4228515625\times10^{-2}$	$-3.5261765040\times10^{-3}$	$-1.1217948718\times10^{-2}$
$\boldsymbol{H}_{-1}$	$3.7500000000\times10^{-3}$	$1.1077148438\times10^{-1}$	$3.5261765040\times10^{-3}$	$-9.1346153846\times10^{-2}$
$\boldsymbol{H}_{0}$	$9.9250000000\times10^{-1}$	$9.6542968750\times10^{-2}$	$-9.9294764699\times10^{-1}$	$1.0256410256\times10^{-1}$
$\widetilde{\boldsymbol{H}}_{-2}$	$1.1875000000\times10^{-2}$	$-1.7500000000\times10^{-2}$	$-1.4862396101\times10^{-2}$	$2.6122346529\times10^{-2}$
$\widetilde{\boldsymbol{H}}_{-1}$	$1.1875000000\times10^{-2}$	$1.4250000000\times10^{-1}$	$1.4862396101\times10^{-2}$	$-2.0336703959\times10^{-1}$
$\widetilde{\boldsymbol{H}}_{0}$	$9.7625000000\times10^{-1}$	$1.6000000000\times10^{-1}$	$-9.7027520780\times10^{-1}$	$1.7724469306\times10^{-1}$
$\boldsymbol{G}_{-2}$	$6.5648882468\times10^{-3}$	$-3.0964377335\times10^{-2}$	$4.3525896101\times10^{-3}$	$-2.6994758871\times10^{-2}$
$\boldsymbol{G}_{-1}$	$-1.7574894635\times10^{-1}$	$-1.3733503316\times10^{-1}$	$-1.3761875586\times10^{-1}$	$-1.2938385229\times10^{-1}$
$\boldsymbol{G}_{0}$	$1.6918405810\times10^{-1}$	$-9.0080319008\times10^{-1}$	$-1.4197134547\times10^{-1}$	$-9.0975071883\times10^{-1}$
$\widetilde{\boldsymbol{G}}_{-2}$	$4.7800000000\times10^{-3}$	$-2.1363070388\times10^{-2}$	$5.4000000000\times10^{-3}$	$-2.6135973142\times10^{-2}$
$\widetilde{\boldsymbol{G}}_{-1}$	$-9.6082189893\times10^{-2}$	$1.4225207034\times10^{-1}$	$-1.2264656332\times10^{-1}$	$1.7257217778\times10^{-1}$
$\widetilde{\boldsymbol{G}}_{0}$	$9.1302189893\times10^{-2}$	-1.0951449891	$-1.2804656332\times10^{-1}$	-1.0844087112

Table B.1: Matrix filter coefficients for the length 6/6 SABMF, **BSA(6/6)**.

	matrix elements			
	$(1,1)$	$(1,2)$	$(2,1)$	$(2,2)$
$\boldsymbol{H}_{-3}$	$-8.2302083262\times10^{-4}$	$8.2302083262\times10^{-4}$	$9.6022938094\times10^{-4}$	$-9.6022938094\times10^{-4}$
$\boldsymbol{H}_{-2}$	$-4.6234965230\times10^{-3}$	$8.2077182033\times10^{-3}$	$1.3974875495\times10^{-3}$	$-6.4715835076\times10^{-3}$
$\boldsymbol{H}_{-1}$	$-8.8047997154\times10^{-3}$	$1.1514624013\times10^{-1}$	$2.5093985463\times10^{-3}$	$-1.0001233526\times10^{-1}$
$\boldsymbol{H}_{0}$	1.0142513171	$1.0776154276\times10^{-1}$	-1.0212372391	$1.0744414815\times10^{-1}$
$\widetilde{\boldsymbol{H}}_{-2}$	$2.1104701795\times10^{-2}$	$-2.1104701795\times10^{-2}$	$-1.9985994269\times10^{-2}$	$1.9985994269\times10^{-2}$
$\widetilde{\boldsymbol{H}}_{-1}$	$2.6799308895\times10^{-2}$	$1.4610470180\times10^{-1}$	$1.4291387169\times10^{-2}$	$-1.8943281291\times10^{-1}$
$\widetilde{\boldsymbol{H}}_{0}$	$9.5209598931\times10^{-1}$	$1.6720940359\times10^{-1}$	$-9.4294419016\times10^{-1}$	$1.6944681864\times10^{-1}$
$\boldsymbol{G}_{-3}$	$1.7829230862\times10^{-2}$	$-1.7829230862\times10^{-2}$	$-2.3452016066\times10^{-2}$	$2.3452016066\times10^{-2}$
$\boldsymbol{G}_{-2}$	$-1.5253945168\times10^{-1}$	$1.8475712670\times10^{-2}$	$1.5061297307\times10^{-1}$	$1.4559887080\times10^{-2}$
$\boldsymbol{G}_{-1}$	$1.3471022082\times10^{-1}$	$-7.5869505647\times10^{-1}$	$1.7406498914\times10^{-1}$	$9.7927575289\times10^{-1}$
$\widetilde{\boldsymbol{G}}_{-4}$	$-1.5234945394\times10^{-3}$	$1.5234945394\times10^{-3}$	$1.0805555534\times10^{-3}$	$-1.0805555534\times10^{-3}$
$\widetilde{\boldsymbol{G}}_{-3}$	$-9.8235980627\times10^{-3}$	$-3.9031124881\times10^{-4}$	$8.9939643797\times10^{-3}$	$-2.3321146392\times10^{-3}$
$\widetilde{\boldsymbol{G}}_{-2}$	$-1.2411843813\times10^{-1}$	$-1.3651668127\times10^{-2}$	$1.1172069054\times10^{-1}$	$6.2836197534\times10^{-4}$
$\widetilde{\boldsymbol{G}}_{-1}$	$1.3546553073\times10^{-1}$	-1.2695994976	$1.0380728171\times10^{-1}$	$9.8579043639\times10^{-1}$

Table B.2: Matrix filter coefficients for the length 8/6 SABMF, **BSA(8/6)**.

	matrix elements			
	$(1,1)$	$(1,2)$	$(2,1)$	$(2,2)$
$\boldsymbol{H}_{-3}$	$2.4344641344\times10^{-3}$	$8.0308636460\times10^{-3}$	$-1.4303031405\times10^{-2}$	$-2.2484089612\times10^{-2}$
$\boldsymbol{H}_{-2}$	$-5.9584095876\times10^{-2}$	$-6.7345319191\times10^{-2}$	$9.1258095290\times10^{-2}$	$1.3679981195\times10^{-1}$
$\boldsymbol{H}_{-1}$	$4.9756553587\times10^{-1}$	$3.3899626889\times10^{-1}$	$-6.4314887462\times10^{-1}$	$-5.8587445386\times10^{-1}$
$\boldsymbol{H}_{0}$	1.1191681918	0	0	$9.4311746304\times10^{-1}$
$\widetilde{\boldsymbol{H}}_{-4}$	$4.6532884089\times10^{-4}$	$1.3454909684\times10^{-3}$	$-1.5859468567\times10^{-3}$	$-1.5100595484\times10^{-3}$
$\widetilde{\boldsymbol{H}}_{-3}$	$4.6032430899\times10^{-3}$	$9.2260003323\times10^{-3}$	$5.5684633005\times10^{-3}$	$7.9169461102\times10^{-4}$
$\widetilde{\boldsymbol{H}}_{-2}$	$-7.4100607596\times10^{-2}$	$-4.2124994014\times10^{-2}$	$1.0567420381\times10^{-1}$	$5.6400334801\times10^{-2}$
$\widetilde{\boldsymbol{H}}_{-1}$	$4.9539675691\times10^{-1}$	$3.0730677897\times10^{-1}$	$-6.3082688988\times10^{-1}$	$-4.1173356874\times10^{-1}$
$\widetilde{\boldsymbol{H}}_{0}$	1.1472705575	0	0	$7.1210319776\times10^{-1}$
$\boldsymbol{G}_{-5}$	$7.2032111363\times10^{-4}$	$2.1517368074\times10^{-3}$	$-2.4185700065\times10^{-3}$	$-5.5906496284\times10^{-3}$
$\boldsymbol{G}_{-4}$	$-1.5277852687\times10^{-2}$	$-1.7573761065\times10^{-2}$	$3.4174427829\times10^{-2}$	$4.1879446146\times10^{-2}$
$\boldsymbol{G}_{-3}$	$8.4831158796\times10^{-2}$	$1.1194381191\times10^{-1}$	$-1.0636149261\times10^{-1}$	$-1.1607913261\times10^{-1}$
$\boldsymbol{G}_{-2}$	$-4.0497214731\times10^{-1}$	$-5.5246405542\times10^{-1}$	$3.2584646973\times10^{-1}$	$3.6022717769\times10^{-1}$
$\boldsymbol{G}_{-1}$	$6.6939704018\times10^{-1}$	0	0	1.0475528122
$\widetilde{\boldsymbol{G}}_{-4}$	$-3.6769171486\times10^{-2}$	$-3.9466446717\times10^{-2}$	$1.9861584220\times10^{-2}$	$1.9330180380\times10^{-2}$
$\widetilde{\boldsymbol{G}}_{-3}$	$9.8299274500\times10^{-2}$	$-2.8355176261\times10^{-2}$	$-6.6840892804\times10^{-2}$	$-5.5235811329\times10^{-3}$
$\widetilde{\boldsymbol{G}}_{-2}$	$-5.5811482613\times10^{-1}$	$-7.8764710492\times10^{-1}$	$3.5809909597\times10^{-1}$	$6.0239546446\times10^{-1}$
$\widetilde{\boldsymbol{G}}_{-1}$	$9.9316944624\times10^{-1}$	0	0	1.2544984519

Table B.3: Matrix filter coefficients for the length 8/6 SABMF, **BSA(8/6)**.

Bibliography

[1] Tham, J.Y., "Detail Preserving Image Compression Using Wavelet Transform," Undergraduate Thesis, Department of Electrical Engineering, National University of Singapore, 1995. [This thesis was also awarded the 1995 NUS Engineering Innovation Award (Merit Prize); and the Best Student Paper (UG Category) of IEEE Region 10 Student Paper Contest 1995.]

[2] Tham, J.Y., "Wavelet-Based Scalable Video Compression," Report (Proposal for the conversion from M.Eng. to Ph.D.), Electrical Engineering, National University of Singapore, (76 pages), Oct. 1996.

[3] Tham, J.Y., Ranganath, S., and Kassim, A.A., "Scalable Very Low Bit Rate Video Compression Using Motion Compensated 3-D Wavelet Decomposition," *IEEE Int. Workshop on Intelligent Signal Process. and Commun. Syst. (ISPACS)*, vol. 3, pp. 38.7.1-38.7.5, Nov. 1996.

[4] Tham, J.Y., Ranganath, S., Ranganath, M., and Kassim, A.A., "A Novel Unrestricted Center-Biased Diamond Search for Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 369-377, Aug. 1998.

[5] Tham, J.Y., Ranganath, S., and Kassim, A.A., "Highly scalable wavelet-based video codec for very low bit rate environment," *IEEE J. Select. Areas Commun. – Special Issue on Very Low Bit-rate Video Coding*, vol. 16, no. 1, pp. 12-27, Jan. 1998.

[6] Tham, J.Y., Shen, L., Lee, S.L., and Tan, H.H., "Good multifilter properties: A new tool for understanding multiwavelets," *Int. Conf. Imag. Sci., Syst., Tech. (CISST'98)*,

pp. 52-59, Jul. 1998.

[7] Tham, J.Y., Ranganath, S., and Kassim, A.A., "Scalable multiwavelet-based image and video compression for multimedia applications," *4th. Int. Conf. Info. Syst., Analysis, Synthesis (SCI'98/ISAS'98)*, vol. 3, pp. 195-202, Jul. 1998.

[8] Tham, J.Y., Ranganath, S., Kassim, A.A., and Tan, S.Y., "Non-iterative adaptive postprocessing for ringing artifact suppression in compressed images," *4th. Int. Conf. Info. Syst., Analysis, Synthesis (SCI'98/ISAS'98)*, vol. 3, pp. 203-210, Jul. 1998.

[9] Shen, L., Tham, J.Y., Lee, S.L., and Tan, H.H., "A special class of orthonormal wavelets: theory, implementations, and applications," *IEEE ICASSP*, vol. 3, pp. 1225-1228, Mar. 1999.

[10] Tham, J.Y., Shen, L., Lee, S.L., and Tan, H.H., "A new multifilter design property for multiwavelet image compression," *IEEE ICASSP*, vol. 3, pp. 1229-1232, Mar. 1999.

[11] Tham, J.Y., Shen, L., Lee, S.L., and Tan, H.H., "A general approach for analysis and application of discrete multiwavelet transforms," *IEEE Trans. Signal Process.*, vol. 48, no. 2, pp. 457-464, Feb. 2000.

[12] Shen, L., Tan, H.H., and Tham, J.Y., "Symmetric-antisymmetric orthonormal multiwavelets and related scalar wavelets," *Applied and Comput. Harmonic Analysis*, vol. 8, no. 3, pp. 258-279, May 2000.

[13] Tan, H.H., Shen, L., and Tham, J.Y., "New biorthogonal multiwavelets for image compression," *Signal Process. J.*, vol. 79, no. 1, pp.45-65, 1999.

[14] Lee, S.L., Shen, L., Tan, H.H., and Tham, J.Y. "Multiwavelets: Theory and applications," *2nd. Int. Workshop on Transforms and Filter Banks* (invited paper), Second International Workshop on Transforms and Filter Banks, Brandenberg, Germany, Mar. 1999.

[15] Shen, L., Tan, H.H., and Tham, J.Y., "Some properties of symmetric-antisymmetric orthonormal multiwavelets," *IEEE Trans. Signal Process.*, vol. 48, no. 7, pp. 2161-2163, Jul. 2000.

[16] Wang P.J., Shang F.C., Ding P., Tham, J.Y., and Lee, S.L., "Scalable multimedia information distribution over heterogeneous network," *ICCS 2000, The 7th. Int. Conf. on Commun. Syst.*, Nov. 2000.

[17] Wang P.J., Shang F.C., Ding P., Tham, J.Y., and Lee, S.L., "Scalable multimedia communication over heterogeneous networks," *2000 IEEE Int. Symp. on Intelligent Signal Process. and Commun. Syst. (ISPACS)*, vol. 1, pp. 16-21, Nov. 2000.

[18] Tham, J.Y., Ranganath, S., Kassim, A.A., and Lee, S.L., "Resolution and rate scalable color image compression using multiwavelets," (in preparation for submission to *IEEE Trans. Image Process.*), 2001.

[19] Tham, J.Y., Ranganath, S., Kassim, A.A., and Lee, S.L., "A flexible multi-scalable color video compression architecture," (in preparation for submission to *IEEE Trans. Circuits Syst. Video Technol.*), 2001.

[20] Tham, J.Y., Ranganath, S., Kassim, A.A., and Lee, S.L., "Multi-scalable video compression via segmented overlay mapping of multiwavelet coefficients," (in preparation for submission to *IEEE Trans. Circuits Syst. Video Technol.*), 2001.

[21] Aldroubi, A., "Oblique and hierarchical multiwavelet bases," *Applied and Comput. Harmonic Analysis*, vol. 4, pp. 231-263, 1997.

[22] Ananda, P., "Measuring Visual Motion from Image Sequences," Ph.D. Dissertation, COINS TR 87-21, Univ. of Massachusetts, Amherst, MA, 1987.

[23] Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I., "Image coding using wavelet transform," *IEEE Trans. on Image Process.*, vol. 1, no. 2, pp. 205-220, Apr. 1992.

[24] Barron, J.L., Fleet, D.J., and Beauchemin, S.S., "Systems and experiments: Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43-77, 1994.

[25] Bernard, C.P., "Discrete wavelet analysis for fast optic flow computation," CMAP Internal Report 415, Centre de Mathematiques Appliquees, Ecole Polytechnique, France, Mar. 1999 (http://www.cmap.polytechnique.fr/ bernard/Publications/).

[26] Bierling, M., "Displacement estimation by hierarchical block matching," *SPIE Vis. Commun. and Image Process.'88*, vol. 1001, pp. 942-951, 1988.

[27] Boroczky, L., "Pel-recursive motion estimation for image coding," Ph.D. Dissertation, Delft University of Technology, 1991.

[28] Chalidabhongse, J., and Kuo, C.-C.J., "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 477-488, Jun. 1997.

[29] Cheng, P.Y., Li, J., and Kuo, C.-C.J., "Multiscale video compression using wavelet transform and motion compensation," *Intern. Conf. on Image Process.*, pp. 606-609, 1995.

[30] Chow, K.H.-K., and Liou, M.L., "Generic motion search algorithm for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 440-445, Dec. 1993.

[31] Chow, K.Y.-R, and Tham, C.K., "Scalable video delivery to handheld-based clients," *Proceeding of IEEE Int. Conf. Networks 2000*, pp. 93-98, Sept. 2000.

[32] Chrysafis, C., and Ortega, A., "Efficient context-based entropy coding fo lossy wavelet image coding," *Data Compression Conf.*, Mar. 1996.

[33] Chui, C.K., and Lian, J., "A study on orthonormal multiwavelets," *Appl. Numer. Math.*, vol. 20, no. 3, pp. 273-298, 1996.

[34] Chung, W.C., Kossentini, F., and Smith, M.J.T., "Rate-distortion-constrained statistical motion estimation for video coding," *IEEE Int. Conf. Image Process.*, vol. 3, pp. 184-187, Oct. 1995.

[35] Cohen, A., Daubechies, I., and Plonka, G., "Regularity of refinable function vectors," *J. Fourier Anal. Appl.*, vol. 3, pp. 295-324, 1997.

[36] Dahmen, W., and Micchelli, C.A., "Biorthogonal wavelet expansions," *Constr. Approx.*, vol. 13, pp. 293-328, 1997.

[37] Daubechies, I., "Ten Lectures on Wavelets," CBMS-NSF Regional Conf. Series on Appl. Math., vol. 61, SIAM, Philadelphia, PA, 1992.

[38] Deering, S.E., "Multicast Routing in a Datagram Internetwork," Ph.D. Dissertation, Stanford University, Dec. 1991.

[39] Dimitrios, T., Strintzis, M.G., and Sahinolou, H., "Evaluation of multiresolution block matching techniques for motion and disparity estimation," *Signal Processing: Image Commun.*, vol. 6, pp. 56-67, 1994.

[40] Donovan, G., Geronimo, J.S., Hardin, D.P., and Massopust, P.R., "Construction of orthogonal wavelets using fractal interpolation functions," in *SIAM, J. Math. Analysis*, vol. 27, pp. 1158-1192, 1996.

[41] Evangelista, G., "Wavelet transforms and wave digital filters," Wavelets and Applications (Y. Meyer, Ed.), Springer-Verlag, pp. 396-412, 1992.

[42] CCITT SGXV, "Recommendation H.261: Video codec for audio-visual services at p×64 kbits/sec.," The International Telegraph and Telephone Consultative Committee, 1990.

[43] Geronimo, J.S., Hardin, D.P., and Massopust, P.R., "Fractal functions and wavelet expansions based on several scaling functions," *J. Approx. Theory*, vol. 78, pp. 373-401, 1994.

[44] Ghanbari, M., "The cross-search algorithm for motion estimaton," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 950-953, Jul. 1990.

[45] Gharavi, H., and Mills, M., "Block matching motion estimation algorithms: New results," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 37, pp. 649-651, May 1990.

[46] Girod, B., "Rate-constrained motion estimation," *SPIE Vis. Commun. and Image Process.'94*, vol. 2308, pp. 1026-1034, 1994.

[47] Goh, S.S., Jiang, Q., and Xia, T., "Construction of biorthogonal multiwavelets using the lifting scheme," *preprint*, Wavelets Strategic Research Programme, National University of Singapore, 1998.

[48] Goodman, T.N.T., Lee, S.L., and Tang, W.S., "Wavelets in wandering subspaces," *Trans. Amer. Math. Soc.* vol. 338, pp. 639-654, 1993.

[49] Goodman, T.N.T., and Lee, S.L., "Wavelets of multiplicity r," *Trans. Amer. Math. Soc.*, vol. 342, pp. 307-324, 1994.

[50] Hardin, D.P., and Roach, D.W., "Multiwavelet prefilter I: orthogonal prefilters preserving approximation order $p \leq 2$," *IEEE Trans. Circuits Syst. Video Technol. II*, vol. 8, pp. 1106-1112, 1998.

[51] HDTV, "Grand alliance HDTV system specification," Version 2.0, Dec. 1994.

[52] Heeger, D.J., "Optical flow using spatiotemporal filters," *Intern. J. Comput. Vis.*, vol. 1, pp. 279-302, 1988.

[53] Heil, C., Strang, G., and Strela, V., "Approximation by translates of refinable functions," *Numerische Mathematik*, 1996.

[54] Hirohisa, J., Kamikura, K., Kotera, H., and Watanabe, H., "Two-stage motion compensation using adaptive global MC and local affine MC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 75-84, Feb. 1997.

[55] Höetter, M., and Thoma, R., "Image segmentation based on object oriented mapping parameter estimation," *Signal Process*, vol. 15, no. 3, pp. 315-334, 1988.

[56] Höetter, M., "Differential estimation of the global motion parameters zoom and pan," *Signal Process.*, vol. 16, pp. 249-265, 1989.

[57] Horn, B.K.P., and Schunck, B.G., "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185-204, 1981.

[58] Horn, U., Girod, B., and Belzer, B., "Scalable vdieo coding with multiscale motion compensation and unequal error protection," *Proc. Int. Symp. Multimedia Commun. Video Coding*, New York, NY, Oct. 1995.

[59] ISO/IEC CD 11172-2 (MPEG-1 Video), "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s : Video," 1993. (http://www.cselt.it/mpeg/standards/mpeg-1/mpeg-1.htm)

[60] ISO/IEC CD 13818-2 — ITU-T H.262 (MPEG-2 Video), "Information technology - Generic coding of moving pictures and associated audio information: Video," 1995. (http://www.cselt.it/mpeg/standards/mpeg-2/mpeg-2.htm)

[61] ISO/IEC 14496-2 (MPEG-4 Video), "Information technology - Coding of audio-visual objects: video," Nov. 1997. (http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm)

[62] ITU Telecommunication Standardization Sector LBC - 95, Study Group 15, Working Party 15/1, Expert's Group on Very Low Bitrate Visual Telephony, from *Digital Video Coding Group, Telenor Research and Development*, or via URL: "http://www.nta.no/brukere/DVC/tmn5".

[63] Illgner, K., and Müller, F, "Spatially scalable video compression employing resolution pyramids," *IEEE J. Select. Areas Commun.*, vol. 15, no. 9, pp. 1688-1703, Dec.. 1997.

[64] Jain, J.R., and Jain, A.K., "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, pp. 1799-1808, Dec. 1981.

[65] Jiang, Q., and Shen, Z., "On existence and weak stability of matrix refinable functions," to appear in *Constructive Approx.*

[66] Jiang, Q., "Orthogonal multiwavelets with optimum time-frequency resolution," *IEEE Trans. on Signal Process.*, vol. 46, no. 4, pp. 830-844, Apr. 1998.

[67] Joo, S., Muramatsu, S., and Hikuchi, H., "A fast motion estimation in the wavelet transform domain for video compression," *2000 IEEE Int. Symp. on Intelligent Signal Process. and Commun. Syst. (ISPACS)*, pp. 68-71, Nov. 2000.

[68] Kappagantula, S., and Rao, K.R., "Motion compensated interframe image prediction," *IEEE Trans. Commun.*, vol. COM-33, pp. 1011-1015, Sep. 1985.

[69] Keesman, G.J., "Motion estimation based on a motion model incorporating translation, rotation and zoom," *Signal Process. IV: Theories and Appl.*, New York: Elsevier, pp. 31-34, 1988.

[70] Kim, J.-S., and Park, R.-H., "A fast feature-based block matching algorithm using integral projections," *IEEE J. Select. Areas Commun.*, vol. 10, pp. 968-971, Jun. 1992.

[71] Koga, T., Ilinuma K., Hirano, A., Iijima, Y., and Ishiguro, T., "Motion-compensated interframe coding for video conferencing," *Proc. NTC 81*, pp. C9.6.1-9.6.5, New Orleans, LA, Nov./Dec. 1981.

[72] Lawton, W., Lee, S.L., and Shen, Z., "An algorithm for matrix extension and wavelet construction," *Math. Comp.,* vol. 65, pp. 723-737, 1996.

[73] Lee, L.W., Wang, J.F., Lee, J.Y., and Shie, J.D., "Dynamic search-window adjustment and interlaced search for block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 85-87, Feb. 1993.

[74] Lee, X., and Zhang, Y.Q., "A fast hierarchical motion-compensation scheme for video coding using block feature matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 6, pp. 627-635, Dec. 1996.

[75] Lee, J.Y., Kim, T.H., and Ko, S.J., "Motion prediction based on temporal layering for layered video coding," *Proc. of ITC-CSCC 1998*, vol. 1, pp. 245-248, July 1998.

[76] Lee, J.Y., Kim, T.H., and Ko, S.J., "Layered video coding for multimedia applications," *Proc. SPIE Conf. on Multimedia Syst. and Appl.* , vol. 3528, pp. 509-517, Nov. 1998.

[77] Lee, J.Y., "A Novel Approach to Multi-Layer Coding of Video for Playback Scalability," Ph.D. Dissertation, Korea University, Jul. 1999.

[78] Li, R., Zeng, B., and Liou, M.L., "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438-442, Aug. 1994.

[79] Liang, K.C., Li, J., and Kuo, C.C.J., "Image compression with embedded multiwavelet coding," *Proc. SPIE.*, vol. 2762, pp. 165-176, 1996.

[80] Liu, B., and Zaccarin, A., "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 2, pp. 148-157, Apr. 1993.

[81] Liu, L.K., and Feig, E., "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 419-422, Aug. 1996.

[82] Jianhua, L., and Liou, M.L., "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 429-433, Apr. 1997.

[83] Lucas, B.D., "Generalized Image matching by the Method of Differences," Ph.D. Dissertation, Dept. of Computer Science, Carnegie-Mellon University, 1984.

[84] Mallat. S., "Multifrequency channel decompositions of images and wavelet models," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, pp. 2091-2110, 1989.

[85] Martucci, S.A., Sodagar, I., Chiang, T., and Zhang, Y.Q., "A zerotree wavelet video coder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 109-118, Feb. 1997.

[86] Mathew, R., and Arnold, J.F., "Layered coding using bitstream decomposition with drift correction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 6, pp. 882-891, Dec. 1997.

[87] Meyer, F.G., Averbuch, A.Z., and Coifman, R.R., "Motion compensation of wavelet coefficients for very low bit rate video coding," *Intern. Conf. Image Process.*, Oct. 1997.

[88] Meyer, F.G., Averbuch, A.Z., and Coifman, R.R., "Motion compensation of wavelet coefficients with wavelet packet based motion residual coding," *Technical Report* (http://noodle.med.yale.edu/ meyer/profile.html), Oct. 1997.

[89] Nagel, H.-H., "On the estimation of optical flow: Relations between different approaches and some new results," *Artificial Intelligence*, vol. 33, pp. 299-324, 1987.

[90] Nguyen-Phi, K., and Weinrichter, H., "A fast wavelet image coder based on contextual coefficient coding," *Intern. Conf. Information, Comm. Signal Process. (ICICS'97)*, vol. 1, pp. 480-484, Sep. 1997.

[91] Nosratinia, A., and Orchard, M., "Multi-resolution backward video coding," *Intern. Conf. Image Process.*, vol. 2, pp. 563-566, Oct. 1995.

[92] Nosratinia, A., and Orchard, M., "Optimal unified approach to warping and overlapped block estimation in video coding," *SPIE Vis. Commun. and Image Process.'96*, Orlando, Florida, Mar. 1996.

[93] Papadopoulos, C.A., and Clarkson, T.G., "Motion compensation using second-order geometric transformations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, No. 4, pp. 319-331, Aug. 1995.

[94] Po, L.M., and Ma, W.C., "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313-317, Jun. 1996.

[95] Podilchuk, C.I., and Jayant, N.S., "Three-dimensional subband coding of video," *IEEE Trans. Image Process.*, vol. 4, no. 2, pp. 125-139, Feb. 1995.

[96] Puri, A., Hang, H.-M., and Schilling, D.L., " An efficient block-matching algorithm for motion compensated coding," *Proc. IEEE ICASSP*, pp. 25.4.1-25.4.4, 1987.

[97] Said, A., and Pearlman, W.A., "New, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243-250, Jun. 1996.

[98] Sezan, M.I., and Lagendijk, R.L., *Motion analysis and image sequence processing*, Kluwer Academic Publisher: Boston, 1993.

[99] Shapiro, J.M., "Embedded image coding using zerotrees of wavelets coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445-3462, Dec. 1993.

[100] Shen, Z., "Refinable function vectors," *SIAM J. Math. Anal.*, vol. 29, pp. 234-249, 1998.

[101] Shi, Y.Q., and Xia, X., "A thresholding multiresolution block matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 437-440, Apr. 1997.

[102] Singh, A., "Optic Flow Computation: A Unified Perspective," IEEE Computer Society Press, 1992.

[103] Srinivasan, R., and Rao, K.R., "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. COM-33, pp. 888-896, Aug. 1985.

[104] Strang, G., and Strela, V., "Short wavelets and matrix dilation equations," *IEEE Trans. Signal Process.*, vol. 43, pp. 108-115, 1995.

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

[106] Strang, G., "Eigenvalues of $(\downarrow 2)H$ and convergence of cascade algorithm," *IEEE Trans. Signal Process.*, vol.44, pp. 233-238, 1996.

[107] Strela, V., Heller, P.N., Strang, G., Topiwala, P., and Heil, C., "The application of multiwavelet filter banks to image processing," *IEEE Trans. on Image Process.*, vol. 8, no. 4, pp. 548-563, 1999.

[108] Sweldens, W., and Piessens, R., "Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions," *SIAM J. Numer. Anal.*, vol. 31, pp. 1240-1264, 1994.

[109] Talluri, R., Oehler, K., Bannon, T., Courtney, J.D., Das, A., and Liao, J., "A robust, scalable, object-based video compression technique for very low bit-rate coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 329-354, Aug. 1996.

[110] Taubman, D., and Zakhor, A., "Multirate 3-D subband coding of video," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 572-588, Sep. 1994.

[111] Taubman, D., and Zakhor, A., "A common framework for rate and distortion based scaling of highly scalable compressed video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 329-354, Aug. 1996.

[112] Trane, P., and Welin, J. , "A New Method for Motion Compensation in MPEG using Hierarchical Mean Calculation ," Online Report, Dec. 1995 (http://www.axis.se/se/axisprize/96pr.htm).

[113] Vaidyanathan P.P., and Hoang, P.Q., "Lattice structures for optimal design of two-channel perfect-reconstruction QMF banks," *IEEE Trans. ASSP*, vol. 36, no. 1, pp. 81-94, Jan. 1988.

[114] Vaidyananthan, P.P., Multirate systems and filter banks, Englewood Cliffs, NJ: Prentice Hall, 1993.

[115] Vetterli, M., and Herley, C., "Wavelets and filter banks: theory and design," *IEEE Trans. Signal Process.*, vol. 40, no. 9, pp. 2207-2232, Sep. 1992.

[116] Villasenor, J., Belzer, B., and Liao, J., "Wavelet filter evaluation for image compression," *IEEE Trans. Image Process.*, vol. 2, pp. 1053-1060, Aug. 1995.

[117] Walker, D.R., and Rao, K.R., "Improved pel-recursive motion compensation," *IEEE Trans. Commun.*, vol. COM-32, pp. 1128-1134, Oct. 1983.

[118] Wang, Q., and Ghanbari, M., "Scalable coding of very high resolution video using the virtual zerotree," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 5, pp. 719-726, 1997.

[119] Witten, I., Neal, R., and Cleary, J., "Arithmetic coding for data compression," *Comms. ACM*, vol. 30, pp. 520-540, 1987.

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

[121] Xia, X.G., Geronimo, J.S., Hardin, D.P., and Suter, B.W., "Design of prefilters for discrete multiwavelet transforms," *IEEE Trans. Signal Process.*, vol. 44, no. 1, pp. 25-35, Jan. 1996.

[122] Xia, X.G., "A new prefilter design for discrete multiwavelet transforms," *IEEE Trans. Signal Process.*, vol. 46, no. 6, pp. 1558-1570, Jun. 1998.

[123] Yang, K.H., Lee, S.J., and Lee, C.W., "Motion-compensated wavelet transform coder for very low bit-rate visual telephony," *Signal Process.*, pp. 581-592, 1995.

[124] Zhang, Y.-Q., and Zafar, S., "Motion compensated wavelet transform coding for color video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 3, pp. 285-296, 1992.

[125] Zeng, B., Li, R., and Liou, M.L., "Optimization of fast block motion estimatin algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 6, pp. 833-844, Dec. 1997.