Applications of Multiwavelets to Image Compression

Michael B. Martin

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
(Virginia Tech)
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Dr. Amy E. Bell, Chair
Dr. A. A. (Louis) Beex
Dr. Brian D. Woerner

June, 1999
Blacksburg, Virginia

Keywords: Image Compression, Wavelets, Multiwavelets, Wavelet Packets,
Multiwavelet Packets, Filter Banks

Applications of Multiwavelets to Image Compression

Michael B. Martin

(ABSTRACT)

Methods for digital image compression have been the subject of much study over the past decade. Advances in wavelet transforms and quantization methods have produced algorithms capable of surpassing the existing image compression standards like the Joint Photographic Experts Group (JPEG) algorithm. For best performance in image compression, wavelet transforms require filters that combine a number of desirable properties, such as orthogonality and symmetry. However, the design possibilities for wavelets are limited because they cannot simultaneously possess all of these desirable properties. The relatively new field of multiwavelets shows promise in removing some of the limitations of wavelets. Multiwavelets offer more design options and hence can combine all desirable transform features. The few previously published results of multiwavelet-based image compression have mostly fallen short of the performance enjoyed by the current wavelet algorithms. This thesis presents new multiwavelet transform methods and measurements that verify the potential benefits of multiwavelets. Using a zerotree quantization scheme modified to better match the unique decomposition properties of multiwavelets, it is shown that the latest multiwavelet filters can give performance equal to, or in many cases superior to, the current wavelet filters. The performance of multiwavelet packets is also explored for the first time and is shown to be competitive to that of wavelet packets in some cases. The wavelet and multiwavelet filter banks are tested on a much wider range of images than in the usual literature, providing a better analysis of the benefits and drawbacks of each.

Acknowledgments

I would like to thank my advisor, Dr. Bell. Without the technical and financial assistance she provided I could not have completed this thesis. Her suggestions and corrections for this thesis have improved it enormously. She always helped me keep sight of the "big picture" and focus on those areas that needed the most attention. My thanks also go to my wife, Kim, whose help in proofreading this thesis is much appreciated, and Dr. Brian Woerner for agreeing to serve on my committee at the last minute.

I would like to thank a number of people from the Virginia Tech physics department. In particular, I am indebted to professors Dale Long, John Ficenec, and Joseph Slawny for their support and guidance in my days as a physics student. I am also grateful for the friendship and assistance provided by my colleagues in physics, Mike Kleder and Zoltan Toroczkai.

Without the support of my parents, I could not be where I am now. I am ever grateful for their love and trust in my judgement, which opened the door to my education. And I am forever thankful to Kim, for her kindness, her faith in me, and her continual love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

It has been suggested that "a picture is worth a thousand words." This is all the more true in the modern era in which information has become one of the most valued of assets. Recent technology has introduced the paradigm of digital information and its associated benefits and drawbacks. When the time comes to store a photograph digitally, its worth is put to the test. A thousand words stored on a digital computer requires very little capacity, but a single picture can require much more. A thousand pictures can require a very large amount of storage. While the advancement of computer storage technology continues at a rapid pace, a means for reducing the storage requirements of an image is still needed in most situations. Thus the science of digital image compression has emerged. Current methods of image compression, such as the popular Joint Photographic Experts Group (JPEG) standard, can provide good performance in terms of retaining image quality while reducing storage requirements. But even the popular standards like JPEG have limitations. Research in new and better methods of image compression is ongoing, and recent results suggest that some newer techniques may provide much greater performance than those developed just five years ago. This thesis gives a summary of some of these new advances, presents some

new multiwavelet decomposition and quantization techniques which improve the currently published results, and illustrates their potential for inclusion in new image compression applications and standards.

1.2 Previous Work

Both wavelet theory and methods for its application to image compression have been well developed over the past decade. Even so, the field of wavelets is still sufficiently new that further advancements continue to be reported in many areas. Numerous authors have contributed to the field to make it what it is today, with the most well known pioneer probably being Ingrid Daubechies. Other researchers whose contributions directly influence this work include Stéphane Mallat for the pyramid filtering algorithm, and the team of R. R. Coifman, Y. Meyer, and M. V. Wickerhauser for their introduction of wavelet packets [3]. Much of the current theory of multiwavelets comes from Vasily Strela and members of the Wavelets Strategic Research Programme (WSRP) at the National University of Singapore. Of course, many others have contributed to the development of the theory of multiwavelets over the past several years.

Recent literature on the subject of multiwavelets has focused mostly on development of the basic theory [9, 30, 21, 29, 24], methods of constructing new multifilters [22, 8], and methods for application to denoising and compression [21, 19, 20, 29, 24, 8]. Some authors have already presented brief evaluations of the performance of multiwavelets for image compression using orthogonal multiwavelets [21, 19, 20, 29, 24], and more recently with biorthogonal multiwavelets [24, 8]. While the results of these evaluations have generally shown interesting promise for multiwavelets, they have been limited to a few tests, and often are just a small portion of a larger work. Also, some techniques required for good image compression results, such as symmetric signal extension for linear phase multifilters, have only just recently been developed. With the very recent work on symmetric signal extension for the class of

symmetric-antisymmetric multiwavelets [29], multiwavelets can now be compared to scalar wavelets on equal footing in practical image compression applications.

1.3 Significance of This Work

Multiwavelets are only now beginning to approach the maturity of development of their scalar counterparts. A few papers that have tested the image compression properties of multiwavelets suggest that multiwavelets can sometimes perform as well as, or better than, scalar wavelets [19, 29, 24, 8]. But to date, no researchers have pursued this more thoroughly with the intention of determining whether multiwavelets might be a better choice for image compression than scalar wavelets, at least in some applications.

This thesis presents an evaluation of the performance of state-of-the-art multiwavelet methods for compression of general classes of images. To better determine typical performance on an arbitrary image, a much larger selection of images is tested than in the usual literature. The images used in this work include many popular favorites like Lena and Barbara, which have been chosen for comparison to the results found in the literature. Additionally, some "synthetic" computer-generated images have been chosen for having characteristics quite different from those of the "natural" images normally used to test wavelet algorithms. This thesis presents the following new results:

1. A comparison of the best known multiwavelets is made to the best known scalar wavelets. Both quantitative and qualitative measures of performance are examined for each of several natural and synthetic images.

2. The use of multiwavelet packets is explored. Multiwavelet packets are based on the ideas of wavelet packets, as applied to multiwavelet filter banks, and are defined in Section 4.3.

3. A method is presented for improving the performance of zerotree-like quantization

methods [16, 15].

1.4 Outline of Report

Chapter 2 presents the transform approach to image compression used by most current algorithms. Chapter 3 provides the basic theory of wavelets and wavelet packets, with an introduction to multiwavelets given in Chapter 4. Chapter 5 discusses implementation issues for wavelets and multiwavelets, and Chapter 6 presents the results from a wavelet/multiwavelet coder implementation. Finally, Chapter 7 concludes this work with a summary of results, some conclusions based thereon, and a discussion of future work.

Chapter 2

Image Compression Methodologies

2.1 Overview of Present Techniques

A number of methods have been presented over the years to perform image compression. They all have one common goal: to alter the representation of information contained in an image so that it can be represented sufficiently well with less information. Regardless of the details of each image compression method, the methods can be classified into two general categories: lossy or lossless. For methods in the first category, some information from the original image is lost, even if only a small amount. Conversely, lossless compression methods provide a perfect reproduction of the original image.

Current methods for lossless image compression, such as that used in the Graphical Interchange Format (GIF) image standard, typically use some form of Huffman or arithmetic coder [10, 28] or an integer-to-integer wavelet transform [1]. Unfortunately, even the best current lossless algorithms provide relatively small compression factors compared to the best lossy methods. To achieve a high compression factor, a lossy method must be used.

The most popular current lossy image compression methods use a transform-based scheme, as shown in Figure 2.1. The signal is processed with an invertible transform, such as discrete

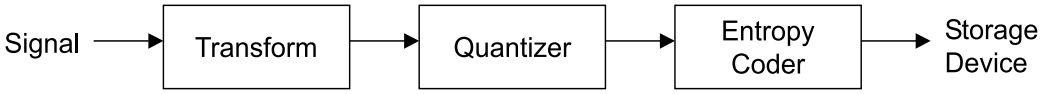

Figure 2.1: Typical image compression system design.

cosine transform (DCT) or wavelet transform. This step is intended to "decorrelate" the input signal by transforming to a representation in which the set of data values is sparser, thereby compacting the information content of the signal into a smaller number of coefficients. The transform coefficients, which may typically be thought of as infinite precision real numbers[1], are then quantized. This step is not reversible and represents the lossy stage in the process. A good quantizer tries to assign more bits for coefficients with more information content or perceptual significance, and fewer bits for coefficients with less information content, based on a given fixed bit budget. The final step is entropy coding, which removes redundancy from the output of the quantizer. Each of these components of the compression process is described in further detail in the following sections.

2.2 Transform

The choice of transform used depends on a number of factors, in particular, computational complexity and coding gain. Computational complexity is measured in terms of the number of multiplications and additions required for the implementation of the transform. Coding gain is a measure of how well the transformation compacts signal energy into a small number of coefficients. The most commonly used transforms today are the DCT, the wavelet transform, and the generalized lapped orthogonal transform (GenLOT). The latter, GenLOT, is a relatively new design of M-channel filter banks which mitigates some of the undesirable properties of the DCT (and of which the DCT is a special case). A good discussion of these different transforms can be found in the book by Strang and Nguyen [18].

[1]Or at least machine precision floating-point numbers.

A good study of the benefits, drawbacks, and performance of DCT and wavelet transforms is presented in a paper by Xiong, Orchard, and Ramchandran [31]. It is noted in that paper that the DCT used in the JPEG standard is less computationally complex than wavelet transforms for the same number of image samples. However, it is also recognized that wavelet transforms can achieve coding gain superior to that of the DCT. Wavelet transforms also allow additional freedom in the selection of the particular wavelet filter used; in contrast, there is only one DCT[2]. GenLOT can also give better compression than the DCT, especially at low bit rate [18]. And, unlike the DCT, GenLOT offers the ability to optimize the filters for certain performance criteria, such as coding gain or stopband attenuation. Wavelet-based transforms perform slightly better than GenLOT at medium bit rate, but it is difficult to select one over the other at high or low bit rates [18].

More sophisticated transform methods are also possible, such as Multiple-Basis Representation (MBR) algorithms. One such approach is to simultaneously decompose an image with different transforms and then select the basis vector yielding the largest coefficient. This process is then repeated as desired on the residual image (which is the difference between the input and reconstructed image). Another approach is to apply two or more different transforms (e.g. DCT, wavelet, etc.) in succession. The initial transform is followed by one or more different transforms that operate iteratively on the residual image of the previous stage. For example, one paper cites good results from the application of a wavelet transform to an image followed by a local cosine transform of the residual [13]. Using multiple transforms (and hence multiple bases) allows the input signal to be better represented than if a fixed basis were used, but the algorithms are more complicated.

[2]There are in fact four transforms that may fall under the name of discrete cosine transform, each with different symmetry properties. Only one, typically denoted "Type IV", has the needed symmetry properties for image compression with symmetric boundary extension [18].

2.3 Quantization

There have been numerous methods proposed to perform quantization of the transform coefficients. Even so, quantization remains an active field of research and some new results show great promise for wavelet-based image compression [32, 33]. Since the properties possessed by the coefficients from the transformation stage depend on the transform used, the choice of a good quantizer depends on the transform that is selected. While transforms and quantizers can be "mixed and matched" to a certain degree, some quantization methods perform better with particular transform methods [31]. Also, perceptual weighting of coefficients in different subbands can be used to improve subjective image quality [18].

Quantization methods used with wavelet transforms fall into two general categories: embedded and non-embedded. Scalar and vector quantizers are common examples of non-embedded quantizers. They determine bit allocations based on a specified bit budget, allocating bits across a set of quantizers corresponding to the image subbands. The most common reference for efficient bit allocation over a set of different quantizers is the paper by Shoham and Gersho [17]. Embedded quantization schemes [16, 15] organize the bits so that, loosely speaking, the most important bits are transmitted first. More pedantically, a quantization method is embedded if, for two different bit budgets N and M, where $N > M$, the first M bits of the quantizer output with bit budget N are identical to those when the budget is just M bits.

2.4 Entropy Coding

Entropy coding substitutes a sequence of codes for a sequence of symbols, where the codes are chosen to have fewer bits when the probability of occurrence of the corresponding symbol is higher. This process removes redundancy in the form of repeated bit patterns in the output of the quantizer. Frequently occurring symbols are replaced with shorter bit patterns while infrequently occurring symbols are replaced with longer bit patterns, resulting in a smaller

bit stream overall.

The most common entropy coding techniques are run-length encoding (RLE), Huffman coding, arithmetic coding [10, 28], and Lempel-Ziv (LZ) algorithms. With RLE, when a symbol is repeated many times consecutively (a "run"), the coder substitutes for the sequence just the first symbol and the length of the run. This works well when there are many long runs in the quantized data, but does not work well otherwise. Huffman, arithmetic, and LZ codes substitute bit patterns for symbols (which may also be bit patterns) based on the frequency of symbols. The frequency of each symbol may be assumed *a priori* or estimated adaptively. LZ methods avoid the frequency estimation problem by building a dictionary that maps symbols to bits, but these methods are less effective than arithmetic coders [28]. While the Huffman algorithm requires each code to be an integral number of bits, arithmetic coding methods allow for fractional numbers of bits per code by grouping two or more such codes together into a block composed of an integral number of bits. This allows arithmetic codes to outperform Huffman codes, and consequently arithmetic codes are more commonly used in wavelet-based algorithms [16, 15].

Chapter 3

Wavelets and Wavelet Packets

In Chapter 2, it was shown that the most commonly used image compression methods use three steps: transform, quantization, and entropy coding. This chapter will discuss how the first step, the transform, may be accomplished using wavelets[1]. The following is intended as merely a crash course in wavelet theory. The interested reader is referred to other sources for more detailed background. Simple introductions may be found in the books by Strang and Nguyen [18] and Vetterli and Kovačević [26], while readers desiring more mathematical treatment may instead prefer the books by Daubechies [6] or Mallat [11].

3.1 Wavelets and Multiresolution Analysis

The theory of wavelets starts with the concept of a multiresolution analysis. In the analysis of functions, it is convenient to express functions belonging to a certain space as a linear combination of basis functions. In wavelet analysis, we go one step further and stipulate that every basis function be a dilation and translation of a single scaling function of unit norm, denoted $\phi(t)$. We require that integer translations $\{\phi(t-k)\}|_{k\in\mathcal{Z}}$ be linearly independent

[1] In fact, this chapter will present only wavelet transforms for one-dimensional signals. The techniques for handling two-dimensional image data are discussed in Chapter 5.

and produce an orthonormal basis for the subspace V_0. Likewise, for fixed integer scale j, the translations $\{2^{-j/2}\,\phi(2^{-j}t-k)\}|_{k\in\mathcal{Z}}$ form an orthonormal basis for the subspace V_j, such that the subspaces satisfy

$$\cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \tag{3.1}$$

and

$$\overline{\bigcup_{j=-\infty}^{\infty} V_j} = L^2(\mathcal{R}), \qquad \bigcap_{j=-\infty}^{\infty} V_j = \{0\}. \tag{3.2}$$

Then each function $f(t) \in L^2(\mathcal{R})$ can be written as a linear combination of these basis functions with weights $\alpha_{j,k}$,

$$f(t) = \sum_{j=-\infty}^{\infty} 2^{-j/2} \sum_{k=-\infty}^{\infty} \alpha_{j,k}\,\phi(2^{-j}t-k), \tag{3.3}$$

where

$$\alpha_{j,k} = \int_{-\infty}^{\infty} f(t)\,2^{-j/2}\,\phi(2^{-j}t-k)\,dt. \tag{3.4}$$

The weights $\alpha_{j,k}$ are called scaling coefficients. It is important to point out that (3.3) is not an orthogonal expansion of $f(t)$ into the V_j subspaces because they are not mutually disjoint (in light of (3.1)). Furthermore, the functions $\{2^{-j/2}\,\phi(2^{-j}t-k)\}|_{j,k\in\mathcal{Z}}$ do not represent a basis for $L^2(\mathcal{R})$ because they are not all linearly independent across scales.

The nesting property of the subspaces implies that the scaling function $\phi \in V_0$ also belongs to V_{-1} and thus satisfies the two-scale dilation equation

$$\phi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k\,\phi(2t-k) \tag{3.5}$$

where the sequence $\{h_k\}$ will become important in the next section. In this context the set $\{h_k\}$ represents coefficients of the orthogonal projection of $\phi(t)$ onto the basis $\{\sqrt{2}\,\phi(2t-k)\}|_{k\in\mathcal{Z}}$ of V_{-1}, but later we will regard them as lowpass filter coefficients in a filter bank.

The multiresolution nature of this analysis lends itself to a graphic interpretation. As the scale j decreases, the support of each basis function $2^{-j/2}\,\phi(2^{-j}t-k)$ decreases as well.

Hence smaller scales may be regarded as representing a finer, or more detailed, resolution. Conversely, larger values of j indicate basis functions with wider support, suggesting coarser scales and less detail. This makes sense in light of (3.1), since $V_j \subset V_{j-1}$ implies that V_{j-1} represents a space of functions at a finer resolution than V_j.

Because we want an orthogonal decomposition of $f(t)$, we now define the "difference" space W_j as the complement of V_j in V_{j-1}, namely

$$V_{j-1} = V_j \oplus W_j, \qquad V_j \bigcap W_j = \emptyset. \tag{3.6}$$

It can be shown that the W subspaces provide a decomposition of $L^2(\mathcal{R})$ into mutually orthogonal subspaces [6], i.e.

$$W_j \perp W_{j'} \text{ if } j \neq j', \quad \text{and} \quad \bigoplus_{j=-\infty}^{\infty} W_j = L^2(\mathcal{R}). \tag{3.7}$$

The basis for the subspace W_0 consists of translations of a new function, $\psi(t)$. Since the W spaces inherit the scaling properties of the V spaces, $\{2^{-j/2}\,\psi(2^{-j}t-k)\}|_{k\in\mathcal{Z}}$ will be a basis for W_j. As with the V spaces, each function $f(t) \in L^2(\mathcal{R})$ can be written as a linear combination of these basis functions with weights $\beta_{j,k}$:

$$f(t) = \sum_j 2^{-j/2} \sum_{k=-\infty}^{\infty} \beta_{j,k}\,\psi(2^{-j}t-k), \tag{3.8}$$

where

$$\beta_{j,k} = \int_{-\infty}^{\infty} f(t)\,2^{-j/2}\,\psi(2^{-j}t-k)\,dt. \tag{3.9}$$

The function $\psi(t)$ is called the wavelet function, and the numbers $\beta_{j,k}$ are called the wavelet coefficients. The difference between (3.3) and (3.8) is that the latter represents an orthogonal expansion of $f(t)$. Also, the functions $\{2^{-j/2}\,\psi(2^{-j}t-k)\}|_{j,k\in\mathcal{Z}}$ do represent a basis for $L^2(\mathcal{R})$. Since $\psi \in W_0$ and $W_0 \subset V_{-1}$, we have

$$\psi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} g_k\,\phi(2t-k) \tag{3.10}$$

with orthogonal projection coefficients $\{g_k\}$. Note from (3.6) that each subspace V_M is the direct sum of V_N and some W subspaces, for $N > M$:

$$V_M = V_N \oplus W_N \oplus W_{N-1} \oplus W_{N-2} \oplus \cdots \oplus W_{M+1}. \tag{3.11}$$

This is another way of saying that a function belonging to the subspace V_M may be written as the linear combination of basis functions of the mutually orthogonal subspaces $V_N, W_N, W_{N-1}, ..., W_{M+1}$. Thus given the function $f(t) \in V_M$ we may write

$$f(t) = 2^{-N/2} \sum_{k=-\infty}^{\infty} \alpha_{N,k}\, \phi(2^{-N}t - k) + \sum_{j=M+1}^{N} 2^{-j/2} \sum_{k=-\infty}^{\infty} \beta_{j,k}\, \psi(2^{-j}t - k). \tag{3.12}$$

In the filter bank representation of the wavelet transform presented in the next section, this selection of subspaces will correspond to the tree-shaped octave-band iteration of the analysis bank.

3.2 Discrete Implementation as Filter Banks

It has been shown that the wavelet analysis relations given by the two-scale equations (3.5) and (3.10) can be expressed in terms of a two-channel perfect reconstruction (PR) filter bank, where the lowpass and highpass filters have impulse responses $\{h_k\}$ and $\{g_k\}$, respectively [6]. Perfect reconstruction means that the output of the filter bank is identical to the input, except for a possible delay and overall scaling factor. A particular signal of interest, $x(t) \in V_0$, can be written as a linear combination of the basis functions $\{\phi(t-k)\}$ with weights $v_{0,k}$:

$$x(t) = \sum_{k=-\infty}^{\infty} v_{0,k}\, \phi(t-k). \tag{3.13}$$

If we pass the coefficients $v_{0,k}$ into a two-channel filter bank, after the first level of filtering and downsampling we obtain the lowpass coefficients

$$v_{1,k} = \sum_{m=-\infty}^{\infty} h_{m-2k}\, v_{0,k} \tag{3.14}$$

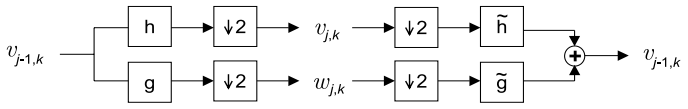

Figure 3.1: Analysis and synthesis stages of a 2-channel single-level biorthogonal PR filter bank.

and highpass coefficients

$$w_{1,k} = \sum_{m=-\infty}^{\infty} g_{m-2k}\, v_{0,k}. \tag{3.15}$$

Similar iterations of the filter bank on the lowpass channel result in the coefficients

$$v_{j,k} = \sum_{m=-\infty}^{\infty} h_{m-2k}\, v_{j-1,k}, \tag{3.16}$$

$$w_{j,k} = \sum_{m=-\infty}^{\infty} g_{m-2k}\, v_{j-1,k}, \tag{3.17}$$

after j stages of the analysis filters, where $v_{j-1,k}$ is the output of the $(j-1)^{th}$ iteration of the filter bank. With the filters $\{h_k\}$ and $\{g_k\}$ as given in (3.5) and (3.10), it turns out that the numbers $\{v_{j,k}\}$ are in fact just the scaling coefficients $\{\alpha_{j,k}\}$, and the numbers $\{w_{j,k}\}$ are actually the wavelet coefficients $\{\beta_{j,k}\}$. Thus the wavelet decomposition of the signal $x(t)$ in the bases for the subspaces V_J and W_J, for $J>0$, can be found by repeated filtering of the scaling coefficients $\{v_{0,k}\}$ corresponding to the subspace V_0.

The synthesis portion of the filter bank reconstructs the sequence $\{v_{j-1,k}\}$ from the sequences $\{v_{j,k}\}$ and $\{w_{j,k}\}$ via

$$v_{j-1,k} = \sum_{m=-\infty}^{\infty} \tilde{h}_{k-2m}\, v_{j,k} + \sum_{m=-\infty}^{\infty} \tilde{g}_{k-2m}\, w_{j,k}\,, \tag{3.18}$$

as is shown in Figure 3.1. Note that the filters in the synthesis stage, with impulse responses $\{\tilde{h}_k\}$ and $\{\tilde{g}_k\}$, are not necessarily the same as those in the analysis stage, which have impulse responses $\{h_k\}$ and $\{g_k\}$. The filters $\{\tilde{h}_k\}$ and $\{\tilde{g}_k\}$ are called the dual[2] filters with respect

[2]The definition of "dual" is beyond the scope of this discussion. The interested reader is referred to Strang and Nguyen [18].

to $\{h_k\}$ and $\{g_k\}$. The dual filters have corresponding dual scaling and wavelet functions, $\tilde{\phi}(t)$ and $\tilde{\psi}(t)$, which generate the dual spaces $\tilde{V}_j$ and $\tilde{W}_j$. The equations relating these dual quantities are identical to those in the last section, with a tilde placed over each quantity.

For an orthogonal PR filter bank, $\tilde{h}_k$ and $\tilde{g}_k$ are just the time reversals of h_k and g_k, respectively. However, an FIR filter bank cannot have both orthogonal and symmetric filters with length greater than two [18]. Consequently, filters having symmetric impulse responses with length greater than 2 must be biorthogonal, in which case $\{\tilde{h}_k\}$ and $\{\tilde{g}_k\}$ will be different from $\{h_k\}$ and $\{g_k\}$. Biorthogonal filter banks give up the orthogonality property to gain symmetric filters. The use of symmetric filters is important in image compression because the best transform methods require symmetric filters to perform symmetric boundary extension of the image. Daubechies [6] gives the "alternating flip" conditions on $\tilde{h}_k$ and $\tilde{g}_k$ for a PR filter bank:

$$g_n = (-1)^{n+1}\,\tilde{h}_{-n+1}, \tag{3.19}$$

$$\tilde{g}_n = (-1)^{n+1}\,h_{-n+1}. \tag{3.20}$$

The requirements on the biorthogonal lowpass and highpass filters to ensure perfect reconstruction of the original signal, expressed in the time domain, are

$$\sum_{n=-\infty}^{\infty} h_n\,\tilde{h}_{n+2k} \;=\; \delta_{k,0}, \tag{3.21}$$

$$\sum_{n=-\infty}^{\infty} h_n\,\tilde{g}_{n+2k} \;=\; 0, \tag{3.22}$$

$$\sum_{n=-\infty}^{\infty} g_n\,\tilde{h}_{n+2k} \;=\; 0, \tag{3.23}$$

$$\sum_{n=-\infty}^{\infty} g_n\,\tilde{g}_{n+2k} \;=\; \delta_{k,0}, \tag{3.24}$$

where k is presumed to be integer.

This is the essence of Mallat's algorithm[3]: starting with the scaling coefficients of a given signal $x(t)$ in subspace V_0, the scaling and wavelet coefficients for subspaces $W_1, W_2, ..., W_{J-1}, W_J$

[3]Mallat's algorithm is also referred to as the pyramid algorithm or the fast wavelet transform.

and V_J are computed directly via (3.16) and (3.17). The inverse computation is performed by repeated application of (3.18). In terms of the multiresolution analysis, Mallat's algorithm starts at a particular scale and proceeds to coarser scales by the filtering process. Conversely, the synthesis bank starts at the coarsest scale and finishes at the finest scale. An example of a three-level analysis filter is shown in Figure 3.2. The corresponding synthesis filter is shown in Figure 3.3.

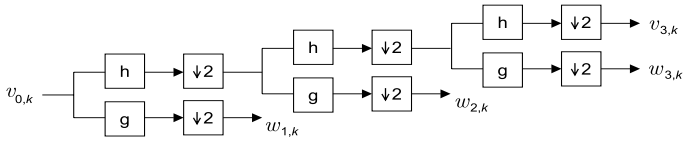

Figure 3.2: Analysis stage of a three-level biorthogonal PR filter bank.

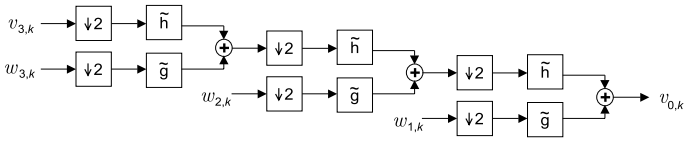

Figure 3.3: Synthesis stage of a three-level biorthogonal PR filter bank.

3.3 Wavelet Packets

The filter bank design associated with the wavelet analysis method involves iterating the lowpass-highpass filtering and downsampling procedure only on the output of the lowpass branch of the previous stage. A question that immediately arises is, "What happens if you iterate on the highpass branch as well? Do you still get corresponding orthonormal bases, and if so, what are they?"

Coifman, Meyer, and Wickerhauser answered these questions by presenting an extension of the octave-band wavelet decomposition to a full tree decomposition [3]. They defined the

new basis functions as follows. Let $u_0(t) \equiv \phi(t)$ and $u_1(t) \equiv \psi(t)$, and define

$$u_{2n}(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \, u_n(2t-k), \tag{3.25}$$

$$u_{2n+1}(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} g_k \, u_n(2t-k). \tag{3.26}$$

Coifman et al. showed that the set $\{u_n(t-k)\}$, where n (the "modulation" parameter) ranges over nonnegative integers and k ranges over all integers, forms an orthonormal basis of $L^2(\mathcal{R})$. By also including dilations by a power of 2, a library of functions $\{2^{-j/2} \, u_n(2^{-j}t-k)\}$ is formed. Note that this library is overcomplete: not all of its members are orthogonal, and it contains many subsets that are a complete basis of $L^2(\mathcal{R})$ themselves. The question remaining is how to select a complete orthonormal basis from the library[4]. The answer is given as a theorem: if the set of integers $\{l, n\}$ is such that the intervals $[2^l n, 2^l(n+1))$ form a disjoint covering of the half line $[0, \infty)$, then the corresponding set of functions $\{2^{l/2} \, u_n(2^l t-k)\}|_{k \in \mathcal{Z}}$ forms a complete orthonormal basis of $L^2(\mathcal{R})$. The basis functions are called wavelet packets.

The selection of a basis can also be viewed in terms of a tree structure. Using tree terminology, the set of elements of each basis corresponds in a one-to-one fashion to a particular set of terminal nodes of a binary tree. Some examples of possible basis selections are shown as trees in Figure 3.4. For each tree in Figure 3.4, a branching indicates that the signal entering from the left passes through two channels. The upper branch is lowpass filtered, while the lower branch is highpass (bandpass) filtered. Each branch is then also downsampled before the next branching point. The right-most points in the tree are the terminal nodes. For example, the tree in Figure 3.4a is the same as that in Figure 3.2. As a second example, consider the two-level tree in Figure 3.4c. The input signal is lowpass filtered and downsampled to obtain the uppermost branch with its corresponding terminal node and basis function $u_0(t)$. The input signal also passes through a lower branch and is highpass filtered and downsampled. The result of that operation is again lowpass filtered and downsampled to produce the next terminal node (with basis function $u_2(t)$) and highpass filtered and downsampled to produce

[4]Note that Multiple-Basis Representation (MBR) methods also provide basis selection adapted to match the input signal, but they do not require orthogonality of the bases.

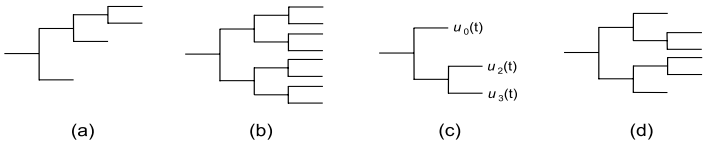

Figure 3.4: Possible wavelet packet filter banks. (a) is the standard wavelet decomposition, (b) is the full tree (Walsh basis), (c) and (d) are other possibilities.

the last terminal node (with basis function $u_3(t)$). Interpretation of Figures 3.4b and 3.4d follows this same reasoning.

As in the previous section, the Mallat algorithm is used to implement this procedure in the form of a filter bank. But there is one key difference for wavelet packets: the highpass output of each branch is also filtered and downsampled, up to a maximum number of decomposition iterations. After computing all the coefficients in the full tree, a basis is chosen by pruning the tree. Pruning involves making a decision at each branch about whether to keep the two "children", or prune them and keep the "parent". The pruning process starts at the terminal nodes at some maximum depth (such as five levels of decomposition) and works back up the tree until all terminal nodes (children) have been chosen instead of their parent nodes. The final basis is given by the remaining unpruned terminal nodes. How the pruning decision is made at each step can vary depending on the application. Note that the pruning operation itself is rather efficient, requiring only $O(N)$ operations where N is the length of the original data signal [4]. The increased computational complexity of wavelet packets comes from the computation of all coefficients in the full tree and then the computation required at each branch to determine whether to prune the tree at that point.

Selection of a "best" basis may be performed in a number of ways. Coifman et al. suggest the use of an additive cost function that is applied to each set of parent and child nodes in the pruning process. If the sum of the costs of the children is greater than the parent's cost, the children are pruned; otherwise the children are kept. The performance of this method depends entirely on the choice of cost functions. Some cost functions that have

been proposed include: Shannon entropy [5], the number of coefficients in the node that are significant compared to (i.e., greater than) some threshold[5] [12], and the number of bits required to represent all the coefficients in the node (introduced in this paper).

Newer methods for selecting a basis approach the problem from a rate-distortion perspective. Ramchandran and Vetterli proposed a method that attempts to select the set of terminal nodes that are optimal in a rate-distortion sense [14]. Their approach involves the minimization at each branch of a Lagrangian "cost function", $J(\lambda) = D + \lambda R$, where D is the average distortion and R is the target average bit rate. The value of λ that minimizes $J(\lambda)$ determines whether to prune and also gives the best quantizer for that node (which is then used for uniform quantization of the coefficients of that node). More recently, Xiong et al. have taken this idea and merged the basis optimization with their space-frequency quantization (SQF) approach, yielding impressive results [32, 33].

The benefit of wavelet packets over the octave-band wavelet decomposition (described in Section 3.1) comes from the ability of the wavelet packets to better represent high-frequency content, and high-frequency oscillating signals in particular. This allows wavelet packets to perform significantly better than wavelets for compression of images with a large amount of texture, such as the commonly used Barbara image. For example, Meyer et al. [12] show that wavelet packet techniques applied to images with textured patterns can give over 0.5 dB improvement in some cases over the SPIHT algorithm results [15]. The authors also point out that the perceived image quality is significantly improved using wavelet packets instead of wavelets, especially in the textured regions of the images. Xiong et al. show similar results using wavelets and wavelet packets both with SPIHT and their own SFQ method [33]. Regardless of the choice of quantizer, they show wavelet packets often outperforming wavelets by 0.5-1.0 dB across bit rates for the Barbara image. The results in these papers confirm the ability of wavelet packets to outperform wavelets in some image compression situations. In the next chapter we will consider an alternative approach to improving wavelet-based image

[5] Usually this threshold is taken to be on the order of the quantization step size.

compression: multiwavelets.

Chapter 4

Multiwavelets and Multiwavelet Packets

The wavelet transform described in Section 3.2 is one type of transform that may be used in image compression. A newer alternative is the multiwavelet transform. Multiwavelets are very similar to wavelets but have some important differences. As indicated in Chapter 3, wavelets may be described in the context of a multiresolution analysis with scaling function $\phi(t)$ and wavelet function $\psi(t)$. In fact, it is possible to have *more than one* scaling (and wavelet) function. This is the idea behind multiwavelets, which are described in this chapter as a natural extension of the wavelets in Chapter 3.

4.1 Wavelets of Multiplicity r

While the very first multiwavelet literature goes back further[1], some of the earliest developed multiresolution theory of multiwavelets can be found in a paper by Goodman et al. [9]. Vasily Strela's Ph.D. thesis [21] extends the theory of multiwavelets even further and presents it in

[1]For more details, see the history and references in Strela's Ph.D. thesis [21].

terms of PR multifilter banks in both the time and frequency domains. The description of multiwavelet theory here follows the organization of the wavelet theory in Chapter 3.

For the remainder of this chapter, let r be a positive integer. Now it is presumed that the subspace V_0 is spanned by translations of $\{\phi_m(t)\}$, $m = 1, 2, \ldots, r$, a set of r normalized and mutually orthogonal scaling functions. The subspace V_j is then spanned by the orthonormal set $\{2^{-j/2}\,\phi_m(2^{-j}t-k)\}$, where $m = 1, 2, \ldots, r$ and k is an integer. Similarly, there are r wavelet functions $\{\psi_m(t)\}$ such that $\{2^{-j/2}\,\psi_m(2^{-j}t-k)\}$ constitutes an orthonormal basis for W_j.

For notational convenience, the set of scaling functions can be written using the vector notation $\Phi(t) \equiv [\phi_1(t)\ \phi_2(t)\ \cdots\ \phi_r(t)]^T$, where $\Phi(t)$ is called the multiscaling function. Likewise, the multiwavelet function is defined from the set of wavelet functions as $\Psi(t) \equiv [\psi_1(t)\ \psi_2(t)\ \cdots\ \psi_r(t)]^T$. When $r = 1$, as in Chapter 3, $\Psi(t)$ is called a *scalar* wavelet, or simply wavelet[2]. While in principle r can be arbitrarily large, all multiwavelets used in this work are only for $r = 2$.

The multiresolution subspace relations in (3.1), (3.2), (3.6), (3.7), and (3.11) hold without modification for multiwavelets. The wavelet two-scale equations, (3.5) and (3.10), have nearly identical multiwavelet equivalents:

$$\Phi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} H_k\, \Phi(2t-k), \tag{4.1}$$

$$\Psi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} G_k\, \Phi(2t-k). \tag{4.2}$$

Note, however, that $\{H_k\}$ and $\{G_k\}$ are *matrix* filters, i.e. H_k and G_k are $r \times r$ matrices for each integer k. The filter bank representation is also mostly unchanged, except now the input and output of every branch in the multifilter bank is a vector. A particular signal of interest, $x(t) \in V_0$, can be written as a linear combination of the basis functions $\{\phi_l(t-k)\}$,

[2]In the rest of this thesis, wavelets are assumed to be scalar unless explicitly denoted multiwavelets.

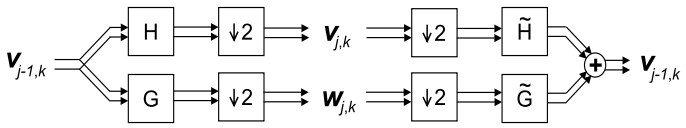

Figure 4.1: Analysis and synthesis stages of a single-level biorthogonal PR multifilter bank.

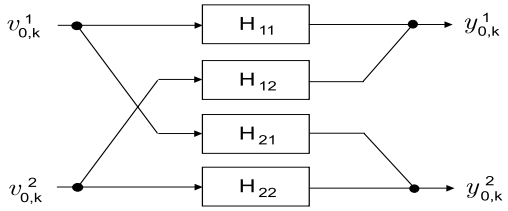

Figure 4.2: View of the H multifilter as a 2-input, 2-output system composed of the scalar filters H_{11}, H_{12}, H_{21}, and H_{22} (for $r=2$).

$l = 1, 2, \ldots, r$, with weight vector $\mathbf{v}_{0,k} \equiv [v^1_{0,k}\ v^2_{0,k}\ \cdots\ v^r_{0,k}]^T$:

$$x(t) = \sum_{k=-\infty}^{\infty} \mathbf{v}^T_{0,k}\, \Phi(t-k) = \sum_{k=-\infty}^{\infty} \sum_{l=1}^{r} v^l_{0,k}\, \phi_l(t-k), \tag{4.3}$$

where

$$v^l_{0,k} = \int_{-\infty}^{\infty} x(t)\, \phi_l(t-k)\, dt. \tag{4.4}$$

In the scalar-valued expression $v^l_{j,k}$, j refers to the scale, k refers to the translation, and l refers to the sub-channel or vector row. Now (3.16) and (3.17) become:

$$\mathbf{v}_{j,k} = \sum_{m=-\infty}^{\infty} H_{m-2k}\, \mathbf{v}_{j-1,k}, \tag{4.5}$$

$$\mathbf{w}_{j,k} = \sum_{m=-\infty}^{\infty} G_{m-2k}\, \mathbf{v}_{j-1,k}. \tag{4.6}$$

This is one stage of a multi-input multi-output (MIMO) filter bank[3], as shown in Figure 4.1 for the $r=2$ case. Each filter block in Figure 4.1 is really a 2-input, 2-output system,

[3]More information about MIMO filter banks may be found in the book by Vaidyanathan [25].

as shown in Figure 4.2. One of the important differences between multiwavelets and scalar wavelets is that each channel in the filter bank has a vector-valued input and a vector-valued output. A scalar-valued input signal must somehow be converted into a suitable vector-valued signal. This conversion is called preprocessing. There are multiple ways to handling preprocessing and they will be discussed in Chapter 5.

The multifilter bank PR conditions that are analogous to the scalar wavelet PR conditions in (3.21)-(3.24) are

$$\sum_{n=-\infty}^{\infty} H_n \, \tilde{H}_{n+2k}^T = I_k, \tag{4.7}$$

$$\sum_{n=-\infty}^{\infty} H_n \, \tilde{G}_{n+2k}^T = 0, \tag{4.8}$$

$$\sum_{n=-\infty}^{\infty} G_n \, \tilde{H}_{n+2k}^T = 0, \tag{4.9}$$

$$\sum_{n=-\infty}^{\infty} G_n \, \tilde{G}_{n+2k}^T = I_k, \tag{4.10}$$

where I_k is the $k \times k$ identity matrix.

For wavelet-based filter banks to be useful for image compression, the filters must have certain key properties. Good scalar filter properties include zeros located at $z = -1$ in the lowpass filter and a few orders of approximation in the highpass filter. These properties are also important for multifilters. Construction of multifilters is generally more difficult than that of scalar wavelets because the multifilters have more degrees of freedom. However, these extra degrees of freedom can be used to impose good filter properties during the construction process.

Early literature [7, 2] presented the Geronimo-Hardin-Massopust (GHM) and Chui-Lian (CL) multiwavelets, but the authors used specific construction methods that did not try to incorporate all the desired multifilter properties. A more general construction technique for orthogonal multiwavelets with optimum time-frequency resolution has been presented in a paper by Xia et al. [29]. In this paper the authors define what they call "good multifilter properties" which equate to the usual desired scalar wavelet features, such as zero lowpass

filter response at $\omega = \pi$ and zero highpass filter response at $\omega = 0$. They proceed to develop a construction method that automatically incorporates these properties into symmetric-antisymmetric multifilters. Strela et al. [22] present a method for constructing biorthogonal multiwavelets based on exchanging the equivalent of zeros between analysis and synthesis multifilters. Their method is similar to the spectral factorization of scalar wavelets. A more recent construction method for biorthogonal symmetric-antisymmetric multiwavelets given in a paper by Goh et al. [8] uses the lifting scheme of Sweldens [1] for computing multifilters with optimal properties.

4.2 Motivation for Multiwavelets

Algorithms based on scalar wavelets have been shown to work quite well in image compression. Consequently, there must be some justification to use multiwavelets in place of scalar wavelets. Some reasons for potentially choosing multiwavelets have been presented in the existing literature and are summarized below.

First, the extra degrees of freedom inherent in multiwavelets can be used to reduce restrictions on the filter properties. For example, it is well known [18] that a scalar wavelet cannot simultaneously have both orthogonality and a symmetric impulse response that has length greater than 2. Symmetric filters are necessary for symmetric signal extension, while orthogonality makes the transform easier to design and implement. Also, the support length and the number of vanishing moments are directly linked to the filter length for scalar wavelets. This means longer filter lengths are required to achieve higher order of approximation at the expense of increasing the wavelet's interval of support (in the time domain). A high order of approximation is desired for better coding gain, but shorter wavelet support is generally preferred to achieve a better localized approximation of the input function. In contrast to the limitations of scalar wavelets, multiwavelets are able to possess the best of all these properties simultaneously. For example, the GHM multiwavelet [7] is orthogonal, has second order

of approximation, has symmetric scaling and wavelet functions (and thus symmetric filters), and has short support for both of its scaling functions ([0,1] and [0,2], respectively). This combination of good properties is impossible with scalar wavelets. The 4-tap Daubechies filter, for example, also is orthogonal, has second order of approximation, and scaling function support on [0,3]. But, because this scalar filter is orthogonal, it does not possess the important property of symmetry. The biorthogonal 9/7 wavelet has symmetric filters, fourth order of approximation (in both analysis and synthesis filters), and scaling function support on [0,9], but it does not possess orthogonality.

Second, one desirable feature of any transform used in image compression is the amount of energy compaction achieved. A filter with good energy compaction properties can decorrelate a fairly uniform input signal into a small number of scaling coefficients containing most of the energy and a large number of sparse wavelet coefficients. This becomes important during quantization since the wavelet coefficients are typically represented with significantly fewer bits on average than the scaling coefficients. Therefore, better performance is obtained when the wavelet coefficients have values clustered about zero with little variance, to avoid as much quantization noise as possible. Some previous literature [30] cites numerical energy compaction results[4] showing that some multiwavelets achieve significantly better energy compaction than some scalar wavelets. Thus multiwavelets have the potential to offer better reconstruction quality at the same bit rate.

Third, previous literature has shown promising results in the application of multiwavelets to image compression. Image compression results presented in a paper by Strela and Walden [20] show that the popular Bi9/7 scalar wavelet gives better results than some older multiwavelets on images like Lena. However, more recent results show that newer multifilters can be competitive with some of the better scalar filters like Bi9/7 [24, 8, 29]. Also, a paper by Strela et al. [19] presents results in which at least one multiwavelet dramatically outperformed scalar wavelets on a synthetic test image.

[4]The authors of this paper define an energy compaction ratio as the ratio of energy in the bandpass parts to the total energy in the signal and give some numerical results.

Finally, there is the question of computational complexity. At first glance it would seem that scalar wavelets have the clear advantage since each branch in a multiwavelet filter bank has two channels and 2-input, 2-output filters (recall Figure 4.2). However, each of the scalar filters in a symmetric-antisymmetric multifilter has the same kind of symmetry that makes the symmetric biorthogonal scalar wavelets efficient. Also, each filter in a multifilter system processes less data than a filter in a scalar filter bank at the same level (due to the preprocessing discussed in Chapter 5). Table 4.1 lists the computational complexities of both the scalar wavelets and multiwavelets. With all other factors equal, it is apparent that the multiwavelets require roughly twice as much computation. However, multiwavelets still compare favorably because they can give performance comparable to scalar wavelets with shorter filters. For example, as Xia et al. [29] points out, the Bi9/7 scalar wavelet (with $M_1 = 7$ and $M_2 = 9$) requires 4.5 multiplies and 7 additions per input sample. This is slightly greater than the 4 multiplies and 7 additions required per sample by the length-4 multifilters SA4 and ORT4 (both with $M_1 = 4$ and $M_2 = 4$), and yet their performance is comparable in many of the test results in Chapter 6. Also, the Bi22/14 scalar filter from the paper by Wei et al. [27] requires 9 multiplies and 17 additions per input sample. In contrast, the longest multifilter used here, the BSA9/7, requires only 8 multiplies and 14 additions per input sample. As the compression results in Chapter 6 will show, multiwavelets can achieve the same level of performance as scalar wavelets with similar computational complexity.

4.3 Multiwavelet Packets

In practice, (4.5) and (4.6) are used to filter discrete signals using the Mallat algorithm that was presented in Chapter 3. Just as with scalar wavelets, this procedure involves iterating the filtering operation on the lowpass channel of the filter bank. And, just as with scalar wavelets, new basis functions can be produced by iterating on the highpass (bandpass) channels as well. This approach combines the wavelet packet decomposition with multiwavelet filters and hence we call it multiwavelet packets. While the idea seems simple and obvious, this

Table 4.1: Comparison of computational complexities of symmetric wavelets and symmetric-antisymmetric multiwavelets for one level of analysis. M_1 and M_2 are the lowpass and highpass filter lengths and L is the length in samples of the scalar-valued input signal. The derivation of these expressions is given in Appendix B.

Filter Type	Multiplies	Additions
Scalar Wavelet, odd length	$\frac{L\,(M_1+M_2+2)}{4}$	$\frac{L\,(M_1+M_2-2)}{2}$
Scalar Wavelet, even length	$\frac{L\,(M_1+M_2)}{4}$	$\frac{L\,(M_1+M_2-2)}{2}$
Multiwavelet, odd length	$\frac{L\,(M_1+M_2)}{2}$	$L\,(M_1+M_2-2)$
Multiwavelet, even length	$\frac{L\,(M_1+M_2)}{2}$	$L\,(M_1+M_2-1)$

author has not seen any mention of multiwavelet packets in previous literature. We define them in a manner analogous to the wavelet packets of Chapter 3.

Let $U_0(t) \equiv \Phi(t)$ and $U_1(t) \equiv \Psi(t)$, and define

$$U_{2n}(t) \;=\; \sqrt{2} \sum_{k=-\infty}^{\infty} H_k\, U_n(2t-k), \tag{4.11}$$

$$U_{2n+1}(t) \;=\; \sqrt{2} \sum_{k=-\infty}^{\infty} G_k\, U_n(2t-k). \tag{4.12}$$

Note the similarity between (4.11) and (4.12) and the equivalent (3.25) and (3.26) from Chapter 3. In fact, the tree structures representing bases for multiwavelet packets look just like those in Figure 3.4, except that the $u_n(t)$ functions in Figure 3.4c would be replaced by the corresponding vector-valued functions $U_n(t)$ and each line would. For example, the wavelet packet tree in Figure 3.4 has a multiwavelet version that is shown in Figure 4.3.

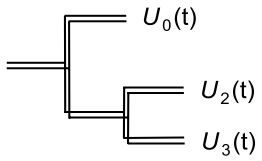

Figure 4.3: Possible multiwavelet packet filter bank. Compare to Figure 4.3(c).

The basis selection algorithms and cost functions used to prune the resulting tree structure are identical to those of the scalar wavelet packet case, with one exception. The difference between wavelet packets and multiwavelet packets is that each branching in the tree structure creates four new channels (assuming $r=2$) instead of just two, due to the dual-channel nature of multiwavelet filter banks. Since the multiwavelet packet tree then has four children for each parent, the computational complexity for multiwavelet packets may be higher than for wavelet packets. Cost function based methods will be essentially unaffected because they just operate on all the pixels corresponding to each node; with multiwavelet packets there are four nodes instead of two, but each node represents half as much data so the net effect is zero. However, methods that perform some form of rate-distortion optimization will require more computation due to the increased number of nodes. With this caveat, the motivation to use multiwavelet packets still holds as it does for scalar wavelets: to better capture high-frequency content and oscillations in the original image data, while retaining the benefits of multiwavelet filters (to be shown in Chapter 6). Since no published literature has yet tested multiwavelet packets, we do so here.

Chapter 5

Implementation

5.1 Image Processing with Wavelet and Multiwavelet Transforms

5.1.1 2-D Algorithms Using 1-D Transforms

The wavelet and multiwavelet transformations presented in Chapters 3 and 4 are directly applicable only to one-dimensional (1-D) signals. But images are two-dimensional (2-D) signals, so we must find a way to process them with a 1-D transform. The two main categories of methods for doing this are separable and non-separable algorithms. Separable methods simply work on each dimension in series. The typical approach is to process each of the rows in order and then process each column of the result. Non-separable methods, such as the factored scalar wavelet method in a paper by Meyer et al. [12], work in both image dimensions at the same time. While non-separable methods can offer benefits over separable methods, such as a savings in computation [12], they are generally more difficult to implement.

5.1.2 Preprocessing for Multiwavelets

Aside from decomposition concerns, there is another issue to be addressed when multiwavelets are used in the transform process. As mentioned in Chapter 4, multiwavelet filter banks require a vector-valued input signal. There are a number of ways to produce such a signal from 2-D image data. Perhaps the most obvious method is to use adjacent rows and columns of the image data; this has already been attempted [21]. However, this approach does not work well for general multiwavelets and leads to reconstruction artifacts in the lowpass data after coefficient quantization [21]. This problem can be avoided by constructing "constrained" multiwavelets, which possess certain key properties. Unfortunately, the extra constraints are somewhat restrictive; image compression tests show that constrained multiwavelets do not perform as well as some other multifilters [19].

Another approach is to first split each row or column into two half-length signals, and then use these two half signals as the channel inputs into the multifilter. A naive approach is to simply take the odd samples for one signal and the even samples for the second signal. As Strela points out [21], this approach doesn't work well because it destroys the assumed characteristics of the input signal. It is generally presumed that image data will be locally well-approximated by low-order polynomials, usually constant, linear, or quadratic. The highpass filters are designed to give a uniformly zero output when the input has this form. Taking alternating data points as the filter inputs alters the character of the input signal; hence the filter output will no longer be forced to zero, reducing compression performance. But there is a way around this problem: one may first *prefilter* the two half-length signals before passing them into the multifilter.

An early form of prefiltering appears in a paper by Xia et al. [30], and it is refined in later papers [21, 20, 24]. The prefilter step adjusts the input signal properties so that one scalar signal is split into two half-length signals in such a way that the orders of approximation built into the multifilter are utilized. The prefiltering is generally performed by taking the two signals as a $2 \times N$ matrix (where the original 1-D signal had length $2N$) and then left-

multiplying by one or more 2×2 prefilter matrices. Note that the earlier methods [30, 21, 20] have some limitations, such as being tied to a specific multifilter or requiring more than one prefilter matrix. Tham et al. [24] present a method that requires only a single orthogonal prefilter matrix for any given multifilter. Additionally, their method also provides some optimization of the prefilter properties to match any given multifilter. When applied to the class of symmetric-antisymmetric (SA) multiwavelets, the method of Tham et al. produces a prefilter matrix with entries of equal magnitude. The authors point out that if the overall constant were absorbed into the multifilter itself, then the preprocessing operation would require no multiplications and only two additions for each input vector. Naturally, there is a matching postfilter operation in the synthesis stage that exactly undoes the effects of the prefilter.

5.1.3 Symmetric Signal Extension

There is one remaining obstacle to overcome before multiwavelets can be competitive with scalar wavelets for image compression. The final issue is symmetric signal extension. It has been shown that symmetric extension is the best way to handle signal boundaries[1]. Of course, symmetric signal extension requires symmetric or antisymmetric filters. This implies using biorthogonal scalar wavelets and SA multiwavelets (either orthogonal or biorthogonal). The method for performing symmetric signal extension for scalar wavelets is well known [18]. One method of symmetric signal extension for multiwavelets has been reported [19], but it only works for the GHM multiwavelet. Only very recently has a more general method been presented that works for the entire class of SA multiwavelets [29]. Perhaps the best feature of this new method is that the prefiltering operation is built into the extension method, reducing the computational complexity of the preprocessing and extension steps to just that of the prefiltering.

[1] Strang and Nguyen give a good illustration of this in Section 10.1 of their book [18].

5.1.4 Iteration of Decomposition

Since multiwavelet decompositions produce two lowpass subbands and two highpass subbands in each dimension, the organization and statistics of multiwavelet subbands differ from the scalar wavelet case. A closer examination of the differences suggests a method for improving the performance of multiwavelets in image compression applications. During a single level of decomposition using a scalar wavelet transform, the 2-D image data is replaced with four blocks corresponding to the subbands representing either lowpass or highpass in both dimensions. These subbands are illustrated in Figure 5.1a. The subband labels in this Figure indicate how the subband data was generated. For example, the data in subband LH was obtained from highpass filtering of the rows and then lowpass filtering of the columns[2]. The multiwavelets used here have two channels, so there will be two sets of scaling coefficients and two sets of wavelet coefficients. Since multiple iterations over the lowpass data are desired, the scaling coefficients for the two channels are stored together. Likewise, the wavelet coefficients for the two channels are also stored together. The multiwavelet decomposition subbands are shown in Figure 5.1b. For multiwavelets, the L and H labels have subscripts denoting the channel to which the data corresponds. For example, the subband labeled L_1H_2 corresponds to data from the second channel highpass filter in the horizontal direction and the first channel lowpass filter in the vertical direction.

This shows how a single level of decomposition is done. In practice, more than one decomposition is performed on the image data. Successive iterations are performed on the lowpass coefficients from the previous stage to further reduce the number of lowpass coefficients. Since the lowpass coefficients contain most of the original signal energy, this iteration process yields better energy compaction. After a certain number of iterations, the benefit gained in energy compaction becomes rather negligible compared to the extra computational effort. Usually five levels of decomposition are used in current wavelet-based compression schemes [15, 33]. Experiments performed for this thesis indicate that three levels are suf-

[2]The ordering convention used here is that of operators, in which subsequent operations are added to the *left* of previous ones.

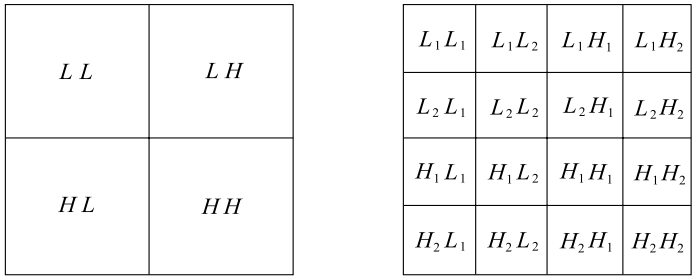

Figure 5.1: Image subbands after a single-level decomposition, for (a) scalar wavelets and (b) multiwavelets.

ficient for multiwavelets, with gains in PSNR diminishingly rapidly[3] with decomposition depth increasing above 3. As will be discussed next, a single level of decomposition with a symmetric-antisymmetric multiwavelet is roughly equivalent to two levels of a wavelet decomposition. Thus a 3-level multiwavelet decomposition effectively corresponds to a 6-level scalar wavelet decomposition. Since tests indicate that the improvement from depth 5 to depth 6 for scalar wavelets is negligible[4], a 3-level multiwavelet decomposition can be considered comparable to a 5-level scalar wavelet decomposition.

Scalar wavelet transforms give a single quarter-sized lowpass subband from the original larger subband, as seen in Figure 5.1a. In previous multiwavelet literature, multi-level decompositions are performed in the same way. The multiwavelet decompositions iterate on the lowpass coefficients from the previous decomposition, as shown in Figure 5.2. In the case of scalar wavelets, the lowpass quarter image is a single subband. But when the multiwavelet transform is used, the quarter image of lowpass coefficients is actually a 2×2 block of subbands (the L_iL_j subbands in Figure 5.1b). Due to the nature of the preprocessing and symmetric extension method, data in these different subbands becomes intermixed during iteration of

[3]For example, the PSNR values for the Man image compressed with the SA4 multiwavelet at 16:1 compression and depths 1, 2, 3, and 4 are, respectively, 31.05 dB, 32.11 dB, 32.13 dB, and 32.13 dB. In this case there was exactly no improvement from the fourth level of decomposition.

[4]The PSNR value for the Lena image compressed with the Bi97 wavelet at 16:1 compression, for example, only increases by 0.01 dB when increasing the decomposition depth from 5 to 6.

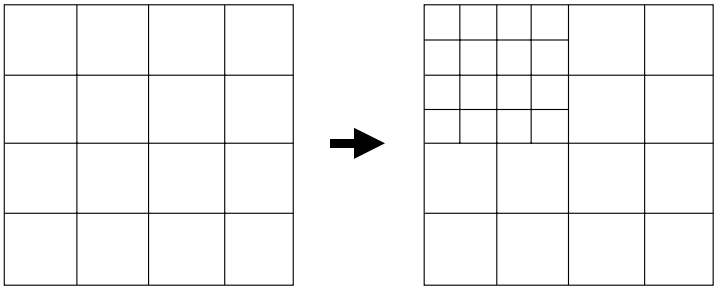

Figure 5.2: Conventional iteration of multiwavelet decomposition.

the multiwavelet transform. The intermixing of the multiwavelet lowpass subbands leads to suboptimal results, as will be discussed further.

Consider the multiwavelet transform coefficients resulting from a single-level decomposition using a symmetric-antisymmetric (SA) multifilter. As an example, the multiwavelet coefficients produced by applying the SA4 multifilter to the 512×512 Lena image are shown in Figure 5.3. It can be readily observed that the 2×2 "lowpass" block (upper left corner) actually contains one lowpass subband and three bandpass subbands. The L_1L_1 subband resembles a smaller version of the original image, which is a typical characteristic of a true lowpass subband. In contrast, the L_1L_2, L_2L_1, and L_2L_2 subbands seem to possess characteristics more like those of highpass subbands. This is most likely due to the fact that the second multiwavelet channel in a SA multifilter corresponds to an antisymmetric wavelet. Hence a smooth signal will typically have small coefficient values for that channel because the antisymmetric filter has small magnitude response at DC (due to having a zero at $z = 1$). Also, only the L_1L_1 subband contains coefficients with a large DC value and a relatively uniform distribution. An illustration of the subband coefficient distribution is given in Figure 5.4. This figure shows the maximum absolute value across each row, in other words as if looking at Figure 5.3 from the left side. The L_1, L_2, H_1, and H_2 subbands, measured along the vertical direction, are the 128-coefficient blocks in order, from left to right (i.e. 0 to 511 on the independent axis). Note that the L_2 subband (rows 128-255) looks more like the

highpass bands H_1 and H_2 (rows 256-511) than the L_1 subband (rows 0-127). Examination of other images yields similar results.

Figure 5.3: Transform coefficients of Lena image after one level of decomposition with the SA4 multifilter. Values near zero are shown in gray, with positive values increasing toward white and negative values decreasing toward black.

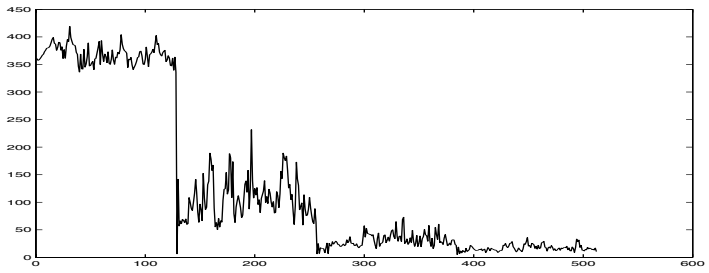

Figure 5.4: Maximum absolute value across each row of 512×512 Lena image after one level of decomposition with SA4 multifilter (see Figure 5.3).

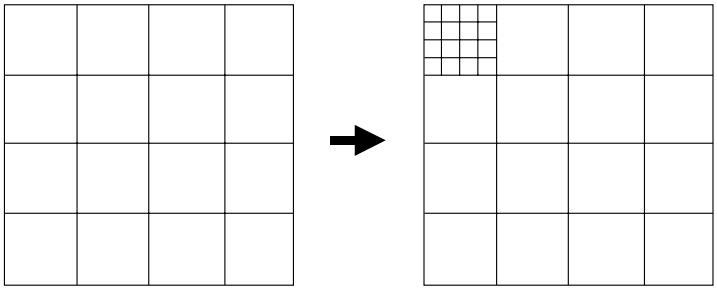

Figure 5.5: Proposed iteration method for multiwavelet decomposition. Compare to Figure 5.2.

A couple of conclusions may be drawn from these observations. First, since these four LL subbands possess different statistical characteristics, mixing them together using the multiwavelet decomposition described previously results in further subbands with mixed data characteristics. This implies that typical quantization schemes that assume the statistics in each subband are either lowpass or highpass will not give the best possible results. Second, since only the L_1L_1 subband actually has lowpass characteristics, we only need to perform further iterations on that one subband. Thus, this thesis proposes to do just this. Results presented in Chapter 6 demonstrate that iterating only on the L_1L_1 subband at each stage in the decomposition does yield better performance than iterating on the entire LL subband. It is also worth noting that iterating only on the L_1L_1 subband requires one quarter of the computational complexity as iteration over the entire LL subband, thus improving run-time performance as well. This new improved multiwavelet decomposition method is illustrated in Figure 5.5.

5.2 Quantization Issues

The quantization method used to generate all the results in this thesis is the SPIHT[5] quantizer developed by Said and Pearlman [15]. It is an embedded coder that refines the ideas presented in Shapiro's embedded zerotree wavelet (EZW) coder [16]. EZW and SPIHT achieve good performance by exploiting the spatial dependencies of pixels in different subbands of a scalar wavelet transform. The SPIHT coder was chosen for the experiments in this thesis due to its good objective and computational performance. To fully understand the results in Chapter 6, it is necessary to better understand how SPIHT works. This section gives an introduction to the operational ideas behind SPIHT and a method for improving its performance for multiwavelet compression methods.

It has been noted [16] that there exists a spatial dependence between pixels in different subbands in the form of a child-parent relationship. In particular, each pixel in a smaller subband has four children in the next larger subband in the form of a 2×2 block of adjacent pixels. This relationship is illustrated in Figure 5.6, which shows a three-level scalar wavelet decomposition and some sample pixel relations. In this figure, each small square represents a pixel and each arrow points from a particular parent pixel to its 2×2 group of children. The importance of the parent-child relation in quantization is this: if the parent coefficient has a small value, then the children will most likely also have small values; conversely, if the parent has a large value, one or more of the children might also.

Coders like SPIHT exploit this spatial dependence by partitioning the pixel values into parent-descendent groups. The coder starts with a threshold value that is the largest integer power of two that does not exceed the largest pixel value. Pixels are evaluated in turn to see if they are larger than the threshold; if not, these pixels are considered *insignificant*. If a parent and all of its descendents are insignificant, then the coder merely records the parent's coordinates. Since the children's coordinates can be inferred from those of the parent, those coordinates are not recorded, resulting in a potentially great savings in the output bitstream.

[5] SPIHT stands for "Set Partitioning in Hierarchical Trees."

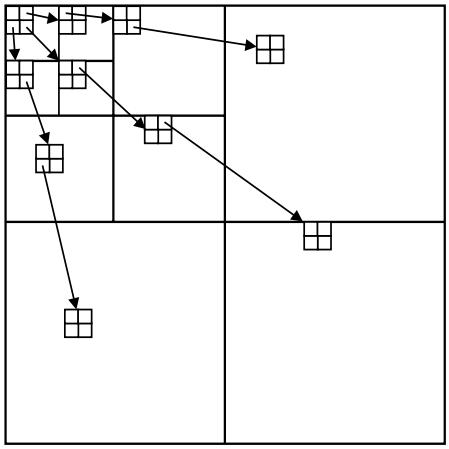

Figure 5.6: Illustration of the parent-child relationship in a 3-level iterated wavelet decomposition.

After locating and recording all the significant pixels for the given threshold, the threshold is reduced by a factor of two and the process repeats. By the end of each stage, all coefficients that have been found to be significant will have their most significant bits (when considered as binary integers) recorded. As further passes occur, more precision is added to the value stored for each pixel. In this manner, the SPIHT algorithm performs a rough sorting of pixel values by magnitude and records their values one bit at a time. It is this separation of bit planes that makes SPIHT an embedded coder: at any point in the output data stream, only the most significant bits for any given pixel are transmitted.

The assumptions that the SPIHT quantizer makes about spatial relations between subbands hold well for scalar wavelets, but they do not hold for multiwavelets. More specifically, the three largest highpass subbands in a scalar wavelet transform are each split into a 2×2 block of smaller subbands by the multiwavelet transform, destroying the parent-child relationship that SPIHT presumes. To work around this limitation, this thesis presents a new quantization method that allows multiwavelet decompositions to receive most of the benefits of using a quantizer like SPIHT. The basic idea is to try to restore the spatial features that

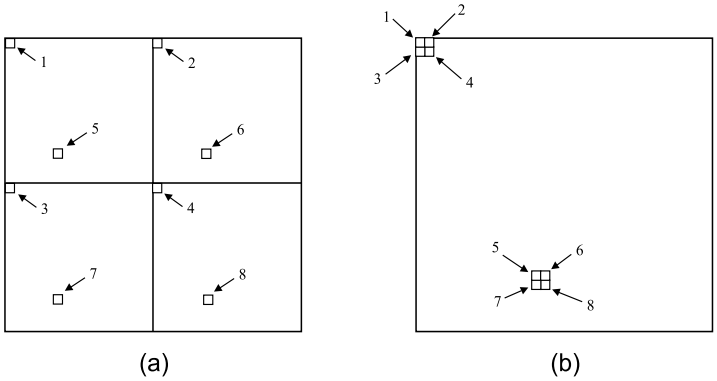

Figure 5.7: Illustration of coefficient shuffling method. Selected pixels are numbered to indicate correspondence. (a) Before shuffling. (b) After shuffling.

SPIHT requires for optimal performance. Examination of the coefficients in a single-level multiwavelet transform reveals that there generally exists a large amount of similarity in each of the 2×2 blocks that compose the L_iH_j, H_iL_j, and H_iH_j subbands, where $i = 1, 2$ and $j = 1, 2$.

This observation suggests the following procedure: rearrange the coefficients in each 2×2 block so that coefficients corresponding to the same spatial locations are placed together. This procedure will be referred to as *shuffling*. A clearer picture of this is given in Figure 5.7. Figure 5.7a shows one of the 2×2 blocks resulting from a multiwavelet decomposition. Eight pixels (two from each subband) are highlighted and given a unique numeric label. Figure 5.7b shows the same set of pixels after shuffling. Note that pixels 1-4 map to a 2×2 set of adjacent pixels, as do pixels 5-8. This shuffling procedure restores some of the spatial dependence of pixels by placing the pixels that correspond to a certain part of the image where they would be if a scalar wavelet decomposition had been performed.

After shuffling coefficients, a 2-level decomposition iterating only on the L_1L_1 block would look like the one shown in Figure 5.8. Note that the subband boundaries, indicated by dotted lines in that figure, have been removed by the shuffling process. The remaining coefficient

data has the same structure as that of a 4-level scalar wavelet decomposition. Experimental results in Chapter 6 show that this new shuffling scheme can greatly improve multiwavelet performance in many cases.

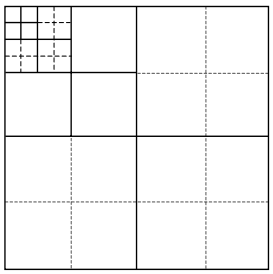

Figure 5.8: Subbands in 2-level multiwavelet decomposition after coefficient shuffling. Solid lines denote new subband boundaries, and dashed lines show subband boundaries that are removed by coefficient shuffling.

5.3 Implementation Details

All wavelet and multiwavelet results in this work were obtained using a separable decomposition of the 2-D image data. For the multiwavelets, the approximation-preserving prefiltering and signal extension approach presented by Xia et al. [29] was used. A block diagram of each level of decomposition, including preprocessing, is shown in Figure 5.9. The 2-D data is processed first in rows, and then columns. The processing of each row or column involves splitting the 1-D signal into even and odd subsets and multiplying the resulting 2×1 vector by the prefilter matrix. The data is then extended symmetrically, filtered, and downsampled. Subsequent iterations for multiwavelets were performed only on the L_1L_1 subband of the previous transform result. Similarly, multiwavelet packets were implemented by iterating only on whole subbands, yielding a "doubly-dyadic" decomposition (i.e., each subband becoming a 4×4 block of smaller subbands). Tests were performed both with and without the use of the coefficient shuffling method. For the wavelet packet and multiwavelet packet

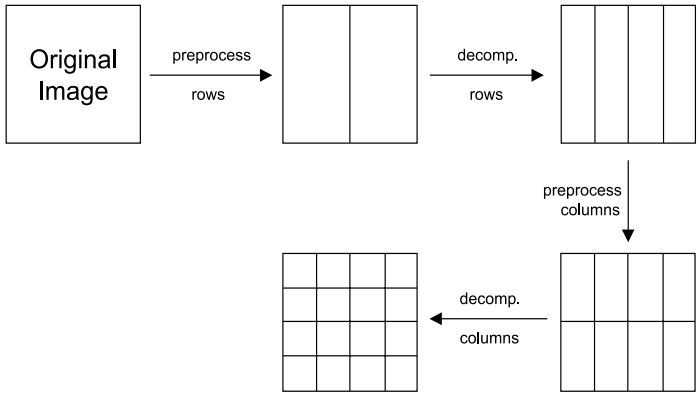

Figure 5.9: Illustration of 2-D multiwavelet filtering with approximation-based preprocessing.

cases, a cost function approach was used (as opposed to a rate-distortion optimized tree search). This choice was made with an eye toward computational complexity, since the rate-distortion searching methods are much more computationally expensive than a simple cost function method [33]. After the transform stage, the coefficients were quantized using the SPIHT coder [15] and written to the output bitstream. No entropy coder was used in these experiments. Since the entropy coding process typically just adds a roughly constant gain in dB to the PSNR [15], entropy coding is not necessary for comparisons of different transform methods and would simply improve all the results in Chapter 6 by a certain amount. The reconstruction method followed the opposite order of steps: read in bitstream, perform inverse SPIHT process to obtain quantized transform coefficients, and perform inverse transform to obtain the reconstructed image.

Chapter 6

Experimental Results

6.1 Preliminary Comments

A relatively small selection of certain test images appears repeatedly in the published literature on image compression. Using only a few standard images has some benefits, such as allowing for direct comparison of results from different compression methods. However, one drawback is that the so-called "standard" images are not necessarily standard. For example, when using a color image with grayscale algorithms, how the color image is converted to grayscale may differ. This means that authors citing results in different papers for the same image may be using slightly different images. Also, using only a few images fails to illustrate more generally how a particular algorithm performs on other image types. For example, some methods work quite well on the popular Lena image while performing poorly on the Barbara image, and vice versa[1]. Lena and Barbara are both examples of "natural" images, on which wavelet-based compression methods are known to work well. Performance of different wavelet and multiwavelet methods changes significantly when "synthetic" test images are used.

[1]The results later in this chapter will show such cases.

The test images used in this work were collected from a variety of sources on the Internet. In addition to the most commonly used natural images, a number of synthetic images were selected to obtain a better overall picture of the performance of the image compression methods described in Chapter 5. The following list of sources indicates the origin of the test images used, which are themselves listed in Table 6.1.

1. MATLAB Image Processing Toolbox
2. Image Compression Lab, UCLA School of Engineering and Applied Sciences
   (`http://www.icsl.ucla.edu/~ipl/psnr_images.html`)
3. Waterloo BragZone
   (`http://links.uwaterloo.ca/bragzone.base.html`)
4. Information Coding Laboratory, UCSD ECE Department
   (`http://www.code.ucsd.edu/~sherwood/image_examples/chan_coded/chan_coded.html`)
5. Home page of François Meyer
   (`http://noodle.med.yale.edu/~meyer/profile.html`)
6. Signal and Image Processing Institute, USC
   (`http://sipi.usc.edu/services/database/Database.html`)
7. UICODER
   (`http://saigon.ece.wisc.edu/~waveweb/QMF/software.html`)

Table 6.1: Listing of test images.

Name	Size	Source[2]	Notes
Barbara	512×512	2	
Barchart	256×256	6	original was named 5.1.13.tiff
Boat	512×512	3	
Finger	512×512	5	
Frog	576×448	3	cropped from 621×498 original
Goldhill	512×512	2	
Gray21	512×512	6	original was named gray21.512.tiff
House	512×512	5	
IC	256×256	1	
Lena	512×512	2	
Lighthouse	512×512	5	
Man	1024×1024	6	original was named 5.3.01.tiff
Mandrill	512×512	4	
Monarch	768×512	3	original was color; converted to grayscale with MATLAB Image Processing Toolbox command RGB2GRAY
Nitf7	512×512	7	
Peppers	512×512	3	
Ruler	512×512	6	original was named ruler.512.tiff
Testpat_1k	1024×1024	6	original was named testpat.1k.tiff
Testpat2	256×256	1	
Yogi	512×512	7	

[2]The numbers in the "Source" column indicate the corresponding entry in the list on the previous page.

Objective results given in the next section are provided in the form of tables of peak signal-to-noise ratio (PSNR) values. Since all tests here are performed on 8-bit grayscale images, the peak signal value is 255. Hence the PSNR values in dB for an $M \times N$ image signal x and its reconstruction $\hat{x}$ are calculated via

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right), \tag{6.1}$$

where the mean square error (MSE) is defined as

$$\text{MSE} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x(m,n) - \hat{x}(m,n)|^2 . \tag{6.2}$$

For each image, a number of wavelet and multiwavelet filters are tested and the compression ratio is varied. Some compression parameters, such as the decomposition depth for a particular type of transform, have fixed values and are mentioned now so that such details can be omitted later. In each table in the next section, the "Type" column specifies the transform type, which has one of the following values:

1. W: wavelet (decomposition depth 5)
2. WP: wavelet packet (maximum decomposition depth 5)
3. MW: multiwavelet (decomposition depth 3)
4. MWP: multiwavelet packet (maximum decomposition depth 3)

In each case, the indicated depth specifies how many iterations of the subband decomposition were performed. For the multiwavelet cases that use the coefficient shuffling scheme presented in Chapter 5, the "MWP" type name is followed by "(sh)". In the wavelet packet and multiwavelet packet cases, the cost function used is specified as a number in parentheses following the type abbreviation. Cost function "1" computes the cost as the number of significant coefficients[3] in the tested node. Cost function "2" computes the cost as the total

[3]In this case, the threshold used for significance testing is simply 0.5, the threshold below which a coefficient will be converted to 0 during integer conversion.

number of bits required in the binary representation of all the coefficients in that node. The Filter column gives the name of the filter used. The third and subsequent columns give the PSNR values in dB for that particular image at various compression/bit rates. The bit rates used here correspond to 8-bit grayscale images, so the number of bits per pixel (bpp) is 8 divided by the compression factor. The test results given in this chapter are all at either 1.0 bpp (8:1), 0.5 bpp (16:1), 0.25 bpp (32:1), or 0.125 bpp (64:1).

Two scalar wavelet filters were used in this work. The first is the popular biorthogonal "Bi9/7" filter [6], which has been used with great success in numerous prior image compression tests [15, 12, 27]. The other scalar filter is the newly presented "Bi22/14" biorthogonal filter [27]. Results show this filter to perform at least as well, or better than, the Bi9/7 and other scalar filters with good image compression performance in many cases [27]. Since best image compression results are obtained when the lowpass synthesis filter is longer than the highpass synthesis filter [18], all biorthogonal filters (and multifilters) used here were implemented according to this rule. The naming convention here is "lowpass synthesis filter length/highpass synthesis filter length". Both orthogonal and biorthogonal multiwavelets were tested, and all are from the class of symmetric-antisymmetric multifilters due to the availability of construction and symmetric extension methods for them. The orthogonal symmetric-antisymmetric multifilters used are "SA4" and "ORT4" [24, 29]. The biorthogonal symmetric-antisymmetric multifilters are "BSA7/5" and "BSA9/7" [8]. The notation for biorthogonal filter lengths used for scalar filters is also applied here, and hence the "7/5" and "9/7" in the filter names refer to the lowpass and highpass synthesis filter lengths.

6.2 Results and Discussion

6.2.1 Preface

A couple of notes apply to the rest of this chapter. First, printed reproductions of the 20 test images used here may be found in Appendix A. Second, for each image and compression rate, the largest PSNR value is highlighted in boldface. This is done for the packet decompositions as well as the non-packet decompositions.

All the multiwavelet results in this chapter were obtained by decomposing only the L_1L_1 subband at each step, as described in Chapter 5. Some justification for this choice is given in Table 6.2. The tests show that the new proposed decomposition that iterates only on the L_1L_1 subband works best at all bit rates for all multifilters and for different types of images. The PSNR values in Table 6.2 were all created without coefficient shuffling. The PSNR values with shuffling (not shown) are comparable and thus still show the improvement for our new decomposition method.

Table 6.2: PSNR results (in dB) comparing multiwavelet decomposition methods. The subband labels are the same as in Figure 5.1.

Image	Multifilter	Comp. ratio	Decomposition iterated on:	
			LL Subband	L_1L_1 Subband
Lena	SA4	16:1	33.50	34.66
	BSA9/7	16:1	33.29	34.96
	SA4	32:1	29.85	31.20
	BSA9/7	32:1	29.88	31.94
Barbara	SA4	16:1	28.82	29.58
	BSA9/7	16:1	28.79	30.25
	SA4	32:1	25.78	26.30
	BSA9/7	32:1	25.60	26.80

6.2.2 Results for Natural Images

6.2.2.1 Lena

We begin analysis of the results with the widely used Lena[4] image. This image is selected from the class of natural images that do not contain large amounts of high-frequency or oscillating patterns. As a result, the standard scalar wavelet methods perform well on it. The PSNR results in Table 6.3 show that the Bi22/14 scalar filter gives the best performance on this image, especially at the higher bit rates. As the bit rate decreases, the BSA9/7 multifilter with shuffling begins to approach the same level, and even surpasses the Bi9/7 filter. Note that the use of coefficient shuffling provided a significant improvement in performance for all multiwavelets. In fact, shuffling brings the multiwavelet performance from well under scalar wavelet performance to being generally on par with the scalar wavelets. It is also interesting to note that in both the scalar and multiwavelet cases, using a packet decomposition *decreased* performance. This shows that the use of packets is not always an improvement, and that the choice of basis is very important when packets are used. In principle, the packet method should produce results at least equal to those of the standard wavelet tree decomposition if an optimized basis searching algorithm is used, such as the ones in the papers by Ramchandran et al. [14, 33]. If a simpler cost function method is used, as in this case, the cost function may not find the best basis and therefore give worse results than the wavelet tree basis[5].

It is well known that PSNR values do not necessarily correspond to perceived image quality at low bit rates. The truth of this statement can be seen in Figures 6.1-6.6. Compared to the original Lena, the reconstructed images at 0.125 bpp using scalar wavelets (Figures 6.2 and 6.3) show a great deal of ringing, in addition to washed out areas. In contrast, the reconstructed images using multiwavelets in Figures 6.4-6.6 show crosshatching and blocking artifacts instead. The reconstruction for SA4 (not shown) is visually almost identical to that for ORT4 in Figure 6.6. While the PSNR values for the Bi22/14 filter (Figure

[4]This image is also sometimes called "Lenna".

[5]The reason for using the cost function approach was given in the last section of Chapter 5.

6.3) and BSA9/7 multifilter (Figure 6.4) are nearly identical, the crosshatching evident in the reconstruction using BSA9/7 makes it appear significantly lower in quality than the Bi22/14 reconstruction. The reconstructions for the SA4 (not shown), ORT4 (Figure 6.6), and BSA7/5 (Figure 6.5) multiwavelets appear worse still with bad blocking artifacts. These blocking artifacts presumably occur for the same reason they occur with the DCT at low bit rate: the short filters do not decay smoothly to zero at the ends. Nonetheless, as the numbers attest, the multiwavelets used here with coefficient shuffling generally perform quite well at high bit rate and have reconstruction quality comparable to the best known scalar wavelets.

Figure 6.1: Original Lena, showing 256 × 256 portion of face.

Figure 6.2: Lena compressed with Bi9/7 wavelet to 0.125 bpp, PSNR=29.44 dB.

Figure 6.3: Lena compressed with Bi22/14 wavelet to 0.125 bpp, PSNR=29.87 dB.

Figure 6.4: Lena compressed with BSA9/7 with multiwavelet with shuffling to 0.125 bpp, PSNR=29.81 dB.

Figure 6.5: Lena compressed with BSA7/5 with multiwavelet with shuffling to 0.125 bpp, PSNR=28.95 dB.

Figure 6.6: Lena compressed with ORT4 with multiwavelet with shuffling to 0.125 bpp, PSNR=29.10 dB.

Table 6.3: PSNR results (in dB) for Lena.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	38.99	35.62	32.31	29.44
W	Bi22/14	**39.23**	**36.06**	**32.79**	**29.87**
MW	SA4	38.31	34.66	31.20	28.34
MW (sh)	SA4	39.06	35.39	31.97	29.08
MW	ORT4	38.35	34.71	31.24	28.38
MW (sh)	ORT4	39.09	35.43	32.01	29.10
MW	BSA9/7	38.06	34.96	31.94	29.30
MW (sh)	BSA9/7	38.49	35.55	32.62	29.81
MW	BSA7/5	38.49	34.83	31.29	28.38
MW (sh)	BSA7/5	39.08	35.43	31.96	28.95
WP (1)	Bi9/7	37.84	34.55	31.58	28.78
WP (2)	Bi9/7	38.51	35.40	32.21	29.35
WP (1)	Bi22/14	37.99	34.93	32.09	29.53
WP (2)	Bi22/14	**38.82**	**35.82**	**32.72**	**29.81**
MWP (1)	SA4	37.82	34.26	30.90	28.13
MWP (2)	SA4	38.22	34.53	30.99	28.15
MWP (1)	ORT4	37.83	34.29	30.94	28.18
MWP (2)	ORT4	38.28	34.57	31.04	28.19
MWP (1)	BSA9/7	37.36	34.39	31.60	29.20
MWP (2)	BSA9/7	37.86	34.65	31.69	29.24
MWP (1)	BSA7/5	38.21	34.65	31.10	28.08
MWP (2)	BSA7/5	38.44	34.68	31.11	28.14

6.2.2.2 Peppers

Another image that has little high-frequency content is Peppers. Not surprisingly, the PSNR values in Table 6.4 show a pattern similar to that for Lena. While the SA4, ORT4, and BSA7/5 multiwavelets match the performance of the scalar wavelets at high bit rate, they do not keep up at low bit rate. Conversely, the BSA9/7 multiwavelet, which gives poor results at high bit rate, outperforms all but the Bi22/14 scalar wavelet at low bit rate. While wavelet packets and multiwavelet packets both give relatively poor results at all bit rates, it is interesting to note that the multiwavelet packet results are very close to the multiwavelet results without shuffling.

6.2.2.3 Monarch

The Monarch image contains proportionally much more low-frequency content than the other images tested in this thesis, although there are a few localized areas of high-frequency content. The scalar wavelets, and to a lesser extent the wavelet packets, give the best performance on this image by a significant margin. Further results given later in this chapter will confirm that scalar wavelets generally outperform multiwavelets for images with very little high-frequency content. At 1.0 bpp scalar wavelets lead multiwavelets by about 2 dB, but this lead decreases as the bit rate decreases. For both the wavelets and multiwavelets, using a packet-based decomposition gave slightly lower results than for the standard decomposition. It is also interesting to note that the coefficient shuffling method dramatically decreased performance for the multiwavelet filters. This performance decrease is most likely due to the fact that the Monarch image has relatively little structure to the high-frequency content, which means that there is little spatial structure for the shuffling to enhance.

Table 6.4: PSNR results (in dB) for Peppers.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	37.09	34.75	31.90	28.89
W	Bi22/14	37.03	**34.82**	**32.10**	**29.10**
MW	SA4	36.58	33.71	30.67	27.54
MW (sh)	SA4	37.14	34.38	31.42	28.35
MW	ORT4	36.60	33.74	30.70	27.57
MW (sh)	ORT4	**37.15**	34.39	31.44	28.37
MW	BSA9/7	36.13	33.71	31.20	28.53
MW (sh)	BSA9/7	36.49	34.21	31.72	28.94
MW	BSA7/5	36.66	33.84	30.86	27.81
MW (sh)	BSA7/5	37.10	34.34	31.30	28.24
WP (1)	Bi9/7	35.98	33.22	30.54	27.99
WP (2)	Bi9/7	36.71	**34.17**	31.37	28.36
WP (1)	Bi22/14	35.92	33.23	30.75	28.05
WP (2)	Bi22/14	**36.75**	34.12	**31.47**	**28.65**
MWP (1)	SA4	36.26	33.40	30.52	27.43
MWP (2)	SA4	36.66	33.70	30.65	27.51
MWP (1)	ORT4	36.20	33.22	30.31	27.28
MWP (2)	ORT4	36.69	33.74	30.68	27.54
MWP (1)	BSA9/7	35.42	33.01	30.59	27.98
MWP (2)	BSA9/7	35.92	33.39	30.96	28.36
MWP (1)	BSA7/5	36.63	33.79	30.82	27.71
MWP (2)	BSA7/5	36.73	33.84	30.86	27.79

Table 6.5: PSNR results (in dB) for Monarch.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	41.55	35.38	30.73	27.38
W	Bi22/14	**41.81**	**35.50**	**31.02**	**27.55**
MW	SA4	39.52	33.93	29.75	26.38
MW (sh)	SA4	38.69	33.52	29.43	26.15
MW	ORT4	39.58	33.97	29.77	26.40
MW (sh)	ORT4	38.73	33.56	29.47	26.17
MW	BSA9/7	39.38	34.14	30.36	27.13
MW (sh)	BSA9/7	38.38	33.55	29.91	26.84
MW	BSA7/5	39.79	34.11	29.92	26.54
MW (sh)	BSA7/5	38.82	33.65	29.55	26.31
WP (1)	Bi9/7	38.74	33.65	29.83	26.87
WP (2)	Bi9/7	41.12	35.20	30.73	27.28
WP (1)	Bi22/14	39.10	34.09	30.17	27.14
WP (2)	Bi22/14	**41.37**	**35.46**	**31.07**	**27.57**
MWP (1)	SA4	38.89	33.47	29.22	25.83
MWP (2)	SA4	39.32	33.75	29.52	26.10
MWP (1)	ORT4	38.92	33.47	29.18	25.78
MWP (2)	ORT4	39.38	33.79	29.54	26.12
MWP (1)	BSA9/7	38.18	33.32	29.73	26.70
MWP (2)	BSA9/7	38.67	33.67	30.17	27.09
MWP (1)	BSA7/5	39.38	33.73	29.28	25.98
MWP (2)	BSA7/5	39.57	33.94	29.63	26.25

6.2.2.4 Barbara

Results for the Barbara image are presented in Table 6.6. Barbara is a popular choice from the class of natural test images that exhibit large amounts of high-frequency and oscillating patterns. It is therefore not surprising that the overall results are somewhat different than for the Lena and Peppers images. Here the wavelet packets show their advantage, with the Bi22/14 wavelet packets giving by far the best results at all bit rates. Not only are the PSNR values large for Bi22/14 wavelet packets, but the visual quality is also superior, with the textured regions better preserved than with other filters. It is interesting to note here that the coefficient shuffling method actually makes the multiwavelet results worse in this case, although the loss in PSNR is generally rather small. This is presumably due to the fact that the shuffling method is intended to help images with a large amount of overall structure. Images like Barbara have relatively little structure, especially in the high-frequency subbands, and thus do not benefit from shuffling. Also, as with Lena and Peppers, the BSA9/7 multifilter generates lower PSNR values at high bit rates than Bi22/14, but closes the gap as the bit rate decreases.

A remarkable feature of multiwavelets is shown in Figures 6.7-6.12. A close-up view of the subject's right leg is shown in Figure 6.7. The same close-up of the leg is then shown with five different choices of wavelet transform at 0.25 bpp in Figures 6.8-6.12. Note that the scalar Bi9/7 (Figure 6.8) and Bi22/14 (Figure 6.9) filters lose much of the textured pattern in the pants and scarf, while the pattern is fairly well preserved when those same filters are used in a wavelet packet decomposition (Figures 6.11 and 6.12). This texture test is a standard comparison used to show the benefits of multiwavelets and has been performed previously [12]. What is interesting here is that the BSA9/7 multiwavelet (Figure 6.10), without shuffling and without a packet decomposition, manages to preserve substantially more of the texture than the Bi9/7 and Bi22/14 scalar wavelets. And this occurs even though the PSNR value for the BSA9/7 result is slightly *lower* than that for the Bi22/14 filter. The scalar wavelet packets still obtain the best results, both in PSNR and visual quality. But

the BSA9/7 multiwavelet gives a result with intermediate quality despite a PSNR value that is at least 1 dB lower than the wavelet packet results. This same phenomenon can also be seen in close-ups of the tablecloth (not shown).

While the use of packets dramatically improved results for the scalar wavelets, there was much less improvement for the multiwavelet packets over the multiwavelets. In fact, multiwavelet packets only showed performance similar to that of the multiwavelets and lacked the substantial improvement that the use of packets gave to the scalar wavelets. The relatively low performance of multiwavelet packets is most likely due to the fact that a multiwavelet packet decomposition deviates significantly from the spatial structure that the SPIHT quantizer assumes, thus reducing performance noticeably. The lack of performance is not necessarily a problem with multiwavelet packets but an expression of the fact that they don't work well with zerotree-based quantizers. The use of a different quantization method that exploits the subband structure of a multiwavelet packet decomposition would presumably give much better performance.

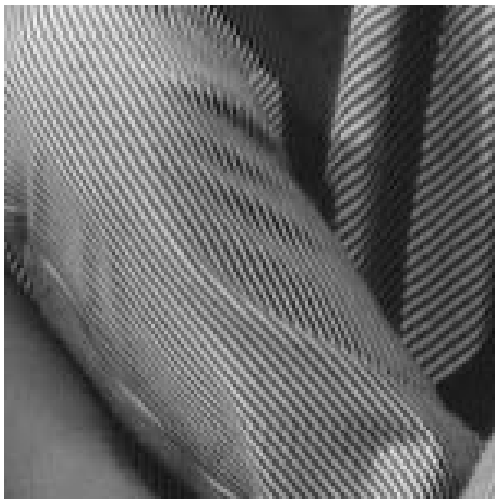

Figure 6.7: Original Barbara, showing a close-up of the leg.

Figure 6.8: Barbara at 0.25 bpp with Bi9/7 wavelet; PSNR=26.35 dB.

Figure 6.9: Barbara at 0.25 bpp with Bi22/14 wavelet; PSNR=26.85 dB.

Figure 6.10: Barbara at 0.25 bpp with BSA9/7 multiwavelet; PSNR=26.80 dB.

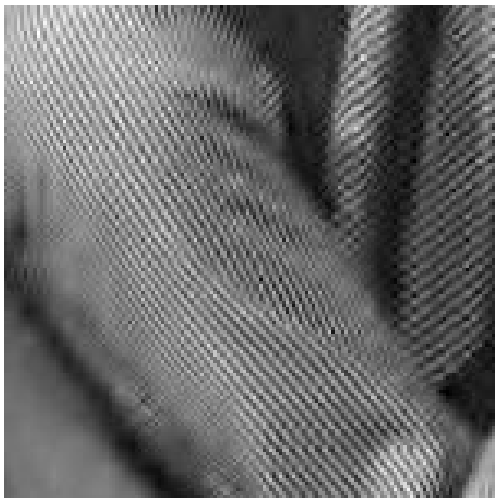

Figure 6.11: Barbara at 0.25 bpp with Bi9/7 wavelet packets; PSNR=27.83 dB.

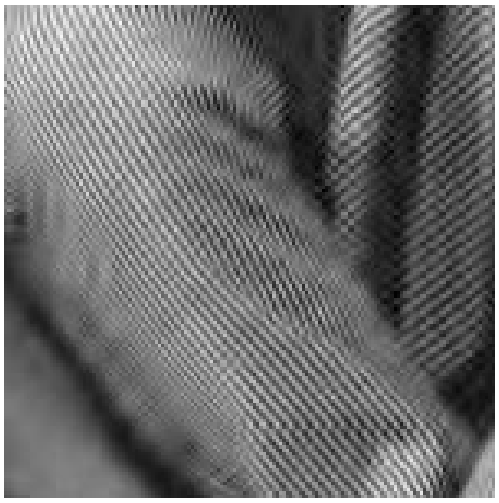

Figure 6.12: Barbara at 0.25 bpp with Bi22/14 wavelet packets; PSNR=28.30 dB.

Table 6.6: PSNR results (in dB) for Barbara.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	34.58	29.74	26.35	23.81
W	Bi22/14	**35.30**	**30.32**	**26.85**	24.00
MW	SA4	34.60	29.58	26.30	23.82
MW (sh)	SA4	34.59	29.50	26.27	23.84
MW	ORT4	34.66	29.64	26.33	23.86
MW (sh)	ORT4	34.65	29.55	26.29	23.83
MW	BSA9/7	34.71	30.25	26.80	**24.31**
MW (sh)	BSA9/7	34.67	30.01	26.60	24.05
MW	BSA7/5	34.92	29.85	26.48	23.85
MW (sh)	BSA7/5	34.91	29.74	26.42	23.74
WP (1)	Bi9/7	35.02	30.67	27.37	24.89
WP (2)	Bi9/7	35.84	31.30	27.83	25.18
WP (1)	Bi22/14	35.71	31.23	27.85	25.21
WP (2)	Bi22/14	**36.42**	**31.84**	**28.30**	**25.50**
MWP (1)	SA4	34.45	29.52	26.48	24.02
MWP (2)	SA4	34.34	29.62	26.75	24.37
MWP (1)	ORT4	34.50	29.59	26.54	24.08
MWP (2)	ORT4	34.45	29.68	26.78	24.41
MWP (1)	BSA9/7	33.67	29.26	26.38	24.35
MWP (2)	BSA9/7	34.61	30.02	26.78	24.24
MWP (1)	BSA7/5	34.92	29.85	26.58	23.87
MWP (2)	BSA7/5	34.69	29.92	26.98	24.54

6.2.2.5 Finger

Like Barbara, the Finger image contains a large amount of texture. Hence, as the results in Table 6.7 show, the packet-based decompositions work best on this image. The best results at all bit rates are given by the scalar wavelet packets. However, in this case the multiwavelet packets also perform well and give the next-best results. As with Barbara, the use of coefficient shuffling decreased performance for the multiwavelets. It would seem that shuffling does not improve compression of images that contain large amounts of high-frequency energy.

The performance of the SA4, ORT4, and BSA7/5 multiwavelets on the Finger image is unimpressive, but the BSA9/7 multiwavelet without shuffling outperforms both scalar wavelets at all bit rates. In fact, while the BSA9/7 multiwavelet performed poorly compared to other multiwavelets in most of the images tested, its performance on this image is comparable to that of the Bi9/7 scalar wavelet packets. Since the BSA9/7 also performed well at capturing the textures in the Barbara image, it seems likely that this particular multifilter works best on images with a large amount of texture and oscillating patterns. The subclass of natural images that contain large textured regions might be a good field of application for the BSA9/7 multiwavelet.

Table 6.7: PSNR results (in dB) for Finger.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	32.73	28.47	24.78	22.36
W	Bi22/14	33.92	29.21	25.53	23.13
MW	SA4	33.26	28.09	24.22	22.10
MW (sh)	SA4	33.02	28.08	24.45	22.06
MW	ORT4	33.38	28.13	24.22	22.12
MW (sh)	ORT4	33.13	28.13	24.47	22.07
MW	BSA9/7	**34.49**	**29.60**	**25.89**	**23.25**
MW (sh)	BSA9/7	34.23	29.48	25.78	23.12
MW	BSA7/5	33.89	28.39	24.21	22.15
MW (sh)	BSA7/5	33.63	28.36	24.50	22.08
WP (1)	Bi9/7	34.42	29.39	25.67	22.90
WP (2)	Bi9/7	34.59	29.64	25.92	23.26
WP (1)	Bi22/14	35.56	30.63	26.78	23.80
WP (2)	Bi22/14	**35.63**	**30.65**	**26.77**	**23.81**
MWP (1)	SA4	33.25	28.08	24.21	22.10
MWP (2)	SA4	34.10	28.99	25.38	22.85
MWP (1)	ORT4	33.36	28.12	24.21	22.11
MWP (2)	ORT4	34.26	29.06	25.41	22.87
MWP (1)	BSA9/7	34.52	29.55	25.95	23.35
MWP (2)	BSA9/7	34.56	29.75	26.26	23.57
MWP (1)	BSA7/5	34.09	28.70	24.83	22.66
MWP (2)	BSA7/5	34.60	29.27	25.54	22.94

6.2.2.6 Goldhill

Goldhill is another popular test image, perhaps the third most commonly used after Lena and Barbara. It contains both regions of locally smooth content and regions of high-frequency textures and sharp transitions. The high-frequency areas make this image more difficult to compress than an image like Lena or Peppers, because wavelet methods work best on low-frequency signals. The reconstruction PSNR results for Goldhill are given in Table 6.8. Perhaps the most obvious result is that the multiwavelets are able to keep pace with the scalar wavelets in PSNR across bit rates. There is no clear overall winner from the multiwavelets, though, since the best performers at high bit rate are not the ones that perform best at the lower bit rates. For example, the best result at 1.0 bpp is for BSA7/5 with shuffling, showing a lead of 0.15 dB over Bi22/14, but at 0.125 it is 0.23 dB behind Bi22/14 and 0.22 dB behind BSA9/7 with shuffling, the best multiwavelet result in this case. In any event, these PSNR variations are fairly small and show that the multiwavelets can perform at the same level as the scalar wavelets. Careful inspection of the reconstructed images verifies this performance similarity, as is it becomes difficult to select a single best image from among the various non-packet filters at each fixed bit rate. It was noted before that the multiwavelets generally performed worse than scalar wavelets for low-frequency images (like Lena) but better for high-frequency images (like Barbara). The results for Goldhill indicate that when both low- and high-frequency elements are combined, the advantages of multiwavelets roughly cancel out the disadvantages, resulting in performance very similar to scalar wavelets.

The use of coefficient shuffling with the multiwavelet transforms improved performance for Goldhill, although the gain it is not as dramatic as it was with Lena. It is also worth noting that while the wavelet and multiwavelet packets don't show any improvement over their standard-decomposition counterparts, they don't experience a large performance deficit either, as they did with Lena. Oddly enough, the Bi22/14 scalar filter performance was improved by using packets at the middle bit rates (0.5 and 0.25 bpp) but was reduced at the highest and lowest bit rates (1.0 and 0.125 bpp). This inconsistency is most likely

attributable to variation in optimality of the basis selected in each case. While the PSNR values tend to be lower for the packet-based methods with this image, visual inspection of the reconstructed images shows that the packet reconstructions are nearly identical to those without packets, although some fine texture details (such as the tile roofing on the houses) are slightly better preserved by using the packets, regardless of the PSNR.

6.2.2.7 Boat

Like Goldhill, the Boat image contains significant amounts of both low and high-frequency regions, hence the PSNR results are very similar. The multiwavelets with shuffling slightly out-performed the scalar wavelets. The scalar wavelet packets gave the best results by a slight margin, while the multiwavelet packets results were worse than the multiwavelet results without shuffling.

Table 6.8: PSNR results (in dB) for Goldhill.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	35.11	31.78	29.33	27.60
W	Bi22/14	35.20	31.86	29.34	**27.74**
MW	SA4	35.19	31.73	29.08	27.30
MW (sh)	SA4	35.30	31.89	29.34	27.54
MW	ORT4	35.20	31.75	29.10	27.32
MW (sh)	ORT4	35.31	31.89	29.35	27.55
MW	BSA9/7	35.00	31.78	29.27	27.61
MW (sh)	BSA9/7	35.03	31.83	**29.46**	27.73
MW	BSA7/5	35.28	31.82	29.15	27.29
MW (sh)	BSA7/5	**35.35**	**31.90**	29.35	27.51
WP (1)	Bi9/7	34.74	31.60	29.26	27.37
WP (2)	Bi9/7	35.07	31.95	29.49	27.57
WP (1)	Bi22/14	34.89	31.78	29.40	27.61
WP (2)	Bi22/14	35.17	**32.01**	**29.52**	**27.75**
MWP (1)	SA4	34.88	31.58	29.02	27.13
MWP (2)	SA4	35.06	31.73	29.09	27.18
MWP (1)	ORT4	34.75	31.46	29.02	27.16
MWP (2)	ORT4	35.05	31.75	29.10	27.18
MWP (1)	BSA9/7	34.13	31.25	28.99	27.16
MWP (2)	BSA9/7	34.84	31.65	29.25	27.58
MWP (1)	BSA7/5	34.94	31.64	29.11	27.21
MWP (2)	BSA7/5	**35.23**	31.82	29.16	27.27

Table 6.9: PSNR results (in dB) for Boat.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	37.16	32.67	29.27	26.71
W	Bi22/14	37.35	32.87	29.37	26.72
MW	SA4	36.80	32.46	28.88	25.97
MW (sh)	SA4	37.53	32.96	29.45	26.74
MW	ORT4	36.85	32.49	28.90	25.98
MW (sh)	ORT4	**37.56**	**33.00**	29.47	26.75
MW	BSA9/7	36.53	32.44	29.06	26.41
MW (sh)	BSA9/7	37.01	32.83	**29.48**	**26.88**
MW	BSA7/5	36.97	32.59	29.06	26.15
MW (sh)	BSA7/5	37.52	32.97	29.45	26.62
WP (1)	Bi9/7	36.62	32.49	29.22	26.64
WP (2)	Bi9/7	37.33	32.91	29.50	26.81
WP (1)	Bi22/14	36.88	32.60	29.21	26.73
WP (2)	Bi22/14	**37.48**	**33.11**	**29.56**	**26.90**
MWP (1)	SA4	36.39	32.24	28.84	25.96
MWP (2)	SA4	36.81	32.45	28.88	25.96
MWP (1)	ORT4	36.49	32.27	28.86	25.96
MWP (2)	ORT4	36.85	32.49	28.90	25.98
MWP (1)	BSA9/7	35.83	32.04	28.84	26.29
MWP (2)	BSA9/7	36.21	32.19	28.88	26.36
MWP (1)	BSA7/5	36.85	32.54	29.05	26.12
MWP (2)	BSA7/5	36.88	32.55	29.05	26.14

6.2.2.8 Lighthouse

Another good example of an image with mixed smooth and high-frequency regions is Lighthouse. The results for Lighthouse, shown in Figure 6.10, are quite interesting. As with Lena and Goldhill, the use of coefficient shuffling improves the performance of the multiwavelets. In fact, it enables all the multiwavelets tested to achieve *better* objective results than the scalar wavelets at all bit rates. Due to the amount of high-frequency content in Lighthouse in the fence, the use of wavelet packets raises the PSNR values significantly. But the reconstructions with higher PSNR values don't necessarily look better.

Consider Figures 6.13-6.20, which show a close-up of the fence and binoculars. The scalar Bi9/7 (Figure 6.14) and Bi22/14 (Figure 6.15) wavelets lose a lot of the fence texture and produce quite a bit of ringing around the binoculars. Also, the sign on the fence is very blurred, to the point of almost being unidentifiable and indistinguishable from the fence itself. The SA4 (Figure 6.16) and ORT4 (Figure 6.17) multiwavelet results, which are nearly identical, seem much clearer. They capture more of the fence texture and exhibit less of the blurring on and around the binoculars. While the BSA9/7 results (Figure 6.18) are rather similar to the scalar wavelet results, BSA7/5 (Figure 6.19) combines the good features of the SA4 and ORT4 reconstructions with less blocking artifacts. And despite having the best PSNR value, the Bi22/14 wavelet packet reconstruction (Figure 6.20) is rather poor. It captures the fence texture better than many others, but blurs most other objects, including the sign and binoculars, and produces the worst ringing of the bunch. The best-looking reconstruction, in this author's opinion goes to the BSA7/5 multiwavelet with shuffling.

Table 6.10: PSNR results (in dB) for Lighthouse.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	32.46	29.24	26.57	24.08
W	Bi22/14	32.46	29.28	26.72	24.24
MW	SA4	32.58	29.20	26.62	24.13
MW (sh)	SA4	32.92	**29.48**	**26.84**	24.33
MW	ORT4	32.60	29.22	26.63	24.14
MW (sh)	ORT4	**32.93**	**29.48**	26.83	24.34
MW	BSA9/7	32.34	29.03	26.60	24.25
MW (sh)	BSA9/7	32.47	29.29	26.76	**24.39**
MW	BSA7/5	32.67	29.21	26.65	24.20
MW (sh)	BSA7/5	32.88	**29.48**	**26.84**	24.37
WP (1)	Bi9/7	32.15	29.13	26.96	24.86
WP (2)	Bi9/7	32.66	29.48	27.20	25.02
WP (1)	Bi22/14	32.13	29.07	26.89	24.87
WP (2)	Bi22/14	**33.07**	**29.66**	**27.29**	**25.10**
MWP (1)	SA4	32.10	28.84	26.55	24.48
MWP (2)	SA4	32.51	29.27	26.78	24.50
MWP (1)	ORT4	32.08	28.81	26.53	24.41
MWP (2)	ORT4	32.53	29.28	26.79	24.51
MWP (1)	BSA9/7	31.85	28.62	26.41	24.20
MWP (2)	BSA9/7	32.30	29.10	26.70	24.28
MWP (1)	BSA7/5	32.29	28.89	26.53	24.27
MWP (2)	BSA7/5	32.56	29.21	26.80	24.54

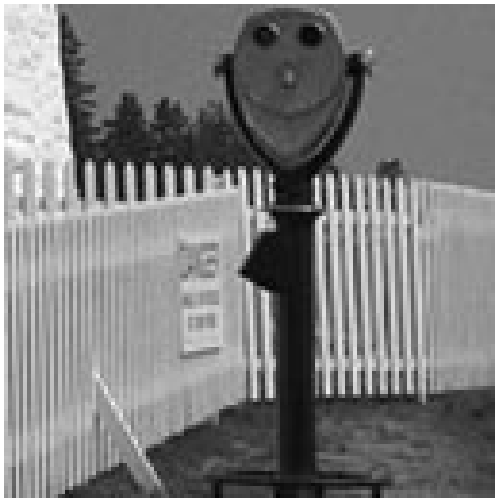

Figure 6.13: Original Lighthouse, showing a close-up of the fence and binoculars.

Figure 6.14: Lighthouse compressed with Bi9/7 wavelet to 0.25 bpp at PSNR=26.57 dB.

Figure 6.15: Lighthouse compressed with Bi22/14 wavelet to 0.25 bpp at PSNR=26.72 dB.

Figure 6.16: Lighthouse compressed with SA4 multiwavelet with shuffling to 0.25 bpp at PSNR=26.84 dB.

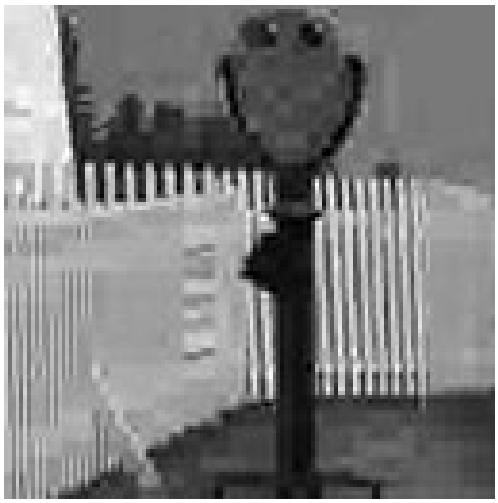

Figure 6.17: Lighthouse compressed with ORT4 multiwavelet with shuffling to 0.25 bpp at PSNR=26.83 dB.

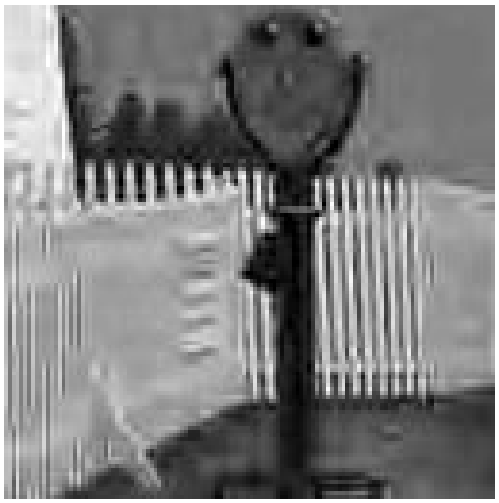

Figure 6.18: Lighthouse compressed with BSA9/7 multiwavelet with shuffling to 0.25 bpp at PSNR=26.76 dB.

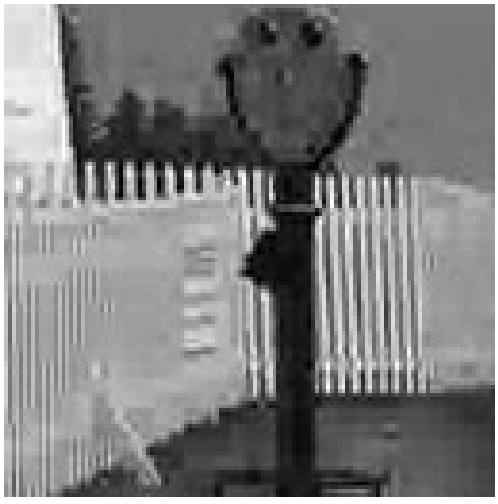

Figure 6.19: Lighthouse compressed with BSA7/5 multiwavelet with shuffling to 0.25 bpp at PSNR=26.84 dB.

Figure 6.20: Lighthouse compressed with Bi22/14 wavelet packets to 0.25 bpp at PSNR=27.29 dB.

6.2.2.9 House

House is another example of an image that is difficult to compress well due to the large amount of high-frequency information it contains. Objective results for the House image are given in Figure 6.11. In this case, the multiwavelets typically produce PSNR values comparable to, or better than, that of the scalar wavelets. In particular, the multiwavelets with shuffling consistently achieve the best PSNR values at high bit rate. Without shuffling, the multiwavelet results are somewhat lower, but they are still competitive at moderate and high bit rates.

It is interesting to note that the wavelet packets and multiwavelet packets perform reasonably well. In particular, the scalar wavelet packet PSNR values are slightly better than scalar wavelets using the standard decomposition. Also note that the multiwavelet packet scores using the second cost function are very close to the those of the multiwavelets without shuffling. As with most of the other images tested, the PSNR values for wavelet packet and multiwavelet packet decompositions produced with the first cost function are generally low.

Figures 6.21-6.28 show a detail of the shutters on the right side of the House image. The horizontal slats in the shutters are completely lost by the scalar wavelets, but all the multiwavelets capture them to some extent. Of course, the wavelet packet decomposition preserves this detail even better. As was seen with the Barbara image, multiwavelets using a standard decomposition can capture some texture details that normally require a packet-based decomposition. This is important to note because a 3-level multiwavelet transform is much less computationally expensive than a typical 5-level scalar wavelet packet transform.

Table 6.11: PSNR results (in dB) for House.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	29.19	25.02	22.01	20.08
W	Bi22/14	29.29	25.00	22.17	20.12
MW	SA4	29.33	25.14	22.00	19.80
MW (sh)	SA4	**29.64**	25.38	22.23	20.15
MW	ORT4	29.34	25.15	22.00	19.81
MW (sh)	ORT4	**29.64**	**25.39**	22.24	20.16
MW	BSA9/7	29.22	25.05	22.05	20.02
MW (sh)	BSA9/7	29.35	25.20	22.22	**20.25**
MW	BSA7/5	29.42	25.21	22.11	19.89
MW (sh)	BSA7/5	29.59	25.37	**22.28**	20.15
WP (1)	Bi9/7	28.66	24.96	22.19	20.06
WP (2)	Bi9/7	**29.59**	**25.49**	**22.40**	**20.27**
WP (1)	Bi22/14	28.92	25.22	22.31	20.11
WP (2)	Bi22/14	29.50	25.42	22.28	20.13
MWP (1)	SA4	29.18	25.17	22.08	19.77
MWP (2)	SA4	29.30	25.14	22.00	19.79
MWP (1)	ORT4	29.30	25.14	21.99	19.78
MWP (2)	ORT4	29.34	25.15	22.00	19.81
MWP (1)	BSA9/7	28.72	24.88	21.98	19.90
MWP (2)	BSA9/7	29.22	25.05	22.05	20.02
MWP (1)	BSA7/5	29.09	25.12	22.18	19.78
MWP (2)	BSA7/5	29.42	25.21	22.11	19.89

Figure 6.21: Original House.

Figure 6.22: House at 16:1 compression using Bi9/7 filter, with PSNR=25.02 dB.

Figure 6.23: House at 16:1 compression using Bi22/14 filter, PSNR=25.00 dB.

Figure 6.24: House at 16:1 compression using SA4 multifilter with shuffling, with PSNR=25.38 dB.

Figure 6.25: House at 16:1 compression using ORT4 multifilter with shuffling, PSNR=25.39 dB.

Figure 6.26: House at 16:1 compression using BSA9/7 multifilter with shuffling, PSNR=25.20 dB.

Figure 6.27: House at 16:1 compression using BSA7/5 multifilter with shuffling, PSNR=25.37 dB.

Figure 6.28: House at 16:1 compression using Bi9/7 filter with packets, PSNR=25.49 dB.

6.2.2.10 Mandrill

The Mandrill image is another popular image that is quite difficult to compress well. The texture of the fur produces a large amount of high-frequency content spread over most of the image. Consequently, the reconstructed images start to show artifacts even at high bit rates. The ability of multiwavelets to capture high-frequency detail better than scalar wavelets is visible in the PSNR results in Table 6.12. The multiwavelets with shuffling typically perform at least as well as the scalar wavelets. PSNR values for the packet-based methods are lower, although the multiwavelet packets now perform as well as, or better than, the scalar wavelet packets. In fact, the multiwavelet packets using the second cost function produce results almost identical to those of the multiwavelets without shuffling. This level of performance is below that of the multiwavelets with shuffling, but not by much.

6.2.2.11 Frog

The results for the Frog image are quite interesting. As Table 6.13 shows, the PSNR values at each bit rate are quite uniform for all transforms. Also interesting is the fact that coefficient shuffling lowers the PSNR for the multiwavelet transforms, although the difference becomes negligibly small at low bit rates.

Table 6.12: PSNR results (in dB) for Mandrill.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	27.86	24.43	22.30	21.16
W	Bi22/14	28.07	24.61	22.35	21.21
MW	SA4	27.84	24.40	22.24	21.04
MW (sh)	SA4	28.15	24.61	22.47	21.16
MW	ORT4	27.86	24.41	22.25	21.05
MW (sh)	ORT4	**28.17**	**24.63**	**22.48**	21.16
MW	BSA9/7	27.69	24.41	22.22	21.16
MW (sh)	BSA9/7	27.97	24.55	22.42	**21.23**
MW	BSA7/5	27.91	24.49	22.23	21.14
MW (sh)	BSA7/5	**28.17**	24.62	22.46	21.20
WP (1)	Bi9/7	26.97	23.98	21.96	20.95
WP (2)	Bi9/7	27.53	24.37	22.23	**21.16**
WP (1)	Bi22/14	27.08	24.08	22.06	20.97
WP (2)	Bi22/14	27.73	**24.45**	**22.28**	21.14
MWP (1)	SA4	27.21	24.06	21.96	20.94
MWP (2)	SA4	27.76	24.36	22.21	21.04
MWP (1)	ORT4	27.24	24.06	21.96	20.95
MWP (2)	ORT4	27.73	24.32	22.20	21.04
MWP (1)	BSA9/7	27.27	23.58	21.73	20.75
MWP (2)	BSA9/7	27.66	24.41	22.22	**21.16**
MWP (1)	BSA7/5	27.30	24.13	22.03	21.02
MWP (2)	BSA7/5	**27.84**	24.44	22.21	21.14

Table 6.13: PSNR results (in dB) for Frog.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	27.52	25.72	24.66	23.46
W	Bi22/14	27.57	25.64	**24.70**	**23.50**
MW	SA4	27.56	25.68	24.53	23.19
MW (sh)	SA4	27.44	25.61	24.48	23.19
MW	ORT4	27.56	25.69	24.53	23.20
MW (sh)	ORT4	27.45	25.62	24.48	23.19
MW	BSA9/7	27.17	25.58	24.61	23.37
MW (sh)	BSA9/7	27.09	25.52	24.53	23.32
MW	BSA7/5	**27.59**	**25.77**	24.58	23.26
MW (sh)	BSA7/5	27.47	25.68	24.53	23.22
WP (1)	Bi9/7	27.03	25.30	24.22	23.15
WP (2)	Bi9/7	27.20	25.46	24.44	23.28
WP (1)	Bi22/14	27.11	25.29	24.25	23.14
WP (2)	Bi22/14	27.30	25.50	24.48	23.34
MWP (1)	SA4	27.20	25.23	24.10	22.98
MWP (2)	SA4	27.43	25.56	24.43	23.14
MWP (1)	ORT4	27.22	25.27	24.13	22.88
MWP (2)	ORT4	27.44	25.57	24.44	23.14
MWP (1)	BSA9/7	26.26	24.87	23.78	22.97
MWP (2)	BSA9/7	27.10	25.50	**24.57**	**23.36**
MWP (1)	BSA7/5	27.28	25.36	24.16	22.95
MWP (2)	BSA7/5	**27.53**	**25.71**	24.56	23.25

6.2.2.12 Man

Like Mandrill, the image titled Man contains a large amount of non-repetitive high-frequency content. The results in Table 6.14 show that objective measures of performance for wavelet and multiwavelet filters are quite close. Visual inspection of the reconstructed images reveals that the images are nearly indistinguishable from each other at 0.25 bpp; the exception is for multiwavelets without coefficient shuffling. The use of shuffling provides significant gains for all multifilters in this case, and as with Lena, raises PSNR values from below those of scalar wavelets to being fully competitive. At low bit rate, there is still little evidence to prefer scalar wavelets or multiwavelets, since it results in a trading off the bluriness of Bi9/7, Bi22/14, and BSA9/7 for the blockiness of SA4, ORT4, and BSA7/5.

Wavelet and multiwavelet packets show slightly worse performance on the Man image. As in previous examples, this is presumably due to poor basis selection. Packet-based methods tend to work best on textured images, like Barbara and Lighthouse, where they can isolate high-frequency patterns. The high-frequency components of the Man image contain very little pattern and hence lose the advantage of packets.

6.2.2.13 Nitf7

The final example from the class of natural images is Nitf7. This image contains a large number of fine lines, sharp edges, and other high-frequency elements. The PSNR results in Table 6.15 indicate a slight performance advantage for the multiwavelets but uniformly low PSNR values at each bit rate. Since much of the high-frequency content of this image is non-repetitive, the packet-based decompositions do not show any advantage and in fact trail the objective results of the scalar wavelets and multiwavelets. As usual, the use of coefficient shuffling improves multiwavelet performance significantly.

Table 6.14: PSNR results (in dB) for Man.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	35.88	32.60	29.74	27.37
W	Bi22/14	35.99	**32.78**	29.87	27.63
MW	SA4	35.60	32.13	29.16	26.77
MW (sh)	SA4	36.02	32.61	29.64	27.28
MW	ORT4	35.64	32.16	29.19	26.79
MW (sh)	ORT4	36.04	32.64	29.66	27.30
MW	BSA9/7	35.38	32.36	29.60	27.30
MW (sh)	BSA9/7	35.68	32.73	**30.01**	**27.68**
MW	BSA7/5	35.76	32.33	29.31	26.88
MW (sh)	BSA7/5	**36.09**	32.71	29.67	27.30
WP (1)	Bi9/7	35.27	32.08	29.35	27.06
WP (2)	Bi9/7	35.81	32.63	29.80	27.41
WP (1)	Bi22/14	35.38	32.26	29.54	27.29
WP (2)	Bi22/14	**35.91**	**32.76**	**29.96**	**27.52**
MWP (1)	SA4	35.37	32.00	29.12	26.76
MWP (2)	SA4	35.56	32.11	29.16	26.77
MWP (1)	ORT4	35.43	32.04	29.15	26.79
MWP (2)	ORT4	35.60	32.14	29.18	26.79
MWP (1)	BSA9/7	34.93	31.95	29.37	27.18
MWP (2)	BSA9/7	35.27	32.21	29.54	27.29
MWP (1)	BSA7/5	35.52	32.22	29.28	26.87
MWP (2)	BSA7/5	35.71	32.31	29.30	26.87

Table 6.15: PSNR results (in dB) for Nitf7.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	24.50	21.44	19.31	17.58
W	Bi22/14	24.38	21.27	19.26	17.53
MW	SA4	23.87	20.82	18.92	17.19
MW (sh)	SA4	24.62	21.44	19.24	17.45
MW	ORT4	23.90	20.85	18.94	17.20
MW (sh)	ORT4	**24.64**	21.46	19.27	17.46
MW	BSA9/7	23.95	21.08	19.16	17.38
MW (sh)	BSA9/7	24.42	**21.55**	**19.45**	**17.59**
MW	BSA7/5	24.00	20.94	19.02	17.32
MW (sh)	BSA7/5	24.59	21.46	19.29	17.52
WP (1)	Bi9/7	23.31	20.57	18.61	17.16
WP (2)	Bi9/7	**24.02**	**21.17**	19.13	**17.54**
WP (1)	Bi22/14	23.22	20.54	18.71	17.21
WP (2)	Bi22/14	23.98	21.11	19.05	17.46
MWP (1)	SA4	23.08	20.22	18.49	16.94
MWP (2)	SA4	23.87	20.82	18.91	17.19
MWP (1)	ORT4	23.17	20.42	18.69	17.05
MWP (2)	ORT4	23.90	20.85	18.94	17.20
MWP (1)	BSA9/7	22.92	20.43	18.50	17.13
MWP (2)	BSA9/7	23.95	21.08	**19.16**	17.38
MWP (1)	BSA7/5	23.37	20.43	18.47	17.04
MWP (2)	BSA7/5	24.00	20.94	19.02	17.32

6.2.3 Results for Synthetic Images

6.2.3.1 Gray21

Now we examine the performance of multiwavelets for the class of synthetic images. We start with Gray21, a simple array of solid blocks that form a gradient with twenty-one shades of gray. The PSNR results for Gray21 in Figure 6.16 show that the multiwavelets seriously outperform the scalar wavelets, especially when the SA4 and ORT4 multifilters are used. In contrast, the BSA9/7 multifilter gives the worst results at all bit rates. The scalar wavelet packets and multiwavelet packets performed poorly, most likely due to a poor choice of basis. This is not surprising since packet-based decompositions work best on images with oscillating patterns, of which Gray21 has none.

The results for Gray21 confirm other authors' results in which multiwavelets performed very well on a geometric test pattern [19], even to the point of one multifilter giving lossless reconstruction at 1.0 bpp. As the results for 0.5 bpp show in Table 6.16, the SA4, ORT4, and BSA7/5 multifilters were able to reconstruct the original image perfectly, regardless of whether shuffling was used. This suggests that the multiwavelets are able to capture high-frequency detail, including the sharp transitions of Gray21, better than scalar wavelets.

6.2.3.2 Testpat2

Very similar results occur for the next test image, Testpat2. Testpat2 is a geometric set of rectangles of decreasing sizes. The objective results shown in Figure 6.16 indicate that multiwavelets again outperformed scalar wavelets, except for the BSA9/7 multiwavelet. In particular, reconstruction error was completely absent at 1.0 bpp for the SA4, ORT4, and BSA7/5 multifilters with the standard decomposition, and also for the SA4 and BSA7/5 multifilters with a multiwavelet packet decomposition.

Table 6.16: PSNR results (in dB) for Gray21. A PSNR value of ∞ means that the MSE is exactly zero.

Type	Filter	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	88.17	59.04	49.45
W	Bi22/14	86.19	57.98	48.61
MW	SA4	∞	67.11	52.44
MW (sh)	SA4	∞	68.01	**53.54**
MW	ORT4	∞	68.43	52.39
MW (sh)	ORT4	∞	**69.63**	53.16
MW	BSA9/7	54.62	47.63	45.21
MW (sh)	BSA9/7	48.61	47.21	44.32
MW	BSA7/5	∞	63.26	50.73
MW (sh)	BSA7/5	∞	63.30	50.75
WP (1)	Bi9/7	72.64	59.41	49.66
WP (2)	Bi9/7	71.34	58.32	49.56
WP (1)	Bi22/14	71.57	57.01	49.41
WP (2)	Bi22/14	70.31	56.11	49.42
MWP (1)	SA4	**77.50**	65.56	52.19
MWP (2)	SA4	76.90	64.01	**52.27**
MWP (1)	ORT4	77.39	**65.77**	52.01
MWP (2)	ORT4	76.63	63.94	52.11
MWP (1)	BSA9/7	68.28	55.84	48.71
MWP (2)	BSA9/7	68.31	56.92	49.56
MWP (1)	BSA7/5	77.12	62.99	50.75
MWP (2)	BSA7/5	73.92	60.16	50.98

Table 6.17: PSNR results (in dB) for Testpat2. A PSNR value of ∞ means that the MSE is exactly zero.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	81.52	67.35	62.45	59.44
W	Bi22/14	71.55	66.57	61.65	58.79
MW	SA4	∞	70.12	63.23	60.00
MW (sh)	SA4	∞	70.06	62.66	59.40
MW	ORT4	∞	70.43	**63.28**	**60.02**
MW (sh)	ORT4	∞	70.38	62.66	59.41
MW	BSA9/7	71.55	54.67	46.63	45.51
MW (sh)	BSA9/7	48.77	45.94	45.25	45.06
MW	BSA7/5	∞	**71.73**	63.20	59.79
MW (sh)	BSA7/5	∞	71.24	62.37	58.97
WP (1)	Bi9/7	81.11	75.12	66.05	61.27
WP (2)	Bi9/7	79.67	75.16	66.00	61.43
WP (1)	Bi22/14	73.42	69.19	65.04	60.44
WP (2)	Bi22/14	73.44	69.28	65.13	60.38
MWP (1)	SA4	∞	78.23	70.51	63.42
MWP (2)	SA4	∞	78.44	70.60	63.42
MWP (1)	ORT4	96.30	77.97	70.66	**63.46**
MWP (2)	ORT4	96.30	77.54	70.81	**63.46**
MWP (1)	BSA9/7	87.84	79.86	70.52	62.32
MWP (2)	BSA9/7	87.84	80.73	70.98	62.69
MWP (1)	BSA7/5	∞	**82.32**	70.67	62.86
MWP (2)	BSA7/5	∞	81.82	**71.12**	62.86

6.2.3.3 Ruler

Ruler is an example of an image that is difficult to compress because it contains a large amount of repeated high-frequency patterns. For this same reason, it is best compressed using a packet-based decomposition. The PSNR values in Table 6.18 confirm the improvement using packets, with the Bi9/7 scalar wavelet packets giving the best results at high bit rate. At lower bit rate, the multiwavelet packets give the best performance. After packet-based decompositions, the SA4 and ORT4 multiwavelets give the best PSNR values. The Bi22/14 scalar filter, which does very well on smooth images, gives some of the worst results for this image. Note that in this case, the packets show their strength most at the lower bit rates; while the best packet-based results only lead by the best non-packet-based results by about 2.5 dB at 1.0 bpp, this difference widens to about 10 dB at 0.25 bpp. This result reinforces the idea that standard wavelet decompositions perform poorly at low bit rates on images with large amounts of high-frequency oscillation.

6.2.3.4 Barchart

The next synthetic test image used is Barchart, the results for which are shown in Table 6.19. As with the Gray21 image, the multiwavelets outperform the scalar wavelets at all bit rates. It is interesting to note that the use of coefficient shuffling dramatically improves the PSNR values for the multiwavelets, especially at higher bit rates, with nearly a 3.0 dB improvement for SA4 and ORT4 at 1.0 bpp. While the packet decompositions give lower PSNR values at high bit rates, the multiwavelet packets give results comparable to those of the multiwavelets at the lowest bit rates.

Table 6.18: PSNR results (in dB) for Ruler.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp
W	Bi9/7	26.09	17.07	13.69
W	Bi22/14	23.83	18.64	12.19
MW	SA4	26.64	18.02	13.45
MW (sh)	SA4	**28.13**	18.03	13.32
MW	ORT4	26.60	18.02	13.46
MW (sh)	ORT4	28.06	18.02	13.33
MW	BSA9/7	26.62	**19.39**	**15.03**
MW (sh)	BSA9/7	25.55	16.92	11.97
MW	BSA7/5	26.31	18.29	13.26
MW (sh)	BSA7/5	26.70	18.64	13.20
WP (1)	Bi9/7	**30.68**	**26.69**	23.46
WP (2)	Bi9/7	**30.68**	**26.69**	23.46
WP (1)	Bi22/14	28.16	24.92	22.72
WP (2)	Bi22/14	28.18	24.92	22.73
MWP (1)	SA4	29.98	25.74	**23.78**
MWP (2)	SA4	29.98	25.74	**23.78**
MWP (1)	ORT4	30.01	25.75	23.77
MWP (2)	ORT4	30.01	25.75	23.77
MWP (1)	BSA9/7	28.27	25.26	22.90
MWP (2)	BSA9/7	28.49	25.37	23.03
MWP (1)	BSA7/5	29.99	26.06	23.77
MWP (2)	BSA7/5	29.99	26.06	23.77

Table 6.19: PSNR results (in dB) for Barchart.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	35.03	27.45	22.18	18.82
W	Bi22/14	34.96	27.16	22.50	18.92
MW	SA4	34.40	27.46	22.31	19.03
MW (sh)	SA4	**37.35**	**28.88**	**23.25**	19.32
MW	ORT4	34.36	27.47	22.37	19.05
MW (sh)	ORT4	37.17	28.85	23.24	19.33
MW	BSA9/7	33.19	26.94	22.50	19.06
MW (sh)	BSA9/7	34.40	27.50	22.63	19.15
MW	BSA7/5	34.32	27.39	22.36	19.34
MW (sh)	BSA7/5	36.08	28.43	23.01	**19.43**
WP (1)	Bi9/7	33.96	27.09	22.16	18.78
WP (2)	Bi9/7	**35.03**	27.45	22.18	18.81
WP (1)	Bi22/14	34.95	27.15	**22.49**	18.91
WP (2)	Bi22/14	34.93	27.12	22.47	18.90
MWP (1)	SA4	34.40	27.45	22.30	19.01
MWP (2)	SA4	34.40	27.45	22.30	19.01
MWP (1)	ORT4	34.36	**27.46**	22.33	19.03
MWP (2)	ORT4	34.36	**27.46**	22.33	19.03
MWP (1)	BSA9/7	32.27	25.86	21.81	18.78
MWP (2)	BSA9/7	32.27	26.14	22.09	18.98
MWP (1)	BSA7/5	34.32	27.38	22.34	**19.32**
MWP (2)	BSA7/5	34.32	27.38	22.34	**19.32**

6.2.3.5 Testpat_1k

The Testpat_1k image is very interesting. This image is mostly a combination of synthetic elements, such as gradients and checkerboard and line patterns, but also has a small 256×256 version of Lena in the center. The results for Testpat_1k are shown in Table 6.20. As with the other synthetic images, the SA4, ORT4, and BSA7/5 multifilters substantially outperform the scalar filters and the BSA9/7 multifilter. Another interesting point is that the use of coefficient shuffling gives somewhat mixed results here. For example, BSA9/7 and BSA7/5 perform consistently better without shuffling. At 1.0 bpp, shuffling improves the PSNR values for SA4 and ORT4 by more than 1 dB, yet at lower bit rates shuffling performance by a similar amount. Even so, for the cases in which shuffling decreases performance, the multiwavelets generally still outperform the scalar wavelets.

6.2.3.6 IC

An example of an image that is somewhat between natural and synthetic is IC. Unlike Testpat_1k, which is a composite of both natural and synthetic images in different parts of the image, IC combines features of both image types simultaneously. The PSNR numbers in Table 6.21 match the visual quality of the reconstructed images. Figures 6.29-6.34 illustrate the results at 0.5 bpp. It can be observed that the multiwavelet methods, which achieve higher PSNR values, generally produce less ringing around edges and sharper transitions. As with the other synthetic images, the sole exception is the BSA9/7 multifilter, which generally seems to perform poorly on such images.

Table 6.20: PSNR results (in dB) for Testpatk_1k.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	46.81	36.25	27.25	22.04
W	Bi22/14	48.47	36.67	30.04	24.12
MW	SA4	52.51	**41.38**	**32.88**	27.14
MW (sh)	SA4	53.68	40.15	31.80	26.22
MW	ORT4	52.25	41.15	32.83	**27.20**
MW (sh)	ORT4	**53.89**	39.70	31.96	26.33
MW	BSA9/7	45.54	35.70	29.18	25.72
MW (sh)	BSA9/7	45.09	35.15	28.05	23.46
MW	BSA7/5	53.05	41.20	32.03	26.98
MW (sh)	BSA7/5	51.83	38.61	31.15	25.39
WP (1)	Bi9/7	45.86	35.95	29.05	24.42
WP (2)	Bi9/7	46.16	36.57	30.45	26.14
WP (1)	Bi22/14	47.28	35.99	29.89	24.06
WP (2)	Bi22/14	47.05	37.40	30.80	26.68
MWP (1)	SA4	52.51	**41.38**	32.88	27.14
MWP (2)	SA4	50.45	40.86	**33.08**	28.19
MWP (1)	ORT4	52.25	41.15	32.83	27.20
MWP (2)	ORT4	50.31	40.45	**33.08**	**28.21**
MWP (1)	BSA9/7	46.04	35.42	29.20	25.63
MWP (2)	BSA9/7	45.82	35.64	29.40	25.74
MWP (1)	BSA7/5	**53.05**	41.20	32.03	26.98
MWP (2)	BSA7/5	50.92	40.90	32.66	27.78

Table 6.21: PSNR results (in dB) for IC.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	35.68	30.38	25.85	22.45
W	Bi22/14	35.82	30.62	26.03	22.28
MW	SA4	35.70	30.94	26.10	21.87
MW (sh)	SA4	36.38	31.90	**27.03**	22.86
MW	ORT4	35.71	30.97	26.15	21.91
MW (sh)	ORT4	**36.39**	**31.91**	**27.03**	**22.89**
MW	BSA9/7	35.06	30.30	26.07	22.33
MW (sh)	BSA9/7	35.39	30.70	26.45	22.72
MW	BSA7/5	35.72	30.96	26.38	21.91
MW (sh)	BSA7/5	36.25	31.50	26.92	22.58
WP (1)	Bi9/7	34.01	29.82	25.83	22.21
WP (2)	Bi9/7	35.09	30.55	26.10	22.29
WP (1)	Bi22/14	34.28	29.91	26.07	**22.36**
WP (2)	Bi22/14	35.01	30.43	25.94	21.80
MWP (1)	SA4	34.64	29.97	25.55	21.61
MWP (2)	SA4	35.47	30.89	26.01	21.63
MWP (1)	ORT4	34.65	29.99	25.62	21.67
MWP (2)	ORT4	35.48	30.93	26.06	21.72
MWP (1)	BSA9/7	33.31	29.49	25.91	22.08
MWP (2)	BSA9/7	34.86	30.28	26.07	22.25
MWP (1)	BSA7/5	34.72	29.78	25.74	22.20
MWP (2)	BSA7/5	**35.72**	**30.95**	**26.37**	21.89

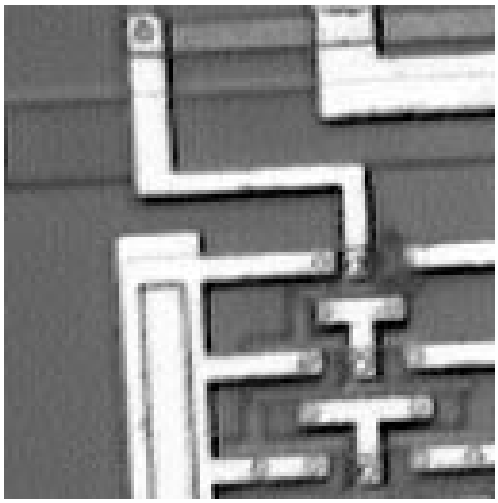

Figure 6.29: 128×128 portion of original IC image.

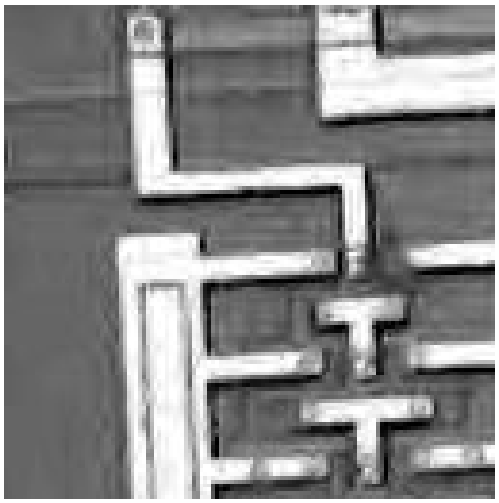

Figure 6.30: IC at 0.5 bpp using Bi9/7 scalar filter, PSNR=30.38 dB.

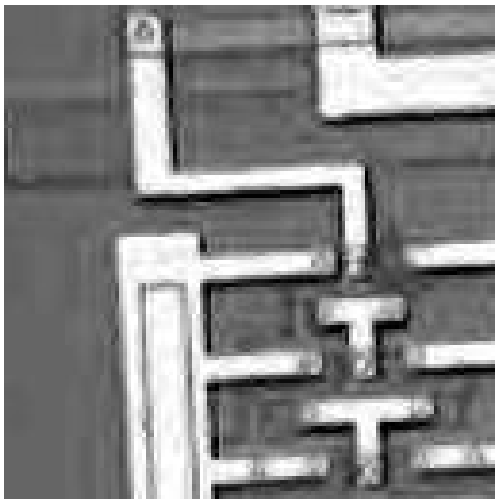

Figure 6.31: IC at 0.5 bpp using Bi22/14 scalar filter with shuffling, PSNR=30.62 dB.

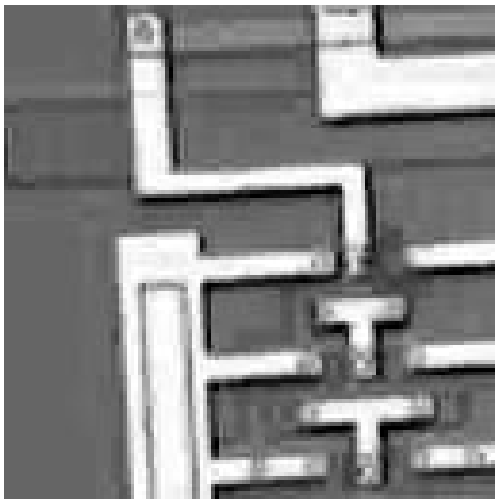

Figure 6.32: IC at 0.5 bpp using SA4 multifilter with shuffling, PSNR=31.90 dB.

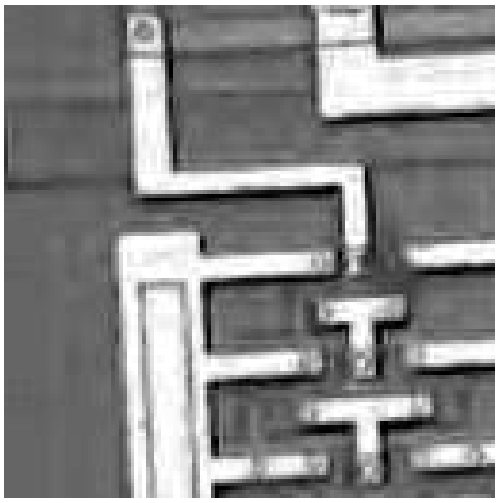

Figure 6.33: IC at 0.5 bpp using BSA9/7 multifilter with shuffling, PSNR=30.70 dB.

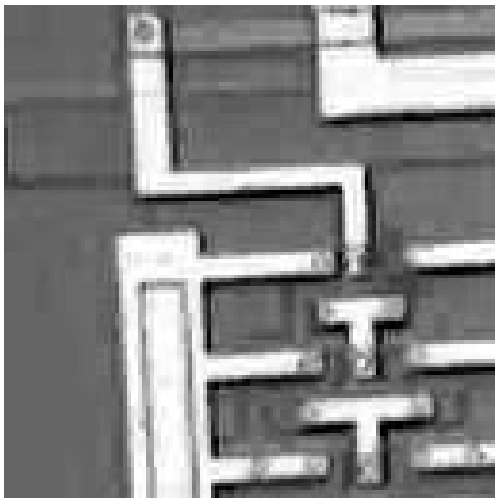

Figure 6.34: IC at 0.5 bpp using BSA7/5 multifilter with shuffling, PSNR=31.50 dB.

6.2.3.7 Yogi

The final test image, Yogi, is similar to IC in the sense that it has features of both natural and synthetic images. Multiwavelets with shuffling produce the best objective results, as shown in Table 6.22. It is interesting to note that in this case the multiwavelets perform poorly without shuffling, yielding results as much as 5 dB less than without shuffling and as much as 3 dB below the scalar wavelet results. Inspection of the reconstructions shows that SA4, ORT4, and BSA7/5 show much less ringing than other filters, which no doubt contributes to their better PSNR values. Due to the lack of high-frequency details and the subsequent poor basis selection, wavelet packets and multiwavelet packets perform poorly.

Table 6.22: PSNR results (in dB) for Yogi.

Type	Filter	1.000 bpp	0.500 bpp	0.250 bpp	0.125 bpp
W	Bi9/7	38.67	29.84	24.84	21.91
W	Bi22/14	38.43	29.73	25.00	22.09
MW	SA4	35.54	28.77	24.13	21.24
MW (sh)	SA4	**40.49**	**31.27**	**25.66**	22.38
MW	ORT4	35.55	28.76	24.15	21.26
MW (sh)	ORT4	40.42	31.19	25.65	**22.39**
MW	BSA9/7	34.97	28.24	24.06	21.50
MW (sh)	BSA9/7	37.94	29.80	25.01	22.17
MW	BSA7/5	35.48	28.68	24.13	21.26
MW (sh)	BSA7/5	39.17	30.35	25.12	22.18
WP (1)	Bi9/7	**38.67**	**29.84**	24.83	21.91
WP (2)	Bi9/7	**38.67**	**29.84**	24.83	21.91
WP (1)	Bi22/14	38.43	29.73	**25.00**	**22.09**
WP (2)	Bi22/14	38.43	29.73	**25.00**	**22.09**
MWP (1)	SA4	35.54	28.77	24.13	21.23
MWP (2)	SA4	35.54	28.77	24.13	21.23
MWP (1)	ORT4	35.55	28.76	24.15	21.25
MWP (2)	ORT4	35.55	28.76	24.15	21.25
MWP (1)	BSA9/7	33.36	27.59	23.74	21.31
MWP (2)	BSA9/7	34.97	28.24	24.05	21.50
MWP (1)	BSA7/5	35.48	28.68	24.13	21.26
MWP (2)	BSA7/5	35.48	28.68	24.13	21.26

Chapter 7

Conclusion

7.1 Summary of Results

7.1.1 Remarks about Multiwavelets

A number of conclusions regarding the image compression performance of the wavelet and multiwavelet filters tested may be made based upon the results in Chapter 6. First, the performance of multiwavelets in general depends greatly on the image characteristics. For images with mostly low-frequency content (like Lena or Monarch), scalar wavelets generally give better performance. However, multiwavelets appear to excel at preserving high-frequency content. In particular, multiwavelets better capture the sharp edges and geometric patterns that occur in the synthetic images like Gray21 and Testpat_1k. For images that contain both low- and high-frequency areas, as do many natural images, multiwavelets with shuffling generally give performance that is competitive to scalar wavelets. For some cases, such as Barbara and Lighthouse, multiwavelets give slightly better visual quality with very similar PSNR values. In fact, tests on images that contain large textured regions (like Barbara and Finger) demonstrate that multiwavelets can attain some of the benefits of wavelet packets, by preserving high-frequency patterns that are lost by scalar wavelets with a non-

packet decomposition. High-frequency content that is spread over a large image region or which exhibits oscillations (as in the Barbara image) is currently best preserved with wavelet packets in general, but multiwavelet packets perform moderately well in some cases.

Addressing individual wavelets, the SA4, ORT4, and BSA7/5 multiwavelets tend to perform best on synthetic images. In particular, images with only sharp transitions, such as Gray21 and Testpat2, are best compressed with either of the SA4 and ORT4 multiwavelets. It is interesting to note that the orthogonal multiwavelets, SA4 and ORT4, show nearly identical performance in most situations. The BSA7/5 multiwavelet performed best on "mixed" images, like Peppers, Goldhill, and Mandrill. Like the Bi9/7 and Bi22/14 scalar wavelets, the BSA9/7 multiwavelet performed best on natural images. However, while the Bi9/7 and Bi22/14 scalar wavelets perform best on smooth images like Lena and Monarch, BSA9/7 performs better on images like Finger and Nitf7 which have a large amount of structure (and hence some high-frequency patterns) in the image.

Evaluating performance over all the test images, it is difficult to select any particular wavelet or multiwavelet over any of the others. However, the Bi22/14 wavelet showed the most consistently good results on natural images, performing best on the smoother images while still giving respectable results on the less-smooth natural images. Also, the Bi22/14 wavelet packets were nearly always the best of the packet-based methods on natural images. For synthetic images, the SA4 and ORT4 multiwavelets gave the most consistently good results, especially for the images that are more synthetic. There was no clear best filter for packet-based decompositions on the synthetic images. Hopefully new multifilters constructed in the future will show more consistent image compression performance.

In general, the shorter multifilters work well for synthetic images, while the longer multifilters work best for natural images. This relation between filter length and performance is the same for scalar wavelets. Shorter filters tend to capture high-frequency detail better because they do not decay smoothly to zero at the ends, and hence better match the local discontinuities of high-frequency image data. On the other hand, longer filters generally decay more rapidly

to zero at the ends and thus better match locally smooth regions in low-frequency image data.

For all types of images, the PSNR advantage demonstrated by each one of the transform methods decreased with the reconstruction bit rate. This is a natural and expected result, since at low bit rate the coding gain of the transform is overpowered by insufficient bits for representation. If too few bits are used to represent an image, the reconstruction quality will be poor regardless of the transform used. At high bit rate, enough bits are available to allow the different transforms to show their strengths (and weaknesses).

7.1.2 Remarks about Multiwavelet Packets

One of the new contributions of this thesis is the implementation and testing of multiwavelet packets. The results from Chapter 6 show mixed results for multiwavelet packets. For most natural images, the Bi22/14 scalar wavelet packets with our new cost function "2" (which counts the number of bits required to represent the coefficients of each node) produced the best results of all packet-based transforms. In fact, for highly textured images like Barbara and Finger, the Bi22/14 wavelet packet results were the best achieved at each bit rate of any of the tested transforms; in those cases the multiwavelet packets did not perform well. However, on some images with both high-frequency and low-frequency content, the multiwavelet packet performance was equal to, if not better than, scalar wavelet packets. Examples of this include Mandrill, Frog, and Nitf7. On many natural images with mixed content, such as Goldhill, Lighthouse, House, and Man, wavelet packets outperformed multiwavelet packets, but the difference in PSNR was often rather small (0.3 dB or less).

Relative to wavelet packets, multiwavelet packets performed better on synthetic images. For Testpat2, the multiwavelet packet results are the best overall at most bit rates and uniformly surpass the scalar wavelet packet results. This image is one of the few examples in which multiwavelet packets give better results than multiwavelets without the packet decomposition. The ruler image had similar results; scalar wavelet packets gave the best

results at most bit rates, but the multiwavelet packets gave consistently much better results than the multiwavelets. In other synthetic images, like Barchart, Testpat_1k, and IC, multiwavelet packets gave the best packet-based decomposition performance at some bit rates, but not all. For those images, the multiwavelet packets sometimes performed better than non-packet multiwavelets, but not always. Taken together, these results show that multiwavelet packets have the potential to give performance improvements for images which have any significant amount of high-frequency textures, just as with scalar wavelet packets. However, the multiwavelet packets do not realize their potential in these tests, presumably due to the quantization method used. Recall that multiwavelets (and therefore also multiwavelet packets) have a spatial decomposition structure which does not match the assumptions of a zerotree quantization method. While the coefficient shuffling method presented in this thesis can improve multiwavelet performance with the SPIHT quantizer by trying to restore the spatial structure assumed by SPIHT, no such method exists to improve multiwavelet packet performance with SPIHT. Hence another quantization method would be better suited for use with multiwavelet packets. An optimized uniform scalar quantization method, such as SFQ [33], might give good results with multiwavelet packets.

7.1.3 Remarks about New Decomposition Methods

Two new methods for improving the multiwavelet transform have been proposed in this thesis: a new multiwavelet decomposition that iterates only on the L_1L_1 subband, and a coefficient shuffling method to improve performance with zerotree-based quantizers. Both methods have been shown to improve the performance of multiwavelet image compression in many cases. The improved decomposition iteration gives uniformly better results and was therefore used to generate the results for all the test images. However, the performance gains of shuffling depend upon the image content to a large extent. In particular, shuffling helps most for images with more low-frequency content. For images with more high-frequency content, shuffling typically has no significant effect on performance, and in some cases de-

creases performance. Decreases in performance as a result of shuffling are presumably due to unstructured high-frequency content in images. Shuffling tries to group together pixels corresponding to the same spatial locations in the image. Images like Barbara, which lack a strong structure in the bandpass subbands due to large amounts of high-frequency content, will therefore not see any benefit from shuffling. However performance losses due to shuffling are usually very small and often occur in cases in which the multiwavelets outperform scalar wavelets regardless of shuffling. An example of this is the Finger image, for which shuffling coefficients tended to lower the multiwavelet PSNR results by up to nearly 0.3 dB; even though the shuffled result for the BSA9/7 multifilter is lower than the unshuffled result at all bit rates, the shuffled result is still at least as good as the Bi22/14 scalar wavelet result at all bit rates. In contrast, when shuffling improves performance, the improvement is often quite significant and gives the multiwavelets performance equal to or better than that of scalar wavelets. Hence, while the type of image being compressed has a significant bearing on whether shuffling will be beneficial, it is probably safe to use the shuffling method in general.

7.1.4 Concluding Remarks

It should be pointed out that the scalar wavelets used here represent the best-known filters published after years of study. In contrast, the multifilters used here are still quite new, and many have only been discovered in the past year or so. It seems very likely that new multifilters will be published in the future which give even better performance than those in this thesis. Even so, the multiwavelets used in this thesis give performance equal to the best scalar wavelets in many cases. While the Bi22/14 scalar wavelet gives consistently good performance for natural images, in most cases a multiwavelet can be selected which gives similar performance with lower computational complexity. This makes multiwavelets a viable alternative to the conventional scalar wavelets in many situations. Also, the multi-channel nature of multiwavelets could be exploited to perform multifiltering on each channel in par-

allel on suitable hardware. The ability to perform fast parallel multiwavelet transforms in silicon is important in video applications, because the computational complexity for wavelet transforms have been a disadvantage to date compared to DCT-based algorithms [31]. As better multifilter construction techniques, and hence better multifilters, are developed, multiwavelets could in fact displace scalar wavelets as the transform of choice for many areas of image and video compression.

7.2 Future Work

The multiwavelet techniques presented in this thesis produce some of the best-reported results to date for multiwavelet-based image compression compared to wavelet-based methods. Nonetheless, there is always room for improvement. The following list includes a number of possible future topics for study. Some of the topics involve improvements to multiwavelet decomposition methods; other topics include more advanced applications of multiwavelets.

1. Since multiwavelets are a relatively new subject of study, only a few construction methods have been published previously. It seems likely that future construction methods might yield multiwavelet filters with better image compression properties. While the latest published methods can construct SA multiwavelets with desirable magnitude response characteristics, most current filters have few orders of approximation. For example, the SA4 multiwavelet has only one approximation order, while the Bi9/7 scalar wavelet has four. The biorthogonal construction method of Goh et al. [8] only guarantees a single order of approximation for the BSA multifilters. Future construction methods that add higher orders of approximation while preserving the good features of the current methods could give multifilters that perform better for image compression.

2. It is important to note that all compression results in this thesis use the zerotree-based SPIHT quantization method. Since zerotree methods are designed to exploit the spatial properties of wavelet decompositions and multiwavelet decompositions have different

spatial properties, zerotree methods do not work as well for multiwavelet transforms. While the shuffling method helps standard multiwavelet decompositions work better with zerotree-based quantizers, multiwavelet packets are still at a disadvantage. A quantization method that exploits the spatial properties of multiwavelet packet decompositions could dramatically improve compression performance. A logical choice for such a quantization method is an optimized scalar quantizer [14, 33]. Also, the SQF method [32] applied to multiwavelets could give better results than the shuffling method and SPIHT as used here.

3. While the computational complexity of multiwavelets is not significantly higher than that of scalar wavelets, the development of ways to further reduce computation would be helpful. Methods for reducing the computation rate include factoring the multifilter into a cascade of shorter filters (as Meyer et al. do for scalar wavelets [12]), implementation of the multifilter via the lifting scheme[1], and construction of multifilters which possess repeated filter coefficients (in addition to the usual symmetry of the SA multifilters). An additional possibility is to apply factored multiwavelets to non-separable 2-D multifiltering, as has already been done for scalar wavelets [12].

4. The algorithms presented and employed in this thesis are designed to work only on grayscale images. Based on the good performance demonstrated in Chapter 6, multiwavelet-based compression of color images should perform well compared to methods based on wavelet and DCT transforms. An extension to the compression of color images could be performed in at least two ways. Since most digital color image devices use RGB color space to represent image data, the multiwavelet methods used here could be applied individually to each color plane. However, a better method would be to transform the RGB color data to a luminance/chrominance color space (such as YCbCr or YUV). The human eye is more sensitive to high-frequency information in the luminance values (corresponding to the "grayscale" data of the image). Thus the color

[1] Use of the lifting method could also result in multifiltering methods which can be performed "in place".

information in the chrominance planes could be quantized more severely, resulting in excellent compression rates with good visual quality. This is the method used by the current imaging standards, such as JPEG.

5. Multiwavelets have demonstrated considerable success in still image compression, with results comparable to wavelet-based compression methods. It would be a natural extension to attempt multiwavelet-based video compression. One multiwavelet-based video codec has already been proposed by Tham et al. [23]. However, the authors of that paper use the standard method for iteration of multiwavelet decompositions and a zerotree-based quantizer. The new proposed decomposition method and coefficient shuffling method presented in this thesis might yield better results than the multiwavelet transform method proposed by Tham et al. Also, the authors of that paper suggest using a Haar or 4-tap Daubechies filter to perform compression in the time dimension (i.e. filtering each pixel at a fixed image location across successive image frames). One of the short multiwavelet filters, such as SA4 or ORT4, might give better image quality results if higher computational complexity is allowed.

6. Good results have been presented in previous attempts [21, 19, 20] to apply multiwavelets to the denoising of 1-D and 2-D signals. Combined with the success shown here for multiwavelet image compression, it seems likely that multiwavelets may work well for compression of noisy images. It should be noted that the Mandrill image used in this thesis contains some significantly noisy regions, and multiwavelets gave the best compression results in that case. This may be due to the ability of multiwavelets to better isolate high-frequency content, which is how noise may appear in an image. The performance of multiwavelets for the compression of noisy images remains a worthwhile subject to be studied.

Appendix A

Test Images

The test images used in Chapter 6 are displayed here for reference. Electronic versions may be found at the locations given in the first section of that chapter.

Figure A.1: Lena.

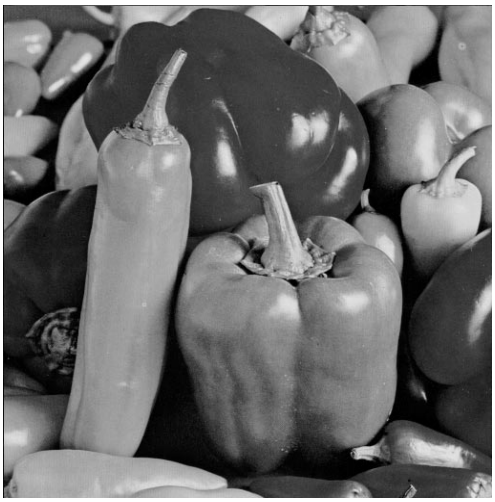

Figure A.2: Peppers.

Figure A.3: Monarch.

Figure A.4: Barbara.

Figure A.5: Finger.

Figure A.6: Goldhill.

Figure A.7: Boat.

Figure A.8: Lighthouse.

Figure A.9: House.

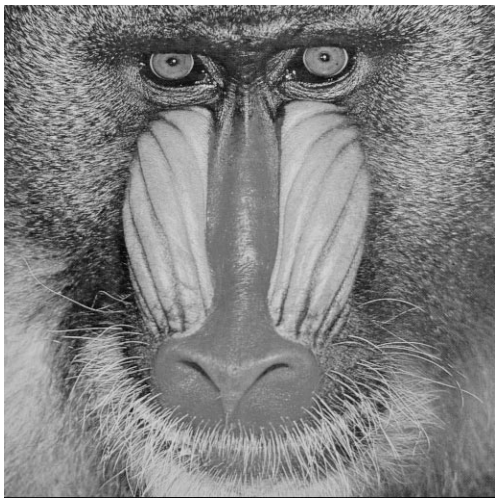

Figure A.10: Mandrill.

Figure A.11: Man.

Figure A.12: Frog.

Figure A.13: Nitf7.

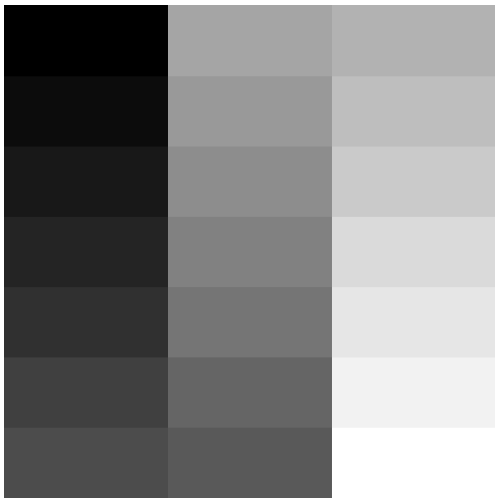

Figure A.14: Gray21.

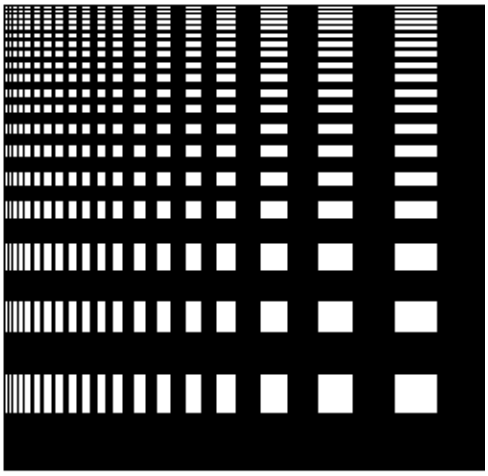

Figure A.15: Testpat2.

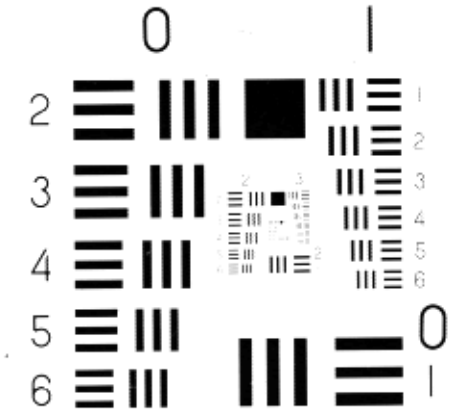

Figure A.16: Barchart.

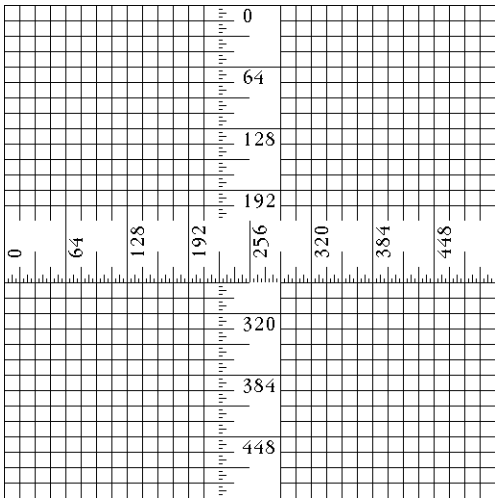

Figure A.17: Ruler.

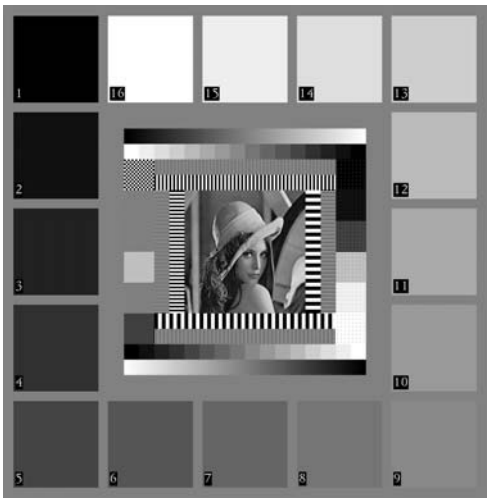

Figure A.18: Testpat_1k.

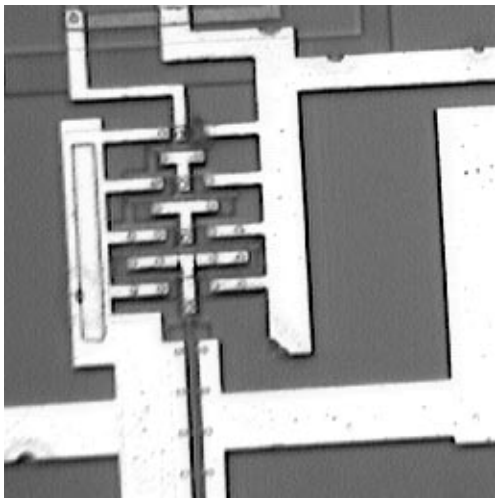

Figure A.19: IC.

Figure A.20: Yogi.

Appendix B

Calculations of Computational Complexity

This section describes the method by which the expressions for computational complexity for symmetric/antisymmetric wavelet and multiwavelet filters were calculated in Table 4.1. First we consider scalar wavelets and assume a biorthogonal filter bank in which M_1 and M_2 are the lowpass (h) and highpass (g) filter lengths. Let L be the length of the (scalar) input signal.

If M_1 and M_2 are even, then each filter has $M_1/2$ and $M_2/2$ unique coefficients. The summation in the lowpass filtering operation for each sample,

$$y = \sum_{k=0}^{M_1-1} h_k\, x_k = \sum_{k=0}^{M_1/2-1} h_k\, (x_k + x_{M_1-1-k}), \tag{B.1}$$

requires $M_1/2$ multiplications, one for each unique filter coefficient, and $M_1/2+(M_1/2-1) = M_1-1$ additions. For the same reasons, the highpass filter requires $M_2/2$ multiplications and M_2-1 additions, for a total of $(M_1+M_2)/2$ multiplications and M_1+M_2-2 additions per output sample for both filter banks. However, the output of each filter is downsampled by 2, so there are $L/2$ samples out of this stage of the decomposition process. Hence the total

numbers of multiplications and additions required are $L(M_1+M_2)/4$ and $L(M_1+M_2-2)/2$, respectively.

If M_1 and M_2 are odd, then the filters have $(M_1+1)/2$ and $(M_2+1)/2$ unique coefficients. The summation in the lowpass filtering operation for each sample,

$$y = \sum_{k=0}^{M_1-1} h_k\, x_k = \sum_{k=0}^{(M_1-1)/2-1} h_k\,(x_k+x_{M_1-1-k}) + h_{(M_1-1)/2}\, x_{(M_1-1)/2}, \tag{B.2}$$

requires $(M_1+1)/2$ multiplications, one for each unique filter coefficient, and $(M_1-1)/2+(M_1-1)/2-1+1 = M_1-1$ additions. Similarly, the highpass filter requires $(M_2+1)/2$ multiplications and M_2-1 additions, for a total of $(M_1+M_2+2)/2$ multiplications and M_1+M_2-2 additions per output sample for both filter banks. Since there are $L/2$ output samples, the total numbers of multiplications and additions required are $L(M_1+M_2+2)/4$ and $L(M_1+M_2-2)/2$, respectively.

Now we consider multifilters. The lowpass (H) and highpass (G) multifilters are composed of four scalar filters, each of which is symmetric or antisymmetric. It should be noted that for a symmetric-antisymmetric multiwavelet, such as those used here, two of the scalar filters are symmetric while the other two are antisymmetric. Here M_1 and M_2 will still refer to the lengths of the scalar filters.

First, suppose M_1 and M_2 are even. Then each of the four scalar filters in H requires $M_1/2$ multiplications and M_1-1 additions, as calculated before for scalar wavelets. Since there are four such filters, the H multifilter requires $4(M_1/2) = 2M_1$ multiplications and $4(M_1-1)+2 = 4M_1-2$ additions. Note that the extra 2 additions come from the addition of two values to create the output value of each channel (see Figure 4.2). Likewise, the G multifilter requires $2M_2$ multiplications and $4M_2-2$ additions. Hence H and G together require $2(M_1+M_2)$ multiplications and $4(M_1+M_2-1)$ additions per output sample. Since the original length-L scalar signal is split into two half-length signals before going into the multifilter and each output channel of the multifilter is downsampled by two, the number of

samples out of each scalar filter is $L/4$. This gives the total requirements of $L(M_1+M_2)/2$ multiplications and $L(M_1+M_2-1)$ additions.

Now suppose M_1 and M_2 are odd. As for old-length scalar filters before, the two symmetric filters composing H require $(M_1+1)/2$ multiplications and M_1-1 additions per output sample. Since the two antisymmetric scalar filters composing H have odd length, their central coefficient must be zero. Hence, they require only $(M_1+1)/2-1$ multiplications and M_1-2 additions per output sample. Thus the total numbers of multiplications and additions required for the H multifilter, including the extra addition for each channel, are $2[(M_1+1)/2]+2[(M_1+1)/2-1] = 2M_1$ and $2(M_1-1)+2(M_1-2)+2 = 4(M_1-1)$, respectively. Similarly, the G multifilter requires $2M_2$ multiplications and $4(M_2-1)$ additions per output sample, and therefore, H and G together require $2(M_1+M_2)$ multiplications and $4(M_1+M_2-2)$ additions for each output sample. Multiplying by the $L/4$ output samples from each scalar filter gives a total of $L(M_1+M_2)/2$ multiplications and $L(M_1+M_2-2)$ additions.

Bibliography

[1] R. C. Calderbank, Ingrid Daubechies, Wim Sweldens, and Boon-Lock Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, 5(3):332–369, 1998.

[2] C. K. Chui and J. Lian. A study on orthonormal multi-wavelets. *J. Appl. Numer. Math*, 20:273–298, 1996.

[3] R. R. Coifman, Y. Meyer, and M. V. Wickerhauser. Wavelet analysis and signal processing. In *Wavelets and their Applications*, pages 153–178. Jones and Bartlett, Boston MA, 1992.

[4] R. R. Coifman and M. V. Wickerhauser. Best-adapted wave packet bases. *preprint (http://www.math.yale.edu/pub/wavelets/papers/bestbase.tex).*

[5] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Trans. on Information Theory*, 38(2):713–718, March 1992.

[6] Ingrid Daubechies. *Ten Lectures on Wavelets.* SIAM, Philadelphia PA, first edition, 1992.

[7] J. Geronimo, D. Hardin, and P. Massopust. Fractal functions and wavelet expansions based on several scaling functions. *J. Approx. Theory*, 78:373–401, 1994.

[8] Say Song Goh, Qingtang Jiang, and Tao Xia. Construction of biorthogonal multiwavelets using the lifting scheme. *preprint*, 1998.

[9] T. N. T. Goodman and S. L. Lee. Wavelets of multiplicity r. *Trans. of the Amer. Math. Society*, 342(1):307–324, March 1994.

[10] Glen G. Langdon Jr. An introduction to arithmetic coding. *IBM J. Res. Develop.*, 28(2):135–149, March 1984.

[11] Stéphane Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego CA, 1998.

[12] F. G. Meyer, A. Z. Averbuch, and J. O. Strömberg. Fast adaptive wavelet packet image compression. *preprint*, 1998.

[13] F. G. Meyer, A. Z. Averbuch, J. O. Strömberg, and R. R. Coifman. Multi-layered image representation: Application to image compression. *Proceedings of the IEEE International Conference on Image Processing*, 1998.

[14] Kannan Ramchandran and Martin Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. on Image Proc.*, 2(2):160–175, April 1993.

[15] Amir Said and William A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circ. and Syst. for Video Tech.*, 6(3):243–250, June 1996.

[16] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Image Proc.*, 41(12):3445–3462, December 1993.

[17] Yair Shoham and Allen Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, 36(9):1445–1453, September 1988.

[18] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Wellesley MA, first edition, 1996.

[19] V. Strela, P. N. Heller, G. Strang, P. Topiwala, and C. Heil. The application of multi-wavelet filter banks to image processing. *preprint*, 1998.

[20] V. Strela and A. T. Walden. Orthogonal and biorthogonal multiwavelets for signal denoising and image compression. *Proc. SPIE*, 3391:96–107, 1998.

[21] Vasily Strela. *Multiwavelets: Theory and Applications*. PhD thesis, Massachusetts Institute of Technology, 1996.

[22] Vasily Strela. A note on construction of biorthogonal multi-scaling functions. *Contemporary Mathematics*, 216:149–157, 1998.

[23] J. Y. Tham, S. Ranganath, and A. A. Kassim. Scalable multiwavelet-based image and video compression for multimedia applications. *4th. International Conference on Information Systems, Analysis and Synthesis*, 3:195–202, 1998.

[24] Jo Yew Tham, Li-Xin Shen, Seng Luan Lee, and Hwee Huat Tan. A general approach for analysis and application of discrete multiwavelet transforms. *preprint*, 1998.

[25] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, Englewood Cliffs, NJ, first edition, 1993.

[26] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[27] Dong Wei, Hung-Ta Pai, and Alan C. Bovik. Antisymmetric biorthogonal coiflets for image coding. *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 1998.

[28] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commications of the ACM*, 30(6):520–540, June 1987.

[29] Tao Xia and Qingtang Jiang. Optimal multifilter banks: Design, related symmetric extension transform and application to image compression. *preprint*, 1998.

[30] Xiang-Gen Xia, Jeffrey S. Geronimo, Douglas P. Hardin, and Bruce W. Suter. On computations of multiwavelet transforms. *Proc. SPIE*, 2569:27–38, 1995.

[31] Zixiang Xiong, Michael T. Orchard, and Kannan Ramchandran. A comparative study of DCT and wavelet based coding. *Proc. SPIE*, 3164:271–278, 1997.

[32] Zixiang Xiong, Kannan Ramchandran, and Michael T. Orchard. Space-frequency quantization for wavelet image coding. *IEEE Trans. on Image Proc.*, 6(5):677–693, May 1997.

[33] Zixiang Xiong, Kannan Ramchandran, and Michael T. Orchard. Wavelet packet image coding using space-frequency quantization. *IEEE Trans. on Image Proc.*, 7(6):892–898, June 1998.

Vita

Michael B. Martin was born on July 25, 1973, in Blacksburg, VA. After growing up in King George County, he left high school two years early in 1989 to attend Simon's Rock College in Great Barrington, MA. Shortly after beginning studies at Simon's Rock, he decided to pursue a degree in physics. In 1991 he transferred to Grinnell College to complete his physics program. After earning his Bachelors degree in 1993, Michael decided to return closer to home for graduate school and enrolled in the Masters program in physics at Virginia Tech. With interests in both math and physics, he quickly settled on the idea of pursuing a Ph.D. in mathematical physics. Nearly four years of coursework later, he realized that the career prospects in mathematical physics had become unappealing. Drawing upon his long-standing interests in electronics and computer programming, he decided to change academic direction. He applied his math and physics coursework to obtain Masters degrees in both fields and then began pursuit of a Masters degree in electrical engineering.

Michael's academic interests span many areas of mathematics, physics, computers, and electronics, but currently focus on image processing, wavelets, filter banks, multirate DSP, and adaptive filtering. When not working on various programming projects, he may be found outside hiking, biking, or rock climbing.