RICE UNIVERSITY

Adaptive Wavelet Transforms via Lifting

by

Roger L. Claypoole, Jr.

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

DOCTOR OF PHILOSOPHY

Approved, Thesis Committee:

Richard G. Baraniuk, Chairman
Associate Professor of Electrical and Computer Engineering

C. Sidney Burrus
Professor of Electrical and Computer Engineering

Raymond O. Wells, Jr.
Professor of Mathematics

Houston, Texas

October, 1999

Adaptive Wavelet Transforms via Lifting

Roger L. Claypoole, Jr.

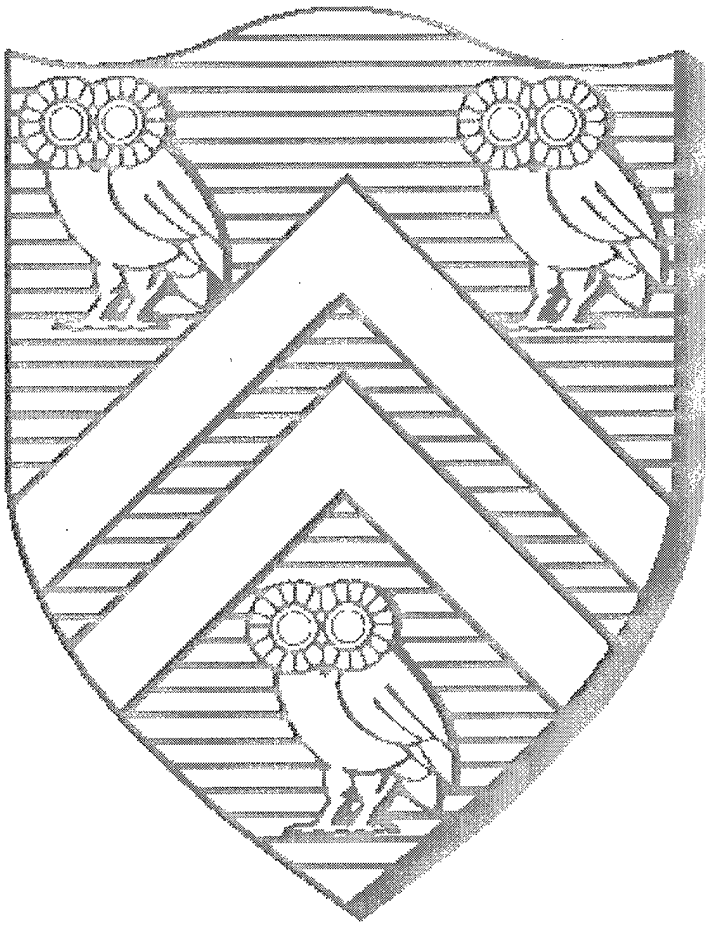
Abstract

Wavelet transforms have proven very useful for a variety of signal and image processing tasks. The wedgelet transform also shows promise for certain edge-dominated images. However, in many applications we desire to introduce adaptivity and nonlinearities into the transforms. These are powerful extensions, but difficult to control within the wavelet framework. The *lifting scheme* provides a new, spatial intuition into the wavelet transform that simplifies the introduction of adaptivity.

In this thesis, we develop several new adaptive wavelet transforms and adaptive multiresolution wedgelet transforms. The lifting construction permits control over the multiresolution properties of these transforms despite the adaptivity. We demonstrate the power of our new adaptive lifted transforms with successful applications to signal denoising and compression problems.

Adaptive Wavelet Transforms via Lifting

Roger L. Claypoole, Jr.



Thesis: DOCTOR OF PHILOSOPHY
Electrical and Computer Engineering
Rice University, Houston, Texas (October 1999)

Acknowledgments

First, I would like to thank the members (past and present) of the Rice DSP Group, especially office-mate Dr. Matt Crouse, Justin Romberg, coauthor Prof. Robert Nowak, Dr. Charlotte Gruner, Jim Lewis, Dr. Jan Odegard, Dr. Katrin Berkner, Felix Fernandes, Clay Scott, Dr. Hyeokho Choi, Brent Hendricks, Ramesh (Neelsh) Neelamani, Vinay Ribeiro, Rohit Gaikwad and Prof. Don Johnson. Their insights and support (not to mention just being there to bounce ideas around) were invaluable.

Next, I want to give special thanks to my advisor, Prof. Richard Baraniuk. Without his support, patience, technical expertise, and ability to see the "big picture," this thesis would never have been completed. It was a pleasure to be part of the "Baraniuk research family."

Thanks to the National Science Foundation (grant nos. MIP-9457438 and CCR-9973188), the Office of Naval Research (grant no. N00014-99-1-0813), the Defense Advanced Research Projects Agency/Air Force Office of Scientific Research (grant no. F49620-97-1-0513), and Texas Instruments for their generous financial support. Thanks to thesis committee members Dr. C. Sidney Burrus and Dr. Raymond O. Wells for providing useful commentary, criticism, and insights. Also, thanks to coauthor Dr. Geoffrey Davis for introducing me to the lifting scheme, and coauthor Dr. Wim Sweldens.

An additional word of thanks to Dr. John Bryan, Jeff and Risa Myers, and Tom and Veronica Hoge for making my stay in Houston enjoyable. "Without friends, no one would choose to live, though he had all other goods."

Last but most importantly, I thank my wife, Lisa, for her patience and understanding during my PhD program. She is my "emotional capacitor;" without her, I would never have survived the PhD experience.

Contents

Illustrations

Tables

Chapter 1

Introduction

1.1 Problem Statement

In his classic treatise on the workings of the human visual system, Marr focused on the importance of the *representation* of information for various cognitive tasks [1]. The way in which information is represented brings out certain types of features while hiding others. Signal compression and estimation applications also rely heavily on having an efficient representation of image data; we would like to approximate a signal with a small number of parameters. Therefore, we seek a transform which yields an efficient representation while bringing out the desired features of the signal.

The discrete wavelet transform (DWT) provides such an efficient representation for a broad range of real-world signals. This property has been exploited to develop powerful signal denoising and estimation methods [2] and extremely low-bit-rate compression algorithms [3]. However, the DWT struggles with discontinuities along contours (edges). The recently developed wedgelet transform [4] provides a more efficient representation for certain classes of these edge-dominated images. Our problem: *Can we improve these transforms with adaptivity?*

1.2 Our Goal

We claim the answer is a resounding yes; the goal of this thesis is to prove it. Lifting [5, 6] provides a framework to introduce adaptivity into the wavelet transform. Within this framework, we build several new adaptive wavelet and wedgelet transforms. We improve signal representation while preserving the multiresolution properties of the

transforms.

1.3 Thesis Layout

We provide the background for this thesis in Chapter 2. We present a brief overview of the wavelet transform, and its relation to perfect reconstruction filter banks. We also describe the wedgelet transform.

In Chapter 3 we present our interpretation of the lifting construction, and its relation to the wavelet transform. We continue in Chapter 4 by building several adaptive wavelet transforms within the lifting framework. We also construct an adaptive multiresolution wedgelet transform.

In Chapter 5 we apply our new transforms to problems in signal estimation and image compression. We conclude in Chapter 6 with a summary of our results and recommendations for future research.

Chapter 2

Background

2.1 The Wavelet Transform

We provide here a brief overview of the wavelet transform. For a more thorough analysis, see [3, 7, 8, 9].

2.1.1 Construction of wavelet spaces

One of the primary themes in multiresolution analysis is the successive projection of a function into "smaller" orthogonal subspaces. These subspaces will be formed from shifts and dilations of a low pass scaling function $\phi(t)$ and a band pass wavelet function $\psi(t)$.

Let $\{V_m\}_{m\in\mathcal{Z}}$ be a sequence of nested subspaces constructed in $\mathrm{L}^2(\Re)$ such that $V_m \subset V_{m-1} \ \forall\ m \in \mathcal{Z}$. Let f be a function which exists in the subspace V_{m-1} for some m. Let P_m be the projection operator which takes the function $f \in V_{m-1}$ into the nested subspace $V_m \subset V_{m-1}$. This projection operator eliminates the portion of f which is not in V_m, while it does not disturb the portion of f which lies in V_m.

Let $\{W_m\}_{m\in\mathcal{Z}}$ be another sequence of nested subspaces such that $W_m \subset V_{m-1} \ \forall\ m \in \mathcal{Z}$. Further, let the span of $V_m \cup W_m$ equal V_{m-1}, and let V_m and W_m be orthogonal subspaces. Thus, $V_{m-1} = V_m \oplus W_m$ and $V_m \cap W_m = \phi$. An example of such a set of nested subspaces is shown in Figure 2.1.

Let $Q_m = \mathbf{I} - P_m$ (where $\mathbf{I}$ is the identity operator) be the orthogonal projection operator which takes the function f into the nested subspace $W_m \subset V_{m-1}$. Thus, $P_m Q_m = Q_m P_m = \mathbf{0}$, where $\mathbf{0}$ is the zero operator.

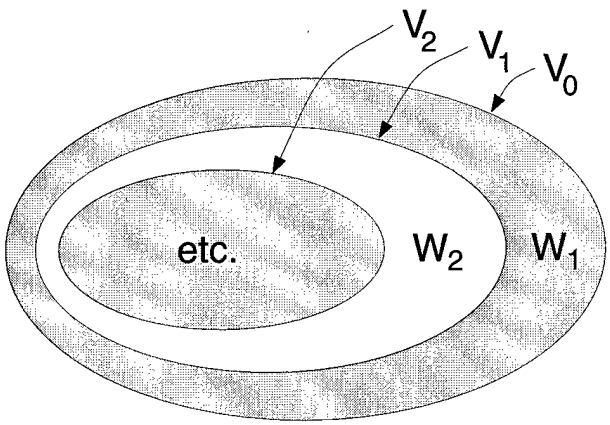


Figure 2.1 : *Nested subspaces of the orthogonal wavelet transform. The V_m are spanned by shifts and dilations of a low pass scaling function, while the W_m are spanned by shifts and dilations of a band pass wavelet function.*

2.1.2 The wavelet recursion relation

Let the set of functions $\{\phi_{m,l}\}_{l\in\mathcal{Z}}$ be an orthonormal basis for V_m, and let the set of functions $\{\psi_{m,l}\}_{l\in\mathcal{Z}}$ be an orthonormal basis for W_m. Since $V_m \subset V_{m-1}$, we can expand $\phi_{m,l}$ in terms of $\{\phi_{m-1,k}\}$, that is,

$$\phi_{m,l}(x) = \sum_k c_{m-1,l}(k)\phi_{m-1,k}(x)\,.$$

To find a particular coefficient $c_{m-1,l}(k)$, we take the inner product of $\phi_{m,l}$ with the basis function $\phi_{m-1,k}$. The inner product of two arbitrary real valued functions f and g is defined by

$$\langle f, g\rangle = \int_{-\infty}^{\infty} f(t)g(t)dt\,.$$

Let $h_l(k) = \langle \phi_{m,l}, \phi_{m-1,k}\rangle$. The expansion of $\phi_{m,l}$ becomes

$$\phi_{m,l}(x) = \sum_k h_l(k)\phi_{m-1,k}(x). \tag{2.1}$$

The set of functions $\phi_{m,l}$ must satisfy Equation [2.1] for all m if we intend to form an orthogonal wavelet transform. If we desire a discrete wavelet transform (DWT),

then we require that each $\phi_{m,l}$ be constructed from integer shifts (l) and translates (2^m) of a single scaling function $\phi(x)$. In this case, Equation [2.1] becomes

$$\phi(x-l) = \sum_k h(k-2l)\phi(2x-k)\,. \tag{2.2}$$

In similar fashion, we have

$$\psi(x-l) = \sum_k g(k-2l)\phi(2x-k)\,, \tag{2.3}$$

where $g(k-2l) = \langle \psi_{m,l}, \phi_{m-1,k} \rangle$. Equations [2.2] and [2.3] are known as the *wavelet recursion relations* [3].

2.1.3 Construction of the discrete wavelet transform

In section 2.1.1, we constructed a set of embedded orthogonal spaces to permit the multiresolution decomposition of a signal f. To perform this decomposition, we first assume that the signal f exists entirely in the space V_{m-1} for some $m \in \mathcal{Z}$ (typically $m = 1$). That is, f can be completely represented as a linear combination of the basis functions for V_{m-1}.

The projection of f into V_m can be expanded in terms of the basis functions for V_m, and the projection of f into W_m can be expanded in terms of the basis functions for W_m, resulting in the following equalities:

$$\begin{aligned} [P_m f](x) &= \sum_l c_m(l)\phi_{m,l}(x) \\ [Q_m f](x) &= \sum_l d_m(l)\psi_{m,l}(x)\,, \end{aligned}$$

with $c_m(l) = \langle P_m f, \phi_{m,l} \rangle$ and $d_m(l) = \langle P_m f, \psi_{m,l} \rangle$.

However, since $P_m + Q_m = \mathbf{I}$, we can write

$$f = P_m f + Q_m f,$$

and any particular decomposition coefficient in V_m can be written as

$$\begin{aligned} c_m(l) &= \langle P_m f, \phi_{m,l} \rangle \\ &= \langle (f - Q_m f), \phi_{m,l} \rangle \\ &= \langle f, \phi_{m,l} \rangle - \langle Q_m f, \phi_{m,l} \rangle \\ &= \langle f, \phi_{m,l} \rangle \ . \end{aligned}$$

If f is expanded in terms of the basis functions for V_{m-1} and the results substituted into the expression for $c_m(l)$, then

$$\begin{aligned} c_m(l) &= \langle f, \phi_{m,l} \rangle \\ &= \langle (\textstyle\sum_k c_{m-1}(k) \phi_{m-1,k}) \, , \phi_{m,l} \rangle \\ &= \textstyle\sum_k c_{m-1}(k) \, \langle \phi_{m-1,k} \, , \phi_{m,l} \rangle \ . \end{aligned}$$

In similar fashion, it can be shown that

$$d_m(l) = \sum_k c_{m-1}(k) \, \langle \phi_{m-1,k} \, , \psi_{m,l} \rangle \ .$$

However, as mentioned in Section 2.1.2, for the case of the discrete wavelet transform these inner products take the form of digital filters independent of decomposition level, i.e.,

$$\begin{aligned} \langle \phi_{m,l}, \phi_{m-1,k} \rangle &= h(k - 2l) \\ \langle \psi_{m,l}, \phi_{m-1,k} \rangle &= g(k - 2l) \, . \end{aligned}$$

Thus, if the coefficients $c_{m-1}(n)$ of $f \in V_{m-1}$ are known, then the coefficients of the projection of f into V_m and W_m are given respectively by

$$c_m(l) = \sum_k c_{m-1}(k) h(k - 2l) \tag{2.4}$$

$$d_m(l) = \sum_k c_{m-1}(k) g(k - 2l) \, . \tag{2.5}$$

Equations [2.4] and [2.5] lead to the filter bank implementation of the discrete wavelet transform (which will be described in Section 2.2).

Once the coefficients of the projection of f into V_m are known, the function can be further decomposed into the orthogonal subspaces of V_m, namely V_{m+1} and W_{m+1}, with the same formulas, since the discrete wavelet filters h and g are independent of the decomposition level m. This leads to a recursive decomposition routine, breaking f down into its projections onto smaller orthogonal subspaces.

Thus, the DWT decomposition is multiscale: it consists of a set of *scaling coefficients* $c_M[n]$, which represent coarse signal information at scale $m = M$, and a set of *wavelet coefficients* $d_m[n]$, which represent detail information at scales $m = 1, 2, \ldots, M$. This recursive algorithm is the heart of the modern multiresolution wavelet decomposition algorithm.

2.1.4 Reconstruction (synthesis)

In the above sections, we demonstrated how a signal can be decomposed. Now, we show how to take those components and reconstruct the original signal.

For one level of reconstruction (starting at scale $m = 1$), we know that the signal f exists in V_0, and thus can be represented as $f(x) = \sum_k c_0(k)\phi_{0,k}(x)$. We are given the decomposition coefficients $c_1(k)$ and $d_1(k)$. These coefficients represent the projection of f into V_1 and W_1 respectively. $V_1 \oplus W_1 = V_0$ and $V_1 \cap W_1 = \phi$, so

$$f(x) = \sum_k c_1(k)\phi_{1,k}(x) + \sum_k d_1(k)\psi_{1,k}(x)\,. \tag{2.6}$$

We take the inner product of both sides of equation [2.6] with $\phi_{0,n}(x)$. Since the $\{\phi_{0,n}\}$ are orthonormal, $\langle f, \phi_{0,n}\rangle = c_0(n)$. From Section 2.1.2, we have

$$\begin{aligned} \langle \phi_{1,k}, \phi_{0,n}\rangle &= h(n-2k) \\ \langle \psi_{1,k}, \phi_{0,n}\rangle &= g(n-2k)\,. \end{aligned}$$

Thus, equation [2.6] becomes

$$c_0(n) = \sum_k c_1(k)h(n-2k) + \sum_k d_1(k)g(n-2k)\,. \tag{2.7}$$

where $c_0(n)$ are the coefficients of $f \in V_0$. Equation [2.7] leads to the filter bank implementation of the inverse DWT (which will also be described in Section 2.2). Note that, for the orthogonal DWT, the digital filters h and g used in the inverse transform are identical to the filters used for the forward transform.

2.1.5 Generalization to biorthogonal wavelets

The above analysis was based on the orthogonal decomposition of our space. As noted, this forces the forward and inverse transform filters to be identical. We now loosen this restriction, and form a biorthogonal wavelet transform [3, 10]. We introduce dual spaces $\widetilde{V}_m$ and $\widetilde{W}_m$, with $V_m = \text{span}\{\phi_{m,k}\}$, $\widetilde{V}_m = \text{span}\{\widetilde{\phi}_{m,k}\}$, etc. The spaces must be *biorthogonal*, that is, $V_m \cap \widetilde{W}_m = \phi$ and $\widetilde{V}_m \cap W_m = \phi$. In addition, the basis vectors must satisfy the *biorthogonality condition*,

$$\left\langle \phi_{m,k} , \widetilde{\phi}_{m,l} \right\rangle = \delta(k-l) \tag{2.8}$$

$$\left\langle \psi_{m,k} , \widetilde{\psi}_{m,l} \right\rangle = \delta(k-l)\,. \tag{2.9}$$

Figure 2.2 demonstrates such a biorthogonal decomposition of $V_0 = \Re^3$. V_1 is spanned by ϕ_0 and ϕ_1, while $\widetilde{V}_1$ is spanned by $\widetilde{\phi}_0$ and $\widetilde{\phi}_1$. V_1 is orthogonal to its dual space, $\widetilde{W}_1$, while $\widetilde{V}_1$ is orthogonal to W_1. All the basis functions satisfy the biorthogonality condition, equations [2.8] and [2.9].

As before, we require recursion relations

$$\begin{aligned} \phi_{m,l}(x) &= \sum_k h_l(k)\phi_{m-1,k}(x) \\ \psi_{m,l}(x) &= \sum_k g_l(k)\phi_{m-1,k}(x)\,. \end{aligned}$$

However, to find h, we must take the inner product of $\phi_{m,l}$ with the *dual* function

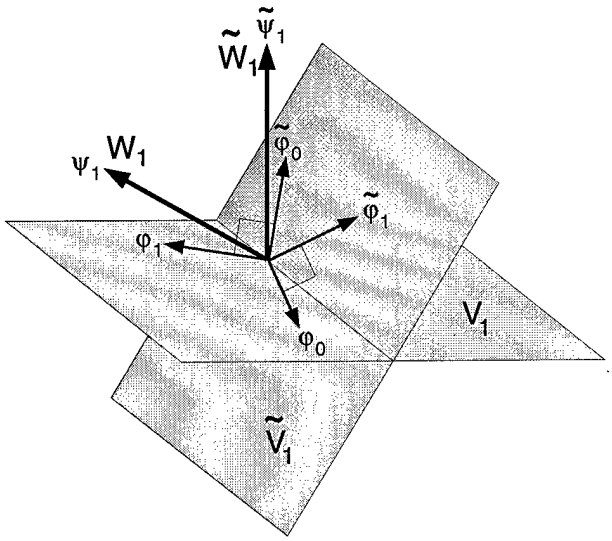


Figure 2.2 : *Example of biorthogonal wavelet spaces in* $\Re^3$. *Span* $\{V_1 \cup W_1\} = \Re^3$ *and span* $\left\{\widetilde{V}_1 \cup \widetilde{W}_1\right\} = \Re^3$. $V_1 \perp \widetilde{W}_1$ *and* $\widetilde{V}_1 \perp W_1$. *All basis functions satisfy the biorthogonality condition.*

$\widetilde{\phi}_{m-1,k}$:

$$\begin{aligned}\left\langle \phi_{m,l}\,,\,\widetilde{\phi}_{m-1,k}\right\rangle &= \sum_n h_l(n)\left\langle \phi_{m-1,n}\,,\,\widetilde{\phi}_{m-1,k}\right\rangle \\ &= \sum_n h_l(n)\,\delta(k-n) \\ &= h_l(k)\,.\end{aligned}$$

In similar fashion, $g_l(k) = \left\langle \psi_{m,l}\,,\,\widetilde{\phi}_{m-1,k}\right\rangle$. We have the equivalent recursion relations with $\widetilde{h}$ and $\widetilde{g}$ for the dual basis.

2.1.6 Construction of the biorthogonal discrete wavelet transform

Given these dual systems, we intend to decompose a signal f which exists entirely in V_{m-1}; $f(x) = \sum_k c_{m-1}(k)\widetilde{\phi}_{m-1,k}(x)$. We desire a decomposition

$$f(x) = \sum_k c_m(k)\widetilde{\phi}_{m,k}(x) + \sum_k d_m(k)\widetilde{\psi}_{m,k}(x)\,.$$

Clearly, a coefficient $c_m(k)$ is found by taking the inner product of f with $\phi_{m,k}$ (due to the biorthogonality condition). This yields

$$\begin{aligned} c_m(k) &= \langle f, \phi_{m,k} \rangle \\ &= \sum_l c_{m-1}(l) \left\langle \phi_{m,k} , \widetilde{\phi}_{m-1,l} \right\rangle \\ &= \sum_l c_{m-1}(l) h_k(l) , \end{aligned} \tag{2.10}$$

and

$$d_m(k) = \sum_l c_{m-1}(l) g_k(l) . \tag{2.11}$$

The digital filters h and g are independent of scale. These equations determine the *analysis* portion of the biorthogonal wavelet transform, and are identical to the orthogonal wavelet transform equations (albeit with different filters h and g).

2.1.7 Reconstruction with the biorthogonal wavelet transform

To complete our analysis, we reconstruct our original signal with the biorthogonal wavelet transform. We are given the biorthogonal wavelet coefficients from scale m, that is,

$$f(x) = \sum_k c_m(k) \widetilde{\phi}_{m,k}(x) + \sum_k d_m(k) \widetilde{\psi}_{m,k}(x).$$

We know that f exists in V_{m-1}, and can be represented as

$$f(x) = \sum_k c_{m-1}(k) \widetilde{\phi}_{m-1,k}(x) .$$

We take the inner product of these equations with $\phi_{m-1,k}$ and find that

$$\begin{aligned} \langle f, \phi_{m-1,l} \rangle &= \sum_k c_{m-1}(k) \left\langle \widetilde{\phi}_{m-1,k} , \phi_{m-1,l} \right\rangle \\ &= c_{m-1}(l) \\ &= \sum_k c_m(k) \left\langle \widetilde{\phi}_{m,k} , \phi_{m-1,l} \right\rangle + \sum_k d_m(k) \left\langle \widetilde{\psi}_{m,k} , \phi_{m-1,l} \right\rangle \\ &= \sum_k c_m(k) \widetilde{h}_k(l) + \sum_k d_m(k) \widetilde{g}_k(l) . \end{aligned} \tag{2.12}$$

This is identical to the reconstruction requirements for the orthogonal transform (Equation [2.7]), except that we use the dual filters $\widetilde{h}$ and $\widetilde{g}$. Equation [2.10], equation [2.11], and equation [2.11] lead to the filter bank implementation of the biorthogonal discrete wavelet transform.

2.2 Filter Banks

In its most basic form, a filter bank is simply a collection of filters with a common input (or output). The analysis bank takes a common input to M channels, and the synthesis bank returns these channels to a single output [11, 12]. However, we are interested in maximally decimated filter banks, i.e., filter banks which do not expand the number of input samples. This is done by incorporating decimators and expanders into each channel of the filter bank, as shown in Figure 2.3. For maximal decimation, we must have $\sum_k \frac{1}{R_k} = 1$. If $R_k = M \ \forall k$, then we have a maximally decimated *uniform* filter bank [11].

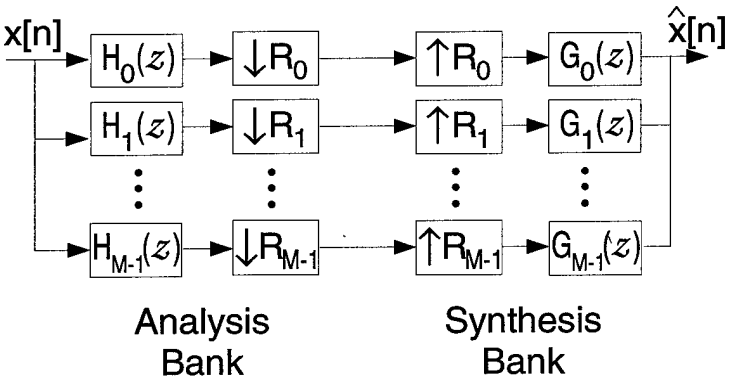


Figure 2.3 : *M-channel filter bank. The analysis bank has one input and M outputs, while the synthesis bank takes the M channels to one output.*

Under what conditions is the output of the filter bank $\widehat{x}[n]$ equal to the input signal $x[n]$? To simplify this question, we use the *polyphase notation* [13]. We write

each filter $H_m(z)$ as:

$$H_m(z) = \sum_{k=0}^{M-1} E_{m,k}(z^M)\, z^{-k},$$

where $E_{m,k}(z) = \sum_l h_m(Ml+k)z^{-l}$. However, applying $E_{m,k}(z^M)$ to a signal and then downsampling by M is equivalent to downsampling the signal by M and then applying $E_{m,k}(z)$ to the result [11]. Thus, we can redraw the filter bank of Figure 2.3 as the polyphase filter bank of Figure 2.4 (using similar decompositions for the synthesis filters). Clearly the bank of delays and decimators simply splits the input signal into its polyphase components, and the bank of expanders and advances reassembles those components. Thus, if we desire *perfect reconstruction* (PR), i.e., $\hat{x}[n] = x[n]$, then we require $R(z)E(z) = \mathbf{I}$ [1] [11, 12].

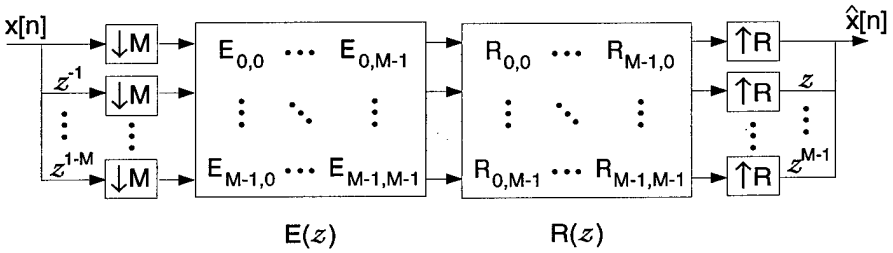


Figure 2.4 : *Polyphase representation of an M-channel, maximally decimated uniform filter bank. $E(z)$ and $R(z)$ are both $M \times M$.*

In general the inverse of $E(z)$ may not exist. If $(E(z))^{-1}$ does exist, the PR system will not be realizable with finite impulse response (FIR) filters unless the determinant of E(z) is a delay. In special cases, we have

$$(E(z))^{-1} = E^H(z^{-1})\,.$$

When $E(z)$ has this property and $R(z)$ is chosen as $E^H(z^{-1})$, our PR system is called *paraunitary* [11, 12].

[1] This requirement is somewhat over-restrictive, but acceptable for our discussions here. For a more general analysis, see [11].

2.3 Filter Bank Implementation of the Wavelet Transform

Armed with this understanding of filter banks, we employ them to implement the discrete wavelet transform. As shown in section 2.1.3, at decomposition level m the coefficients $c_m(n)$ and $d_m(n)$ can be written in terms of the coefficients at the next higher level as:

$$c_m(n) = \sum_k c_{m-1}(k)h'(k-2n) \tag{2.13}$$

$$d_m(n) = \sum_k c_{m-1}(k)g'(k-2n)\,, \tag{2.14}$$

where h' and g' are filters which satisfy the wavelet recursion equations

$$\phi(x-l) = \sum_k h'(k-2l)\phi(2x-k)$$

$$\psi(x-l) = \sum_k g'(k-2l)\phi(2x-k)\,.$$

(We add the notation h' and g' so that we can use h and g for the filters which will ultimately be implemented in our filter bank. We hope that this change in notation does not overly confuse.)

We create new filters h and g which are "flipped" versions of the filters h' and g', that is, $h(n) = h'(-n)$ and $g(n) = g'(-n)$. Substituting these filters $h(n)$ and $g(n)$ into equations [2.13] and [2.14] yields

$$c_m(n) = \sum_k c_{m-1}(k)h(2n-k)$$

$$d_m(n) = \sum_k c_{m-1}(k)g(2n-k)\,.$$

Note that if $y(n) = (x(n))\downarrow_2$, then $y(n) = x(2n)$. Thus, $c_m(n)$ and $d_m(n)$ can be written as

$$c_m(n) = \left(\sum_k c_{m-1}(k)h(n-k)\right)\downarrow_2$$

$$d_m(n) = \left(\sum_k c_{m-1}(k)g(n-k)\right)\downarrow_2\,.$$

Clearly, this is simply the wavelet filters $h(n)$ and $g(n)$ convolved with $c_{m-1}(n)$ and then downsampled by two. Thus, the forward discrete wavelet transform can be implemented as a filter bank with wavelet filters h and g, as shown in Figure 2.5. The filters $h(n)$ and $g(n)$ are traditionally chosen to be low pass and high pass filters, respectively. Thus, the coefficients $c_m(n)$ are referred to as "low pass coefficients," "coarse coefficients," or "scaling coefficients." The coefficients $d_m(n)$ are referred to as "high pass coefficients," "wavelet coefficients," or "detail coefficients" [3]. The transform is typically iterated on the output of the low pass band ($c[n]$) to create the series of detail coefficients at different scales.

In similar fashion, reconstruction (from Equation [2.7]) is governed by

$$c_{m-1}(n) = (c_m(n))\uparrow_2 * \widetilde{h}(n) + (d_m(n))\uparrow_2 * \widetilde{g}(n)\,.$$

Thus, the inverse DWT can be implemented as a filter bank as shown in Figure 2.6. As discussed in Section 2.1.5, h, g, $\widetilde{h}$, and $\widetilde{g}$ must satisfy the biorthogonality and wavelet recursion requirements if the filter bank is to form a wavelet system. For special choices of h and g, we have $\widetilde{h} = h$ and $\widetilde{g} = g$, and the underlying wavelet and scaling functions form an orthonormal wavelet system [3], as described in Section 2.1.1.

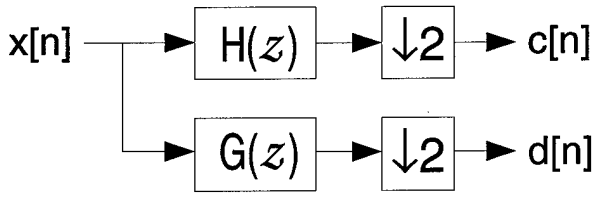


Figure 2.5 : *Filter bank implementation of the wavelet transform. Transform is iterated on the scaling coefficients* $c[n]$.

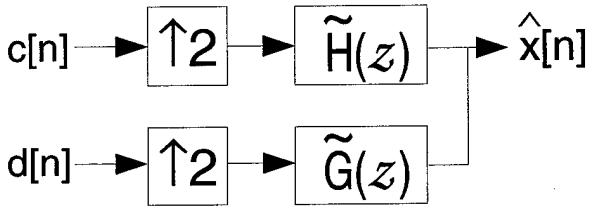


Figure 2.6 : *Filter bank implementation of the inverse wavelet transform. With appropriate choices of $\widetilde{H}$ and $\widetilde{G}$, the transform will yield a perfectly reconstructed output sequence.*

2.4 Wedgelets

Wavelets provide a very economical representation for a broad class of signals. However, they do not perform well near edges in images. Therefore, wavelets are not well suited for images which fall into the *horizon class* [4], that is, binary images with an edge along a contour. An example of such a horizon image is shown in Figure 2.7.

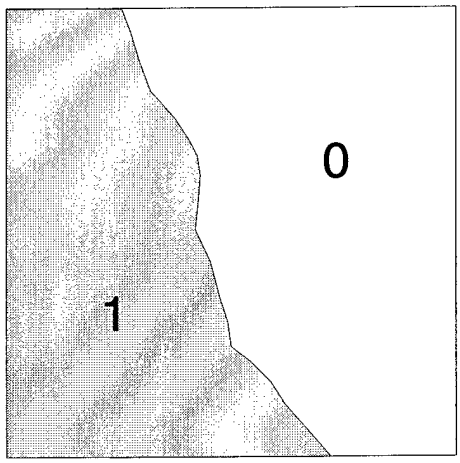


Figure 2.7 : *Horizon image example. A binary image with a single edge along a contour.*

To address this problem, Donoho created the *wedgelet* representation [4], which is near optimal for images in the horizon class. Each wedgelet is defined on a dyadic

square as a binary image with an edge along an *edgelet.* An edgelet is a straight line connecting two vertices. To keep the computation tractable, the edgelets are restricted to a discrete set of vertices along the border, typically the length of the border divide by some constant, M. An example of a wedgelet constructed in this fashion is shown in Figure 2.8. For this square, we have $M = 3$ vertices per side. Thus, our total number of wedgelets is 42, along with the complete square (no wedgelet).

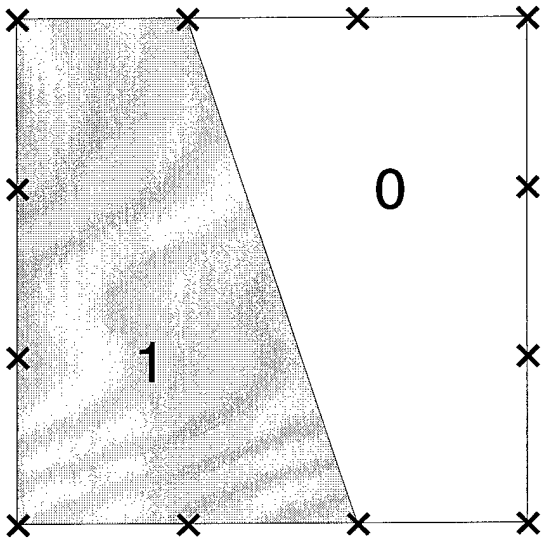


Figure 2.8 : *Wedgelet example on an arbitrary dyadic square. Each edgelet (which defines the wedgelet) is formed by connecting the vertices along the border (in this case, $M = 3$ vertices per side.)*

To approximate an image in the horizon model, we perform a dyadic decomposition of the image. If a wedgelet (or an entire square) in a particular dyadic square is a good approximation to the horizon image, then we do not decompose the square any further. However, if the approximation does not meet our criteria, then we decompose the dyadic square into four smaller squares. We then compute the wedgelets for these squares, and determine if any of the approximations are acceptable. This process continues until the overall wedgelet approximation meets our criteria. Figure

2.9 shows a wedgelet approximation to the horizon image of Figure 2.7. Note that the wedgelets are only applied at the lowest level; we only perform splits on dyadic squares (and not other polygons). We can approximate any image in the horizon class to any desired approximation error with enough dyadic splits [4].

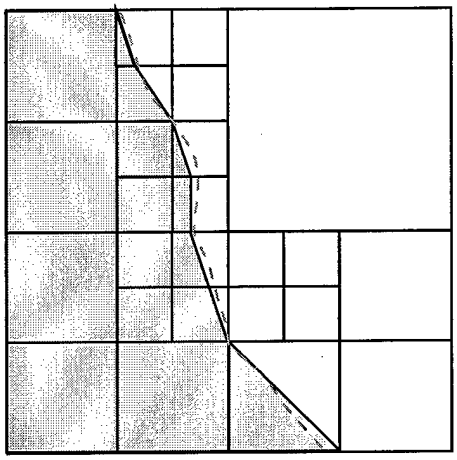


Figure 2.9 : *Wedgelet approximation to a horizon function. The underlying horizon function is shown with the dashed line. The dyadic decomposition continues until the approximation error between our string of edgelets and the contour is acceptably small.*

Chapter 3

Lifting

The economy of the wavelet transform stems from the fact that the DWT tends to compress real-world signals into just a few coefficients of large magnitude. Compression follows from the "vanishing moments" property of wavelets, which guarantees that the wavelet coefficients of low-order polynomial signals are zero [3]. Thus, if a signal is exactly polynomial, it can be completely described using scaling coefficients alone. In more realistic situations, the signal will not be polynomial, but may be well-approximated by a *piecewise* polynomial function. Because wavelet functions also have localized support, most of the wavelet coefficients of such a signal will be zero except those corresponding to wavelets having support near the breakpoints of the polynomial segments.

It is fruitful to view the DWT as a prediction-error decomposition. The scaling coefficients at a given scale (m) are "predictors" for the data at the next higher resolution or scale ($m-1$). The wavelet coefficients are simply the "prediction errors" between the scaling coefficients and the higher resolution data that they are attempting predict. This interpretation has led to a new framework for DWT design known as the *lifting scheme* [5, 6]. Our analysis builds on this interpretation.

3.1 The Lifting Concept

Lifting is a spatial (or time) domain construction of biorthogonal wavelets developed by Sweldens [5, 6]. We present here an overview of our interpretation of the lifting concept.

3.1.1 Lifting operations

Lifting consists of iteration of the following three basic operations (see Figure 3.1):

Split: Divide the original data into two disjoint subsets. Although any disjoint split is possible, in the standard lifting scheme we split the original data set $x[n]$ into $x_e[n] = x[2n]$, the even indexed points, and $x_o[n] = x[2n+1]$, the odd indexed points.

Predict: Generate the wavelet coefficients $d[n]$ as the error in predicting $x_o[n]$ from $x_e[n]$ using prediction operator $\mathcal{P}$:

$$d[n] \;=\; x_o[n] - \mathcal{P}(x_e[n])\,. \tag{3.1}$$

Update: Combine $x_e[n]$ and $d[n]$ to obtain scaling coefficients $c[n]$ that represent a coarse approximation to the original signal $x[n]$. This is accomplished by applying an update operator $\mathcal{U}$ to the wavelet coefficients and adding to $x_e[n]$:

$$c[n] \;=\; x_e[n] + \mathcal{U}(d[n])\,. \tag{3.2}$$

These three steps form a *lifting stage.* Iteration of the lifting stage on the output $c[n]$ creates the complete set of DWT scaling and wavelet coefficients $c_j[n]$ and $d_j[n]$.[1] At each scale, we weight the $c_j[n]$ and $d_j[n]$ with k_e and k_o respectively, as shown in Figure 3.1. This normalizes the energy of the underlying scaling and wavelet functions.

The lifting steps are easily inverted, *even if $\mathcal{P}$ and $\mathcal{U}$ are nonlinear, space-varying, or non-invertible.* Rearranging (3.1) and (3.2), we have

$$x_e[n] = c[n] - \mathcal{U}(d[n])\,, \qquad x_o[n] = d[n] + \mathcal{P}(x_e[n])\,.$$

[1] In fact, *all* wavelet transforms can be factored into a series of lifting stages (with perhaps multiple predicts and updates per stage) [14].

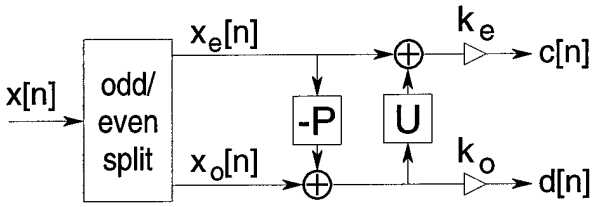


Figure 3.1 : *Lifting stage: Split, Predict, Update.* k_e *and* k_o *normalize the energy of the underlying scaling and wavelet functions.*

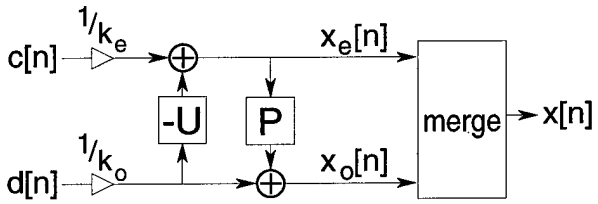


Figure 3.2 : *Typical inverse lifting steps: undo the Update, undo the Predict, and Merge the even and odd samples.*

As long as the same $\mathcal{P}$ and $\mathcal{U}$ are chosen for the forward and inverse transforms, the original signal will be perfectly reconstructed. The inverse lifting stage is shown in Figure 3.2.

3.1.2 Predictor design

In the simplest scenario, the prediction operator $\mathcal{P}$ is a linear shift-invariant filter with z-transform $P(z)$. In Figure 3.3, we illustrate a symmetric, $N = 4$-point predictor $P(z) = p_1\, z^{-1} + p_2 + p_3\, z + p_4\, z^2$. By tracing the contribution of $x_e[n]$ and $x_o[n]$ through the tree to the point $d[n]$, we find the equivalent filter that would be applied to the original data $x[n]$. In vector form, we have

$$\mathbf{g} \;=\; [-p_1,\; 0,\; -p_2,\; 1,\; -p_3,\; 0,\; -p_4]^T.$$

(Note the zeros at the positions corresponding to odd points in the original data, except for the 1 in the center.)

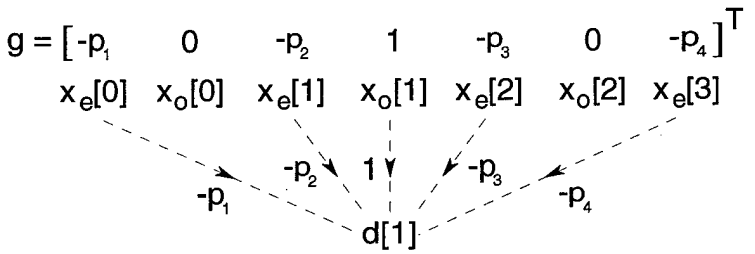


Figure 3.3 : *Prediction filtering. An $N = 4$ point linear prediction filter $P(z)$ yields the prediction vector* **g** *shown across the top.*

If a signal is exactly polynomial, it can be completely described using scaling (coarse) coefficients alone [3]. Our detail coefficients represent the "prediction errors" where our signal could not be completely represented by a low-order polynomial. Thus, the goal of the prediction step is to eliminate all low-order polynomials from $x[n]$; the residuals will be our detail coefficients. For a linear predictor, this is easily accomplished by the following procedure:

Form the $N \times 2N-1$ Vandermonde matrix $\mathbf{V}$ with entries $[\mathbf{V}]_{k,n} = (n-l)^k$, $n = 1, 2, \cdots, 2N-1$, $k = 0, 1, \cdots, N-1$. We adjust the shift l so that the $n - l = 0$ column corresponds to the 1 in $\mathbf{g}$, and then we delete all the even columns. We call the resulting matrix $\mathbf{V}^\diamond$.

Now, for the predictor to suppress all low-order polynomials, we require that $\mathbf{Vg} = 0$. By eliminating the columns which correspond to the 0's in $\mathbf{g}$, and pulling the column which corresponds to the center 1 out to the right, we are left with:

$$\mathbf{V}^\diamond \mathbf{p} = [1 \quad 0 \cdots 0]^T, \tag{3.3}$$

where the entries in $\mathbf{p}$ are the prediction filter coefficients. This set of linear equations is readily solved, since $\mathbf{V}^\diamond$ is Vandermonde and thus always invertible [15, p. 120].

3.1.3 Update design

The (linear) update filter $U(z)$ creates the coarse coefficients $c[n]$ by updating each $x_e[n]$ with the nearest $\widetilde{N}$ wavelet coefficients $d[n]$ from either side. The update order $\widetilde{N}$ can be chosen independently of N; however, the prediction coefficients p_k must be fixed prior to determining the update filter in the standard lifting programme.

In Figure 3.4, we trace the contribution of the original $x_e[n]$ and $x_o[n]$ to each $c[n]$ for an $N = 2$ point prediction followed by an $\widetilde{N} = 4$ point update with $U(z) = u_1\, z^{-2} + u_2\, z^{-1} + u_3 + u_4\, z$. In vector form, we have the equivalent filter $\mathbf{h}$ at the top of the Figure. Note that $\mathbf{h}$ is a function of both the update coefficients u_k and the prediction coefficients p_k.

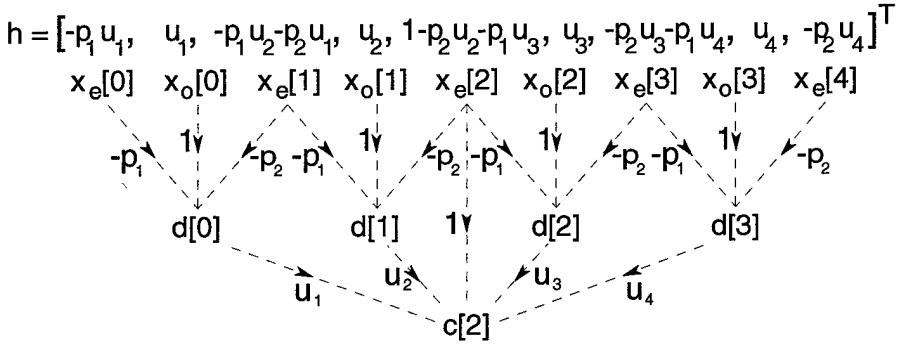


Figure 3.4 : *Update filtering. An $N = 2$ point linear predict followed by an $\widetilde{N} = 4$ point linear update yields the update vector* $\mathbf{h}$ *shown across the top.*

The update filter vector $\mathbf{h}$ should pass low-order polynomials into $c[n]$ while attenuating high-order polynomials. Conversely, we can design the mirror update filter vector $\widetilde{\mathbf{g}}$ (defined as $\widetilde{g}_n = (-1)^n h_n$) to *suppress* low-order polynomials. For the example in Figure 3.4, we have

$$\widetilde{\mathbf{g}} = [-p_1u_1,\ -u_1,\ (-p_1u_2-p_2u_1),\ -u_2,\ (1-p_2u_2-p_1u_3),\ \cdots \\ -u_3,\ (-p_2u_3-p_1u_4),\ -u_4,\ -p_2u_4]^T.$$

Since the $N = 2$ prediction coefficients are already determined, there are $\widetilde{N} = 4$ unknowns (the update coefficients u_k) in $\widetilde{\mathbf{g}}$. Solution of $\mathbf{V}\widetilde{\mathbf{g}} = \mathbf{0}$ as in equation [3.3] yields the update coefficients, which now suppress high-order polynomials.

In summary, in the lifting scheme we design the prediction step to eliminate the low-order polynomial signal structure, leaving only the high-order details. We design the update to preserve the low-order polynomial signal structure at the next coarser scale.

3.2 Equivalence between Polynomial Constraints and Vanishing Moments

3.2.1 Wavelet transform in polyphase form

As discussed in Section 2.1, any biorthogonal wavelet transform can be represented as a perfect reconstruction multirate filter bank. By splitting each wavelet filter into its polyphase components [11], the DWT can be implemented as shown in Figure 3.5.

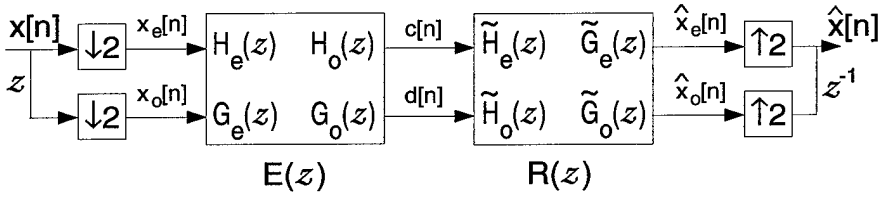


Figure 3.5 : *Wavelet filter bank in polyphase form.*

We have used the fact that $H(z) = H_e(z^2) + z^{-1}H_o(z^2)$, etc., and that filtering a signal with $H_e(z^2)$ and then downsampling by two is equivalent to downsampling the signal by two and then applying $H_e(z)$ [11].

3.2.2 Lifting in polyphase form

The lifted wavelet transform can be written in a similar form. As discussed in Section 3.1, lifting is comprised of a split, predict, and update. If we perform an odd/even split, we are working in the "polyphase domain." In this context, the predict and update steps are represented by the polyphase matrices shown in Figure 3.6. The prediction matrix passes $x_e[n]$ unchanged, but the wavelet coefficients $d[n]$ are the difference between $x_o[n]$ and $P(x_e[n])$, i.e., the failure of the odds to be predicted by the evens. Likewise, the update matrix passes the wavelet coefficients untouched, but uses these coefficients to "update" the $x_e[n]$ and create the scaling coefficients $c[n]$.

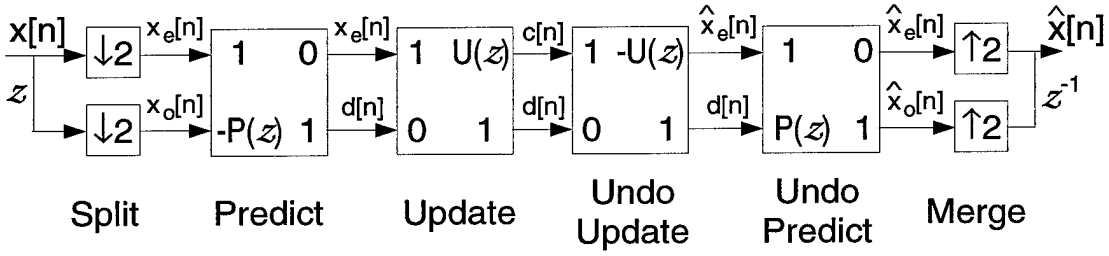


Figure 3.6 : *Polyphase representation of lifting operators.*

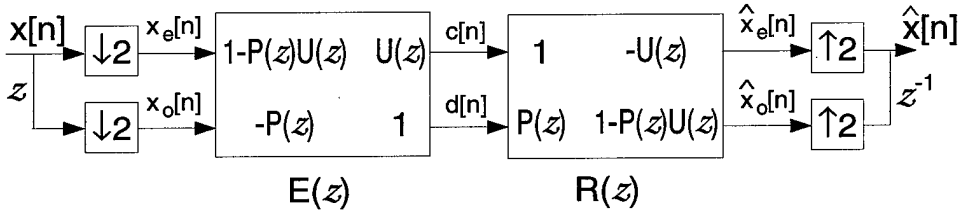


Figure 3.7 : *Polyphase representation of lifting with the prediction and update stages combined into one matrix*

Multiplying these matrices together yields the representation shown in Figure 3.7. Thus, we have taken the lifting process and written it in terms of the polyphase

matrices $E(z)$ and $R(z)$. Note that the one stage implementation of the lifted wavelet transform will not, in general, be orthogonal. Orthogonality requires that $E^H(z^{-1})E(z) = I$ [11], which forces our prediction and update filters to satisfy

$$\begin{aligned} U(z^{-1}) &= \frac{P(z)}{4(1-U(z)P(z))} \\ U(z)U(z^{-1}) &= 1/4\,. \end{aligned}$$

This will only be possible with real finite impulse response filters if $P(z)$ and $U(z)$ are constants (or delays). Therefore, we hope to interpret the general one-stage lifted transform as a biorthogonal wavelet transform. We equate the entries in the lifted polyphase matrices of Figure 3.7 with the polyphase components of the wavelet filters of Figure 3.5. This yields the following relations:

$$\begin{aligned} H_e(z) &= 1 - P(z)U(z) \\ H_o(z) &= U(z) \\ G_e(z) &= -P(z) \\ G_o(z) &= 1\,, \end{aligned}$$

or, equivalently,

$$H(z) = 1 - P(z^2)U(z^2) + z^{-1}U(z^2) \tag{3.4}$$

$$G(z) = -P(z^2) + z^{-1}\,, \tag{3.5}$$

with similar expressions for $\widetilde{H}(z)$ and $\widetilde{G}(z)$.

3.2.3 Vanishing moments and lifting constraints

In a wavelet system such as that shown in Figure 2.5, the analysis filters $h[n]$ and $g[n]$ correspond to a scaling function $\phi(t)$ and a wavelet function $\psi(t)$ respectively. The relationship between the wavelet filters and the wavelet functions is defined by

the *wavelet recursion relations* [3], as shown in Section 2.1.2:

$$\begin{aligned}\phi(t) &= \sum_k h[k]\phi(2t-k) \\ \psi(t) &= \sum_k g[k]\phi(2t-k).\end{aligned}$$

The wavelet filter coefficients $h[k]$ and $g[k]$ can be designed by adding vanishing moments to the underlying scaling and wavelet via the recursion relations. In equations [3.4] and [3.5], we have written the wavelet filters in terms of the prediction and update filters. Thus, we can map the vanishing moments constraints into constraints on our prediction and update filters P and U. For example, if we add a zeroth vanishing moment to $\psi(t)$ we have

$$\int_{-\infty}^{\infty} \psi(t)dt = \int_{-\infty}^{\infty} \sum_k g[k]\phi(2t-k) = 0.$$

We switch the order of the integral and the finite summation, recognize that $\int \phi(2t-k)dt$ is a non-zero constant m_0, and we have

$$m_0 \sum_k g[k] = 0.$$

Thus, the sum of the coefficients $g[k]$ must be zero. But $G(z) = -P(z^2) + z^{-1}$. Forcing $\sum_k g[k] = 0$ yields the constraint

$$p_1 + p_2 \cdots + p_N = 1.$$

This is identical to the lifting constraint that we derived in Section 3.1.2 when we forced the prediction filter to eliminate zeroth order polynomials! Upon further analysis, we find that every vanishing moment we add to the analysis wavelet function $\psi(t)$ is equivalent to eliminating additional polynomials in our prediction step. Once the coefficients of $G(z)$ (and, thus, the prediction filter coefficients) are determined, we add vanishing moments to the synthesis wavelet function $\widetilde{\psi}(t)$ via the relation

$$\int_{-\infty}^{\infty} t^l\, \widetilde{\psi}(t)dt = \int_{-\infty}^{\infty} \sum_k \widetilde{g}[k]\, t^l\, \widetilde{\phi}(2t-k) = 0.$$

It is straightforward to show that each vanishing moment on the synthesis wavelet function $\widetilde{\psi}(t)$ is equivalent to an additional update filter polynomial constraint, as derived in Section 3.1.3.

Thus, designing a biorthogonal wavelet system by adding vanishing moments to the underlying wavelet functions is equivalent to eliminating and preserving polynomials with the predict and update steps, respectively. Both interpretations yield identical constraints on the wavelet filters h, g, $\widetilde{h}$ and $\widetilde{g}$. However, the lifting scheme never explicitly utilizes the polyphase matrix representation or the underlying scaling and wavelet functions, and therefore makes the incorporation of nonlinearities and adaptivity into the wavelet transform more understandable. In addition, when we utilize the prediction and update filters in Section 4.1.1 to satisfy requirements other than the traditional lifting constraints, it is clear that we are sacrificing vanishing moments in the underlying scaling and wavelet functions. Thus, we can exploit the structure of the lifting scheme to build adaptive and nonlinear transforms, while carefully controlling the underlying properties of the wavelet transform.

3.3 The Update/Predict Programme

When the prediction and update operators are constructed via the polynomial lifting constraints, the output of the update step is a coarse approximation (low pass and downsampled) version of our signal. We need this coherent interpretation of the update coefficients, since they will be input to further iterations of the transform. After the first iteration, all subsequent predictions are based on updated coefficients. If we are to make effective prediction throughout the transform, we need some kind of structure in the update. However, if the prediction is performed with a nonlinear operator, it may not be possible to construct an update operator that satisfies the polynomial lifting constraints and provides a low pass interpretation of the updated coefficients.

In addition, we need to ensure that our transform is stable. Lossy processing

introduces errors into the transform coefficients, so it is crucial that the nonlinearities do not unduly amplify these errors. For compression, our goal is to use a high-order predictor in smooth regions and a low-order predictor near edges. In order to avoid sending side information on which predictor was chosen, we need to base the choice only on the $x_e[n]$. However, in lossy compression the decoder only has the quantized even coefficients $\widehat{x}_e[n]$ rather than the original coefficients $x_e[n]$. If we use locally adapted filters, then quantization errors in coarse scales could cascade across scale and cause a series of incorrect filter choices leading to serious reconstruction errors.

In the predict-then-update case, the problem of stability cannot be solved by synchronization alone, i.e., having the encoder make its choice of predictor based on quantized data. The reason is that the reconstructed values $\widehat{x}_e[n]$ are obtained from quantized low pass values $\widehat{c}[n]$. The low pass signal $c[n]$ is a function of the prediction residual signal $d[n]$, which in turn depends on what filters are chosen for prediction, as shown in Figure 3.8. Hence the encoder cannot obtain the quantized values $\widehat{x}_e[n]$ until it selects a predictor, and it cannot select a predictor without obtaining $\widehat{x}_e[n]$. If we are to employ a nonlinear lifting procedure for lossy coding, it is essential that we avoid this catch 22.

A crafty detour (developed by Davis [16]) around these problems is to perform the update step first, followed by the prediction, as shown in Figure 3.9. The relevant equations then become

$$c[n] = x_e[n] + \mathcal{U}(x_o[n]), \qquad d[n] = c[n] - \mathcal{P}(x_e[n]).$$

In the update/predict programme, the roles of the update and prediction filters are reversed. We first design a linear update filter to preserve the first $\widetilde{N}$ low-order polynomials in the data (as in Section 3.1.3). This is equivalent to adding vanishing moments to the dual wavelet function $\widetilde{\psi}(t)$. In the standard predict-first scheme, we first add vanishing moment to the primal wavelet function, as described in Section 3.1.2.

Since the update/predict lifting creates $c[n]$ prior to $d[n]$, the prediction operator

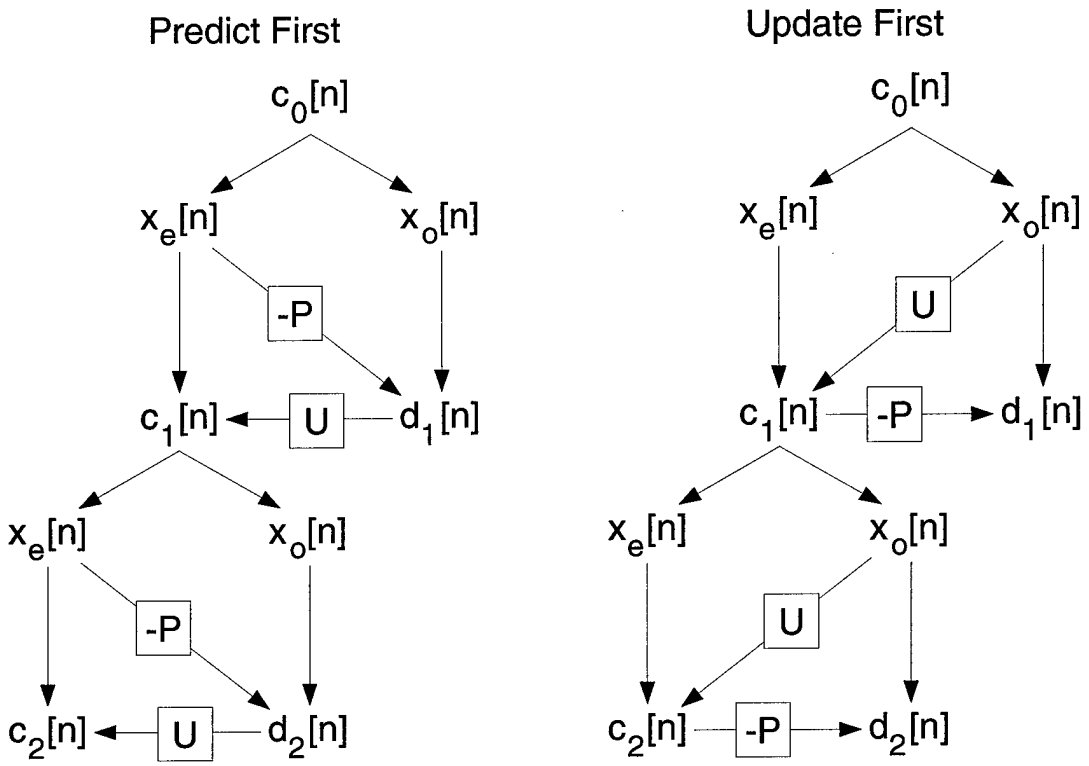


Figure 3.8 : *Two-iteration lifted wavelet transform trees with predict-first (left) and update-first (right). When predicting first, the prediction must be performed prior to construction of the coarse coefficients and iteration to the next scale. When updating first, the prediction operator is outside the loop. The coarse coefficients can be iterated to the lowest scale, quantized, and reconstructed prior to the predictions.*

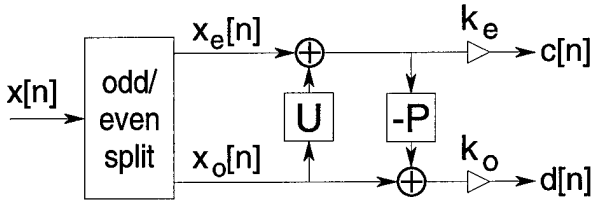


Figure 3.9 : *Update-first lifting sequence.*

can be designed to optimize performance criteria other than polynomial suppression capability, without affecting the coarse approximation $c[n]$. For example, the predictor could be a median filter, or a filter designed to minimize the prediction error energy. In Section 4.1.2, we will exploit this flexibility to design space-varying predictors that adapt to the characteristic of the signal, while completely preserving the low pass interpretation and orthogonal properties of the wavelet transform.

Chapter 4

Adaptive Transforms

4.1 Introducing Adaptivity into the Wavelet Transform

The lifting approach to wavelet design gives us a great deal of flexibility. In principle, we can use any linear, nonlinear, or space-varying predictor or update, and the lifting construction ensures that the resulting transform is invertible. We now investigate the capabilities of the lifting approach for adaptive DWTs that optimize data-based prediction measures to match the characteristics of a given signal. The motivation behind these new transforms is that better predictors will lead to more efficient signal representations. Since the compression abilities of signal transformations are key to successful signal processing algorithms [2], the adaptive transforms derived here have the potential to improve transform-based processing.

4.1.1 Scale-adaptive transforms

In Section 3.1, we derived the lifting construction based on a polynomial signal suppression/preservation argument. However, we alluded to nonlinear schemes based on other than polynomial prediction. For example, we could design a predictor for certain textural components, such as periodic patterns. More generally, we can let the signal itself dictate the structure of the predictor [17, 18].

In the scale-adaptive transform (ScAT), we adapt the predictor in each lifting stage to match signal structure at the corresponding scale. The basic idea is to use a linear N-point predictor, but require that it suppress polynomials only up to order $M < N$. The remaining $N - M$ degrees of freedom can then be used to adapt the predictor to the signal.

Specifically, at each scale we optimize the predictor over the $N - M$ degrees of freedom to minimize the spatially-averaged squared prediction error. This optimization produces predictors that can match both polynomial and non-polynomial signal structure within each scale. For example, if the signal contains a regular texture, then a relatively low-order adaptive predictor of this form may be able to match the texture much better than a pure polynomial predictor of the same order.

The optimization itself is a straightforward N-dimensional constrained least-squares problem, the constraint being that we require the predictor to suppress $M < N^{\text{th}}$-order polynomials. Let $\mathbf{x_o}$ denote the odd-indexed data we wish to predict, let $\mathbf{X_e}$, $[\mathbf{X_e}]_{n,k} = x_e[n-k]$, be a matrix composed of the even-indexed data used in the prediction, and let $\mathbf{p}$ be the vector of prediction filter coefficients. The vector of prediction errors is then given by

$$\mathbf{e} \;=\; \mathbf{x_o} - \mathbf{X_e p}\,.$$

Our objective is to find the prediction coefficients $\mathbf{p}$ that minimize the sum of squared prediction errors $\mathbf{e}^T\mathbf{e}$ while satisfying the $M < N$ polynomial constraints. Thus, we solve

$$\min_{\mathbf{p}} \|\mathbf{x_o} - \mathbf{X_e p}\|^2 \quad \text{subject to} \quad \mathbf{V}^\diamond \mathbf{p} = [1 \quad 0 \cdots 0]^T\,,$$

with $\mathbf{V}^\diamond$ an $M \times N$ matrix determined as in Section 3.1.2. Since $\mathbf{V}^\diamond$ is the first M rows of an $N \times N$ Vandermonde matrix with full rank, we are ensured that our M polynomial constraints are linearly independent. The optimal prediction coefficients for this constrained least squares problem can be efficiently computed using the QR factorization method [15, p. 567].

The optimal predictor attempts to "lock-on" to the dominant signal structure at each scale. The wavelet coefficients $d[n]$ then represent the variations of the signal from this structure. Once the optimal predictor is determined, the update is designed using the methods of Section 3.1.3 to ensure that the dominant coarse-scale (low-

frequency) structure is preserved in the coarse signal approximation that is used at the next scale.

Unfortunately, the effectiveness of this filter is limited, due to the large number of pixels at each scale (especially high scales). Our prediction filter is attempting to minimize the prediction error across the entire scale. However, our signals our non-stationary, with the structure varying significantly within each scale. It is too much to ask of a single filter to mimic the signal structure of an entire scale. Therefore, we seek an adaptive algorithm which adapts point-by-point, not just scale-by-scale.

4.1.2 Space-adaptive transforms

In addition to the scale-by-scale optimization described above, lifting permits us to *instantaneously* adapt the predictor to the signal and change the wavelet basis functions at each point. An example of such a transform is our space-adaptive transform (SpAT) [16, 17, 18]. We employ the update/predict framework of Section 3.3 and choose a predictor from a suite of predictors to minimize each $d[n]$ value.

Our adaptive algorithm performs a $\widetilde{N} = 1$ point update, and then for each n chooses the $N \in \{1, 3, 5, 7\}$ point prediction that minimizes the prediction error $d[n]$. These filters are a branch of the Cohen-Daubechies-Feauveau family [19, 20]. We chose this $(1, N)$ family because it provides the SpAT with a great deal of flexibility [16]. The $(1, 1)$ filter set corresponds to a Haar (box) wavelet transform, while the underlying wavelet functions of the $(1, 7)$ filter set are shown in Figure 4.1. These wavelet functions are relatively smooth and the synthesis functions have small side-bands, making them a good choice for compression and signal estimation.

A demonstration of the SpAT applied to a step edge is shown in Figure 4.2. The transform is able to lock-on to the dominant signal structure at each point, and avoid discontinuities and other high-order polynomial phenomena that would decrease the quality of prediction.

When this algorithm is used for signal estimation, we choose the minimizing filter

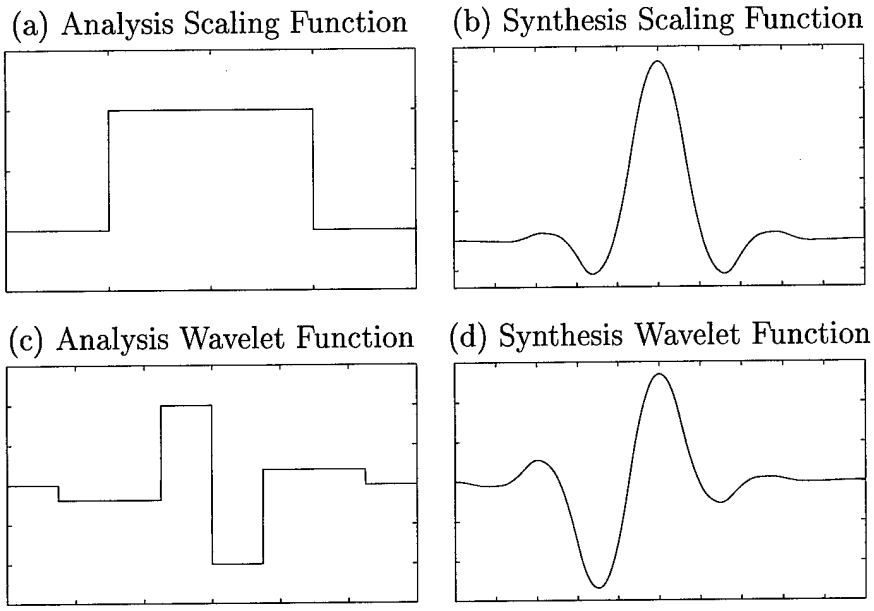


Figure 4.1 : *Top row: (a) analysis and (b) synthesis scaling functions for the order* $(1, 7)$ *Cohen-Daubechies-Feauveau filter used in the SpAT. Bottom row: (c) analysis and (d) synthesis wavelet functions. These basis functions correspond to the update-first form of lifting.*

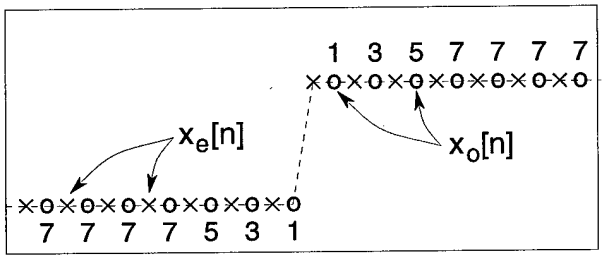


Figure 4.2 : *In the SpAT, the order* N *of the predictor varies with space* n *to minimize the wavelet coefficient value* $d[n]$*. Above each* $x[n]$ *we give the corresponding choice* $N(n)$*. As the predictor approaches an edge, it decreases* N *(chooses wavelets of smaller spatial support) in order to avoid the edge.*

for each pixel, and remember the choices as side information. However, when we perform image compression, we cannot afford this overhead. Therefore, we choose the prediction operator based on the local properties of the image [21]. For each prediction window, we analyze the data to determine if it is well approximated by a low order polynomial. If it is, then we use a high-order predictor with wide support, which corresponds to a smooth basis function. If the data does not meet our smoothness criteria, we determine which pixels in the prediction window contribute to the failure. We classify these pixels as "edge" or discontinuity coefficients. Near these edges we reduce the order of the predictor so that the neighborhood we use for prediction never overlaps the edge. In this manner we maintain high accuracy away from edges, avoid large errors in the presence of edges, and achieve the same filter choices shown in Figure 4.2.

4.1.3 Other adaptive schemes

In the above sections, we described two fundamental algorithms. In the first class (scale adaptive), we created a prediction filter at each scale by minimizing the overall mean-square prediction error at that scale. It was limited due to the low number of free variables, compared to the large number of prediction points. Therefore, the filter could not effectively lock-on to the underlying structure of the image.

In second class (spatially adaptive), we minimized the prediction error at each pixel by choosing the prediction filter from a predetermined family of filters. Without these family constraints, the algorithm could drive the prediction error to zero at each pixel, effectively removing *all* the structure from the signal.

It is possible to combine these two ideas into a single algorithm. Instead of limiting the filter to a family, we instead design the prediction filter with the ScAT algorithm (free variables used to minimize the prediction error). However, whereas the ScAT created one filter for each scale, we now design a new filter for each pixel. To prevent the filter from removing all the information from our signal, we analyze a block of

data around each pixel, and allow this local data to influence the design criteria. This makes each prediction filter more heavily dependent on local data, instead of data across the entire scale as in the ScAT.

4.1.4 Median filtering

Median filters are well known nonlinear filters, and have been successfully incorporated into wavelet-like decompositions by Goutsias and Heijmans [22], Hampson and Pesquet [23], and de Queiroz et al [24]. These decompositions show great promise for image compression, and all have structures similar to our lifting construction. Therefore, we hope to incorporate and interpret the median filter within our adaptive lifting scheme.

Consider the application of an N-point median filter predictor. For each odd coefficient $x[2n+1]$, the predictor will analyze the N nearest even coefficients $x[2(n-k)]$ (N-point data window) and choose as the prediction the median value of this data set. The detail coefficients $d[n] = x[2n+1] - P(x[2(n-k)])$ will be the difference between the odd coefficients and these median values for each data window.

Thus, for every neighborhood of N data points, the output of the median filter is a single member of the original data set. If, for example, the median value is the second data point in our window, then the output of the median filter is equivalent to applying the filter $e_2 = [0 \;\; 1 \;\; 0 \;\; \cdots \;\; 0]$ to the N-point data window. In general, let e_i be a length-N filter of zeros with a 1 in the i^{th} position. If the median value of the data is in the i^{th} position of the data window, then the output of the median filter is equivalent to applying the filter e_i to the data window. Thus, median filtering is adaptive linear filtering, with each filter chosen from the family of $\{e_i\}_{i=1}^{N}$.

Using the adaptive lifting ideas developed above, we utilize a median filter in the prediction step and then follow this operation by an adaptive update step, designed to preserve the low pass interpretation of the scaling coefficients $c[n]$. First, we compute the wavelet coefficients $d[n]$ using the median filter. For each n, we remember which

e_i was utilized. Then, a tree can be constructed to trace each scaling coefficient up to the the original data $x[n]$, as shown in Figure 4.3. For each $d[n]$ used to *lift* the coarse coefficient, we have a contribution from only two members of the original data set. This provides an *update vector* $\mathbf{h}$ as described in Section 3.1.3 and shown across the top of Figure 4.3. We apply the appropriate set of linear constraints to solve for the update filter coefficients u_i.

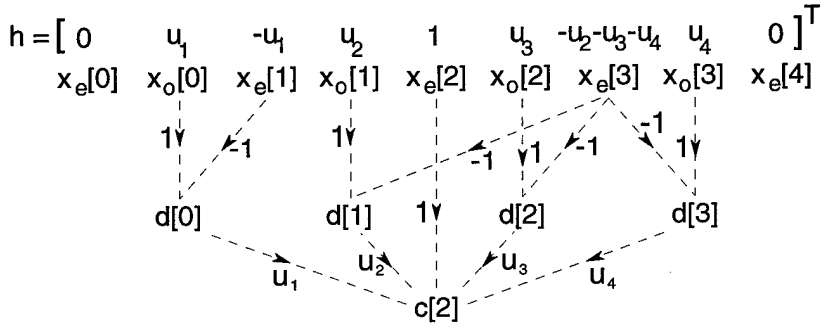


Figure 4.3 : *Median prediction with linear update filtering. An $N = 5$ point median filter prediction followed by an $\widetilde{N} = 4$ point linear update yields the update vector* $\mathbf{h}$ *shown across the top.*

Each scaling coefficient $c[n]$ is constructed as a low-order polynomial approximation to the original data. A new set of update filter coefficients u_k must be found for each n to ensure each $c[n]$ has a valid polynomial interpretation. Thus, despite the application of the nonlinear median filter in the predict step, the update filter coefficients u_i adapt to ensure the scaling coefficients satisfy linear polynomial constraints and are a low pass representation of the original data set. We can now iterate on these coefficients to maintain the recursive, multiscale properties of the wavelet transform.

If we desire an update filter of length $\widetilde{N}$, we must generate $\widetilde{N}$ update constraints. If $\widetilde{N} = 1$, then the one-point update filter coefficient becomes $u_1 = 1/2$ for any choice of median filter and any data.

For $\widetilde{N} > 1$, we typically use all the free variables to satisfy polynomial constraints. However, it is possible that a single data point will be the output of multiple median prediction filters, as shown for $x_e[3]$ in Figure 4.3. Thus, the branches of the update filter tree overlap, and the resulting linear update constraints may be incompatible (the resulting matrix $\mathbf{V}^{\diamond}$ will be poorly conditioned). In this case, we incorporate the first few polynomial constraints, and use the remaining free variables to minimize the energy of the update filter. We have found in practice that this keeps the update filter coefficients from becoming highly unbalanced, even when the update filter tree branches are greatly overlapped.

It is also possible to incorporate a median filter into the update step. Using our adaptive-filter interpretation of the median filter, we know that a median update filter will choose just one detail coefficient to update each coarse coefficient, as demonstrated in Figure 4.4. Regardless of the median choices (for the prediction or the update filters), we form each coarse coefficient by combining the even data point and the output of the update median filter, multiplied by $u_1 = 1/2$. As discussed above, such a scheme is guaranteed to satisfy the 0^{th} polynomial constraint. This is the median filter subband decomposition of Hampson and Pesquet (in their paper, it is referred to as *Nonlinear Filter 2*) [23]; it can be viewed as optimal in the sense of polynomial approximation.

Note that, for the $\widetilde{N} = 1$ point linear update and the median filter update, the output of the update filter is multiplied by $u_1 = 1/2$, regardless of the median filter choices of the prediction filter. However, for the $\widetilde{N} > 1$ point linear update, the update filter must adapt to the prediction choices; therefore, the update filter coefficients (and possibly the median filter choices) must be available as side information for the inverse transform to achieve perfect reconstruction. Thus, the $\widetilde{N} = 1$ point linear update transform and median filter update transform are better suited for image compression, especially lossless image compression [23, 25], since they can be easily modified to accommodate integer-to-integer arithmetic [26].

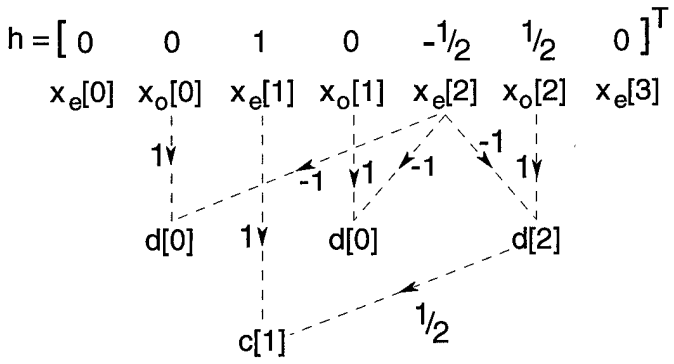


Figure 4.4 : *5-point median filter predict followed by a 3-point median filter update. The update vector* **h** *shown across the top satisfies the* 0^{th} *polynomial constraint regardless of median choices.*

4.2 Redundant Lifting

In many applications, the redundant (shift-invariant) wavelet transform often exhibits superior performance over the non-redundant wavelet transform [27]. For example, thresholding the non-redundant transform coefficients (as is done in denoising) creates pseudo-Gibbs phenomena in the neighborhood of signal discontinuities. The sizes of these artifacts are related to the actual locations of the discontinuities. In the redundant wavelet transform, we average over all possible shifts of the input signal. This averaging usually improves the mean square error (MSE) performance of the transform [27].

Our spatially adaptive lifted transforms are designed to improve performance near discontinuities. Therefore, we conjecture that our algorithms should improve denoising performance, if implemented within the redundant framework. We need only to extend the lifted wavelet transform to the redundant case.

A typical redundant wavelet transform is implemented as shown in Figure 4.5. By removing the time-varying decimators, we ensure that the transform is shift-invariant. Of course, the transform now takes N data points to NL transform coefficients (where L is the number of iterations), and not to N. Subsequent iterations of the transform

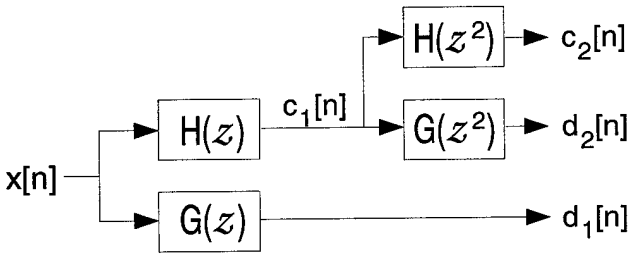


Figure 4.5 : *Two iterations of the undecimated implementation of the redundant wavelet transform. Subsequent iterations at scale l require application of expanded filters $H(z^{2^l})$ and $G(z^{2^l})$ to the coarse coefficients $c_l[n]$.*

at scale l must use versions of the original wavelet filters, expanded by 2^l. This implementation is often referred to as the "undecimated wavelet transform" [28].

However, the first step in any lifting stage is the data split; in the non-redundant transform, this is implemented with decimators, and all the lifting operations are applied to downsampled data. In the redundant transform, we do not have direct access to this data, since the decimators have been removed! Therefore, we must implement the redundant wavelet transform with the decimators intact, as shown in Figure 4.6. Such an implementation, where we compute the wavelet transform for all possible shifts and average the results, is know as the "translation invariant wavelet transform," or "cycle-spinning" [27].

In a nutshell, the lifted redundant wavelet transform is simply two non-redundant lifted wavelet transforms, intertwined at each scale. The first transforms still predicts the odd coefficients from the even coefficients, as per the standard lifting construction. The second, however, predicts the even coefficients from the odd coefficients. This is accomplished by shifting the input sequence by one and then feeding it into the same lifted transform. The output of this doubly combined lifted transform is equivalent to that from the usual redundant transform. However, by using the lifting scheme in this setting, we are now able to introduce adaptivity into the redundant transform, creating a redundant space adaptive transform using the same ideas developed in

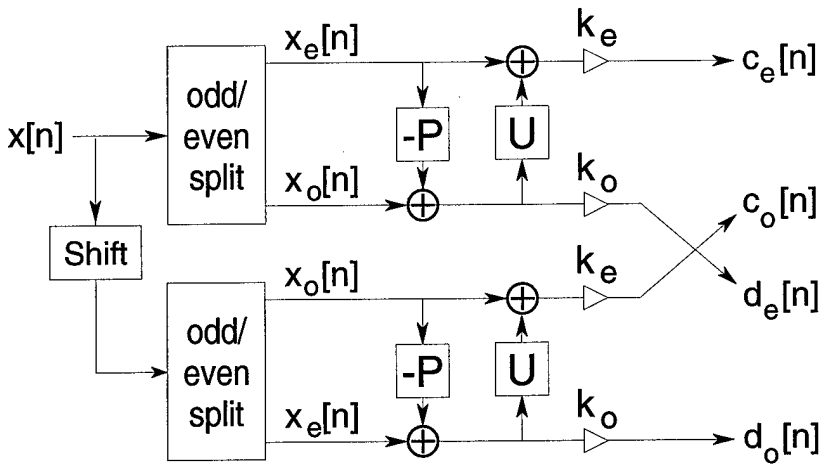


Figure 4.6 : *Lifting implementation of the redundant wavelet transform. Transform is computed on all possible shifts at each scale. Transform is iterated on both sets of coarse coefficients,* $c_e[n]$ *and* $c_o[n]$.

Section 4.1.

4.3 Multiresolution Wedgelet Transform

4.3.1 Construction of a multiresolution transform based on wedgelets

The adaptive algorithms described in section 4.1 are all based on the wavelet transform. However, wavelet coefficients decay slowly near edges, while the wedgelet transform [4] is near optimal for edges. Therefore, we propose an adaptive multiresolution algorithm based on wedgelets. This will require three modifications to the basic wedgelet transform.

First, wedgelets are designed for images in the horizon class [4], that is, binary images with a single edge along a contour. Our images are gray scale with varying amplitudes. Thus, instead of fitting each dyadic square with a single wedgelet, we fit each square with a constant function and a wedgelet. This expands our class of functions to include horizon images of varying amplitudes.

Second, we design each wedgelet to be orthogonal to the constant function. Although the wedgelets themselves do not compose an orthonormal set, we know that for each dyadic square, only a single wedgelet will be chosen. Thus, we are guaranteed that our final approximation (for any dyadic square) is composed of two orthonormal components. This modification makes our wedgelets similar to unbalanced Haar wavelets [29].

Third, the wedgelet transform is not truly multiresolution; it simply decomposes the image in space until a desired approximation error is achieved (by creating deeper and deeper dyadic splits). Therefore, we incorporate wedgelets into the lifting scheme to create a true multiresolution transform. To maintain control over the multiresolution properties of the transform, we again use an update-first architecture (see Section 3.3). Our lifted wedgelet transform has three steps:

Split: Wedgelets are nonseparable, so we begin by dividing the original image $x[n,m]$ into four disjoint subsets. This is performed by rectangular sampling:

$$x_1[n,m] = x[2n,2m], \qquad x_2[n,m] = x[2n+1,2m],$$
$$x_3[n,m] = x[2n,2m+1], \qquad x_4[n,m] = x[2n+1,2m+1].$$

Update: We combine the four polyphase components to represent a coarse approximation to the original image. Similar to the SpAT transform of Section 4.1.2, we perform a Haar update. Thus, each $x_1[n,m]$ is replaced with a coarse coefficient

$$c[n,m] = \left(x_1[n,m] + x_2[n,m] + x_3[n,m] + x_4[n,m]\right)/4.$$

Predict: We now generate the wavelet coefficients $d_k[n,m]$ as the error in predicting $x_k[n,m]$ from $c[n,m]$ for $k = 1,2,3$. For our prediction operator, we analyze a neighborhood of $c[n,m]$ and fit the best wedgelet to this data. This wedgelet is created as described above (it is comprised of a constant function and an orthogonal wedgelet). The projection onto the constant function is simply the

average value of the data. Since each wedgelet is orthogonal to the constant function, the best wedgelet is the one with the largest magnitude projection coefficient. Once this best wedgelet is found, we determine the prediction error between this wedgelet and the polyphase coefficients $x_k[n,m]$, $k = 1,2,3$. These prediction errors (our failure to be well approximated by a wedgelet) are our detail coefficients $d_k[n,m]$, $k = 1,2,3$. Thus, at each scale we find the wedgelet transform which best approximates the low pass coefficients, and use this approximation to predict the missing polyphase components.

Clearly, this algorithm is adaptive, since the *best* wedgelet is chosen to create each detail coefficient at each scale. The transform is multiresolution, due to our update-first lifting architecture. The lifting structure also ensures that our transform is computationally efficient; the transform still requires only $O(N^2)$ operations, although the scaling factor is increased by two orders of magnitude. Our choice of a Haar update preserves the orthogonality of the coarse coefficients. Since wedgelets are near optimal for certain classes of images, we expect that our multiresolution wedgelet transform will also perform well on these images.

We immediately have two minor extensions to this algorithm. First, we can pass the choice of wedgelets (and possibly the wedgelet coefficients) as side information. Second, instead of basing the choice of wedgelet on the coarse coefficients, we could choose the wedgelet which minimizes the prediction error for each set of detail coefficient (similar to the SpAT).

4.3.2 Combined wedgelet/wavelet transform

Although the wedgelet transform is near optimal for edges, it does not match the performance of the wavelet transform in smooth regions. However, it is possible to combine the wedgelet and wavelet transforms to increase the overall performance. We present two methods that achieve this goal.

First, we create a [slightly] redundant transform using both the wavelet and

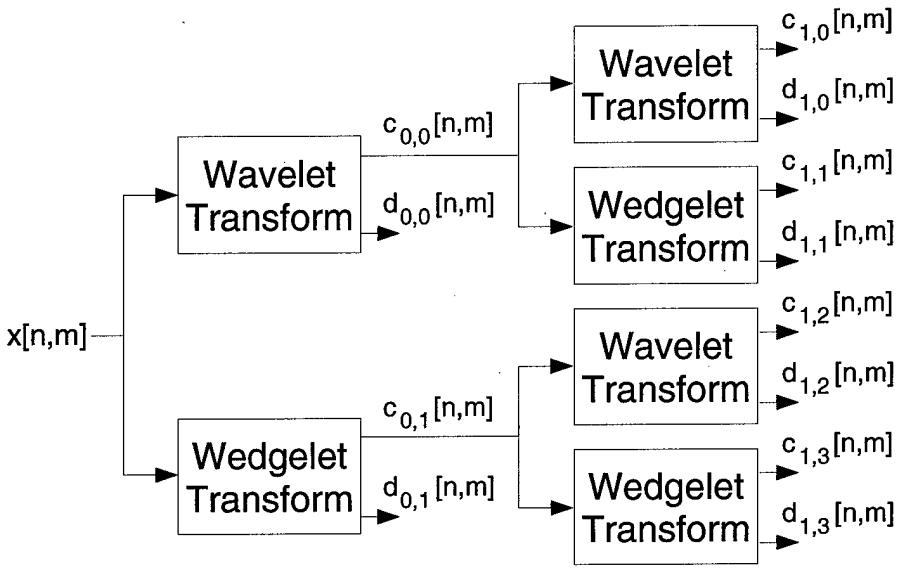


Figure 4.7 : *The [slightly] redundant wedgelet/wavelet transform. At each iteration, we take both the wedgelet and wavelet transform of the coarse coefficients. Total redundancy is approximately four.*

wedgelet basis functions. At scale 0, we take both the wedgelet and wavelet transform of our image. At scale 1, as we iterate on the coarse coefficients, we take both transforms for each set of coarse data. As we invert these transforms, we average the results. Thus, we have a system with redundancy of approximately four. An example of this transform is shown in Figure 4.7.

Second, we create a critically sampled transform which chooses between the wedgelet approximation and wavelet approximation for each pixel. This is similar to the SpAT, which chooses from the $(1, N)$ family of filters at each point. In this case, at each coarse coefficient, we determine if the coefficient of the best wedgelet is sufficiently large. If so, then we are most likely in an image region dominated by a large edge [30]. In this case, we assume that the wedgelet is a good predictor, and use it to create the detail coefficients. If not, then a Haar transform (or a smoother predictor) is used to create the details. This transform has the advantage of increased flexibility without increased redundancy. This makes it potentially well-

suited for certain image compression applications. We can extend the algorithm for image denoising applications by determining the best choices and passing these as side information.

4.3.3 Nonlinear approximation with wedgelets

As shown in [4] and mentioned in Section 4.3.1, wedgelets form a near optimal representation for images in the horizon class. Therefore, our multiresolution wedgelet transform has the potential to create sparse representations for these images.

Figure 4.8 demonstrates this potential. We created a test image constructed of smooth (polynomial) regions separated by discontinuities along contours. We transformed the image using the Haar, Daubechies $(7,9)$, and multiresolution wedgelet transforms, and retained the N largest coefficients. We then used these coefficients to reconstruct the image and measured the mean square error (MSE) between our approximation and the original image. The decay of this approximation error represents the compression potential of the transform [31].

As shown in Figure 4.8, the error in the wedgelet approximation decays faster than both the Haar and Daubechies $(7,9)$ approximations. We empirically determined the error decay for this test image. The Haar and Daubechies $(7,9)$ errors decay as $N^{-1.5}$, while the multiresolution wedgelet error seems to decay as N^{-2}. This is very encouraging.

Figures 4.9 and 4.10 display our test image and approximations by each of the three transforms. The approximations in Figure 4.9 were created by retaining $N = 500$ coefficients, while the approximations in Figure 4.10 were created with $N = 1700$ coefficients. These choices of N were made to keep the PSNR of the approximations in the 30–40 dB range. Clearly, the wedgelet transform provides the best approximation along the discontinuities, as expected.

To further compare the three algorithms, we plotted the magnitude of the approximation error for $N = 350$ (chosen smaller to accentuate the errors) in Figure

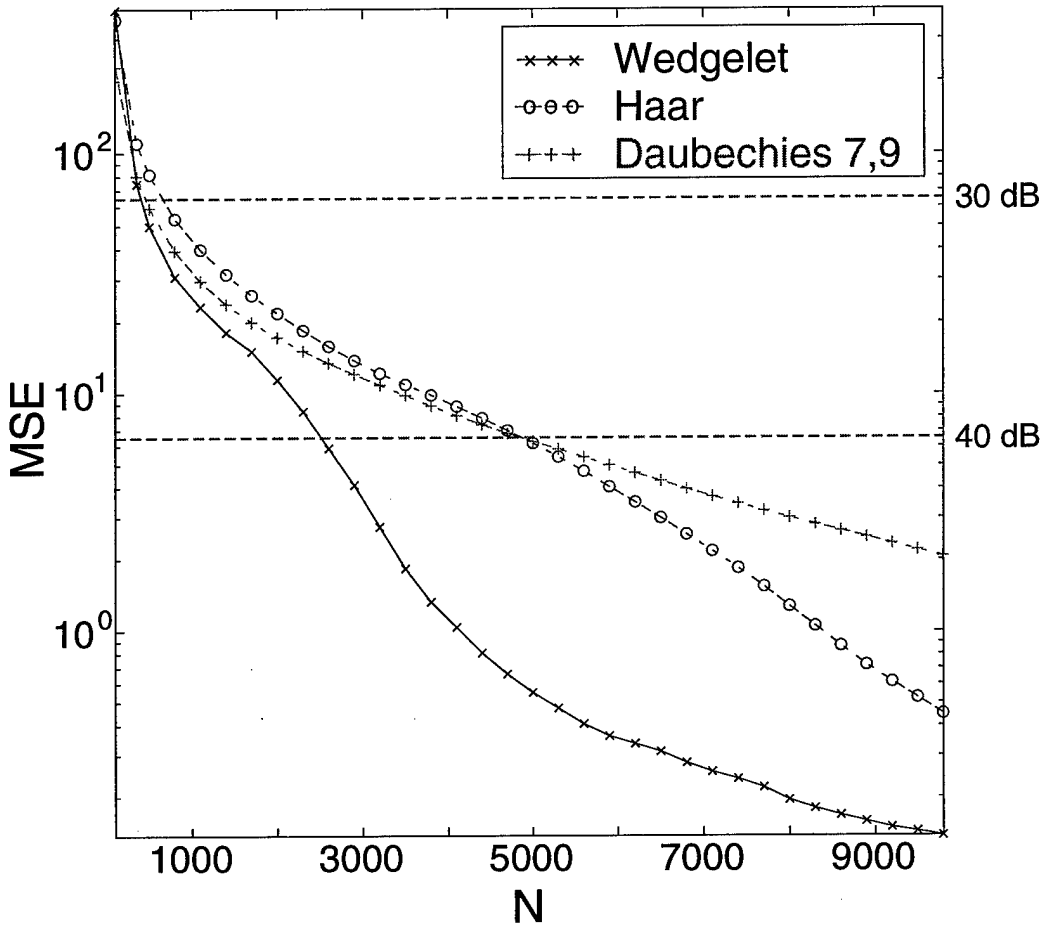
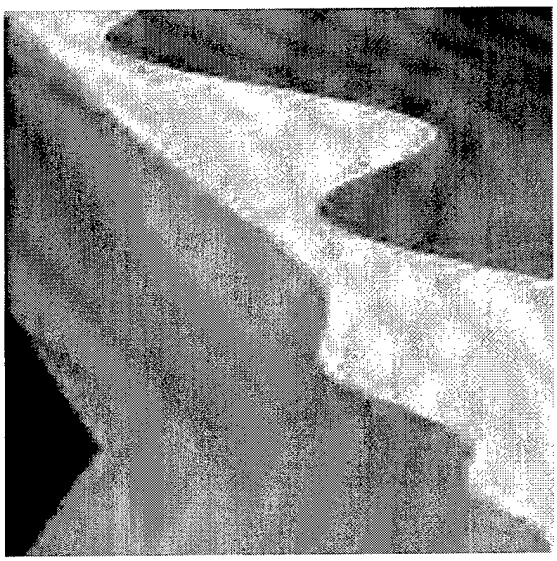


Figure 4.8 : *Mean square error (MSE) of approximations to our test image (smooth regions separated by discontinuities along contours) versus the number of coefficients used in the approximation. The multiresolution wedgelet transform looks very promising.*

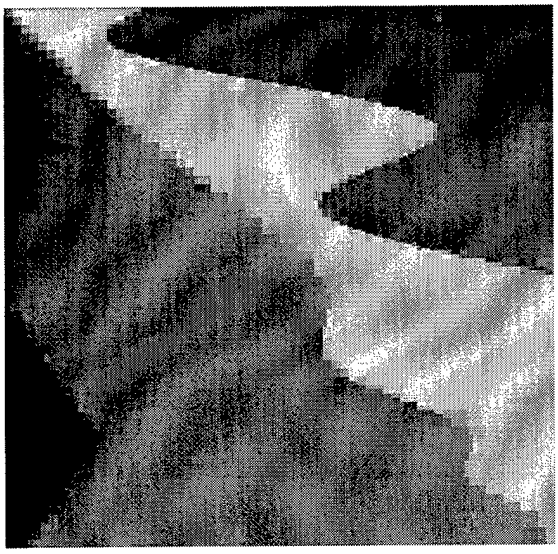
4.11. The wedgelet transform and the Haar transform both struggle in the smooth regions. The wedgelet transform does a better job of reconstructing the edges. The Daubechies $(7, 9)$ transform performs well in the smooth regions, but suffers badly near the discontinuities. This strengthens our hypothesis that the ideal solution is a combination of the wedgelet transform and a smooth predictor, as described in Section 4.3.2.

Original Image

Daubechies $(7, 9)$ *approximation*



Haar approximation



Multiresolution wedgelet approximation

Figure 4.9 : *Approximations to our test image. In each case,* $N = 500$ *coefficients were used for the approximation. The wedgelet transform provides improved MSE performance and does a superior reconstruction job in the vicinity of the discontinuities.*

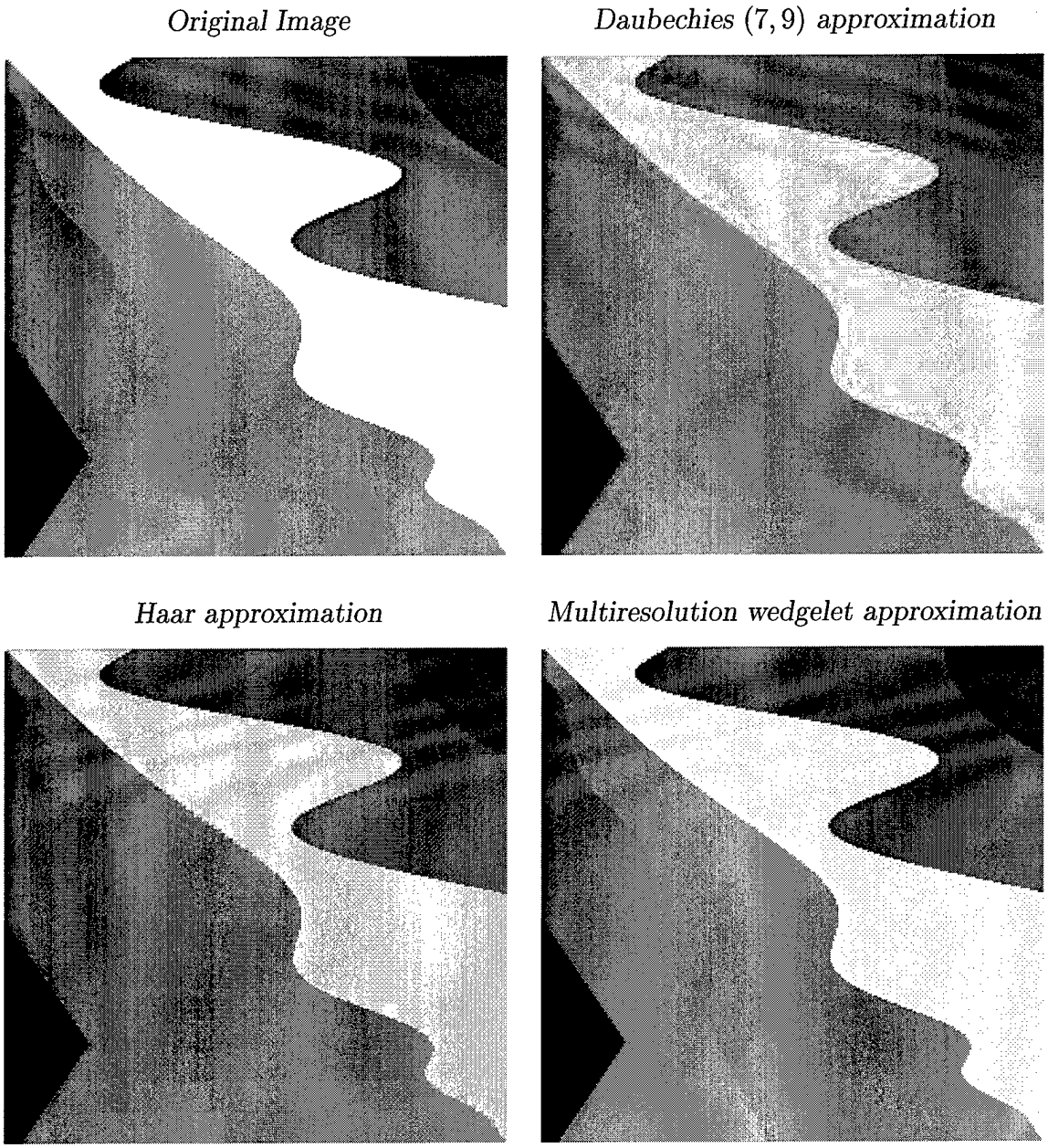


Figure 4.10 : *Approximations to our test image. In each case, $N = 1700$ coefficients were used for the approximation. The wedgelet transform provides improved MSE performance and does a superior reconstruction job in the vicinity of the discontinuities.*

Daubechies $(7,9)$ *error*

Haar error

Multiresolution wedgelet error

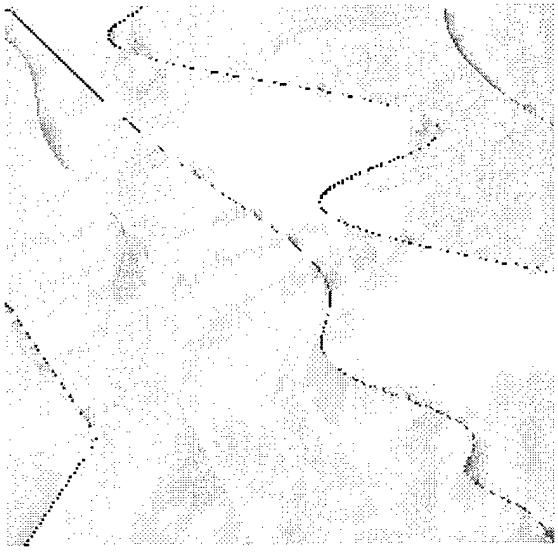


Figure 4.11 : *Approximation errors for our test image. In each case,* $N = 350$ *terms were used in the approximation. The wedgelet and Haar transforms both struggle in the smooth regions, while the wedgelet transform has the best representation of the edges. The Daubechies* $(7,9)$ *transform performs well in the smooth regions, but suffers near the discontinuities.*

4.3.4 Wedgelets for image analysis

As mentioned in Section 4.3.1, to find the "best" wedgelet, we must compute (at each pixel) the projection of the image onto all the wedgelets and choose the one with the largest magnitude coefficient. We also proposed in Section 4.3.2 to use this coefficient to make adaptive decisions. If the coefficient exceeded some threshold, we used wedgelets to perform our prediction; otherwise, we used a smoother wavelet.

However, there is a great deal of information embedded in these wedgelet coefficients. For example, if a region of an image is well modeled by a particular wedgelet, then the coefficient of this wedgelet will be much larger than the other coefficients. However, if the region is not well modeled by *any* wedgelet, then all the wedgelet coefficient will tend to be small. We can compute the variance or entropy of the set of wedgelet coefficients for each point in the image, and use these statistics for analysis. An example of the wedgelet variances for our test image is shown in Figure 4.12. The dark areas represent image regions with high wedgelet variance, while the light areas correspond to low variance. Figure 4.12 demonstrates that the areas of high wedgelet variance are very localized and correspond to the edges of the image, where we expect the wedgelets to perform well.

Thus, by analyzing the statistics of the wedgelet coefficients, we gather additional information about our image. The above example demonstrates how to localize areas near edges and can be used to better adapt our predictor to the image. However, the same approach could be used to determine different types of texture or to perform image segmentation.

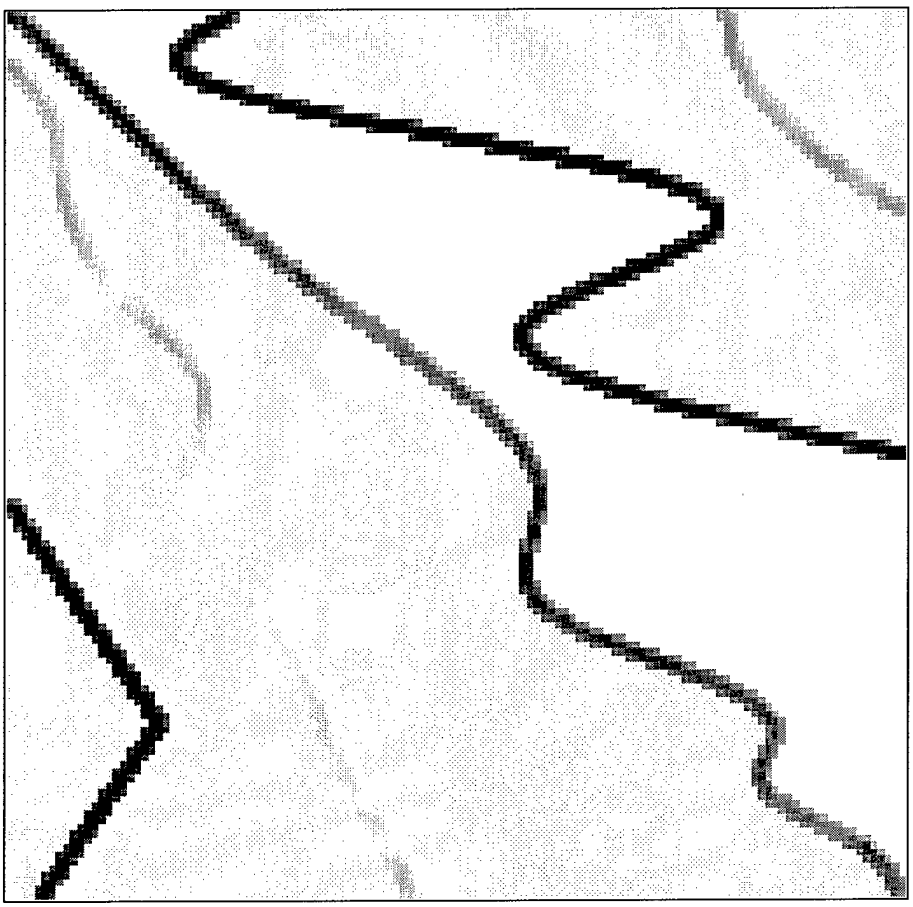


Figure 4.12 : *Wedgelet coefficient variances for the test image of Figure 4.9. Dark areas correspond to high variance, and are localized near the edges of the image. Light areas represent regions of low variance (where wedgelets perform poorly).*

Chapter 5

Applications

In this chapter, we demonstrate the utility of the adaptive algorithms developed earlier. We have two main applications: image compression and signal denoising.

5.1 Image Compression

5.1.1 Compression with the SpAT

Figure 5.1 shows compression results for an edge-dominated test image. This image was constructed by superimposing texture on shapes of different magnitudes and orientations. The original image was transformed and compressed to 0.67 bits-per-pixel (BPP) (12:1 compression) using an embedded zero-tree encoder [32]. For simplicity, we compress the zero-tree symbol stream with a Huffman coder, and make no effort to compress the quantization bit stream.

We notice in Figure 5.1 that images transformed with the Daubechies $(7, 9)$ and linear $(1, 7)$ lift suffer from blurring and ringing around the edges. However, the image transformed with our adaptive lifted algorithm has much sharper edges. Ringing is reduced, edge sharpness is maintained, and the background texture is not significantly corrupted. These improvements are very visible in the closeup shown in Figure 5.2. The reason for these improvements is that edges in our new transform are represented in a more compact fashion, and as a result there is less degradation of the image when we zero out small, non-zero coefficients.

As a performance metric, we computed the peak signal to noise ratio (PSNR),

$$\mathrm{PSNR} = 20 \log_{10} \left(\frac{\max |x_i|}{\sqrt{\sum (x_i - \widehat{x}_i)^2 / N}} \right), \tag{5.1}$$

where x_i is the i^{th} pixel of our original image, $\widehat{x}_i$ is the i^{th} pixel of our reconstructed image, and N is the total number of pixels. The PSNR curve (Figure 5.3) demonstrates that, for this edge-dominated test image, the adaptive algorithm has better PSNR performance than both the Daubechies $(7,9)$ and linear $(1,7)$ lift transforms. The Daubechies $(7,9)$ PSNR curve is shown for reference only; our goal is to improve the performance of the linear $(1,7)$ lift though adaptivity.

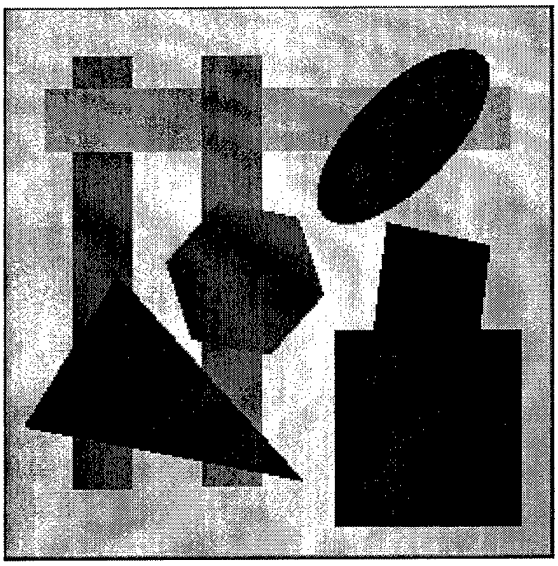
In Figure 5.4, we see the result of our adaptive lifting algorithm on the image *cameraman*, compressed to 0.32 BPP (25:1 compression). Our prediction decisions were based on data quantized to 7 iterations of the zero-tree encoder to ensure that the decoder and encoder were synchronized. While ringing has been reduced in the horizontal and vertical edges, there are still some ringing artifacts in the diagonal direction. The reason for these remaining artifacts is that we are using a separable transform in which we seek to avoid horizontal and vertical edges.

Note in Figure 5.5 the PSNR performance of our adaptive algorithm over the linear $(1,7)$ lift. Each point on the PSNR curve was generated with decoder/encoder synchronization. Again, the performance of the popular Daubechies $(7,9)$ transform is shown for reference. Although our adaptive algorithm does not match the PSNR performance of the Daubechies $(7,9)$ transform, the visual quality of our algorithm is comparable, due to the reduction in edge artifacts. In general the adaptive algorithm results in much sharper decoded images. We conjecture that introducing adaptivity into the Daubechies $(7,9)$ transform (a potential area of future research) would result in further PSNR increases.

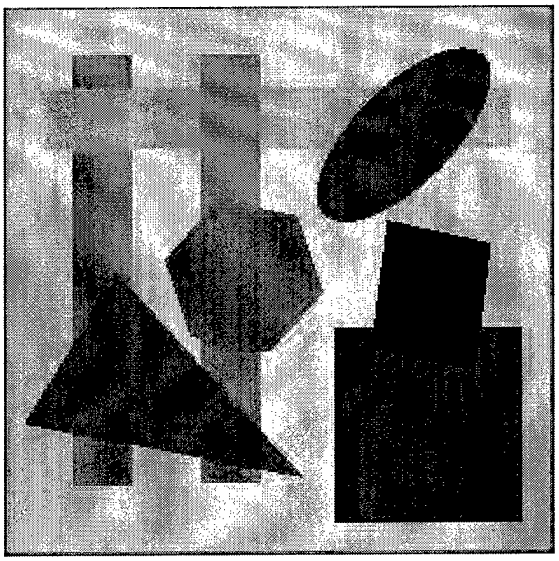
5.2 Signal Denoising

In this section, we compare the performance of the new adaptive transforms with some of the standard Daubechies wavelets for signal denoising applications. First, we compare the entropies of the coefficient distributions of several well-known test signals to assess the level of compaction afforded by the new transforms. Second,

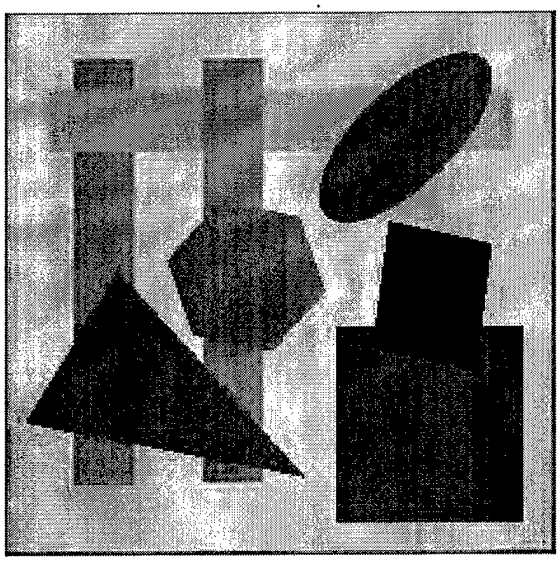
Original Image



Compressed with Daubechies $(7, 9)$



Compressed with Linear $(1, 7)$ *Lift*



Compressed with Adaptive Lift

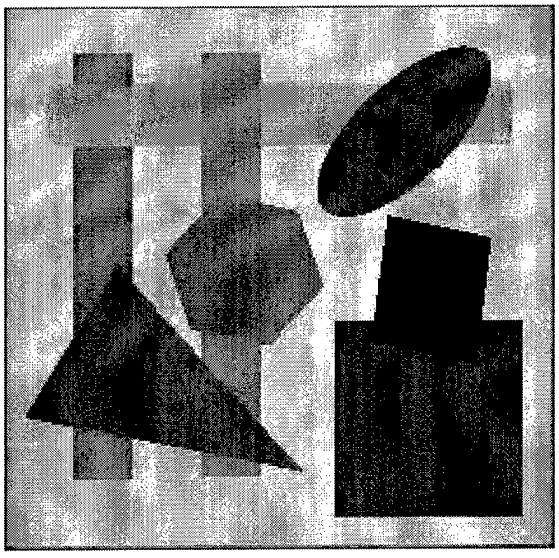


Figure 5.1 : *Edge dominated image with texture, compressed to 0.67 BPP (12:1 compression). Note the ringing around the edges of the square in the Daubechies* $(7, 9)$ *and linear* $(1, 7)$ *lift images that is eliminated by the adaptive lift.*

Compressed with Daubechies $(7, 9)$ *Compressed with Adaptive Lift*

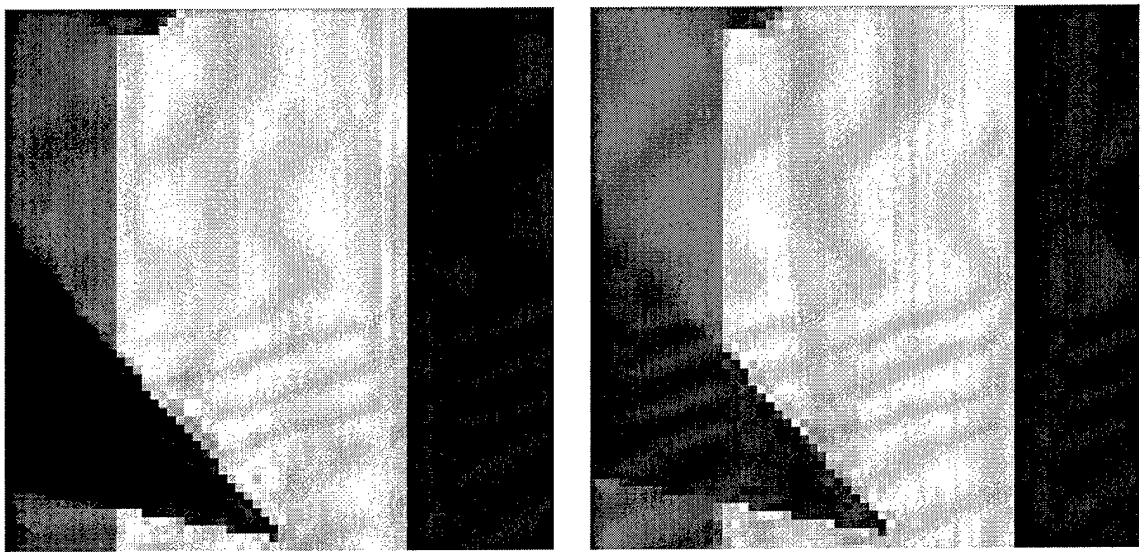


Figure 5.2 : *Close-up of edge dominated image with texture, compressed to 0.67 Bits-Per-Pixel (BPP) (12:1 compression). Note the sharp edges and reduced ringing with the adaptive algorithm.*

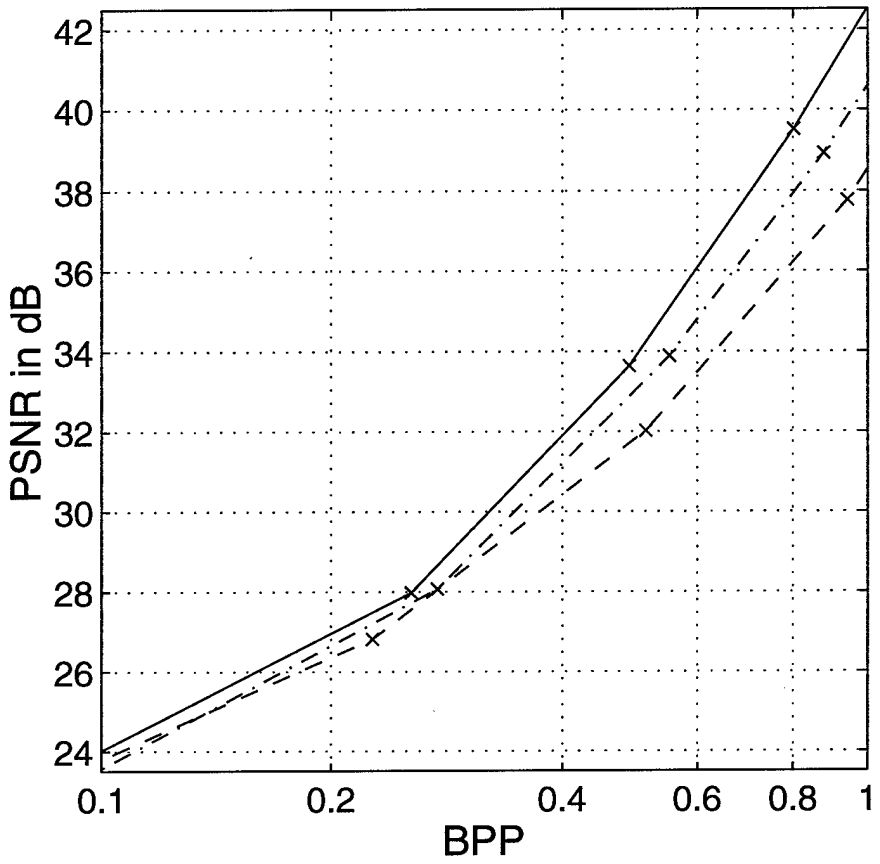


Figure 5.3 : *Peak Signal-to-Noise Ratio (PSNR) curves for the edge-dominated test image of Figure 5.1. This test image was designed to demonstrate the potential gains of the adaptive lift. The adaptive algorithm (solid line) outperforms the Daubechies* $(7,9)$ *transform (dash-dot) and the* $(1,7)$ *linear lift (dash). The encoder and decoder were synchronized for the adaptive algorithm.*

Cameraman Image

Compressed with Daubechies $(7, 9)$

Compressed with Linear $(1, 7)$ *Lift*

Compressed with Adaptive Lift

Figure 5.4 : *Cameraman image compressed to 0.32 BPP (25:1 compression).*

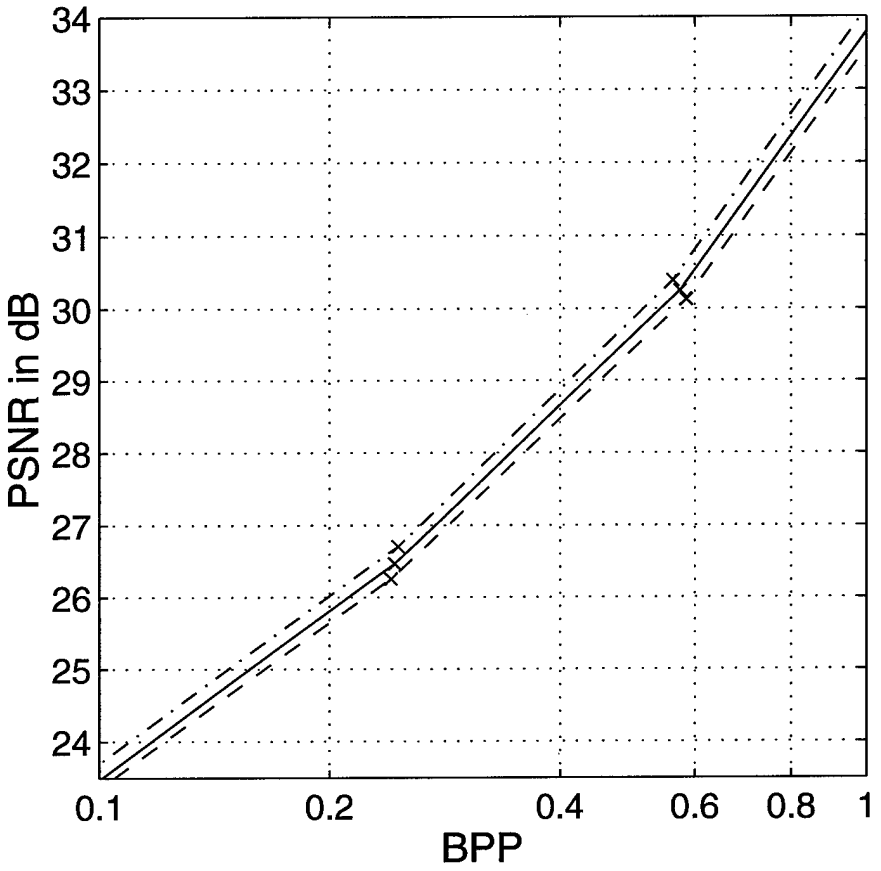


Figure 5.5 : *PSNR curves for the cameraman image. The adaptive algorithm (solid line) outperforms the linear* $(1,7)$ *lift (dash), but it does not meet the PSNR performance of the Daubechies* $(7,9)$ *transform (dash-dot). However, edge artifacts are significantly reduced by the adaptive algorithm. The encoder and decoder were synchronized for the adaptive algorithm.*

we compare the performance of the new transforms in a signal denoising application. Third, we compare the redundant implementation of our adaptive transform against other redundant transforms. Finally, we apply our multiresolution wedgelet transform to image denoising.

5.2.1 Entropy comparison

The entropy of the transform coefficient distribution is a common measure of the efficiency of a signal transform [33]. If we collectively denote the scaling and wavelet coefficients by $\{w_i\}$, then the entropy is defined as

$$H(\mathbf{w}) \;=\; \sum_i |w_i|^2 \log_2 |w_i|^2,$$

assuming the normalization $\sum_i |w_i|^2 = 1$.

Table 5.1 compares the entropies of the Daubechies-8 (D8) and Daubechies-2 (Haar) DWTs to those obtained using the ScAT and SpAT. The ScAT used an $N = \widetilde{N} = 4$ lifting construction, with $M = 3$ vanishing moments enforced. The first four signals, Doppler, Blocks, Bumps, and HeaviSine, are standard test signals introduced in [2]. The last signal, DoppelBlock, is a concatenation of the Doppler signal and the Blocks signal (hence it contains both smooth and edgy signal elements). All signals were 1024 samples long. The entropies in Table 5.1 show that both adaptive transforms perform nearly as well (or better) than the D8 or the Haar in each test case.

5.2.2 Denoising with non-redundant wavelet transforms

Because DWTs provide such a parsimonious representation for wide classes of signals, the DWT has proved to be a powerful tool for noise removal. The basic "wavelet denoising" programme [2] is described as follows. We observe L samples $\{x[n]\}$ of an unknown function f with additive i.i.d. Gaussian noise $\{\eta[n]\}$:

$$x[n] \;=\; f[n] + \eta[n], \qquad n = 0, 1, \ldots, L-1.$$

Table 5.1 : *Entropy results for various signals and transforms.*

Signal	**Entropy**			
	D8	Haar	ScAT	SpAT
Doppler	2.84	3.15	2.88	**2.57**
Blocks	2.60	2.62	2.52	**2.32**
Bumps	3.54	3.65	3.53	**3.23**
HeaviSine	2.26	2.50	**2.25**	2.27
DoppelBlock	3.54	3.60	3.41	**3.15**

We compute the DWT of x and apply a "threshold" nonlinearity to the wavelet coefficients. When a "hard-threshold" is applied, all the small wavelet coefficients (those with magnitude below a given threshold T) are set to zero, while all other coefficients are unaffected. A "soft-threshold" sets very small coefficients to zero and reduces all other coefficients by the threshold amount T. In both cases, the scaling (coarse) coefficients are not modified. The threshold T is chosen in proportion to the standard deviation of the noise. If the data signal is pure Gaussian white noise, then this universal threshold guarantees that asymptotically, as the number of data points increases, the wavelet thresholding estimator tends to the zero function, with probability one. The inverse DWT of the thresholded coefficients produces a "denoised" signal. For more information see [2].

However, in [2] the thresholds were derived only for orthogonal wavelet transforms. In [34], Berkner and Wells propose new thresholds that take into account correlations induced by redundant wavelet transforms and biorthogonal wavelet transforms. As shown in Section 3.2, the SpAT and ScAT are biorthogonal transforms, since each is implemented with one lifting stage. Thus, in our adaptive transforms (and in all our redundant transforms), we adapt the thresholds for each data point and at each scale

Table 5.2 : *Denoising: MSE for various signals and transforms. Each signal is corrupted with additive white Gaussian noise with standard deviation equal to 10% of the peak signal magnitude.*

Signal	**MSE**			
	D8	Haar	ScAT	SpAT
Doppler	0.030	0.049	**0.027**	0.032
Blocks	0.040	**0.029**	0.034	0.032
Bumps	0.032	0.034	**0.028**	0.031
HeaviSine	0.014	0.031	**0.014**	0.016
DoppelBlock	0.046	0.044	0.042	**0.040**

to compensate for noise correlations and variances.

Also, in [35] smaller thresholds than those proposed in [2] were found to perform better (with respect to mean square error) in many denoising applications. Smaller thresholds (and a hard threshold for redundant denoising) are also recommended in [27]. Thus, the experimental thresholds in [35] are used for all of our image denoising and redundant denoising experiments (modified to compensate for noise correlations, where appropriate).

Table 5.2 provides the mean square error (MSE) performance of the four transforms and five signals discussed in Section 5.2.1 above. In this experiment, white Gaussian noise of standard deviation $0.1 \times \max_n |x[n]|$ was added to each of the test signals. The MSEs in Table 5.2 show again that both adaptive transforms perform nearly as well as (or better than) the D8 and the Haar in each test case. An example of the denoised signals is shown in Figure 5.6. These graphs were generated for the DoppelBlock signal with additive white Gaussian noise, standard deviation at 5% of max signal magnitude. The ScAT performs well on the smooth regions, while the SpAT performs well on both the smooth and edge-dominated regions.

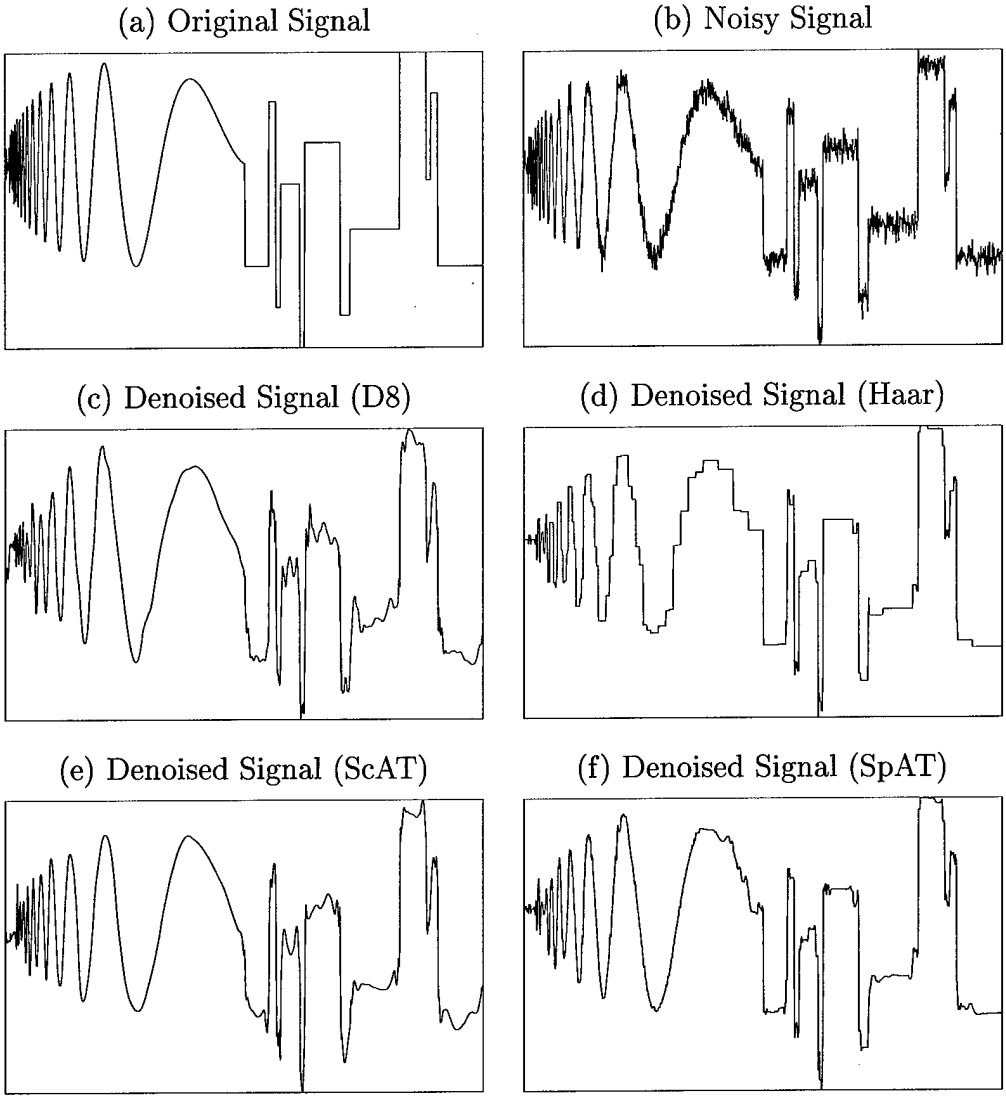


Figure 5.6 : *Top row: (a) original DoppelBlock signal (concatenation of Doppler and Blocks); (b) signal with additive white Gaussian noise, standard deviation equal to 5% of maximum signal magnitude. Second row: (c) signal denoised with the Daubechies 8 orthogonal wavelet transform; (d) signal denoised with the Haar transform. Bottom row: (e) signal denoised with the ScAT transform; (f) signal denoised with the SpAT transform.*

Table 5.3 : *Denoising: PSNR for various images and transforms. Each image is corrupted with additive white Gaussian noise with standard deviation equal to 15% of the peak image magnitude.*

Image	**PSNR (dB)**			
	D8	Haar	ScAT	SpAT
Cameraman	22.9	23.0	**23.6**	23.3
Lenna	23.4	22.7	**24.3**	23.5
Building	22.4	22.8	**23.2**	23.0
Bridge	21.4	21.3	**21.9**	21.6
Fruit	24.6	24.3	**26.2**	25.1

For our next experiment, we applied our algorithms to several standard images. We extended our algorithms to two-dimensions in a separable fashion; at each scale, our transforms are applied to the rows and then the columns of the image. We added white Gaussian noise (at 15% of max image magnitude) and then denoised those images (using the thresholds of [35], modified for our biorthogonal and adaptive transforms). As our performance metric, we again computed the peak signal to noise ratio (PSNR) as defined in equation [5.1]. Our PSNR results are shown in Table 5.3, while an example of the denoised cameraman image is displayed in Figure 5.7. The ScAT demonstrates improved PSNR performance over the other transforms. The SpAT improves PSNR performance over the non-adaptive orthogonal transforms while significantly increasing edge crispness and preserving image texture.

Remark: no results have been presented for our space/scale adaptive transform presented in Section 4.1.3. Although this transform was very interesting, it did not perform well in our denoising experiments. We attribute this to two factors. First, by allowing the prediction filters to adapt to a much smaller set of data, the resulting filters can become highly asymmetric. These filters tend to not converge to

(a) Denoised Image (D8)

(b) Denoised Image (Haar)

(c) Denoised Image (ScAT)

(d) Denoised Image (SpAT)

Figure 5.7 : *Top row: (a) Cameraman image denoised with the Daubechies 8 orthogonal wavelet transform, 21.4 dB PSNR, (b) image denoised with the Haar transform, 21.6 dB PSNR. Second row: (c) image denoised with the Scale Adaptive Transform (ScAT), 22.1 dB PSNR, (d) image denoised with the SpAT transform, 21.9 dB PSNR. The ScAT yields the highest PSNR while the SpAT significantly reducing ringing around edges.*

continuous functions, and they accentuate quantization errors when our signals are reconstructed. Second, following such an adaptive prediction by an adaptive update is very problematic. The update filters tend to also become asymmetrical, again accentuating quantization errors. In addition, in many cases the combination of skewed prediction coefficients leads to lifted update constraints which can not be solved; the update matrix $\mathbf{V}^{\diamond}$ described in Section 3.1.3 becomes ill-conditioned (or singular). Thus, we do not recommend the space/scale adaptive transform for signal denoising at this time, although we intend to pursue future research with this algorithm.

5.2.3 Denoising with redundant transforms

We now denoise our test signals using the redundant wavelet transform. We compare a redundant version of our SpAT against redundant D8 and Haar transforms. Our thresholds must again be increased to account for the correlations introduced by the biorthogonal nature of the SpAT. Also, our thresholds must be increased for all three transforms due to correlations introduced by redundancy [34]. We now use a hard threshold, since this outperforms soft thresholding within the context of redundant transforms [27].

In Table 5.4, we present the denoising performance of the redundant transforms. These signals were corrupted with white Gaussian noise (standard deviation equal to 5% of the maximum signal magnitude). In all cases, a hard threshold was applied. As expected, our adaptive algorithm is competitive in all cases, performing nearly as well as (or better than) the D8 and Haar transform for each test case. An example of the denoised DoppelBlock signal is shown in Figure 5.8.

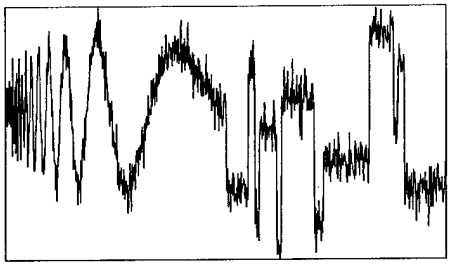
5.2.4 Denoising with multiresolution wedgelet transforms

We now compare the multiresolution wedgelet transform against other transforms in an image denoising environment. We expect the wedgelet transform to perform well near edges, but to have problems in smooth regions and regions of texture. Therefore,

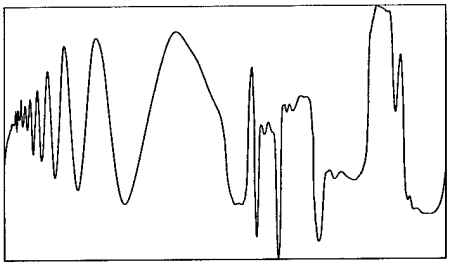
Table 5.4 : *Denoising: MSE for various signals and redundant transforms. Each signal is corrupted with additive white Gaussian noise with standard deviation equal to 5% of the peak signal magnitude. In all cases, a hard threshold was applied.*

Signal	**MSE**		
	redund D8	redund Haar	redund SpAT
Doppler	**0.010**	0.017	0.012
Blocks	0.012	**0.006**	0.008
Bumps	0.012	**0.011**	**0.011**
HeaviSine	**0.007**	0.009	0.008
DoppelBlock	0.015	0.014	**0.013**

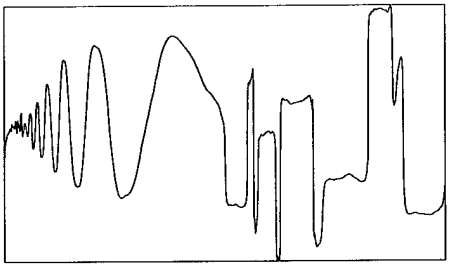
(a) Noisy Signal



(b) Denoised Signal (Redundant D8)



(c) Denoised Signal (Redundant Haar)



(d) Denoised Signal (Redundant SpAT)

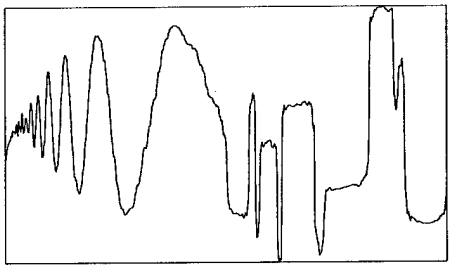


Figure 5.8 : *Top row: (a) DoppelBlock signal with additive white Gaussian noise, standard deviation equal to 5% of maximum signal magnitude; (b) signal denoised with the Daubechies 8 redundant wavelet transform. Second row: (c) signal denoised with the redundant Haar transform; (d) signal denoised with the redundant SpAT transform.*

we use our combined wedgelet/D4 transform; at each scale, the results of denoising with the wedgelet transform are averaged with the results of the D4 transform.

We added white Gaussian noise (at 15% of peak image magnitude) and then denoised those images (using the hard thresholds of [35]). As our performance metric, we again computed the peak signal to noise ratio (PSNR) as defined in equation [5.1], as well as the L^∞ error. As shown in Tables 5.5 and 5.6, the combined transform exceeds the performance of the other non-adaptive transforms used here (the Daubechies 8 wavelet, Daubechies 4 wavelet, and the Haar wavelet). Figure 5.9 shows an example of the denoised "fruit" image. As expected, the wedgelet transform performs very well in the vicinity of edges.

Table 5.5 : *Denoising: PSNR for various images and transforms. Each image is corrupted with additive white Gaussian noise with standard deviation equal to 15% of the peak image magnitude.*

Image	**PSNR (dB)**			
	D8	*D4*	*Haar*	*Combined Wedgelet & D4*
Cameraman	24.4	24.8	24.6	**25.7**
Lenna	24.9	24.8	24.2	**26.0**
Building	24.0	24.1	24.8	**25.3**
Bridge	22.5	22.3	22.2	**23.8**
Fruit	26.2	26.4	25.6	**27.2**

(a) Noisy Image

(b) Denoised Image (D8)

(c) Denoised Image (Haar)

(d) Denoised Image (Wedgelet & D4)

Figure 5.9 : *Top row: (a) noisy "fruit" image (white Gaussian noise with standard deviation equal to 15% of maximum image magnitude, PSNR = 20 dB); (b) image denoised with the Daubechies 8 wavelet transform, 26.2 dB PSNR. Second row: (c) image denoised with the Haar transform, 25.6 dB PSNR; (d) image denoised with the combined Wedgelet/D4 transform, 27.2 dB PSNR. PSNR is improved and ringing is reduced.*

Table 5.6 : *Denoising: L^∞ error for various images and transforms. Each image is corrupted with additive white Gaussian noise with standard deviation equal to 15% of the peak image magnitude.*

Image	**L^∞ by Algorithm**			
	D8	*D4*	*Haar*	*Combined Wedgelet & D4*
Camera	0.71	0.80	0.83	**0.68**
Lenna	0.62	0.63	0.73	**0.57**
Building	0.63	0.68	0.69	**0.58**
Bridge	0.62	0.64	0.68	**0.60**
Fruit	0.57	0.59	0.64	**0.53**

Chapter 6

Discussion And Future Work

6.1 Contributions of this Thesis

In this thesis, we developed two new adaptive wavelet transforms (the SpAT and the ScAT) for signal and image estimation. We also successfully modified the SpAT for image compression and redundant denoising. We built an adaptive transform around a median filter. Finally, we created a multiresolution wedgelet transform, and built several extensions, all of which work particularly well on image edges. These transforms were made possible by our interpretation of the lifting framework. The success of these transforms was demonstrated with applications to signal denoising, image denoising, and image compression.

6.2 Potential for Future Research

There are several areas of potential research that we find interesting. We present here three of the most promising:

6.2.1 Image compression with the wedgelet transform

Due to the encouraging preliminary results of Section 4.3.3, we conjecture that our multiresolution wedgelet transform may be well suited for compression on certain types of images. By combining the wedgelet transform and a "smoother" predictor as discussed in Section 4.3.2, the transform may perform well on more general images than the horizon model. Since the wedgelet transform is constructed with an update-first architecture, we can create and quantize all the coarse coefficients before making

any wedgelet prediction decisions. Thus, the decoder and encoder can be synchronized (as in the SpAT) with no side information, ensuring that incorrect prediction decisions do not contribute to reconstruction error. We intend to make these modifications and determine to what class of images (and to what quantization levels) the wedgelet transform is well suited.

Also, at its most basic level, the wedgelet transform is simply a pruned tree. Our pruning is performed in a greedy fashion, i.e., at each node we choose the wedgelet which is the "best fit" or minimizes our prediction error. It may be possible to find a *global* optimum, looking across scale as we prune.

6.2.2 Signal estimation with redundant transforms

All the image transforms created in this thesis can be implemented as redundant transforms. Since our adaptive transforms demonstrated improved signal denoising performance over non-redundant transforms, we expect redundant versions of our transforms to out-perform redundant versions of our competitors.

6.2.3 Signal estimation/analysis with the wedgelet transform

The combined wedgelet/wavelet transform (described in Section 4.3.2) performed very well in a signal denoising environment. However, it still suffered in smooth regions or regions of texture. At each scale the results of the two transforms are averaged; thus, if the wedgelet transform performs poorly, so will the overall transform.

Our critically sampled wedgelet/wavelet transform (also described in Section 4.3.2) has the potential to solve this problem, since it chooses either wedgelets or smooth wavelets at each pixel. Our current decisions are based on the size of the wedgelet coefficient, or the statistics of those coefficients (as described in Section 4.3.4). The decision for each pixel is made independent of the other pixels. However, better decisions could be made by analyzing the statistics of the wavelet coefficients, both across scale and within scale as done in [36].

Finally, these statistics (and the statistics of the wedgelet coefficients) could be used to improve the processing of the transform coefficients. As described in Section 4.3.4, this information could also be used to determine image texture or perform image segmentation.

Bibliography

[1] David Marr. *Vision.* W. H. Freeman, New York, 1982.

[2] D. L. Donoho. Denoising by soft-thresholding. *IEEE Trans. Inform. Theory,* 41:613–627, 1995.

[3] I. Daubechies. *Ten Lectures on Wavelets.* CBMS-NSF Regional Conf. Series in Appl. Math., Vol. 61. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[4] David Donoho. Wedgelets: nearly-minimax estimation of edges. Technical Report 515, Stanford University, Stanford, CA, October 1997.

[5] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200, 1996.

[6] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.

[7] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer.* Prentice-Hall, Englewood Cliffs, NJ, 1997.

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

[9] H.L. Resnikoff and R.O. Wells. *Wavelet Analysis.* Springer-Verlag, New York, 1998.

[10] M. Vetterli and C. Herley. Wavelets and filter banks: Theory and design. *IEEE Trans. Acoust. Speech Signal Process.*, 40(9):2207–2232, 1992.

[11] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[12] M. Vetterli. A theory of multirate filter banks. *IEEE Trans. Acoust. Speech Signal Process.*, 35(3):356–372, March 1987.

[13] M. Bellanger, G. Bonnerot, and M. Coudreuse. Digital filtering by polyphase network: Application to sample rate alteration and filter banks. *IEEE Trans. Acoust. Speech Signal Process.*, 24(2):109–114, 1976.

[14] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[15] G. Golub and C Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1989.

[16] R. Claypoole, G. Davis, W. Sweldens, and R. Baraniuk. Nonlinear wavelet transforms for image coding. In *Proc. of Asilomar Conf. on Signals, Systems and Computers*, November 1997.

[17] R. Claypoole, R. Baraniuk, and R. Nowak. Adaptive wavelet transforms via lifting. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, May 1998.

[18] R. Claypoole, R. Baraniuk, and R. Nowak. Adaptive wavelet transforms via lifting. *Submitted to IEEE Transactions on Signal Processing*, 1999.

[19] D. L. Donoho. Smooth wavelet decompositions with blocky coefficient kernels. In *Recent Advances in Wavelet Analysis*, pages 259–308. Academic Press, 1993.

[20] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics*, pages 15–87. ACM SIGGRAPH Course Notes, 1996.

[21] R. Claypoole, G. Davis, W. Sweldens, and R. Baraniuk. Nonlinear wavelet transforms for image coding via lifting. *Submitted to IEEE Transactions on Image Processing*, 1999.

[22] J. Goutsias and H. Heijmans. Multiresolution signal decomposition schemes, part 1: Linear and morphological pyramids. *Submitted for Publication, IEEE Transactions on Signal Processing*, 1998.

[23] F. J. Hampson and J.-C. Pesquet. A nonlinear subband decomposition with perfect reconstruction. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc.*, 1996.

[24] R. de Quieroz, D. A. F. Florêncio, and R. W. Schafer. Non-expansive pyramid for image coding using a non-linear filter bank. *IEEE Trans. Image Processing*, 1996. Preprint.

[25] R. Claypoole, R. Baraniuk, and R. Nowak. Lifting construction of non-linear wavelet transforms. In *Proc. Time Frequency/Time Scale Analysis Conference*, October 1998.

[26] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Appl. Comput. Harmon. Anal.*, 5(3):332–369, 1998.

[27] R. R. Coifman and D. L. Donoho. Translation-invariant de-noising. In Anestis Antoniadis and Georges Oppenheim, editors, *Wavelets and Statistics*, pages 125–150. Springer-Verlag, New York, NY, 1995.

[28] M. J. Shensa. Wedding the à trous and Mallat algorithms. *IEEE Trans. Signal Process.*, 40(10):2464–2482, 1992.

[29] M. Girardi and W. Sweldens. A new class of unbalanced Haar wavelets that form an unconditional basis for L_p on general measure spaces. *J. Fourier Anal. Appl.*, 3(4), 1997.

[30] Manfred Heuckel. An operator which locates edges in digitized pictures. *J. Ass. Comput. Mach.*, 18:113–125, 1971.

[31] R. A. DeVore, B. Jawerth, and B. J. Lucier. Image compression through wavelet transform coding. *IEEE Trans. Inform. Theory*, 38(2):719–746, 1992.

[32] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Signal Process.*, 41(12):3445–3462, 1993.

[33] Ronald R. Coifman and Mladen Victor Wickerhauser. Entropy based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 32:712–718, March 1992.

[34] K. Berkner and R. Wells. A correlation-dependent model for denoising via nonorthogonal wavelet transforms. *Rice Univeristy Technical Report*, CML TR 98–07, 1998.

[35] H. Guo, J. E. Odegard, M. Lang, R. A. Gopinath, I. W. Selesnick, and C. S. Burrus. Wavelet based speckle reduction with applications to SAR based ATD/R. In *Proc. ICIP*, November 1994.

[36] Y. Wan and R. Nowak. Overcomplete representation, generalized model selection, and their applications in image processing. IEEE Southwest Symposium on Image Analyis and Interpretation, April 2000. Preprint.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	~~16.Dec.99~~	DISSERTATION

4. TITLE AND SUBTITLE
ADAPTIVE WAVELET TRANSFORMS VIA LIFTING

5. FUNDING NUMBERS

6. AUTHOR(S)
CAPT CLAYPOOLE ROGER L JR

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
RICE UNIVERSITY

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
FY99-441

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
76

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT