# Lossless Image Compression Via the Lifting Scheme

Wade Spires
*University of Central Florida*
wspires@cs.ucf.edu

November 2005

## Abstract

*A study in lossless image compression using the lifting scheme is presented. We first suggest why lossless image compression is an important issue and the general idea behind lifting. A detailed approach to generating biorthogonal wavelets using the lifting scheme is provided, which is followed by a small example. The results from applying lifting to a variety of images are given in order to test the performance of lifting in a practical setting.*

## 1 Introduction

Lossy image compression can provide acceptable image quality while also providing dramatic reductions in image size. However, powerful methods for lossless image compression are still needed. Lossless compression can benefit the following important uses of imaging:

- Medical images,
- Seismic data,
- Satellite images,
- Manuscript images,
- Heavily edited images.

Accuracy in medical imagery is quite literally a matter of life and death. A doctor must discern small details in medical images to help correctly diagnose many diseases such as cancer. Thus, artifacts due to lossy compression cannot be tolerated. Seismic forces such as earthquakes are often preceded by small landscape changes prior to turning destructive, necessitating accurate imagery. Similarly, military decision-makers analyzing satellite images require precise information in order to take informed actions based on this data. Manuscripts by famous authors can be preserved in their original state, thus preserving their study for future scholars. Heavily edited images benefit from lossless compression as many details can be gradually lost if an image is consistently saved in a lossy format. Generally, lossless image compression aids any application that must discern small details amongst a large data set with high accuracy.

The wavelet transform has proved to be an indispensable tool in data compression due to its ability to decorrelate data effectively and efficiently. The lifting scheme is a simple method for designing customized biorthogonal wavelets and offers several advantages:

- Allows a faster implementation of the wavelet transform,
- Saves storage by providing an in-place calculation of the wavelet transform,
- Simplifies determining the inverse wavelet transform,
- Provides a natural way to introduce and think about wavelets.

The main difference with classical constructions is that it does not rely on the Fourier transform. Due to this reformulation, second generation wavelets can be constructed. Second generation wavelets, unlike traditional, first generation wavelets, are not necessarily translates and dilates of a function.
In this way, wavelets can be applied to non-smooth domains and curves or surfaces.

## 2 General Concept of Lifting

The lifting transform at its highest level is very simple. The lifting transform can be performed via two operations: `Predict` and `Update`. Suppose we have the one-dimensional signal $a_0$. Lifting is done by performing the following sequence of operations:

1. Split $a_0$ into $\mathrm{Even}_{-1}$ and $\mathrm{Odd}_{-1}$
2. $d_{-1} = \mathrm{Odd}_{-1} - \mathrm{Predict}(\ \mathrm{Even}_{-1}\ )$
3. $a_{-1} = \mathrm{Even}_{-1} + \mathrm{Update}(\ d_{-1}\ )$

These steps are repeated to construct multiple scales of the transform. The inverse transformation is simple as well. We only reverse the order of operations and change the signs. The even and odd sequences are then merged together to form the original signal. The wire diagram in Figure 1 shows the forward transform visually. The coefficients $a$ are represent the averages in the signal, while the coefficents in $d$ represent the differences in the signal. Thus, these two sets also correspond to the low-pass and high-pass frequencies present in the signal.

To mirror the operations of the wavelet transform, we must formulate the operations `Predict` and `Update`. For the Haar wavelet, we have the following steps:

- `Predict(x) = x`
- `Update(y) = y / 2`

The Haar case is very easy to derive and implement. Further, the operation can be performed on an input signal or image in-place (see Figure 2), making it both time and space efficient. However, Haar has poor results due to the simplicity of the prediction step. Since every wavelet transform has a transform that can be formulated in terms of lifting steps, several other more effective transforms can be derived and used, such as Daubechies and symmetric biorthogonal. The derivation of biorthogonal wavelets in particular is discussed in great detail in the next section.

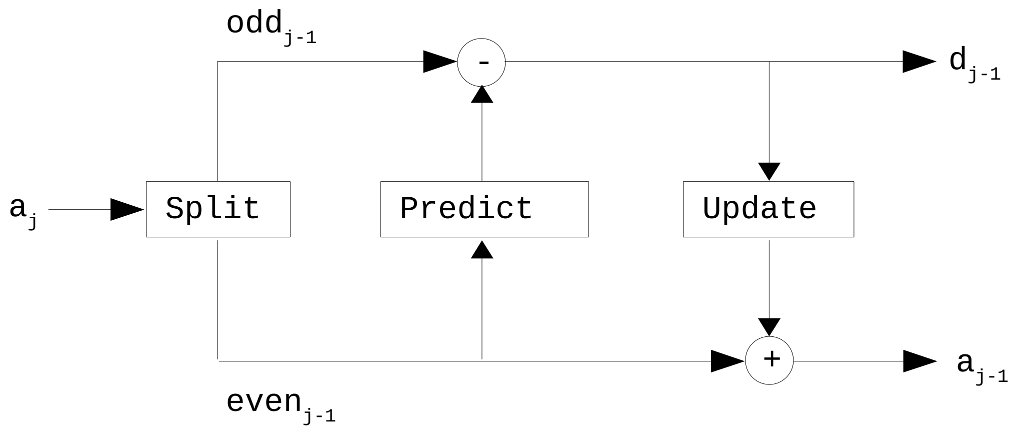

Figure 1: Wire diagram of forward transformation with the lifting scheme

Compression requires two major steps: decorrelation and coding. The process of lifting provides spatial decorrelation of image data, but no actual compression is performed by this step. Hence, a coding step must follow lifting to reduce the amount of data. An entropy coder, such as Huffman or arithmetic, can be used for this purpose.

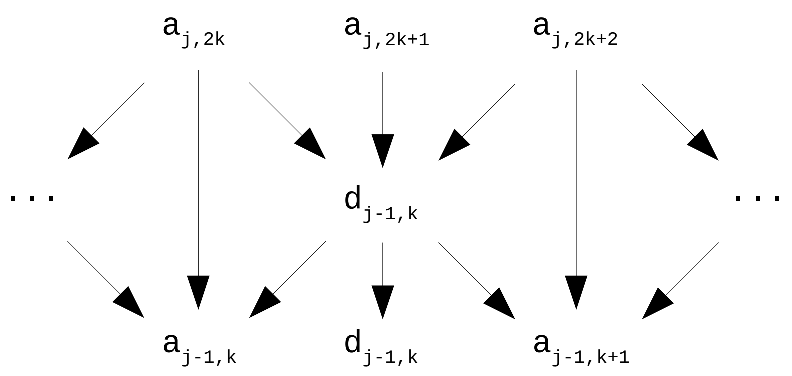

Figure 2: The lifting scheme allows for an in-place transformation on a signal: prediction replaces the odd values and update replaces the even using the prediction results.

## 3 Theory and Technical Approach

This section elaborates on the general ideas presented in Section 2 and provides a detailed look at the construction of a special type of wavelet, biorthogonal, by applying specific constraints to the transformation process. In general, the lifting scheme can be applied with the following algorithm:

```
for j = -1 to -n
  for each position k in $a_{j+1}$
    ($a_{j,k}$, $d_{j,k}$} = split($a_{j+1,k}$)
    $d_{j,k}$ -= predict( $a_{j,k}$ )
    $a_{j,k}$ += update( $d_{j,k}$ )
```
where the `predict()` and `update()` functions are defined according to the above discussion. The inverse transformation is easily derived from the forward by just reversing the direction and the signs:
```
for j = -n to -1
  for each position k in $a_{j+1}$
    $a_{j,k}$ -= update( $d_{j,k}$ )
    $d_{j,k}$ += predict( $a_{j,k}$ )
    $a_{j+1,k}$ = merge($a_{j,k}$, $d_{j,k}$}
```
The basic algorithm again is visually depicted in Figure 1. We will now discuss each step in detail.

## 3.1 Splitting Step

The number of samples can be reduced by simply subsampling the input signal:

$$a_{-1,k} = a_{0,2k}$$

where $a_{0,2k}$ is an even sample ($2k$) at the starting level ($0$) of the transformation and $a_{-1,k}$ is the sample at the next level ($-1$). Similarly, the other samples are partitioned as

$$d_{-1,k} = a_{0,2k+1}$$

where $d_{-1,k}$ are the difference, or wavelet, coefficients. This partition corresponds to the so- called Lazy wavelet. This will clearly not decorrelate the signal (unless of course the coefficients are already small, which is an unreasonable assumption). While this step does not achieve much, the next steps, predict and update, will show more explicitly how this partitioning method can be used to achieve our goal of decorrelating a signal in a recoverable manner.

## 3.2 Prediction Step

This section covers the prediction stage. Three main topics are discussed here. First, we discuss the main idea behind prediction and how this can lead us to linear prediction. Next, we generalize this type to other functions by deriving the steps for cubic prediction. Finally, we discuss the problems that can occur along the boundaries of signals and how to determine filter coefficients, which are used for prediction.

### 3.2.1 Linear Prediction

As previously mentioned, we seek a more compact representation of $a0,k$. Consider the case where $d_{-1,k}$ is 0 for all $k$ (i.e., $d_{-1,k}$ contains no information). In this case, by discarding these values, keeping only $a_{-1,k}$, we can have a more compact representation and still have the ability to reconstruct $a_{0,k}$. Unfortunately, this is unlikely to occur with real signals of interest.

Instead, another approach is tried. We apply a prediction step in order to find the odd samples from the even ones. This step is constructed such that it is based on the correlation of the signal at level 0. That is, we apply a prediction operator $P$:

$$d_{-1,k} = P(a_{-1,k})$$

so that we can then replace the original data set with $a_{-1,k}$. Hence, the missing samples can be predicted in order to recreate $a_{0,k}$. However, it may still not be possible to exactly recreate, or predict, $d_{-1,k}$ from $a_{-1,k}$. We can then replace $d_{-1,k}$ with the difference between itself and its predicted value $P(d_{-1,k})$. Since $P(d_{-1,k})$ and $d_{-1,k}$ are probably similar in value, the difference will be small relative to the original $d_{-1,k}$ values. Hence, we obtain

$$d_{-1,k} = a_{0,2k+1} - P(a_{-1,k})$$

The signal now encodes how much data deviates from the model on which $P$ was built. If the signal is correlated and fits our model, the majority of the wavelet coefficients will be small.

So, in order to exploit data redundancy of the data most effectively, we need the sets $a_{-1,k}$ and $d_{-1,k}$ to be correlated as much as possible. This is why we split between even and odd samples in the splitting stage instead of splitting the signal into left and right halves: to take advantage of spatial and temporal locality that is likely present. As a first approach, we can predict an odd sample $d_{0,2k+1}$ as the average value of its even neighbors from the previous level, that is, $a_{-1,k}$ and $a_{-1,k+1}$. Hence, we have the prediction

$$d_{-1,k} = a_{0,2k+1} - 1/2 * (a_{-1,k} + a_{-1,k+1})$$

This prediction scheme assumes the signal is sampled over intervals of length 2 from a piecewise linear function. In effect, the wavelet coefficients, $d_{-1,k}$, express the degree with which the signal fails to be linear. This is shown in Figure 3. For a signal that fits this model behavior, the wavelet coefficients will be smaller than for a signal that does not.

After applying this procedure, we have the two sets $a_{-1,k}$ and $d_{-1,k}$. We can perform this step again to obtain $a_{-2,k}$ and $d_{-2,k}$ from $a_{-1,k}$. From some level $j$, we move to the next level by successively splitting the signal to yield $a_{-j-1,k}$ and $d_{-j-1,k}$ from $a_{-j,k}$. Finally, we replace the original signal with the set $\{a_{-n,k},\ d_{-n,k},\ \ldots,\ d_{-1,k}\}$ after $n$ steps or levels. Hence, the original signal has been replaced by a single low-pass coefficient and several high-pass wavelet coefficients. Again, if the data fits our model, then we expect the elements of this set to be smaller in magnitude.

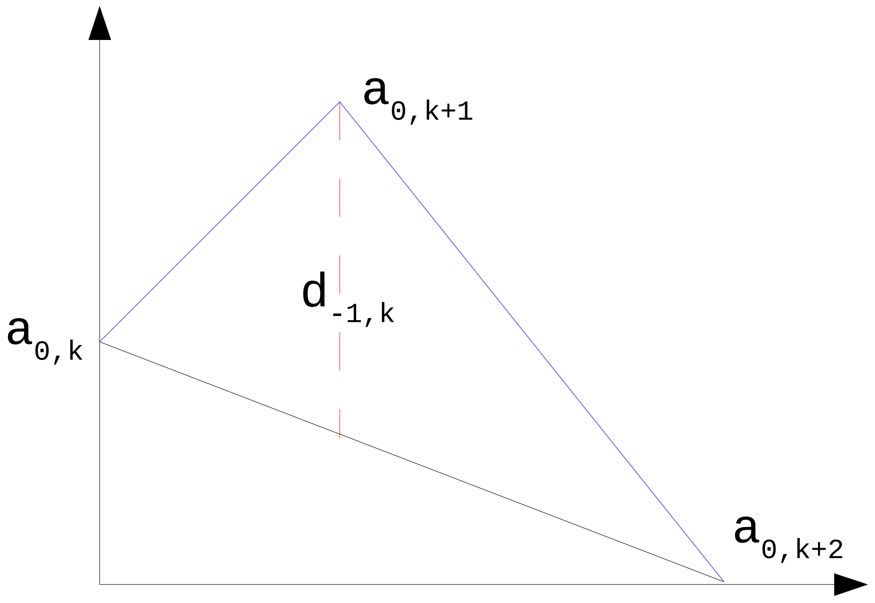

Figure 3: Wavelet coefficient represents failure of signal to be linear

### 3.2.2 Cubic Prediction

Linear prediction is neither necessary nor ideal. The prediction function could also determine the failure to be cubic or any other higher order function. A higher order prediction scheme based on this idea is referred to as interpolating subdivision. As before, we use a recursive procedure for finding the value of an interpolating (prediction) function at every level. The order of the subdivision scheme is denoted by N. For instance, setting N=2 produces the piecewise linear approximation given above. Setting N=4, yields a cubic approximation. The smoothness of the interpolating function used to find the wavelet coefficents is determined by N. Such a function is called the dual wavelet with N determining the number of dual vanishing moments.

Suppose we set N to 4 so that we can apply cubic interpolation. We now use four values, two to the left and two to the right of each $d_{j-1,k}$, to obtain $d_{j,k}$. Since we have four points, we can define a cubic polynomial that passes through each point. That is, there exists a unique cubic polynomial $p(x)$ such that

- $a_{j,k-1} = p(x_{j,k-1})$,
- $a_{j,k} = p(x_{j,k})$,
- $a_{j,k+1} = p(x_{j,k+1})$, and
- $a_{j,k+2} = p(x_{j,k+2})$

where $x_{j,i}$ marks the position of each signal value at the $j^{th}$ stage. So, we find the value of the signal at the next higher stage ($a_{j+1,2k+1}$) at the middle point ($x_{j+1,2k+1}$) by evaluating the polynomial at this point

$$a_{j+1,2k+1} = p(x_{j+1,2k+1}).$$

This can be seen in Figure 4.

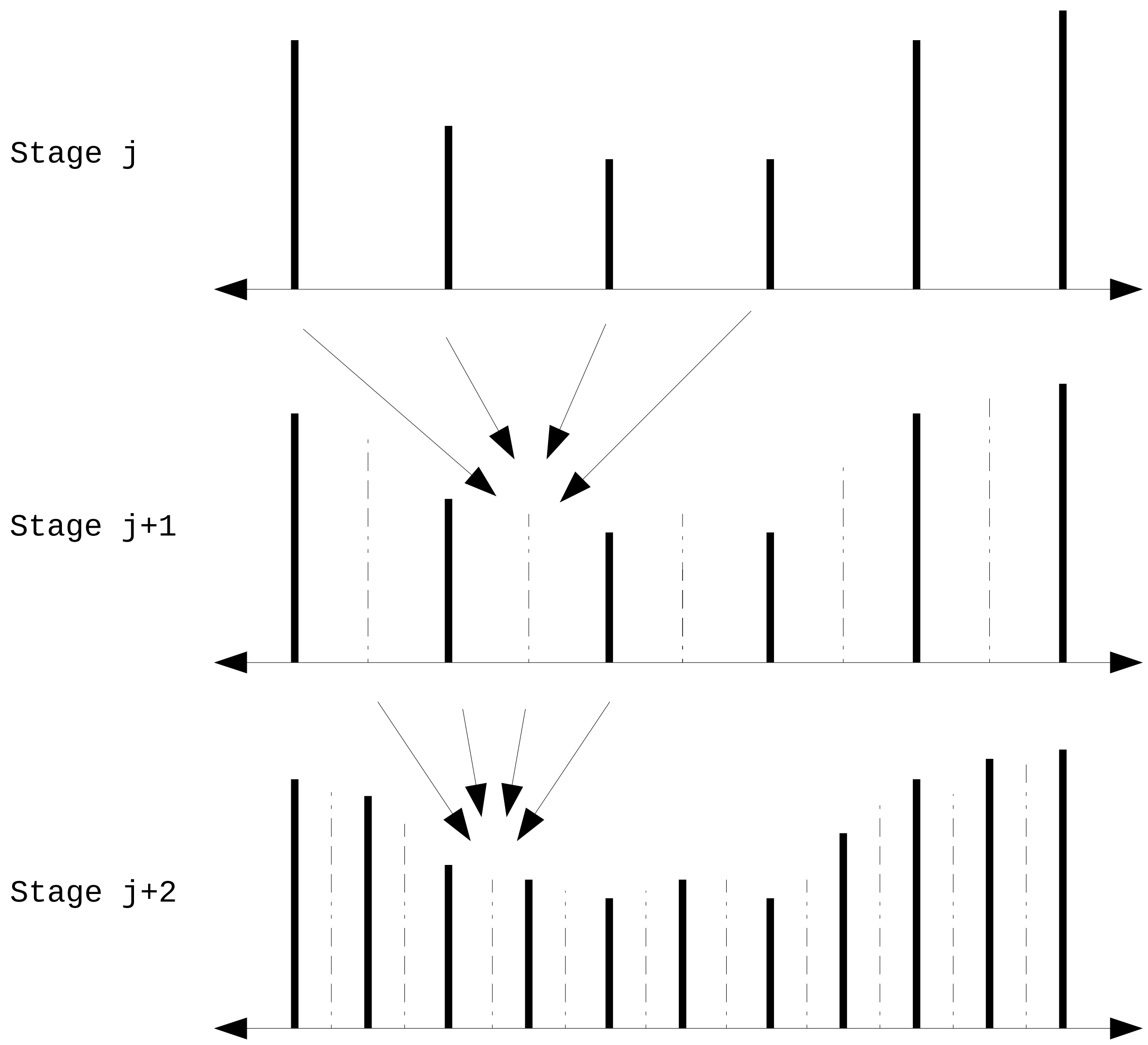

Figure 4: Two neighbors from each side are used to iteratively find the unique cubic polynomial that interpolates these points

The prediction function P uses polynomial interpolation of order N-1 to find the predicted values. If the original data resembles an order N-1 polynomial, then the $d_{j,k}$ will be close to zero as the difference between the original signal and predicted values will be small. The evaluation of this polynomial can be performed efficiently using Neville's algorithm, which recursively generates higher order polynomials from lower order:

$$P_{i(i+1)\dots(i+m)} = \frac{(x - x_{i+m})P_{i(i+1)\dots(i+m-1)} - (x - x_i)P_{(i+1)(i+2)\dots(i+m)}}{x_i - x_{i+m}}$$

### 3.2.3 Border Issues and Filter Coefficients

For the cubic case (or higher), we have a problem when the point we are approximating does not have two points to its left and right. We have the following cases as shown in Figure 5:

- Left Boundary—1 on the left and 3 on the right
- Middle—2 on the left and 2 on the right
- Right Boundary—3 on the left and 1 on the right (or 4 on the left and one on the right)

Note that in the first case with a left boundary the splitting process ensures that an a coefficient always resides in the first position of the signal.

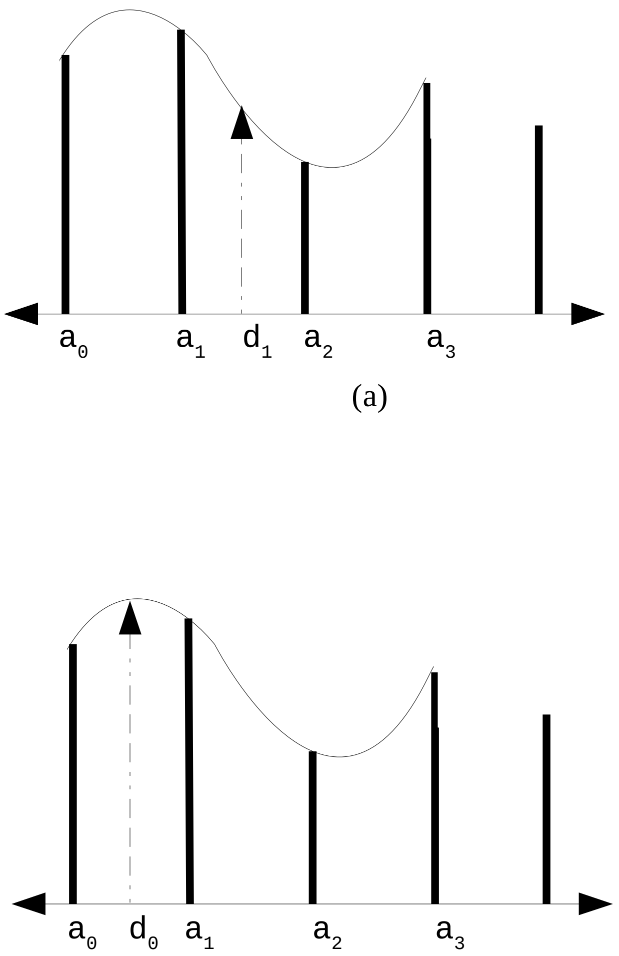

Figure 5: In (a), the middle value can be evaluated using two values on both the left and right. However, in (b), only one value lies to its left, so three values must be used from its right.

An example of how to calculate the coefficients for a cubic interpolation is shown in Figure 6. We want to find a set of coefficients for the cases listed above. That is, when there are two a's on either side and when there are one a on the left and three d's on the right. These occur at positions $x_1$ and $x_2$ in the figure, respectively.

Since N is equal to 4, we will have 4 coefficients for each case. To calculate the first filter coefficient, say $f_1$, for each case, we do the following: set $f_1$'s value to 1 and set the other three values, $f_2$, $f_3$, and $f_4$, to zero. Next, construct the polynomial that interpolates our points (e.g., a cubic polynomial) and evaluate the function at the positions we need. Hence, we evaluate the polynomial where we have two coefficients to the left and two to the right (x1) to get $p(x_1)=-.0625$. Similarly, we evaluate the polynomial at $x_2$, since there is one coefficient on the left and three on the right, to obtain $p(x_2)=.3125$. Now, we have the two first coefficients for our two cases. The procedure is the same for the next coefficient: set $f_2$ and $f_1$, $f_3$, and $f_4$ to zero; construct the interpolating cubic polynomial p; and evaluate p at $x_1$ and $x_2$. This is shown in Figure 6(b). The process is continued for the other coefficients.

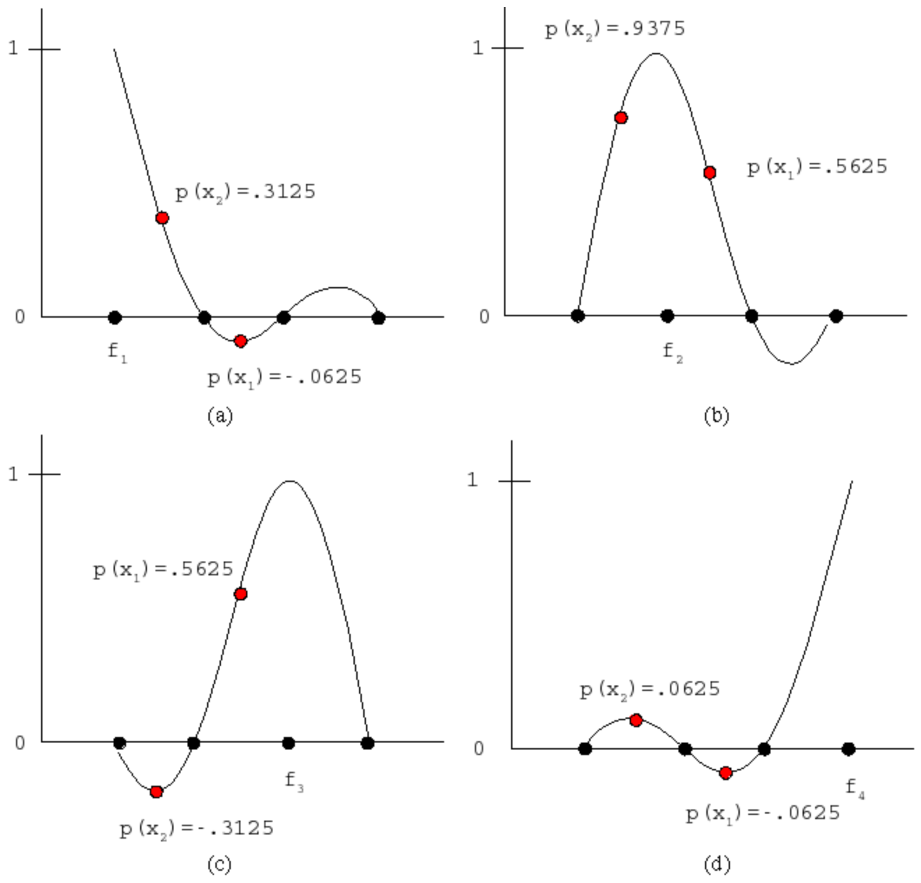

Figure 6: Determining filter coefficients for N=4

Tables 1 and 2 list the filter coefficients needed for the interpolation with N=2 and N=4. We note that with our splitting method the first case, with 0 a's on the left, does not occur as we always have one a coefficient in the first position. The prediction phase thus gets reduced to a lookup in the previous tables in order to be able to calculate the wavelet coefficients. For example, if we want to predict a d value using N=4 and three d coefficients on the left and one on the right, we would perform the following operation:

$$d_{-j,k} = a_{-j+1,k} - (.0625*a_{-j,k-3} - .3125*a_{-j,k-2} + .9375*a_{-j,k-1} + .3125*a_{-j,k+1})$$

The prediction of other d coefficients would be a similar process except that we would use neighboring a coefficients and the filter coefficients corresponding to the case under study. Note that, as seen in both the table and the figure, the polynomial is symmetric, which is due to the fact that the coefficents' positions are equally spaced apart. Hence, we do not have to explicitly compute the cases in which the number of coefficients to the left is greater than the number to the right.

| ***# left*** | ***# right*** | ***k - 3*** | ***k - 1*** | ***k + 1*** | ***k + 3*** |
|---|---|---|---|---|---|
| 0 | 2 | | | -.5 | 1.5 |
| 1 | 1 | | .5 | .5 | |
| 2 | 0 | 1.5 | -.5 | | |

Table 1: Filter coefficients for N=2

| ***# left*** | ***# right*** | ***k - 7*** | ***k - 5*** | ***k - 3*** | ***k - 1*** | ***k + 1*** | ***k + 3*** | ***k + 5*** | ***k + 7*** |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | | | | | 2.1875 | -2.1875 | 1.3125 | -.3125 |
| 1 | 3 | | | | .3125 | .9375 | -.3125 | .0625 | |
| 2 | 2 | | | -.0625 | .5625 | .5625 | -.0625 | | |
| 3 | 1 | | .0625 | -.3125 | .9375 | .3125 | | | |
| 4 | 0 | -.3125 | 1.3125 | -2.1875 | 2.1875 | | | | |

Table 2: Filter coefficients for N=4

## 3.3 Update Step

We have one step left: update. The goal at this stage is to maintain some global properties of the original signal in the reduced set. For example, if the signal is a 2D image, we would like the average pixel intensity value to be the same across all scales of the transform, which implies that the final coefficient at the last scale is the average pixel value of the entire, original image. We design a system to preserve some fixed Ñ moments of the wavelet function, $\psi$ , at each level. Ñ is referred to as the number of real vanishing moments. The first Ñ moments are given by

$$\int \psi(x)\,dx = 0,\ \int x\psi(x)\,dx = 0,\ \int x^2\psi(x)\,dx = 0,\ \ldots,\ \int x^{\tilde{N}-1}\psi(x)\,dx = 0$$

At each scale, we want to preserve up to Ñ- 1 of the a's at every level and use this information to see how much of every d coefficient is needed to update every a. The coefficients used for this update procedure are named lifting coefficients.

To find these values, we apply the following algorithm:

- First, initialize the moments as follows:

| Moment 1 | 0 | 1 | 2 | 3 | 4 | 5 | ... | Ñ |
|---|---|---|---|---|---|---|---|---|
| Moment 2 | 0 | 1 | 4 | 9 | 16 | 25 | ... | $Ñ^2$ |
| Moment 3 | 0 | 1 | 16 | 27 | 64 | 125 | ... | $Ñ^3$ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| Moment Ñ | 0 | 1 | $2^{Ñ}$ | $3^{Ñ}$ | $4^{Ñ}$ | $5^{Ñ}$ | ... | $Ñ^{Ñ}$ |

- For each $d_l$, determine the $a_j$ coefficients that contributed to predicting $d_l$ and the weighting of this, i.e., find the filter coefficients as given above for prediction.
- Update the moments for every $a_j$ at the current level with the following equation
  $$m_{j,k} = m_{j,k} + f_j * m_{l,k}$$
  where j is the index relative to an a coefficient, f(j) is its corresponding filter coefficient (0<j<=N), k is the moment being updated (0<k<=Ñ-1), and l is the index to a d coefficient.
- The moments must be zero at every level. Hence, we can construct a linear system to find the lifting coefficients for every d. The steps are:
  - For a d coefficient, set its value to one and set all remaining d's to zero.
  - Apply an inverse transform one step up to determine the contribution of this d to the a's that update it and create a linear system of ÑxÑ variables as follows:
  $$\begin{bmatrix} m_{l_0,0} & m_{l_1,0} & \cdots & m_{l_N,0} \\ \vdots & \vdots & \cdots & \vdots \\ m_{l_0,\tilde{N}} & m_{l_1,\tilde{N}} & \cdots & m_{l_N,\tilde{N}} \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_{\tilde{N}} \end{bmatrix} = \begin{bmatrix} m_{g_j,0} \\ \vdots \\ m_{g_j,\tilde{N}} \end{bmatrix}$$
  where $c_i$ are the lifting coefficents to be found, $m_{li,k}$ marks the ith order moment with $l_i$ the index relative to the a coefficents at the current level, and $M_{gj}$ is the column from the moments matrix in which the d's value is one. Note that j's value varies from 1 to the length of the signal.
  - Solve the system to yield the set of lifting coefficients for the d coefficient.

For each d at each level, the above procedure is used to produce Ñ lifting coefficients.

After the lifting coefficents have been constructed, the update stage of the lifting scheme can be applied. Suppose Ñ=2, and let $d_{-j,k}$ be a wavelet coefficent at level $-j$ and position $k$. Then we have 2 lifting coefficents associated with $d_{-j,k}$, say $c_1$ and $c_2$. To update the average coefficents, we do the following:

$$a_{-j,k-1} = a_{-j,k-1} + c_1 * d_{-j,k} \text{ and } a_{-j,k+1} = a_{-j,k+1} + c_2 * d_{-j,k}$$

This is done for each $d$.

## 4 Lifting Scheme Example

We now provide a small example to help clarify how the lifting scheme works. Suppose we have a signal $a = [a_0\ a_1\ ...\ a_7\ ]$. Let N=2 and Ñ=2. Since N=2, we use the filter coefficents given in Table 1. First, we apply splitting and prediction:

| $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ |
|---|---|---|---|---|---|---|---|
| | $d_0$ | | $d_1$ | | $d_2$ | | $d_3$ |

The differences, $d_k$, are predicted from the signal as follows:

- $d_0 = d_0 - (.5 * a_0 + .5 * a_2)$
- $d_1 = d_1 - (.5 * a_2 + .5 * a_4)$
- $d_2 = d_2 - (.5 * a_4 + .5 * a_6)$
- $d_3 = d_3 - (1.5 * a_4 + (-.5) * a_6)$

Note that $d_4$ uses different filter coefficients since no coefficients are to its right.

For the update stage, we assume that the lifting coefficents have already been found. The pairing between each $d_k$ and its two lifting coefficents (since Ñ=2) and the setup for update are given here:

| | $d_0$ | | $d_1$ | | $d_2$ | | $d_3$ |
|---|---|---|---|---|---|---|---|
| | 2/5,1/5 | | 0,2/3 | | 4/15,4/5 | | -2/15,2/5 |
| $a_0$ | | $a_2$ | | $a_4$ | | $a_6$ | |

Here we have, for example, the lifting coefficents $c_{1,1} = 2/5$ and $c_{1,2} = 1/5$ associated with $d_1$, $c_{2,1} = 0$ and $c_{2,2} = 2/3$ associated with $d_2$ , etc. The averages, $a_k$, are updated from the differences and lifting coefficients as follows:

- $a_0 = a_0 + (2/5 * d_0)$
- $a_2 = a_2 + (1/5 * d_0 + 0 * d_1)$
- $a_4 = a_4 + (2/3 * d_1 + 4/15 * d_2)$
- $a_6 = a_6 + (4/5 * d_2 + -2/15 * d_3)$

We then go to the next level of the transformation and apply splitting and prediction again but only on the remain averages:

|  |  |  |  |
|---|---|---|---|
| $a_0$ | $a_2$ | $a_4$ | $a_6$ |
|  | $d_0$ |  | $d_1$ |

We obtain the lifting coefficents for the differences at this level to update the remaining averages:

|  |  |  |  |
|---|---|---|---|
|  | $d_0$ |  | $d_1$ |
|  | 1/2,.214286 |  | -1/3,.47619 |
| $a_0$ |  | $a_4$ |  |

For a longer signal, the interior $a_k$ would have had $d_{k-1}$ on its left and $d_k$ on its right, so the lifting coefficents used in the update would have been 1/4 for both $c_1$ and $c_2$. This would have lead to the update rule

$$a_{-1,k} = a_{-1,k} + 1/4 * (d_{-1,k-1} + d_{-1,k})$$

for those points $a_k$ not near the boundary.

## 5 Experiments and Observations

In this section, we present our results from applying the lifting scheme to a various sets of images. We test our system on 3 image sets obtained from Microsoft© Corporation's computer vision research:

- office—various office environments
- signs—close-up of street signs
- trees—trees and plants found outdoors

These were selected to provide a comparison in the lifting scheme's ability to decorrelate varying types data. Each set contains 50 gray-scale images, and each image has dimension of either 640x480 or 480x640.

For each image, we perform a 5-level wavelet transformation using the lifting scheme. As the transform is separable, the rows of the image are filtered first, which is followed by column filtering. We then calculate the 0-order entropy of the image. That is, we compute

$$H(I) = -\sum p(x) * \log_2(p(x))$$

where the summation is over all pixel values $x$ that occur in image $I$.

We give the entropy for several types of lifting:

- The original signal with no lifting applied
- The Haar lifting
- 2 real and 2 dual vanishing moments
- 4 real and 4 dual vanishing moments
- 9 real and 7 dual vanishing moments (corresponds to the wavelet used for fingerprint compression by the FBI)

For each type given above, we record the entropy for each image and provide the following summary data for each image set:

- Average entropy
- Variance of entropy
- Standard deviation of entropy
- Minimum entropy recorded
- Maximum entropy redecorded

A sample image from each set is shown. Alongside each image, we also give the image resulting from application of the lifting scheme to the image. The details of the transformed image can be hard to discern as we have reduced the image size, so we have provided the transformed images in the Appendix as well at the original size to show this detail.

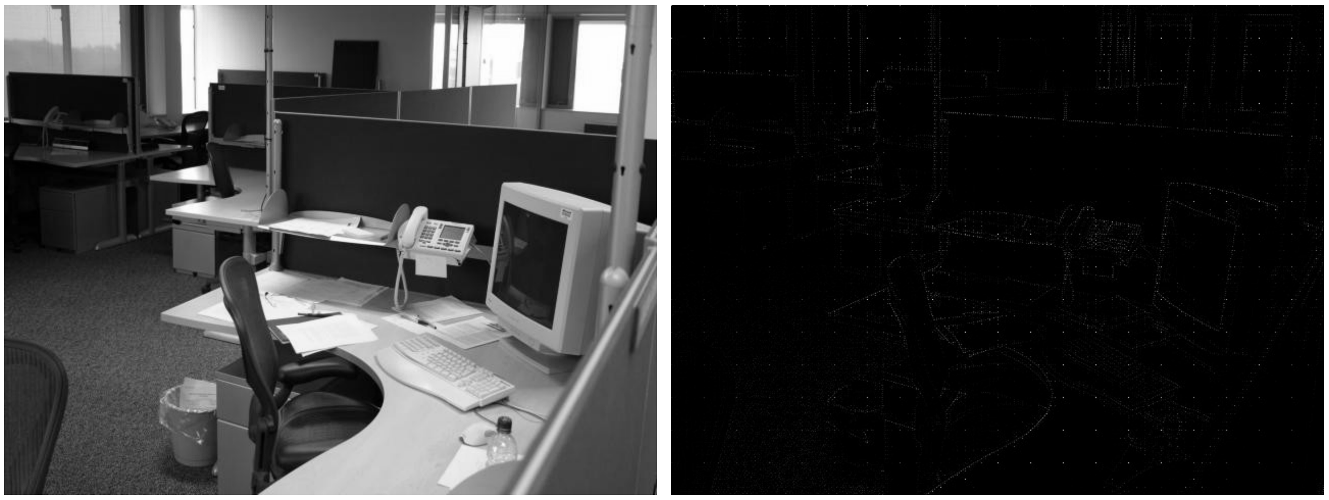

Figure 7: Image from 'office' image set and its lifted representation

| **Lift Type** | **Average** | **Variance** | **Std. Deviation** | **Min** | **Max** |
|---|---|---|---|---|---|
| Orignal | 7.44 | .05 | .22 | 6.87 | 7.88 |
| Haar | 5.18 | .42 | .65 | 4.07 | 6.67 |
| 2, 2 | 4.63 | .28 | .53 | 3.69 | 5.82 |
| 4, 4 | 8.75 | .43 | .66 | 7.57 | 10.18 |
| 9, 7 | 3.94 | .24 | .49 | 3.04 | 5.11 |

Table 3: Entropies for 'office' image set

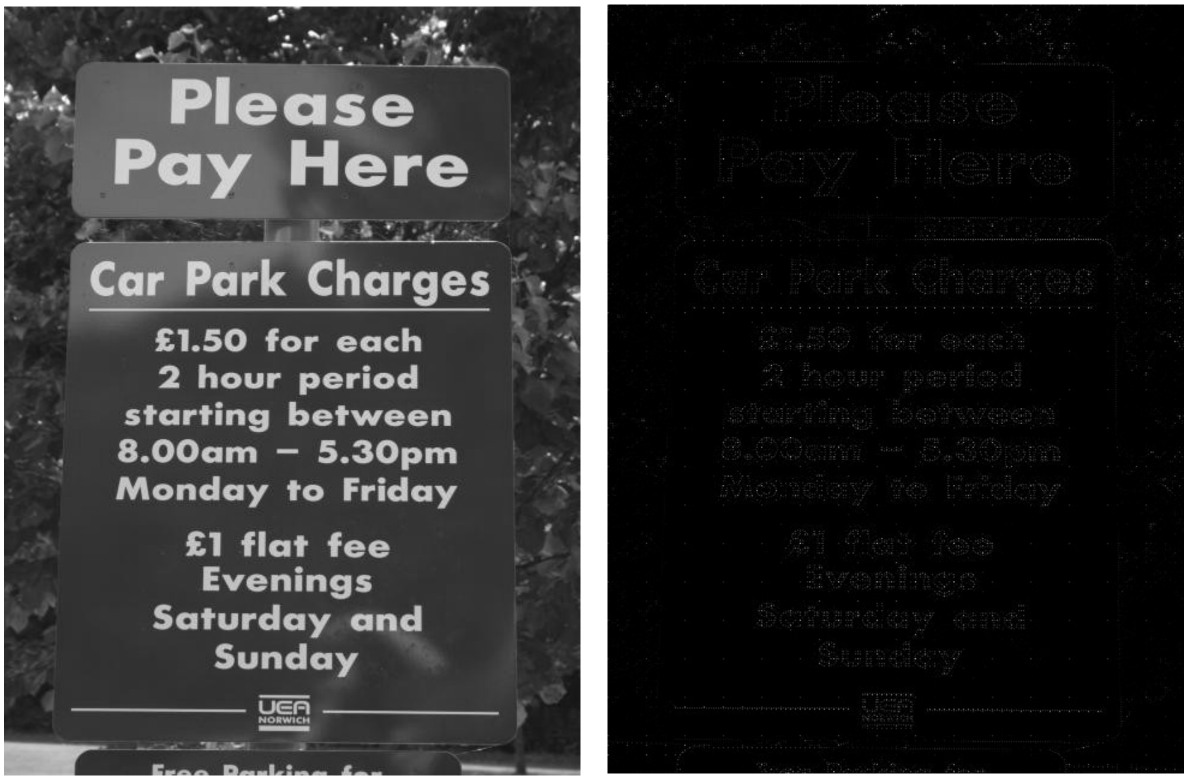

Figure 8: Image from 'signs' image set and its lifted representation

| **Lift Type** | **Average** | **Variance** | **Std. Deviation** | **Min** | **Max** |
|---|---|---|---|---|---|
| Orignal | 6.74 | .34 | .59 | 5.04 | 7.61 |
| Haar | 5.83 | 1.29 | 1.13 | 1.66 | 7.72 |
| 2, 2 | 5.16 | .82 | .91 | 1.87 | 6.88 |
| 4, 4 | 9.37 | 1.40 | 1.18 | 3.97 | 11.27 |
| 9, 7 | 4.42 | .69 | .83 | 1.57 | 5.94 |

Table 4: Entropies for 'signs' image set

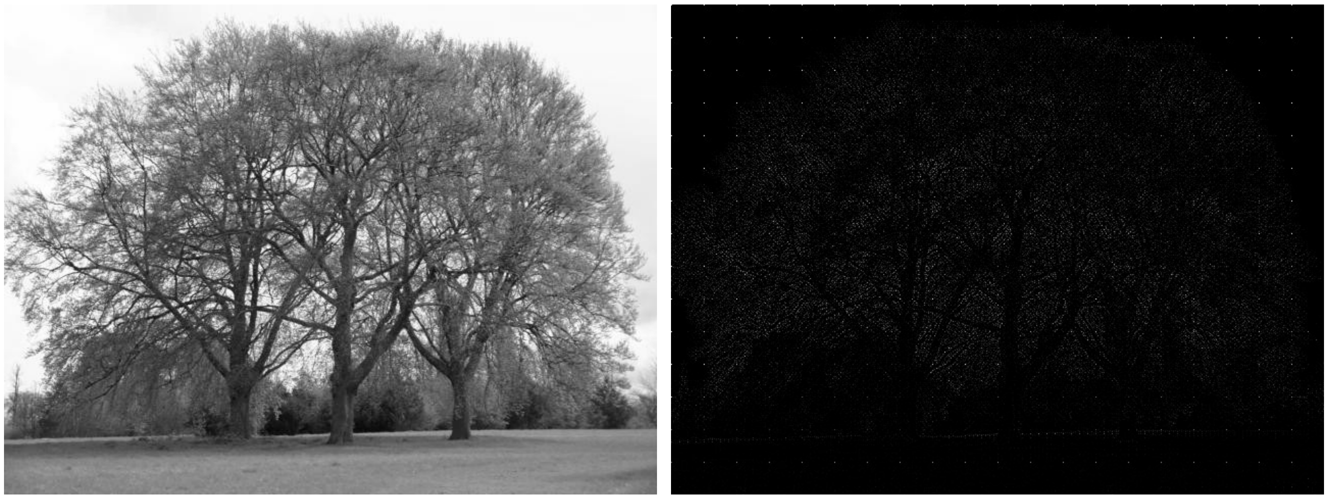

Figure 9: Image from 'trees' image set and its lifted representation

| **Lift Type** | **Average** | **Variance** | **Std. Deviation** | **Min** | **Max** |
|---|---|---|---|---|---|
| Orignal | 7.21 | .27 | .52 | 5.69 | 7.86 |
| Haar | 7.31 | .53 | .73 | 5.82 | 8.93 |
| 2, 2 | 6.68 | .47 | .68 | 5.35 | 8.21 |
| 4, 4 | 10.72 | .81 | .90 | 8.60 | 12.59 |
| 9, 7 | 5.77 | .34 | .58 | 4.59 | 7.11 |

Table 5: Entropies for 'trees' image set

Table 6 summarizes the performance of the lifting scheme applied to these image sets. We observe that for all sets that the popular (9, 7) wavelet yielded the lowest entropy on average. This wavelet has the highest number of vanishing moments. Both the Haar and (2, 2) wavelet helped to decorrelate the image. The (4, 4) wavelet yielded a worse entropy thatn the original image; the reason for this is unclear.

| **Image Set** | **Original** | **Haar** | **2, 2** | **4, 4** | **9, 7** |
|---|---|---|---|---|---|
| office | 7.44 | 5.18 | 4.63 | 8.75 | 3.94 |
| signs | 6.74 | 5.83 | 5.16 | 9.37 | 4.42 |
| trees | 7.21 | 7.31 | 6.68 | 10.72 | 5.77 |

Table 6: Summary of average entropies for each image set

## 6 Summary and Conclusions

Lossless image compression is an important topic with a variety of applications, such as medical imaging and seismic data analysis, as the processing and transmission of very large data sets is required, yet errorless image compression must be maintained. The ability to maximally exploit redundancy in image data is important. The wavelet transform has been shown to provide this task with excellent results.

The lifting scheme is a relatively new approach for constructing wavelets. Unlike classical wavelets, the lifting scheme generates wavelets in the spatial domain instead of the frequency domain. Hence, the transform can be derived and understood independent of the Fourier transform. The lifting scheme is implemented in three stages:

- Split—divide the signal into two sets, even and odd, according to its position in the signal;
- Predict—predict the odd set using the even set as the failure to predict the odd set from the even set in order to calculate the wavelet coefficents;
- Update—update the even set using the wavelet coefficents to compute the scaling function coefficents.

For biorthogonal wavelets, these stages are constructed to ensure certain properties:

- Prediction provides polynomial cancelation for the high-pass frequencies
- Update guarantees preservation of the moments for the low-pass frequencies

The parameter N corresponds to the degree of the polynomial for prediction, while Ñ determines the number of moments preserved in the update stage. Hence, various transformations can be built by modifying the number of dual and real vanishing moments (N and Ñ).

In our experiments, we have seen that the lifting scheme can help decorrelate data. Using the 0-order entropy as a metric, the Haar, (2, 2), and (9, 7) wavelets derived from the lifting scheme reduced the average entropy of our image sets. Hence, applying an entropy encoder, such as arithmetic, after the lifting step should yield improved compression rates over an entropy encoding with no transformation.

Interesting future work utilizing the lifting scheme may involve the following topics:

- Integer transformation,
- Face detection.

Theoretically, the lifting scheme provides for perfect reconstruction of a signal. However, due to the practical limitation of finite precision arithmetic, an image that has been lifted may not be perfectly reconstructed once it has been written to file. Luckily, the lifting scheme can be altered to produce only integers in the transformed image. Hence, from both a theoretical and practical viewpoint, the resulting lifted image can be used reconstruct the original image.
The alteration to the forward algorithm is very simple:

```
for j = -1 to -n
  for each position k in $a_{j+1}$
    ($a_{j,k}$, $d_{j,k}$} = split($a_{j+1,k}$)
        d_{j,k} -= floor( predict( a_{j,k} ) + 1/2 )
        a_{j,k} += floor( update( d_{j,k} ) + 1/2 )
```
That is, we add 1/2 to both the prediction and update steps, apply the `floor()` function to round the result to an integer, and record this value. Unfortunately, initial tests were unable to maintain perfect reconstruction, so more work must be done concerning this matter.

One other possible application of the lifting scheme is object detection. In particular, Schneiderman et al. have developed a highly accurate object detection system using the wavelet tranform and applied it the task of face detection. The lifting scheme may possibly be substituted for the classical wavelet being used to improve the efficiency of the algorithm. Also, the lifting scheme may improve results by correctly handling image boundaries without the introduction of artifacts.

## References

[1] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Losless image compression using integer to integer wavelet transforms. In *International Conference on Image Processing (ICIP), Vol. I,* pages5 96- 599. IEEE Press, 1997.

[2] 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.

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

[4] G. Fernández, S. Periaswamy, and Wim Sweldens. LIFTPACK: A software package for wavelet a transforms using lifting. In M. Unser, A. Aldroubi, and A. F. Laine, editors, *Wavelet Applications in Signal and Image Processing IV,* pages 396- 408. Proc. SPIE 2825, 1996.

[5] David Salomon. *Data Compression: The Complete Reference*. Springer, 2004.

[6] Henry Schneiderman and Takeo Kanade. Object detection using the statistics of parts. International Journal of Computer Vision, pages 151- 177, 2004.

[7] W. Sweldens. Wavelets and the lifting scheme: A 5 minute tour. *Z. Angew.* Math. Mech., 76 (Suppl. 2):41- 44, 1996.

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

[9] Vision at MSR Cambridge. *Microsoft Research Cambridge.* 2004, Microsoft Corporation,. 11 Nov. 2005 <http://www.research.microsoft.com/vision/cambridge/recognition/default.htm>.

## Appendix

For our sample images, we again show the lifted images. However, each image is given at its original size to better show the detail and structure of the wavelet coefficients constructed. Note how the coefficients are not divided into the familiar “octave” representation as the transformation is performed in-place. It is possible, however, to represent a lifted image in this form as well at the cost of some additional work.

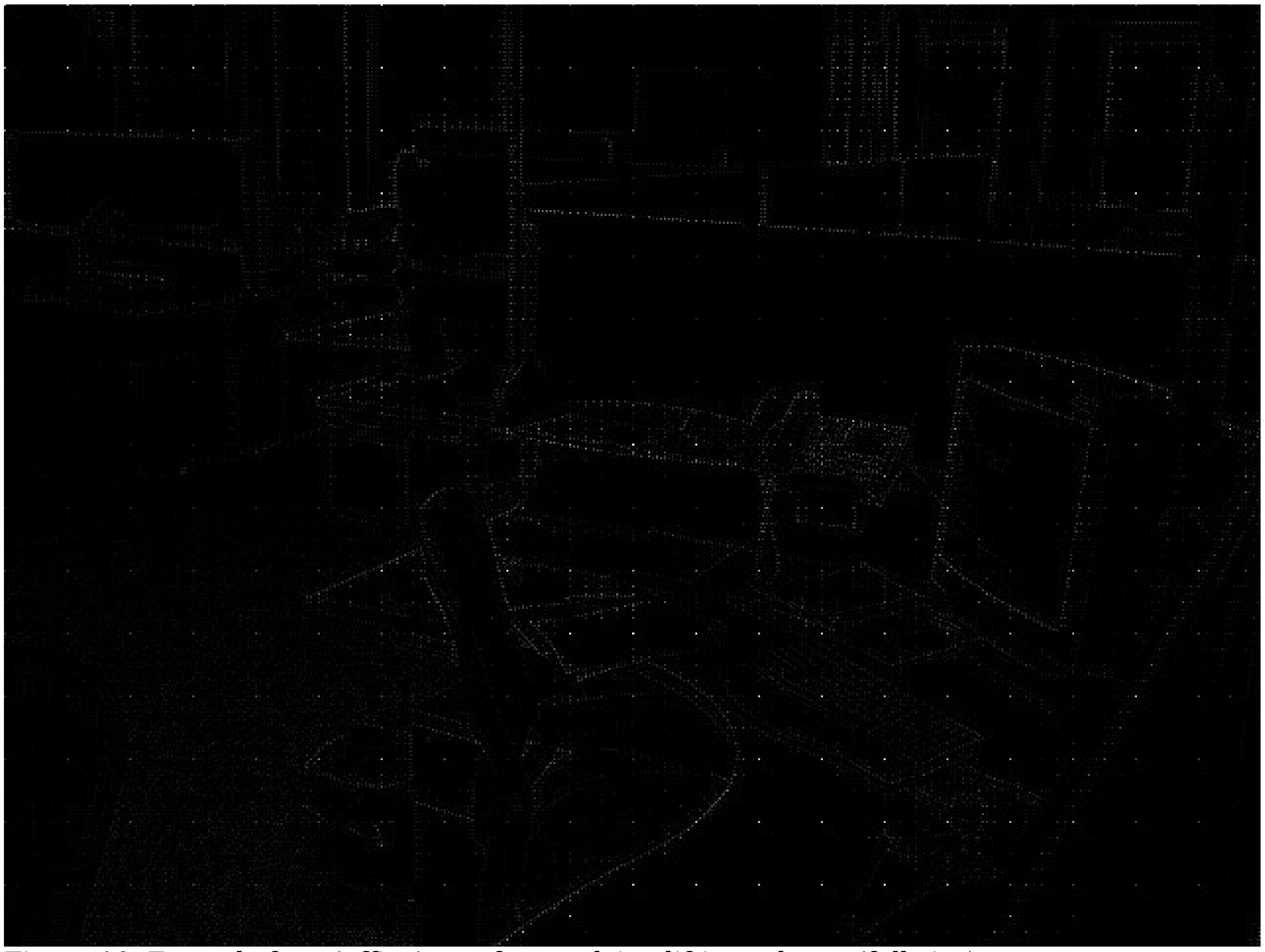
Figure 10: Example from 'office' set after applying lifting scheme (full-size)

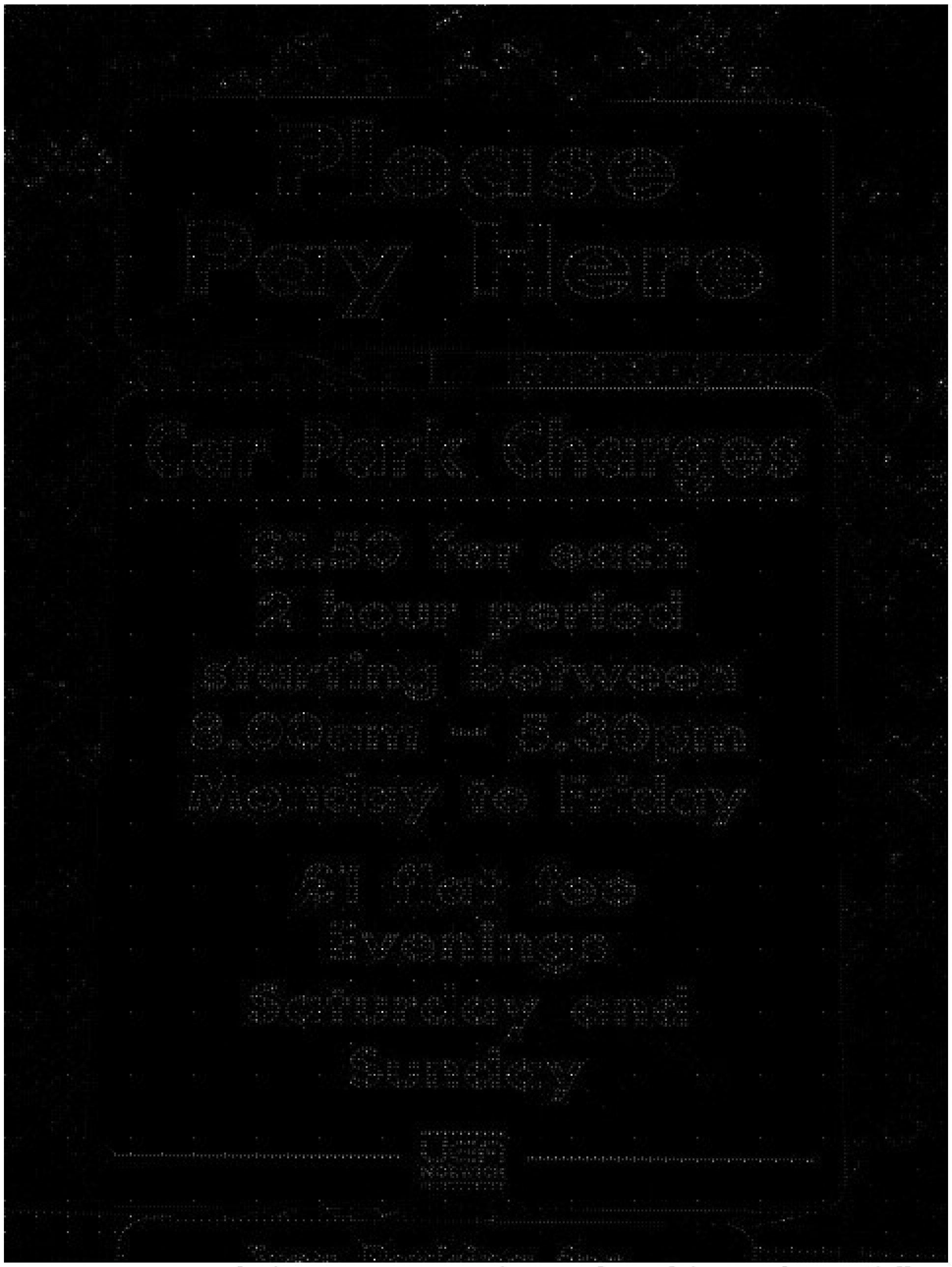

Figure 11: Example from 'signs'' set after applying lifting scheme (full-size)

Figure 12: Example from 'trees'' set after applying lifting scheme (full-size)