Adaptive Wavelet Transforms via Lifting

Roger Claypoole,[†] *Richard Baraniuk*,[†] *and Robert Nowak*[‡] [*]

[†] Department of Electrical and Computer Engineering
Rice University
6100 South Main Street
Houston, TX 77005–1892
Email: clayporl@rice.edu, richb@rice.edu
Web: www.dsp.rice.edu
Fax: (713) 524–5237

[‡] Department of Electrical Engineering
Michigan State University
East Lansing, MI 48824
Email: nowak@egr.msu.edu
Web: www.egr.msu.edu/~nowak/
Fax: (517) 353–1980

Submitted to IEEE Transactions on Signal Processing, May 1999
(also Rice University ECE Technical Report #9304, April 1999)

EDICS #'s 2–FILB and 2–MWAV

September 23, 1999

Abstract

This paper develops new algorithms for adapted multiscale analysis and signal adaptive wavelet transforms. We construct our adaptive transforms with the *lifting scheme*, which decomposes the wavelet transform into prediction and update stages. We adapt the prediction stage to the signal structure and design the update stage to preserve the desirable properties of the wavelet transform. We incorporate this adaptivity into the redundant and non-redundant transforms; the resulting transforms are scale and spatially adaptive. We study applications to signal estimation; our new transforms show improved denoising performance over existing (non-adaptive) orthogonal transforms.

[*]Supported by NSF, grant nos. MIP–9457438 and MIP–9701692, ONR grant no. N00014–95–1–0849, DARPA/AFOSR grant no. F49620-97-1-0513, and Texas Instruments.

1 Introduction

The discrete wavelet transform (DWT) provides a very efficient representation for a broad range of real-world signals. This property has been exploited to develop powerful signal denoising and estimation methods [1] and extremely low-bit-rate compression algorithms [2].

The one-dimensional DWT represents a real-valued discrete-time signal in terms of shifts and dilations of a lowpass scaling function and a bandpass wavelet function [2]. The DWT decomposition is multiscale: it consists of a set of *scaling coefficients* $c^0[n]$, which represent coarse signal information at scale $j = 0$, and a set of *wavelet coefficients* $d^j[n]$, which represent detail information at scales $j = 1, 2, \ldots, J$. The forward DWT has an efficient implementation in terms of a recursive multirate filterbank based upon an appropriately constructed lowpass filter h and highpass filter g, as shown in Figure 1. The inverse DWT employs an inverse filterbank with lowpass filter $\widetilde{h}$ and highpass filter $\widetilde{g}$. For special choices of h and g, we have $\widetilde{h} = h$, $\widetilde{g} = g$, and the underlying wavelet and scaling functions form an orthonormal signal basis. Otherwise, under certain conditions on h, g, $\widetilde{h}$, and $\widetilde{g}$, these functions form a biorthogonal basis [2].

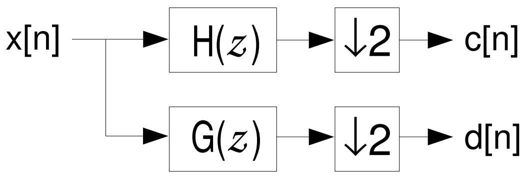

Figure 1: *Filter bank implementation of the wavelet transform. The transform is typically iterated on the scaling coefficients $c[n]$.*

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 [2]. 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 (j) are "predictors" for the data at the next higher resolution or *finer* scale ($j+1$). 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* [3].

In this paper we use lifting to design customized DWTs that adapt to match the signal under consideration. We develop two new multiscale transforms — *scale-adapted transforms* (ScAT) and *space-adapted transforms* (SpAT). The fundamental idea in both cases is to adapt the prediction to minimize a data-based error criterion. While other adaptive transform techniques have been proposed in the literature [4, 5, 6], the adaptive transforms developed here are new, particularly in their use of the lifting programme. We also present adaptive redundant transforms, and a lifting interpretation of median filters. We demonstrate the power of these new transforms with applications to signal denoising problems; in general, our adaptive algorithms outperform existing orthogonal transforms. An example of this improvement is shown in Figure 2, where we compare a proposed adaptive transform (the SpAT, to be discussed in Section 3) against the performance of the Daubechies 8 and Haar transforms. For low signal-to-noise ratios, the SpAt yields a 0.8 dB improvement. At higher ratios, this improvement grows to more than 1 dB.

The paper is organized as follows. In Section 2, we provide a short tutorial on the basic lifting construction and describe a variant of the basic scheme. In Section 3, we develop new adaptive DWTs using the lifting construction, and interpret existing non-linear (median filter based) DWTs within the lifting framework. In Section 4, we apply our adaptive lifting techniques to the redundant wavelet transform. In Section 5, we apply the new DWTs (both redundant and non-redundant) to signal denoising and demonstrate that the adapted DWTs can perform significantly better than standard wavelet denoising methods in several interesting cases. We close in Section 6 with concluding remarks and plans for future work.

2 The Lifting Concept

Lifting is a spatial (or time) domain construction of biorthogonal wavelets developed by Sweldens [3]. We present here an overview of our interpretation of the lifting concept.

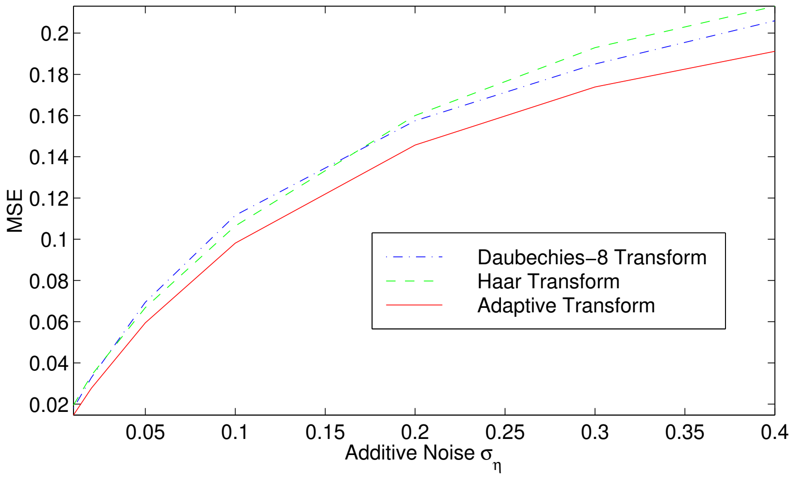

Figure 2: *Mean-Squared Error (MSE) denoising results for a test signal with spatially varying characteristics (the DoppelBlock signal, shown in Figure 17). Results are shown for various levels of input noise and three transforms: Daubechies 8 orthogonal, Haar orthogonal, and the adaptive transform (SpAT) proposed in Section 3. For low signal-to-noise ratios, the SpAT yields a 0.8 dB improvement.*

2.1 Lifting Operations

Lifting consists of iteration of the following three basic operations (see Figure 3):

Split: Divide the original data into two disjoint subsets. Although any disjoint split is possible, we will 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 $\mathcal{P}$:

$$d[n] \;=\; x_o[n] - \mathcal{P}(x_e[n]). \tag{1}$$

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 the result to $x_e[n]$:

$$c[n] \;=\; x_e[n] + \mathcal{U}(d[n]), \tag{2}$$

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. This normalizes the energy of the underlying scaling and wavelet functions.

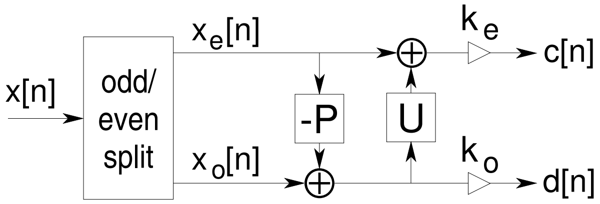

Figure 3: *Lifting stage: Split, Predict, Update. k_e and k_o normalize the energy of the underlying scaling and wavelet functions.*

The lifting steps are easily inverted, *even if $\mathcal{P}$ and $\mathcal{U}$ are nonlinear, space-varying, or non-invertible.* Rearranging (1) and (2), we have

$$x_e[n] = c[n] - \mathcal{U}(d[n]), \qquad x_o[n] = d[n] + \mathcal{P}(x_e[n]). \tag{3}$$

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 4.

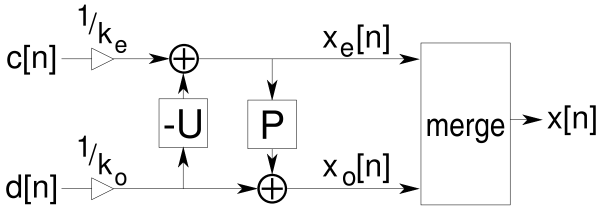

Figure 4: *Typical inverse lifting steps: undo the Update, undo the Predict, and Merge the even and odd samples.*

2.2 Predictor Design

In the simplest scenario, the prediction operator $\mathcal{P}$ is a linear shift-invariant filter with z-transform $P(z)$. In Figure 5, we illustrate a symmetric, $N = 4$-point predictor $P(z) = p_1 z^{-1} + p_2 + p_3 z + p_4 z^2$.

[1] In fact, *all* wavelet transforms can be factored into a series of lifting stages (with perhaps multiple predicts and updates per stage) [7].

By tracing the contribution of $x_e[n]$ and $x_o[n]$ through the tree to the point $d[n]$, we can find the equivalent filter that would be applied to the original data $x[n]$. In vector form, we have

$$\mathbf{g} \;=\; [-p_1,\; 0,\; -p_2,\; 1,\; -p_3,\; 0,\; -p_4]^T. \tag{4}$$

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

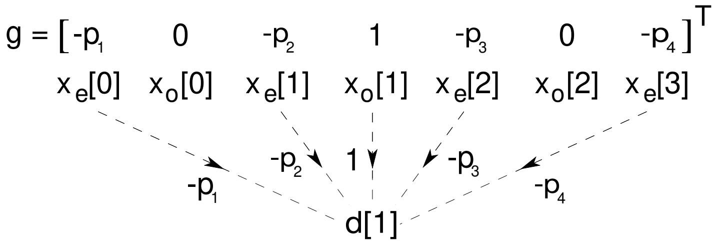

Figure 5: *Prediction filtering. An $N = 4$ point linear prediction filter $P(z)$ yields the prediction vector* $\mathbf{g}$ *shown across the top.*

If a signal is exactly polynomial, it can be completely described using scaling (coarse) coefficients alone [2]. 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 $N \times 2N-1$ matrix $\mathbf{V}$ with entries $[\mathbf{V}]_{k,n} = (n-l)^k$, $k = 0, 1, \cdots, N-1$ (rows), $n = 1, 2, \cdots, 2N-1$ (columns). We define $0^0 \equiv 1$. 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 $N \times N$ 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:

$$\mathbf{V}^\diamond \mathbf{p} = [1 \quad 0 \cdots 0]^T, \tag{5}$$

where entries in $\mathbf{p}$ are the prediction filter coefficients. Note that $\mathbf{V}^\diamond$ is a Vandermonde matrix based on distinct elements $(n-l)^k$. Thus, the set of linear equations in (5) is readily solved, since such a Vandermonde matrix as $\mathbf{V}^\diamond$ is always invertible [8, p. 120].

2.3 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 6, 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.

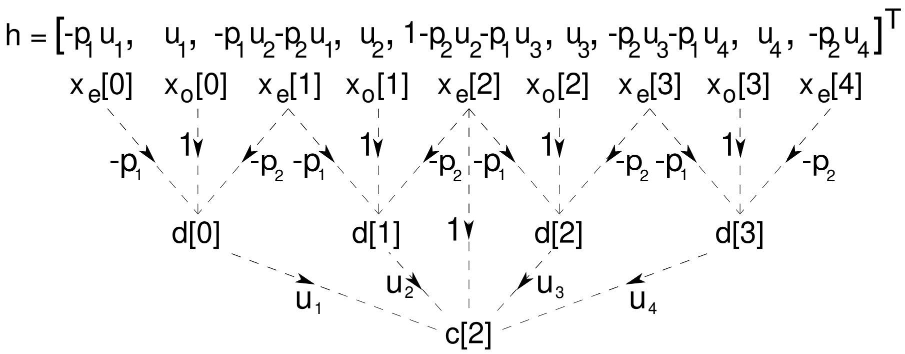

Figure 6: *Update filtering. An $N = 2$ point linear predict followed by an $\widetilde{N} = 4$ point linear update yields the update vector* $\mathbf{h}$ *shown across the top.*

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 6, we have

$$\widetilde{\mathbf{g}} = [-p_1u_1,\, -u_1,\, (-p_1u_2 - p_2u_1),\, -u_2,\, (1 - p_2u_2 - p_1u_3),\, -u_3,\, (-p_2u_3 - p_1u_4),\, -u_4,\, -p_2u_4]^T. \quad (6)$$

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 (5) 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.

2.4 Equivalence between Polynomial Constraints and Vanishing Moments

2.4.1 Wavelet Transform in Polyphase Form

As discussed in Section 1, any biorthogonal wavelet transform can be represented as a perfect reconstruction multirate filter bank. By splitting each wavelet filter into its polyphase components [9], the DWT can be implemented as shown in Figure 7.

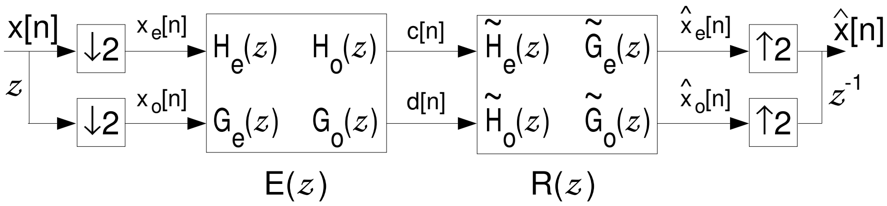

Figure 7: *Wavelet filter bank in polyphase form.*

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 $H_e(z)$ [9].

2.4.2 Lifting in Polyphase Form

The lifted wavelet transform can also be written in polyphase form. As discussed in Section 2, 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 8. The prediction matrix passes $x_e[n]$ unchanged, but the wavelet coefficients $d[n]$ are the difference between $x_o[n]$ and $P(x_e[n])$, ie, 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 $c[n]$.

Multiplying these matrices together yields the representation shown in Figure 9. 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 $E^H(z^{-1})E(z) = I$ [9] , which forces our prediction and update filters to satisfy

$$U(z^{-1}) \quad = \quad \frac{P(z)}{4(1 - U(z)P(z))}$$

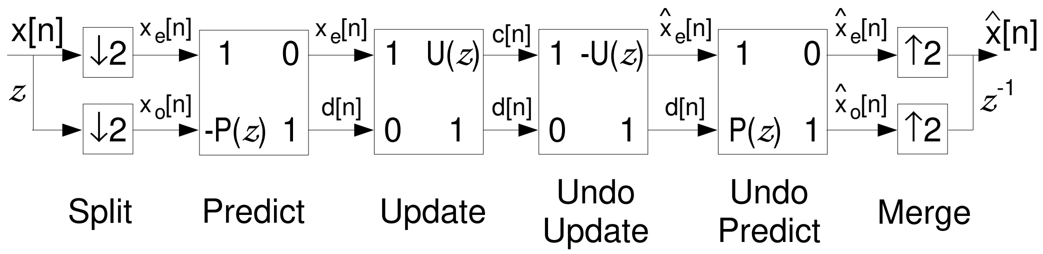

Figure 8: *Polyphase representation of lifting operators.*

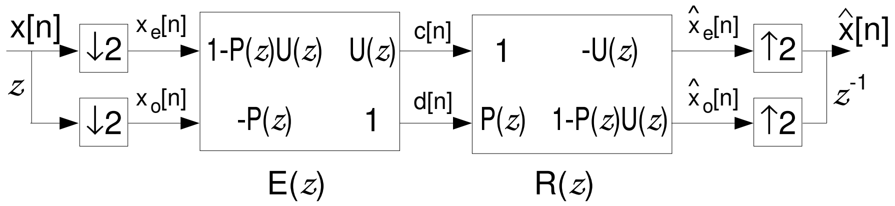

Figure 9: *Polyphase representation of lifting with the Prediction and Update stages combined into one matrix.*

$$U(z)U(z^{-1}) \quad = \quad 1/4.$$

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 9 with the polyphase components of the wavelet filters of Figure 7. 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,

$$H(z) \quad = \quad 1 - P(z^2)U(z^2) + z^{-1}U(z^2), \tag{7}$$

$$G(z) \quad = \quad -P(z^2) + z^{-1}, \tag{8}$$

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

2.4.3 Vanishing Moments and Lifting Constraints

In a wavelet system such as that shown in Figure 1, 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 given by the *wavelet recurrent relations* [2]:

$$\phi(t) = \sum_k h[k]\phi(2t-k), \tag{9}$$

$$\psi(t) = \sum_k g[k]\phi(2t-k). \tag{10}$$

The wavelet filter coefficients $h[n]$ and $g[n]$ can be designed by enforcing vanishing moment conditions on the underlying scaling function $\phi(t)$ and wavelet function $\psi(t)$ via the recurrence relations. We have written the coefficients of the wavelet filters in terms of the lifting coefficients, namely the prediction and update filter coefficients. Thus, we can map the vanishing moments constraints into constraints on our prediction and update filters. 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 integration and 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. However, in equation (8) above, we equated the wavelet filter $G(z)$ to the lifting prediction filter: $G(z) = -P(z^2) + z^{-1}$. 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 2.2 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 can 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 2.3.

Thus, designing a biorthogonal wavelet system by enforcing vanishing moment conditions on 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 representation or the underlying scaling and wavelet functions, and therefore makes the incorporation of non–linearities and adaptivity into the wavelet transform more tractable. In addition, when we utilize the prediction and update filters in Section 3.1 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 non-linear transforms, while carefully controlling the underlying properties of the wavelet transform.

2.5 The Update/Predict Programme

Our goal is to introduce adaptivity into the wavelet transform. In the lifting framework of Figure 3, the update structure depends on the predictor structure. Hence, if $\mathcal{P}$ is space-varying or nonlinear, then so is $\mathcal{U}$, and the design procedure of Section 2.3 becomes unwieldy. A crafty detour around this problem developed by Davis [10] is to perform the update step first, followed by the prediction, as shown in Figure 10. The relevant equations then become

$$c[n] = x_e[n] + \mathcal{U}(x_o[n]), \qquad d[n] = c[n] - \mathcal{P}(x_e[n]). \tag{11}$$

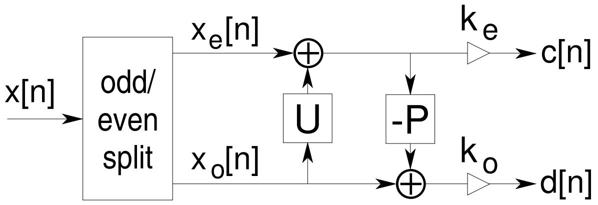

Figure 10: *Update–First Lifting Sequence.*

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 2.3). This is equivalent to adding vanishing moments to the synthesis wavelet function

$\widetilde{\psi}(t)$. In the standard predict-first scheme, we first add vanishing moment to the analysis wavelet function, as described in Section 2.2.

Since the update/predict lifting stage creates $c[n]$ prior to $d[n]$, the prediction operator can be designed to satisfy requirements other than polynomial suppression capability. For example, the predictor could be a median filter, or a filter designed to minimize the prediction error energy. These non-linear filters can now be applied without affecting the coarse approximation $c[n]$. In Section 3.2, we will exploit this flexibility to design space-varying predictors that adapt to the local characteristics of the signal, while completely preserving the low-pass interpretation and orthogonal properties of the wavelet transform.

3 Adaptive Lifting

The lifting construction of the wavelet transform provides a great deal of flexibility. In principle, we can use any linear, nonlinear, or space-varying predictor and update, and the lifting construction ensures that the resulting transform is invertible. We now investigate the capabilities of the lifting approach for adaptive lifting 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 [1], our adaptive transforms have the potential to improve transform-based processing.

3.1 Scale-Adaptive Transform (ScAT)

In Section 2, we derived the lifting construction based on a polynomial signal suppression/preservation argument. However, we alluded to nonlinear schemes based on more general prediction schemes. 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.

In the scale-adaptive transform (ScAT), we adapt the predictor in each lifting stage in order 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 polynomials up to order $M < N$. Let $\mathbf{x_o}$ denote the odd-indexed data we wish to predict and 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. The vector of prediction errors is then given by

$$\mathbf{e} \;=\; \mathbf{x_o} - \mathbf{X_e}\mathbf{p}. \tag{12}$$

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

$$\min_{\mathbf{p}} \|\mathbf{x_o} - \mathbf{X_e}\mathbf{p}\|^2 \quad \text{subject to} \quad \mathbf{V}^{\diamond}\mathbf{p} = [1 \quad 0\cdots 0]^T, \tag{13}$$

with $\mathbf{V}^{\diamond}$ an $M \times N$ matrix determined as in Section 2.2. 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 [8, 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 we determine the optimal predictor, we design the update filter using the methods of Section 2.3 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 samples/pixels at each scale (especially high scales). Our prediction filter attempts to minimize the prediction error across the entire scale. However, many signals are non-stationary, with the structure varying significantly within each scale. A single filter cannot mimic the image structure of an entire scale. Therefore, we seek an adaptive algorithm which adapts point-by-point, not just scale-by-scale.

3.2 Space-Adaptive Transform (SpAT)

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. In our space-adaptive transform (SpAT), we employ the update/predict framework of Section 2.5 and choose a predictor from a suite of predictors to minimize each individual prediction error $d[n]$.

Our adaptive algorithm performs an $\widetilde{N} = 1$ 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 subset of the Cohen-Daubechies-Feauveau family [11, 12]. We chose this $(1,N)$ family because it provides the SpAT with a great deal of flexibility [10]. The $(1,1)$ filter set corresponds to a Haar wavelet transform, while the underlying wavelet functions of the $(1,7)$ filter set are shown in Figure 11. These wavelet functions are relatively smooth and the synthesis functions have small side-bands, making them a good choice for compression and signal estimation.

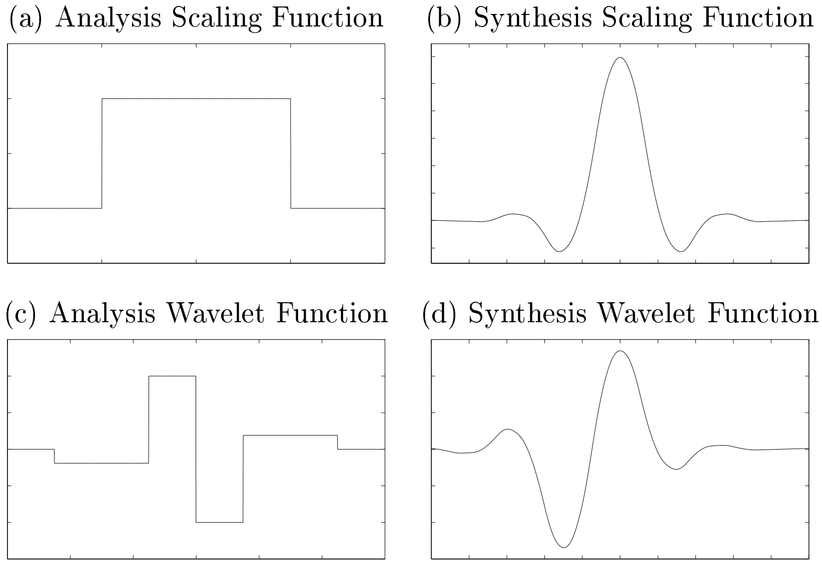

Figure 11: *Top row: analysis (a) and synthesis (b) scaling functions for the order (1,7) Cohen-Daubechies-Feauveau [11, 12] filter used in the SpAT. Bottom row: analysis (c) and synthesis (d) wavelet functions. These basis functions correspond to the update-first form of lifting.*

A demonstration of the SpAT applied to a step edge is shown in Figure 12. The transform is able to lock-on to the dominant signal structure at each point. Thus, we avoid the futile exercise of approximating discontinuities and other non-smooth behavior with low-order polynomials.

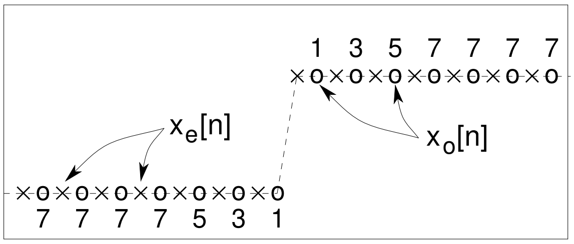

Figure 12: *In the SpAT, the order N of the predictor varies with space n to minimize the wavelet coefficient value $d[n]$. Above each $x[n]$ we give the corresponding choice $N(n)$. As the predictor approaches an edge, it decreases N (chooses wavelets of smaller spatial support) in order to avoid the edge.*

3.3 Other Adaptive Schemes

In the above sections, we described two algorithms. The first (ScAT) 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 signal.

The second algorithm (SpAT) minimized the prediction error at each data point, and was thus spatially varying. Without constraints, the algorithm could drive the prediction error to zero at each point, effectively removing *all* the information from the signal. To prevent this problem, the choice of prediction filters were limited to the $(1, N)$ family of Cohen-Daubechies-Feauveau.

It is possible to combine these two ideas into a single algorithm. Instead of limiting the filter to a family (as with the SpAT), 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 data point, based on minimizing the (possible weighted) sum of prediction errors over a local window about the point. By analyzing a block of data around the data point, we prevent the filter from removing all the information from our signal, while allowing 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.

3.4 Median Filtering

Median filters are well known non-linear filters, and have been successfully incorporated into wavelet-like decompositions by Goutsias and Heijmans [13], Hampson and Pesquet [6], and de Queiroz et al [14]. 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 13. 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 2.3 and shown across the top of Figure 13. We apply the appropriate set of linear constraints to solve for the update filter coefficients u_i.

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 non-linear median

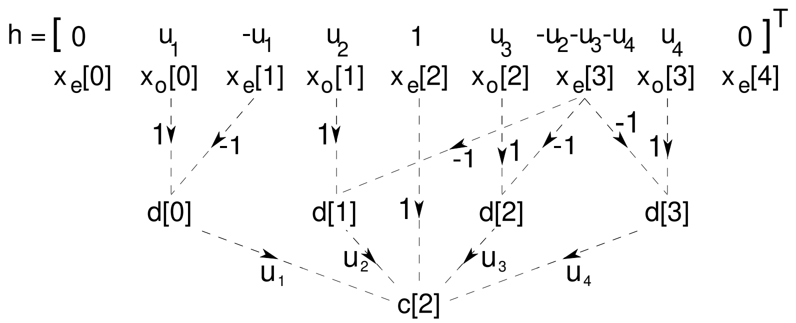

Figure 13: *Median prediction with linear update filtering. An $N = 5$ point median filter prediction followed by an $\widetilde{N} = 4$ point linear update yields the update vector* $\mathbf{h}$ *shown across the top.*

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 13. 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 14. 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) [6]; it can be viewed as optimal in the sense of polynomial approximation.

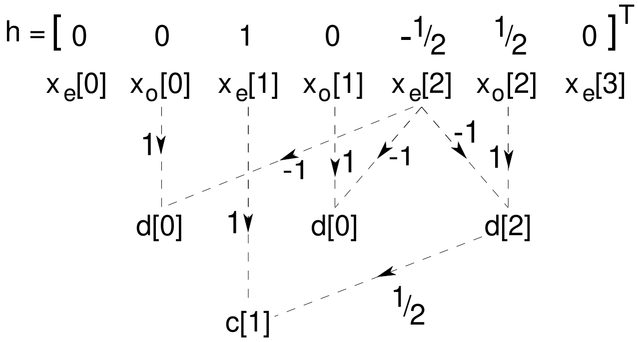

Figure 14: *5-point median filter predict followed by a 3-point median filter update. The update vector* $\mathbf{h}$ *shown across the top satisfies the* 0^{th} *polynomial constraint regardless of median choices.*

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 possible 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 [6, 15], since they can be easily modified to accommodate integer-to-integer arithmetic [16].

4 Redundant Lifting

In many applications, the redundant (shift-invariant) wavelet transform often exhibits superior performance over the non-redundant wavelet transform [17]. For example, thesholding 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-squared error performance of the transform [17].

Our spatially adaptive lifted transforms are designed to improve performance near discontinuities. Therefore, we conjecture that incorporating our adaptive algorithms into the redundant wavelet transform should result in further performance improvements. We need only to extend the lifted wavelet structure to the redundant case.

A typical redundant wavelet transform is implemented as shown in Figure 15. By removing the time-varying decimators, we ensure that the transform is shift-invariant. Of course, the transform now takes L data points to $L(J+1)$ transform coefficients (where J is the total number of iterations or scales), and not to L as in the non-redundant transform. Subsequent iterations at scale j require application of the original wavelet filters expanded by 2^j. This implementation is often referred to as the "non-decimated wavelet transform" [18].

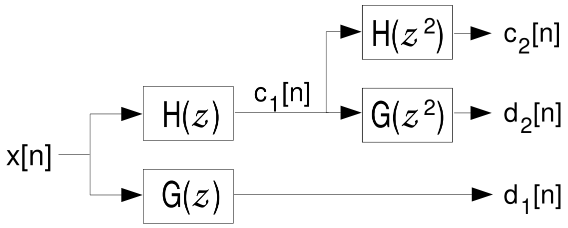

Figure 15: *Two iterations of the undecimated implementation of the redundant wavelet transform. Subsequent iterations at scale j require application of expanded filters $H(z^{2^j})$ and $G(z^{2^j})$ to the coarse coefficients $c[n]$.*

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 down-sampled 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 16. 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"[17].

In a nutshell, the lifted redundant wavelet transform is simply two non-redundant lifted wavelet transforms, intertwined at each scale. The first transform 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

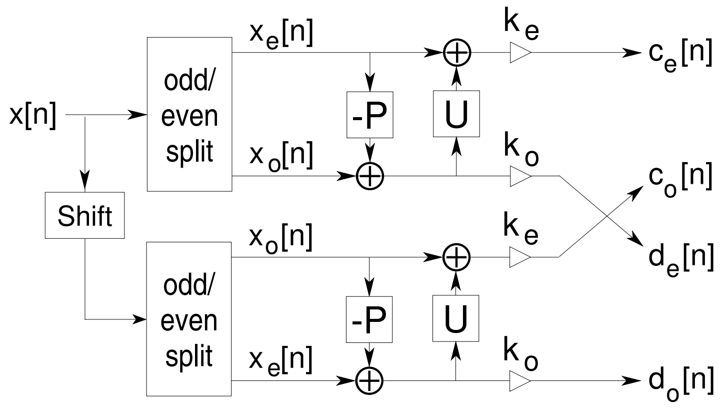

Figure 16: *Lifting implementation of the redundant wavelet transform. Transform is computed on all possible shifts at each scale. Transform is iterated on both sets of coarse coefficients,* $c_e[n]$ *and* $c_o[n]$.

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 Section 3.

5 Numerical Experiments

In this section, we compare the performance of the new adaptive transforms with some of the standard Daubechies wavelets. We perform three experiments. 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, we compare the performance of the new transforms in a signal denoising application. Finally, we compare the redundant implementation of our adaptive transform against other redundant transforms for denoising.

5.1 Entropy Comparison

The entropy of the transform coefficient distribution is a common measure of the efficiency of a signal transform [5]. If we collectively denote the scaling and wavelet coefficients by $\{w_i\}$, then the entropy is defined as

$$H(\mathbf{w}) \;=\; \sum_i |w_i|^2 \log_2 |w_i|^2, \qquad (14)$$

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

Table 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 [1]. 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 1 show that both adaptive transforms perform nearly as well as (or better than) the D8 or the Haar in each test case.

Table 1: *Entropy results for various signals and transforms.*

Signal	**Entropy**			
	D8	Haar	ScAT	SpAT
Doppler	2.837	3.153	2.878	**2.568**
Blocks	2.598	2.618	2.517	**2.318**
Bumps	3.541	3.652	3.532	**3.233**
HeaviSine	2.262	2.500	**2.250**	2.265
DoppelBlock	3.543	3.597	3.414	**3.149**

5.2 Denoising with Nonredundant 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 [1] is described as follows. We observe L samples $\{x[n]\}$ of an unknown function f with additive i.i.d. Gaussian noise $\{\eta[n]\}$:

$$x[n] \;=\; f[n] + \eta[n], \qquad n = 0, 1, \ldots, L-1. \tag{15}$$

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 left intact. 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 [1].

However, in [1], the thresholds were derived only for orthogonal wavelet transforms. In [19], Berkner and Wells propose new thresholds that take into account correlations induced by redundant wavelet transforms and biorthogonal wavelet transforms. As shown in Section 2.4, 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 considered here), the thresholds are adapted for each data point and at each scale to compensate for noise correlations and variances.

Also, in [20], smaller thresholds than those proposed in [1] were found to perform better (with respect to mean-squared error) in many denoising applications. Smaller thresholds (and a hard threshold) are also recommended for redundant denoising in [17]. Thus, the experimental thresholds in [20] are used for all of our image denoising and redundant denoising experiments (modified to compensate for noise correlations, where appropriate).

Table 2: *Denoising: MSE for various signals and transforms. Each signal is corrupted with additive white Gaussian noise with standard deviation equal to 5% of the peak signal magnitude.*

Signal	**MSE**			
	D8	Haar	ScAT	SpAT
Doppler	0.0298	0.0490	**0.0265**	0.0322
Blocks	0.0404	**0.0286**	0.0338	0.0319
Bumps	0.0324	0.0335	**0.0280**	0.0310
HeaviSine	0.0143	0.0313	**0.0139**	0.0164
DoppelBlock	0.0461	0.0444	0.0423	**0.0396**

Table 2 provides the mean-squared error (MSE) performance of the four transforms and five signals discussed in Section 5.1 above. In this experiment, white Gaussian noise of standard deviation $0.05 \times \max_n |s[n]|$ was added to each of the test signals. The MSEs in Table 2 show again that both adaptive transforms perform nearly as well as (or better than) the D8 or the Haar in each

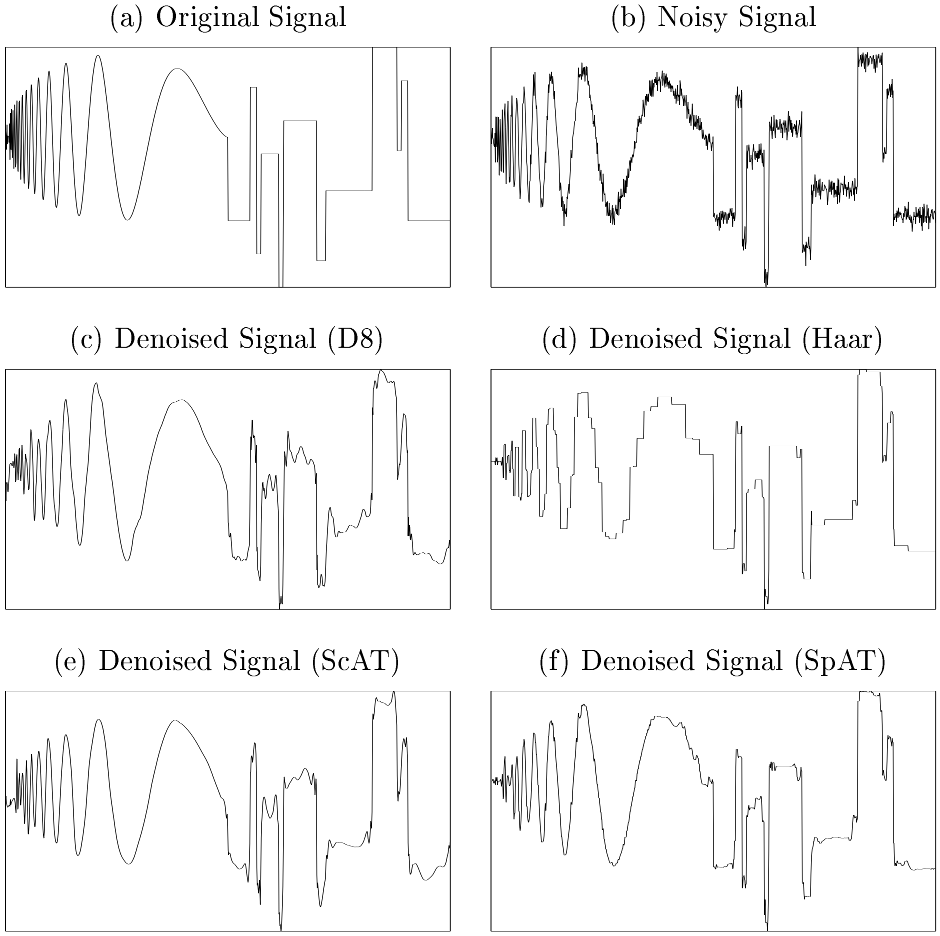

Figure 17: *Top row: original DoppelBlock signal (concatenation of Doppler and Blocks) (a), signal with additive white Gaussian noise, standard deviation equal to 5% of maximum signal magnitude (b). Second row: signal denoised with the Daubechies 8 orthogonal wavelet transform (c), signal denoised with the haar transform (d). Bottom row: Signal denoised with the ScAT transform (e), signal denoised with the SpAT transform (f).*

test case. An example of the denoised signals is shown in Figure 17. These graphs were generated for the DoppelBlock signal with additive white Gaussian noise with 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.

For our next experiment, we applied our algorithms to several standard images. We extended our algorithm to two-dimensions in a separable fashion; at each scale, our transform is 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 [20], modified for our biorthogonal and adaptive transform). As our performance metric, we compute the peak signal to noise ratio (PSNR),

$$\text{PSNR} = 20 \log \left(\frac{\max(x_i)}{\sqrt{\sum (x_i - \hat{x}_i)^2 / N}} \right),$$

where x_i is the i^{th} pixel or our original image, $\hat{x}_i$ is the i^{th} pixel or our denoised image, and N is the total number of pixels. Our PSNR results are shown in Table 3, while an example of the denoised cameraman image is displayed in Figure 18. The SpAT transform demonstrates improved PSNR performance, while significantly increasing edge crispness and preserving image texture.

Table 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. "Blocks" is an artificial, edge-dominated image with texture.*

Image	**PSNR (dB)**		
	d8-dwt	haar	SpAT
Cameraman	67.9	68.0	**68.3**
Lenna	68.4	67.7	**68.5**
Building	67.4	67.8	**68.0**
Bridge	66.4	66.3	**66.6**
Blocks	68.5	**69.0**	**69.0**

Remark: no results have been presented for our space/scale adaptive transform presented in Section 3.3. 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 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 2.3 becomes

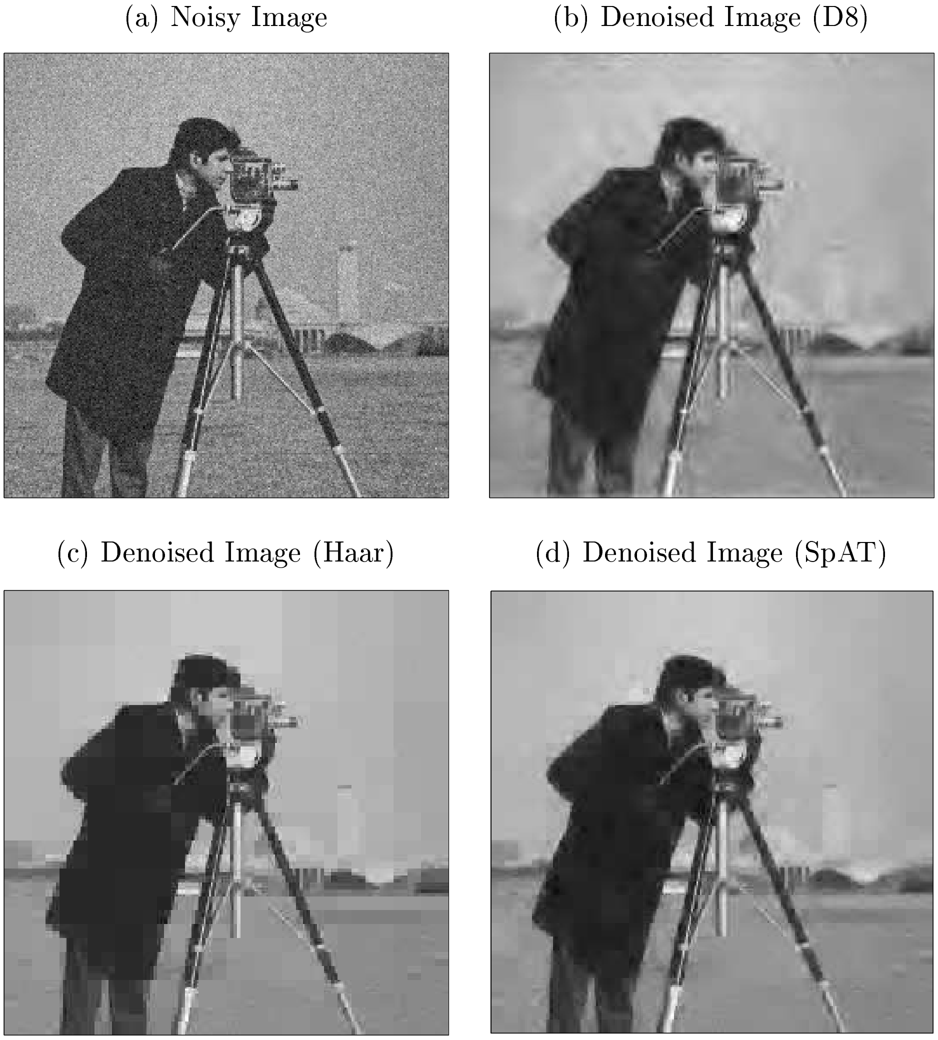

Figure 18: *Top row: (a) Cameraman image with additive white Gaussian noise with standard deviation equal to 15% of the maximum image magnitude, (b) Image denoised with the Daubechies 8 orthogonal wavelet transform, 66.4941 dB PSNR. Second row: (c) Image denoised with the haar transform, 66.5983 dB PSNR, (d) image denoised with the SpAT transform, 66.8278 dB PSNR (d). The SpAT transform yields the highest PSNR while significantly reducing ringing around edges.*

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.

5.3 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. Again, our thresholds must 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 [19]. Here we have used a hard threshold, since this outperforms soft thresholding within the context of redundant transforms [17].

In Table 4, we present the denoising performance of the redundant transforms. These signals were corrupted with white Gaussian noise (standard deviation of 5% max 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 19.

Table 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	**0.0101**	0.0166	0.0120
Blocks	0.0123	**0.0060**	0.0080
Bumps	0.0116	**0.0111**	0.0113
HeaviSine	**0.0066**	0.0092	0.0080
DoppelBlock	0.0146	0.0137	**0.0126**

6 Conclusions

This paper has described several new adaptive DWTs based on the lifting scheme. We used the lifting construction to adaptively match the DWT to a given signal based on data-based error criteria. In addition, we implemented our adaptive transform as a redundant wavelet transform, and we presented a lifting interpretation of median filtering.

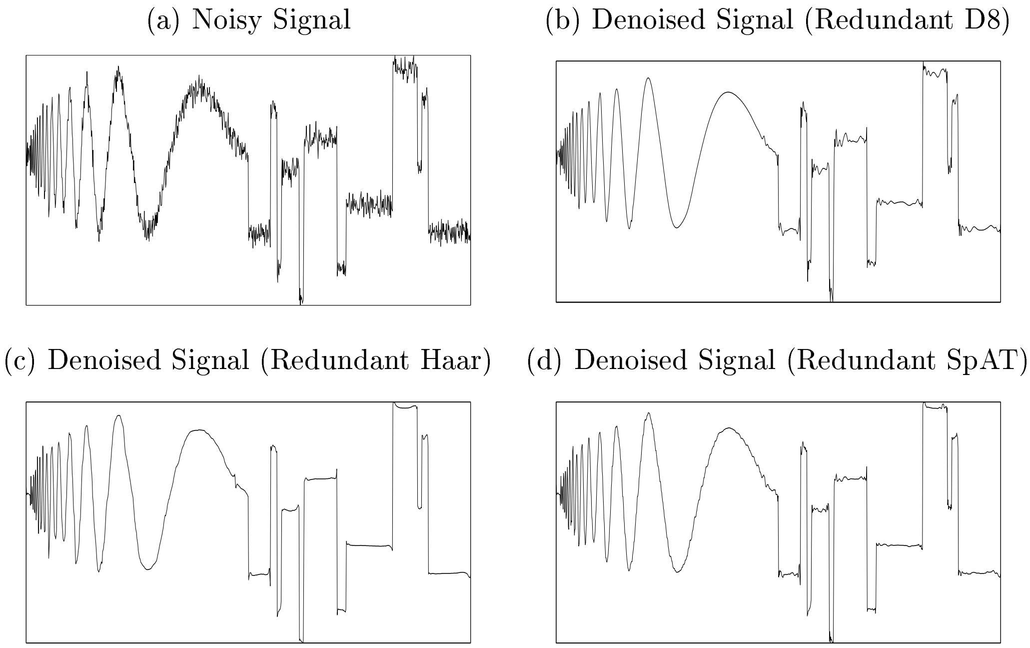

Figure 19: *Top row: (a) DoppelBlock signal with additive white Gaussian noise, standard deviation equal to 5% of maximum signal magnitude, (b) signal denoised with the Daubechies 8 redundant wavelet transform. Second row: (c) signal denoised with the redundant haar transform, (d) signal denoised with the redundant SpAT transform.*

Comparisons in entropy measures and signal denoising demonstrate the potential utility of the new transforms. Our adaptive transforms outperform standard wavelet transforms on signals and images with varying spatial characteristics.

Many variations on and embellishments to the ideas presented here can be made to develop new adaptive DWTs. For example, a logical next step would be to extend our adaptive redundant SpAT to two-dimensions, and to implement the ScAT as both a redundant transform and a two-dimensional transform. We hypothesis that, with further modifications, a combination of the ScAT and SpAT could be a powerful signal and image analysis tool. We also intend to incorporate other non-linear filters into our adaptive lifting structure (as was done with the median filter in Section 3.4 or in the morphology-based work of Heijmans and Goutsias [21]).

Finally, in this paper we have only examined the potential of the new transforms for signal denoising, but they may also improve algorithm performance in other applications, such as signal compression [10], detection, and classification.

Acknowledgments: Thanks to Geoff Davis and Wim Sweldens for introducing us to the lifting paradigm. Thanks also to Geoff for suggesting the update/predict architecture of Section 2.5.

References

[1] D. L. Donoho, "Denoising by soft-thresholding," *IEEE Trans. Inform. Theory*, vol. 41, pp. 613–627, 1995.

[2] 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.

[3] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, 1996.

[4] A. Tewrik, D. Sinha, and P. Jorensen, "On the optimal choice of a wavelet for signal representation," *IEEE Trans. Inform. Theory*, vol. 398, pp. 747–765, 1992.

[5] Ronald R. Coifman and Mladen Victor Wickerhauser, "Entropy based algorithms for best basis selection," *IEEE Transactions on Information Theory*, vol. 32, pp. 712–718, Mar. 1992.

[6] F. J. Hampson and J.-C. Pesquet, "A nonlinear subband decomposition with perfect reconstruction," in *Proc. ICASSP*, 1996.

[7] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 245–267, 1998.

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

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

[10] 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*, Nov. 1997.

[11] D. L. Donoho, "Smooth wavelet decompositions with blocky coefficient kernels," in [22], pp. 259–308. Academic Press, 1993.

[12] W. Sweldens and P. Schröder, "Building your own wavelets at home," in *Wavelets in Computer Graphics*, pp. 15–87. ACM SIGGRAPH Course notes, 1996.

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

[14] 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.

[15] R. Claypoole, R. Baraniuk, and R. Nowak, "Lifting construction of non-linear wavelet transforms," in *Proc. Time Frequency/Time Scale Analysis Conference*, Oct. 1998.

[16] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Appl. Comput. Harmon. Anal.*, vol. 5, no. 3, pp. 332–369, 1998.

[17] R. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," in *Wavelets and Statistics*, Anestis Antoniadis and Georges Oppenheim, Eds. 1995, pp. 125–150, Springer-Verlag, New York, NY.

[18] M. J. Shensa, "Wedding the à trous and Mallat algorithms," *IEEE Trans. Signal Process.*, vol. 40, no. 10, pp. 2464–2482, 1992.

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

[20] 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*, Nov. 1994.

[21] J. Heijmans and J. Goutsias, "Morphology-based perfect reconstruction filter banks," in *Proc. Time Frequency/Time Scale Analysis Conference*, Oct. 1998.

[22] L. L. Schumaker and G. Webb, Eds., *Recent Advances in Wavelet Analysis*, Academic Press, New York, 1993.