An Approach to Integer Wavelet Transformations for Lossless Image Compression

Hongyang Chao, Paul Fisher and Zeyi Hua

September 1996, Last revised December 1997

Abstract. We developed a general way to create integer wavelet transformations that can be used in lossless (reversible) compression of images with arbitrary size. The method, based on some updating techniques such as lifting and correction, allows us to generate a series of reversible integer transformations which have the similar features with the corresponding biorthogonal wavelet transforms and some non-orthogonal wavelet transforms; but need only be calculated with integer addition and bit-shift operations. In addition, the integer wavelet transforms created in this paper possess a property of precision preservation (*PPP*). This property is very useful, in lossless compression, for conserving memory in both compression and decompression, and speeding up the computational process.

§1. Introduction

The wavelet transform has been proven to be one of the most powerful tools in the field of image compression. In general, wavelet based image compression can be divided into three steps: wavelet transform, quantization and entropy coding. Theoretically, the wavelet transformation part is considered lossless since the transformation is reversible in the sense of mathematics. However, most transformations are lossy in practice, because all computers have only finite precision, even if we use floating point calculations. This fact is a limitation for lossless image compression by wavelet transform based algorithms. We are going to develop a general way to create reversible integer transforms, especially wavelets. All of wavelet transforms created here possess the property of precision preservation (*PPP*), which we will explain later in this paper. There are several reasons for creating this kind of reversible wavelets:

The first reason is, of course, for the purpose of lossless image compression. Lossless image compression is very important for images found in such applications as medical and space science. In such situations, the designers of the compression algorithm always try to compress the information as much as possible; and they also must be very careful to avoid discarding any information that may be required or even useful at some later point. Some of lossless image compression algorithms are based on the so called predictive transforms [5, 8-9], which are widely used in practice. However, this method

only produces a single resolution after transform, which retard the progressive transmission or recovery. As we've already known, wavelet is one of the best tool for multiresolution analysis. Therefore, it is an interesting challenge to build integer reversible wavelets for lossless compression.

Second, most transforms expand the range of the pixel values after transformation. For example, we assume there is an 8-bit gray-scale image with values among [-128, 127]. After performing any simple transformation, the values of its coefficients would exceed the previous range. In order to have correct results, we have to allocate the working unit as, at least, short integer type, which requires twice as much memory as the original. In general, the transformed coefficient image has less information entropy than the original, which is also the purpose of transformation. So, it is very appealing, in lossless compression requirement, to have some compression algorithm that just needs to process the data within the same precision as the original images. This fact can also be utilized in the computer with limited precision and limited memory without losing any precision during the computation. For this purpose, the wavelet transform with the property of precision preservation (PPP) is a good choice. This paper will provide a formulation of such wavelets.

Third, even for lossy compression, people prefer to use integer computation instead of the floating-point since integer calculations are faster than floating point for many computers; and integer computations are much easier to implement in hardware, which is more important in some applications. The memory utilization of integers is also a positive consideration. Therefore, the integer reversible wavelet, which avoids the propagation of the rounding error, is definitely very useful in the matter. From the academic point of view, it is also very interesting to have a compression scheme which has very fast performance, and which can exactly reconstruct the image when necessary.

We draw on the work of several other authors who have already contributed to this area [2-3], where some specific examples were developed. However, this paper presents a more general method, which allows one to see several new results as well as those presented and acknowledged prior to this work.

This paper is organized as follows: Section 2 and 3 give some examples of integer wavelet transforms. The examples in section 2 are the starting point for our approach, and we will indicate several interesting ideas (including PPP) there. The examples in Section 3 show the steps and motivation of our general method. Section 4 indicates how one can use the lifting technique to create an integer biorthogonal wavelet transform. The Correction technique to generate more general integer wavelet transforms is described in Section 5. Section 6 describes how to process boundaries in order to apply the integer calculation in finite sized images or signals. In the last section, Section 7, we prove the integer wavelet transforms developed by both the lifting and correction method possess the property of precision preservation (PPP). Some applications and discussion are also shown in this section.

For the sake of convenience, length, and simplicity, we only discuss the algorithm for the one-level decomposition and reconstruction and only for the

one-dimensional signals. The extension to two dimensions is immediate as the rows and columns can be treated into a sequence of the one-dimensional signals. In the entire paper, assume that$\{c_n^0\}$ is the original signal where the superscript indicates level and the subscript indicates a particular point in the signal. Also,$\{c_n^1\}$ and $\{d_n^1\}$ are its decomposition parts at the first level. $\{c_n^1\}$ and $\{d_n^1\}$ is its low frequency (L) and high frequency (H) parts, respectively. For multi-levels, we just treat $\{c_n^1\}$ as $\{c_n^0\}$ and repeat the procedure again. If the signal has finite number of non-zero elements, we will deal with the boundary in the relevant sections.

§2. Some Simplest integer wavelets and the analysis

In this section, we will describe two integer wavelets. Although these wavelets are very simple, they are good start points. Also, we will introduce the concept of *PPP* with the aid of these examples.

Example 1. *Modified Haar wavelet transform.* It has been already known that the low and high pass analysis (decomposition) filters of *Haar wavelet* are given as

n	**1**	**1**
$\tilde{h}_n$	$1/2$	$1/2$
$\tilde{g}_n$	$1/2$	$-1/2$

The normalized *Haar wavelet* transform takes scaling factor $\sqrt{2}$ for both low and high frequency filtering, and the formula for forward transformation is:

$$\begin{cases} c_k^1 = \sqrt{2}(\frac{c_{2k}^0 + c_{2k+1}^0}{2}) \\ d_k^1 = \sqrt{2}(\frac{c_{2k}^0 - c_{2k+1}^0}{2}) \end{cases}$$

Obviously, it is impossible to obtain either forward or inverse transformations above preciously in a computer even if we use floating point computing. However, if we make the following modifications, we would be able to get a reversible integer transformation that still keeps the most features of *Haar wavelet*.

(a) Decomposition:

$$\begin{cases} d_k^1 = c_{2k}^0 - c_{2k+1}^0 \\ c_k^1 = c_{2k+1}^0 + \mathrm{Int}(\frac{d_k^1}{2}) \end{cases} \tag{2.1}$$

Here, $\mathrm{Int}(x)$ is an arbitrary rounding function, which may have different interpretations. For example, $\mathrm{Int}(x)$ can be the integer that is nearest to x; or $\mathrm{Int}(x)$ may be any integer that satisfies $x - 1 < \mathrm{Int}(x) \leq x$, etc. If all entries in the original signal $\{c_k^0\}$ are integer, those in both $\{c_k^1\}$ and $\{d_k^1\}$ are integers after (2.1). From (2.1), we can also easily obtain the following integer reconstruction algorithm:

(b) Reconstruction:

$$\begin{cases} c^0_{2k+1} = c^1_k - \mathrm{Int}(\dfrac{d^1_k}{2}) \\ c^0_{2k} = d^1_k + c^0_{2k+1} \end{cases} \tag{2.2}$$

The first modification we made was choosing proper scaling factors which are different from those in the *normalized Haar wavelet* transform. The factors for low-pass and high-pass filtering are 1 and 2, respectively. With these factors, the summation of all entries in low-pass filter is 1, and the filtering acts as a convex combination of the original signal. Therefore, after one-level decomposition, the range of the transformed values is still the same as that of original signal. Since the decomposition will be repeated in the low-pass phase, this feature will be kept to the end of the decomposition. The second modification is that the calculation order has been changed. The quantity $\{d^1_k\}$, obtained in the first step of (2.1), is utilized in the second step. Also, we use the rounding operation $\mathrm{Int}(x)$ where necessary. These facts will be applied to the method of creating integer wavelets later in this paper.

It should also be noticed that since (2.1)-(2.2) are not linear, because of the rounding operation, the transformation order becomes significant. For instance, if the decomposition was applied first to the columns and then to the rows of an image, the inverse transformation must first be applied to the rows and then to the columns.

Example 2: *Lazy wavelet transform.* This transform does not really do anything. However, it illustrates an important concept which can be seen in next section. The corresponding forward transform is simply dividing the original signal into two separate parts. Decomposition and reconstruction can use the same formula as follows:

$$\begin{aligned} c^1_k &= c^0_{2k}, \\ d^1_k &= c^0_{2k+1}. \end{aligned}$$

Neither Example 1 nor 2 are good transforms for image compression. However, much better transforms can be achieved from these two. As suggested at the beginning, we consider them only as a starting point for our method to create the integer, reversible wavelet transform.

We must mention that there is another interesting property in the above two transforms that may not be easily seen, but meets one of the needs we mentioned in Section 1. If the values of the signal or image pixels are represented by a finite number of bits, say one bit or one byte, we can still use the same number of bits to represent the result of the forward transform within the computer itself because of the complementary code and modulo (a fixed number) arithmetic features of computers. While, from the reconstruction algorithm, the computer will get back the exact original signal through the same features. We call this a *Property of Precision Preservation* (*PPP*) for the above wavelets.

Before we give any further proof about the wavelets in this section having the *Property of Precision Preservation*, we would like to explain a little about the complementary code and modulo arithmetic of the computer. Consider the computation of the difference of two integers given as $c = b - a$ and its inverse computation of $a = b - c$. Suppose that the same number of bits, say q bits, be used to represent all of the variables, which means the values of a, b and c have to be within the range $[-2^{q-1}, 2^{q-1} - 1]$. If q equals to certain number such as 1, 8, 16 and so on, then the calculation of a and c inside a computer can be specified as follows:

$$c_m = \begin{cases} b - a & \text{for } -2^{q-1} \leq b - a < 2^{q-1} \\ b - a - 2^q & \text{for } b - a \geq 2^{q-1} \\ 2^q + b - a & \text{for } b - a < -2^{q-1} \end{cases} \tag{2.3}$$

and

$$a_m = \begin{cases} b - c_m & \text{for } -2^{q-1} \leq b - c_m < 2^{q-1} \\ b - c_m - 2^q & \text{for } b - c_m \geq 2^{q-1} \\ 2^q + b - c_m & \text{for } b - c_m < -2^{q-1} \end{cases} \tag{2.4}$$

Here the subscript m indicates the internal representation. If the real mathematical value of c is outside the range $[-2^{q-1}, 2^{q-1} - 1]$, its internal representation of c_m appears as the number c modulo 2^q. Therefore, the representation may not be the same as the external representation of c. However, the same complementary code and modulo arithmetic for a_m will cause the internal representation to be identical to the external representation for a. For example, if $b = 2$ (00000010) and $a = -127$ (10000001), c_m has the internal binary value of (10000001) when $q = 8$ with a value of -127, which is different from the external representation of $c = 129$. However, the internal inverse value for $a_m = b - c_m$ will just be the same as a.

With above observation, we can easily show the integer wavelets in Example 1 and 2 have the *PPP* feature. In fact, for Example 2, this property is obviously true. Here we only have to prove the integer wavelet in Example 1 has the same property. Suppose the range of the pixel values is within the interval $[-2^{q-1}, 2^{q-1}]$. We still assign q bits to represent the result of the transformation, which means the value of transform coefficients will also be within the same interval. Due to the nature of complementary code and modulo computation on a machine, (2.1)-(2.2) will be automatically implemented within the machine as follows:

$$d_k^1 = \begin{cases} c_{2k}^0 - c_{2k+1}^0 & \text{for } -2^{q-1} \leq c_{2k}^0 - c_{2k+1}^0 < 2^{q-1} \\ c_{2k}^0 - c_{2k+1}^0 - 2^q & \text{for } c_{2k}^0 - c_{2k+1}^0 \geq 2^{q-1} \\ c_{2k}^0 - c_{2k+1}^0 + 2^q & \text{for } c_{2k}^0 - c_{2k+1}^0 < -2^{q-1} \end{cases} \tag{2.5}$$

$$c_k^1 = \begin{cases} \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 & \text{for } -2^{q-1} \le \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 < 2^{q-1} \\ \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 - 2^q & \text{for } \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 \ge 2^{q-1} \\ \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 + 2^q & \text{for } \mathrm{Int}(\frac{d_k^1}{2}) + c_{2k+1}^0 < -2^{q-1} \end{cases} \tag{2.6}$$

While, the computer itself will implement the reconstruction algorithm (2.2) as:

$$c_{2k+1}^0 = \begin{cases} c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) & \text{for } -2^{q-1} \le c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) < 2^{q-1} \\ c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) - 2^q & \text{for } c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) \ge 2^{q-1} \\ c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) + 2^q & \text{for } c_k^1 - \mathrm{Int}(\frac{d_k^1}{2}) < -2^{q-1} \end{cases} \tag{2.7}$$

$$c_{2k}^0 = \begin{cases} d_k^1 + c_{2k+1}^0 & \text{for } -2^{q-1} \le d_k^1 + c_{2k+1}^0 < 2^{q-1} \\ d_k^1 + c_{2k+1}^0 - 2^q & \text{for } d_k^1 + c_{2k+1}^0 \ge 2^{q-1} \\ d_k^1 + c_{2k+1}^0 + 2^q & \text{for } d_k^1 + c_{2k+1}^0 < -2^{q-1} \end{cases} \tag{2.8}$$

It is not difficult to check that (2.7)-(2.8) are just the mathematical inverse transform of (2.5)-(2.6). This means, if we properly take advantage of the bound in the coefficient size mentioned above, the algorithm can be implemented using a minimal amount of storage for lossless compression.

It is important to indicate that *PPP* is not applicable for lossy compression.

§3. More Examples and Additional Analysis

In this section we provide another two examples which will show the motivation for our general approach. We are only going to describe the decomposition algorithm for the following examples. The reconstruction is identical to the decomposition, except it is now running "backwards".

Example 3: *A* $(2,6)$ *integer wavelet.* This transformation is similar to the one used in [2]. The corresponding analysis filters are listed below:

n	**−2**	**−1**	**0**	**1**	**2**	**3**
$\tilde{h}_n$	0	0	1/2	1/2	0	0
$\tilde{g}_n$	−1/16	−1/16	1/2	−1/2	1/16	1/16

and we choose the scaling factors 1 for the low-pass part and 2 for the high-pass part.

The integer decomposition by using above filters starts with *modified Haar wavelet* at step (1), and then upgrades the high frequency components at step (2):

(1) Modified Haar wavelet transform

$$\begin{cases} d_k^{1,0} = c_{2k}^0 - c_{2k+1}^0 \\ c_k^1 = c_{2k+1}^0 + \mathrm{Int}(\dfrac{d_k^1}{2}) \end{cases}$$

(2) Update the high-frequency components

$$d_k^1 = d_k^{1,0} - \mathrm{Int}(\frac{c_{k-1}^1 - c_{k+1}^1}{4}).$$

Example 4: *A* $(5,3)$ *integer wavelet.* This transformation is also similar in function to using the biorthogonal analysis filters. It is given by

n	**-2**	**-1**	**0**	**1**	**2**
$\tilde{h}_n$	-1/8	1/4	3/4	1/4	-1/8
$\tilde{g}_n$	0	0	-1/4	1/2	-1/4

The scaling factors in this example are the same as Example 3.

The integer decomposition related to the above filters starts with the *Lazy wavelet* at step (1), and updates the high-frequency and low-frequency components, respectively, at step (2) and (3):

(1) Lazy wavelet

$$c_k^{1,0} = c_{2\,k}^0$$
$$d_k^{1,0} = c_{2\,k+1}^0$$

(2) Update the high-frequency components

$$d_k^1 = d_k^{1,0} - \mathrm{Int}(\frac{c_k^{1,0} + c_{k+1}^{1,0}}{2})$$

(3) Update the low-frequency components

$$c_k^1 = c_k^{1,0} + \mathrm{Int}(\frac{d_{k-1}^1 + d_k^1}{4})$$

The *PPP* feature mentioned at the end of the previous section is also applicable for these two examples. The detailed explanation can be found in section 7. It is obvious that these transformations are not truly linear, but they are similar to the one using the corresponding filters given above. Particularly, the filters in the above examples belong to, with minor modification, the group of the best biorthogonal filters for image compression according to both our experience and the conclusion of [4].

From the above two examples and the examples in the previous section, we observed the following:

1. For some given wavelet filters, we can begin with an existing integer (either linear or nonlinear) transformation, such as *Haar* or *Lazy* wavelets, and then use some proper upgrading formulas to obtain an expected integer wavelet transformation.

2. If we look carefully at the examples given previously, we will see that the integer wavelet transform can also be generated in following manner: first, divide the original signal into two separate parts, called "low-pass" and "high-pass" parts, by a simple method such as down-sampling or simple combination. Then, update every element in each part according to the information obtained previously.

Now, the key problem is: What kind of deductive formulas should be used in the updating steps? We provide an answer to this question in the following two sections, Section 4 and Section 5.

§4. Lifting Scheme and Integer Biorthogonal Filtering

In this section, we will discuss how to generate an integer reversible scheme for a pair of given biorthogonal filters with the aid of the *lifting* approach. The Lifting scheme, discovered by Sweldens [1], is a new approach for constructing biorthogonal wavelets with compact support. However, the most interesting part of this method for us is: it can be used, with minor modification, to create integer wavelets which are "similar" to the given biorthogonal wavelets with compact support. Here, "similar" means the decomposition results from the integer wavelet are the same as those of the corresponding biorthogonal wavelet, except the rounding error and some scaling factors. We will describe it below.

The following lemma is the main result of the lifting scheme [1] reported as corollary 6 in that paper.

Lemma 4.1. *Take an initial set of finite biorthogonal filters* $\{h, \tilde{h}^0, g^0, \tilde{g}\}$, *then a new set of finite biorthogonal filters* $\{h, \tilde{h}, g, \tilde{g}\}$ *can be found as*

$$\begin{aligned} \tilde{h}(\omega) &= \tilde{h}^0(\omega) + \tilde{g}(\omega)\overline{s(2\omega)} \\ g(\omega) &= g^0(\omega) - h(\omega)s(2\omega) \end{aligned} \tag{4.1a}$$

Similarly, if we take as an initial set of biorthogonal filters $\{h^0, \tilde{h}, g, \tilde{g}^0\}$, *a new set of finite biorthogonal filters* $\{h, \tilde{h}, g, \tilde{g}\}$ *can be found as*

$$\begin{aligned} h(\omega) &= h^0(\omega) + g(\omega)\overline{\tilde{s}(2\omega)} \\ \tilde{g}(\omega) &= \tilde{g}^0(\omega) - \tilde{h}(\omega)\tilde{s}(2\omega) \end{aligned} \tag{4.1b}$$

Here, $s(\omega)$ *is a trigonometric polynomial and the corresponding filter* s *is finite, and so is* $\tilde{s}(\omega)$. *Equations (4.1a) and (4.1b) are called "Lifting" and "Dual Lifting", respectively.*

Actually, regarding the filters, (4.1) is equivalent to

$$\begin{aligned} \tilde{h}_k &= \tilde{h}^0_k + \sum_l s_l \tilde{g}_{k+2l} \\ g_k &= g^0_k - \sum_l s_l h_{k+2l} \end{aligned} \tag{4.2a}$$

or

$$\begin{aligned} h_k &= h^0_k + \sum_l \tilde{s}_l g_{k+2l} \\ \tilde{g}_k &= \tilde{g}^0_k - \sum_l \tilde{s}_l \tilde{h}_{k+2l} \end{aligned} \tag{4.2b}$$

Next, we modify the lifting scheme a little to create a corresponding integer-biorthogonal wavelet. Suppose $\{c^0_k\}$ is an original signal, $\{c^1_k\}$ and $\{d^1_k\}$ are again its low and high frequency decomposition parts, obtained by

using the filters $\{h,\ \tilde{h},\ g,\ \tilde{g}\}$. If we use filters $\{\tilde{h},\ \tilde{g}\}$ for the purpose of analysis, the corresponding decomposition algorithm is

$$\begin{cases} c_k^1 = \alpha_c \sum\limits_n c_n^0 \tilde{h}_{n-2k}, \\ d_k^1 = \alpha_d \sum\limits_n c_n^0 \tilde{g}_{n-2k}; \end{cases}$$

and the reconstruction algorithm will be

$$c_n^0 = 2 \sum_k \left(\frac{c_k^1 h_{n-2k}}{\alpha_c} + \frac{d_k^1 g_{n-2k}}{\alpha_d}\right),$$

which is related to the synthesis filter $\{h,\ g\}$. Here, the scaling factors α_c and α_d are positive constants with $\alpha_c \cdot \alpha_d = 2$. For example, in the situation of normalized biorthogonal decomposition and reconstruction, $\alpha_c = \alpha_d = \sqrt{2}$; and for the integer wavelets given by *Example* 1 through *Example* 4 above, $\alpha_c = 1$ and $\alpha_d = 2$.

If the set of filters $\{h,\ \tilde{h},\ g,\ \tilde{g}\}$ is from (4.2a), then the decomposition can also be accomplished as follows:

1. Decomposition by analysis filter set $\{\tilde{h}^0,\ \tilde{g}\}$

$$\begin{cases} c_k^{1,0} = \alpha_c \sum\limits_n c_n^0 \tilde{h}_{n-2k}^0 \\ d_k^1 = \alpha_d \sum\limits_n c_n^0 \tilde{g}_{n-2k} \end{cases} \tag{4.3}$$

2. Update the low-frequency part $\{c_k^{1,0}\}$

$$c_k^1 = c_k^{1,0} + \frac{\alpha_c}{\alpha_d} \sum_l d_{k-l}^1 s_l \tag{4.4}$$

The relative reconstruction scheme will be:

1. Calculate

$$c_k^{1,0} = c_k^1 - \frac{\alpha_c}{\alpha_d} \sum_l d_{k-l}^1 s_l \tag{4.5}$$

2. Calculate

$$c_n^0 = 2 \sum_k \left(\frac{c_k^{1,0} h_{n-2k}}{\alpha_c} + \frac{d_k^1 g_{n-2k}}{\alpha_d}\right) \tag{4.6}$$

Here, equation (4.3) and (4.6) are just the forward and inverse wavelet transforms by using biorthogonal filters $\{h,\ \tilde{h}^0,\ g^0,\ \tilde{g}\}$. While (4.4) and (4.5) are forward and backward upgrading formulas. For the sake of clarity, we haven't considered the boundary situation, but we will address this later.

Corollary 4.1. *Suppose biorthogonal filters* $\{h,\ \tilde{h},\ g,\ \tilde{g}\}$ *are from initial filters* $\{h,\ \tilde{h}^0,\ g^0,\ \tilde{g}\}$ *or*$(\{h^0,\ \tilde{h},\ g,\ \tilde{g}^0\})$ *by the lifting scheme (4.1a) (or (4.2b)). If the decomposition and reconstruction that are related to filters* $\{h,\ \tilde{h}^0,\ g^0,\ \tilde{g}\}$ *can be accomplished only by integer calculation, such as Example 1 and Example 2, we also can create a corresponding integer wavelet decomposition and reconstruction scheme, which is very "close" to the original transformations (4.3)-(4.6) by using filters* $\{h,\ \tilde{h},\ g,\ \tilde{g}\}$. *Here the word "close" means that the difference of the two decomposition schemes is just some rounding error, and this rounding error will be recovered by the integer perfect reconstruction scheme.*

In fact, $\{c_k^{1,0}\}$ and $\{d_k^1\}$ are integer after (4.3) according to the hypothesis of the Corollary. We can update the low-pass part $\{c_k^1\}$ (*lifting*) by

$$c_k^1 = c_k^{1,0} + \mathrm{Int}\left(\frac{\alpha_c}{\alpha_d}\sum_l s_l d_{k-l}^1\right) \tag{4.7}$$

instead of (4.4). Here $\mathrm{Int}(x)$, as described in Section 2, is an arbitrary rounding up function, which satisfies $x-1 \le \mathrm{Int}(x) \le x+1$. It is obvious that (4.7) is very "close" to (4.4), and the exact reconstruction scheme can easily be obtained from

$$c_k^{1,0} = c_k^1 - \mathrm{Int}\left(\frac{\alpha_c}{\alpha_d}\sum_l d_{k-l}^1 s_l\right) \tag{4.8}$$

and (4.6).

There will be a similar result, if the set of biorthogonal filters $\{h,\ \tilde{h},\ g,\ \tilde{g}\}$ is obtained from the initial set of filters $\{h^0,\ \tilde{h},\ g,\ \tilde{g}^0\}$ by using (4.2b). In this case, the formula for updating (*dual lifting*) the high-pass components will be:

$$d_k^1 = d_k^{1,0} + \mathrm{Int}\left(\frac{\alpha_d}{\alpha_c}\sum_l \tilde{s}_l c_{k+l}^1\right) \tag{4.9}$$

Also, if the filters $\{h,\ \tilde{h},\ g,\ \tilde{g}\}$ is obtained from a set of filters $\{h^0,\ \tilde{h},\ g,\ \tilde{g}^0\}$ by the lifting scheme, and the set $\{h^0,\ \tilde{h},\ g,\ \tilde{g}^0\}$ is again obtained from a filter set $\{h^0,\ \tilde{h}^0,\ g^0,\ \tilde{g}^0\}$, we can repeatedly use Corollary 4.1 to get a "close" integer wavelet transformation, as we have done for *Example* 4.

We now refer to a paper by Daubechies-Sweldens [10]. According to their conclusion, any biorthogonal wavelet with finite support can be obtained with a finite number of lifting and dual lifting steps and a scaling starting from the *Lazy wavelet* in Example 2. In other words, for a given biorthogonal analysis filters $\{\tilde{h},\ \tilde{g}\}$, we can get a series of lifting factors s_l and dual lifting factors $\tilde{s}_l$ with the aid of the Euclidean algorithm described in their paper, which allows us to get the given filters, starting with the filters given in Example 2, by using (4.2a) or (4.2b) repeatedly. This fact also means that we can generate the corresponding integer biorthogonal wavelet by the method developed in this section.

§5. The Correction Method for Integer Wavelets

The last section solved the problem of how to build an integer wavelet which is "close" to a given biorthogonal wavelet. Look at the updating formula (4.9) (or (4.7)). We only use the previous low-pass (high-pass) information to modify the current high-pass (low-pass) coefficient d_k^1 (c_k^1). However, if we also use the high-pass (low-pass) information $d_{k+j}^{1,0}$ $(j>0)$ ($c_{k+j}^{1,0}$ $(j>0)$) obtained in the previous step, or even further use the information from d_j^1 $(j<k)$ (c_j^1 $(j<k)$) which is calculated before d_k^1 (c_k^1) in the current step, we will have more freedom to generate the integer wavelet which may satisfy some conditions we are expecting. In this section, we will describe another approach for obtaining integer wavelets by using the so-called *Correction method*. The motivation for this method is from the S+P transform, and we will now generalize this approach. Actually, the lifting scheme for generating biorthogonal wavelets can be considered as a special case of the *correction method*. From this method we can even get some complicated filters with fast decomposition and reconstruction abilities.

Suppose that we already have a simple integer wavelet transform, such as Examples 1 through 4, and the decomposition and reconstruction scheme can be formulated as follows:

Decomposition

$$\begin{cases} c_k^{1,0} = df_c\left(\{c_n^0\}\right) \\ d_k^{1,0} = df_d\left(\{c_n^0\}\right) \end{cases} \tag{5.1}$$

Reconstruction

$$c_n^0 = rf\left(\{c_k^{1,0}\}, \{d_k^{1,0}\}\right) \tag{5.2}$$

Here, (5.1) and (5.2) can be the same as (4.3) and (4.6) or other algorithms.

In general, after the above decomposition, one may not be satisfied with the result. There may still be some correlation among the high-pass components because of the aliasing from the low-pass components, or the low-pass components do not carry enough of the expected information from the original signal. Hence, we could make an improvement by putting some correction part on the high-pass components or low-pass components. There are many ways to accomplish this. However, for the sake of the integer calculation, we prefer to use the following correction method. For example, if we want to make a correction for the high-pass part, the corresponding formula would be:

$$d_k^1 = d_k^{1,0} - \mathrm{Int}(\delta d_k^1). \tag{5.3}$$

Here, δd_k^1 is a correction quantity for d_k^1

$$\delta d_k^1 = \sum_{i=S_0}^{S_1} \sigma_i c_{k+i}^{1,0} + \sum_{j=1}^{T} \tau_j d_{k+j}^{1,0} + \sum_{l=1}^{R} \beta_j d_{k-l}^1 \tag{5.4}$$
$$k = \cdots, m, m+1, m+2, \cdots$$

and, $\{\sigma_i\}_{i=S_0}^{S_1}$, $\{\tau_j\}_{j=1}^{T}$ and $\{\beta_l\}_{l=1}^{R}$ are parameters which have been chosen for the user's purpose, such as reducing the redundancy among high-pass components, increasing the vanishing moments of analysis (or synthetic) high-pass filter, or some other special requirement. We are not going to discuss how to choose these parameters, but one can refer to the references [3, 5, 6] for clarification of this process. In general, we recommend choosing $\beta_j = 0$ since it makes the choice of the parameters less complicated. We need to mention that, for the sake of the integer calculation, it is better to choose every entry in the sets $\{\sigma_i\}_{i=S_0}^{S_1}$, $\{\tau_j\}_{j=1}^{T}$ and β_l to be rational numbers with denominators being power of 2.

Being aware of the calculation order, it follows from (5.1), (5.3) and (5.4) that the perfect reconstruction algorithm can be

$$d_k^{1,0} = d_k^1 + \text{Int}\left(\delta d_k^1\right), \ k = \cdots, m+2, m+1, m, \cdots \tag{5.5}$$

combined with (5.2).

As mentioned above, the *lifting scheme* is a special case of the *correction method*. Examples 3 and 4 can also be considered as the illustrations of this method. *S+P wavelet* [3], which does not result in a closed form of compact supported biorthogonal filters, is a typical example of *correction method*. In that case, the decomposition starts with the *modified Haar wavelet*, then it updates the high-pass components by (5.4), where the parameters are as follows: $S_0 = -1$, $S_1 = 1$, $T = 1$, and

$$\sigma_{-1} = -\frac{1}{4},\ \sigma_0 = -\frac{1}{8},\ \sigma_1 = \frac{3}{8},\ \tau_1 = \frac{1}{4}.$$

We next give another example to show the flexibility of the *correction method*.

Example 5. In this example, the first two steps of the decomposition are the same as those in *Example 4*:

Step 1. Lazy wavelet

$$c_k^{1,0} = c_{2\,k}^0$$
$$d_k^{1,0} = c_{2\,k+1}^0$$

Step 2. Update the high-frequency components

$$d_k^1 := \text{Int}(\frac{c_k^{1,0} + c_{k+1}^{1,0}}{2}) - d_k^1$$

In the third step to update the low-pass components, we use following correction formula:

Step 3. Update the low-pass part by formula

$$c_k^1 = c_k^{1,0} + \text{Int}(\delta c_k^1),$$

where

$$\delta c_k^1 = \sigma_{-1} d_{k-1}^1 + \sigma_0 d_k^1 + \tau_1 c_{k+1}^{1,0} + \tau_2 c_{k+2}^{1,0}.$$

Case 1. *If* $\sigma_{-1} = \sigma_0 = \frac{-1}{4}$ *and* $\tau_1 = \tau_2 = 0$, *the result is just Example 4;*

Case 2. *If* $\sigma_{-1} = 0$, $\sigma_0 = -\frac{1}{2}$ *and* $\tau_1 = \tau_2 = 0$, *we get a non-symmetric biorthogonal wavelet, in which both analysis low-pass and high-pass filters have only three non-zero elements. This wavelet has almost the same computing complexity as Haar wavelet, but is better for image compression. Without truncation, its analysis high-pass filter has two vanishing moments, and the synthetic high-pass filter has one vanishing moment. The corresponding analysis filters are list below:*

n	**-1**	**0**	**1**
$\tilde{h}_n$	*3/4*	*1/2*	*-1/4*
$\tilde{g}_n$	*1/4*	*-1/2*	*1/4*

Case 3. *If we expect higher vanishing moments for the synthetic high-pass filter without truncation, we can choose*

$$\sigma_{-1} = -\frac{3}{16},\ \sigma_0 = -\frac{5}{16},\ \tau_1 = -\frac{1}{8},\ \tau_2 = \frac{1}{8};$$

which is the solution to the system of functions

$$\begin{cases} \tau_1 + \tau_2 = 0, \\ 1 + 2\sigma_{-1} + 2\sigma_0 = 0, \\ 1 + 4\sigma_0 + 2\tau_2 = 0. \end{cases}$$

The corresponding analysis filters are

n	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**
$\tilde{h}_n$	*-3/32*	*3/16*	*3/4*	*5/16*	*-9/32*	*0*	*1/8*
$\tilde{g}_n$	*0*	*0*	*1/4*	*-1/2*	*1/4*	*0*	*0*

§6. Boundary Conditions

In the previous two sections, we did not show how to process the boundaries of the signal during the creation of the integer wavelet. We will consider it in this section. Assume that $\{c_n^0\}_{n=0}^{N-1}$ is the original signal. Also, $\{c_n^1\}_{n=0}^{N_1-1}$ and $\{d_n^1\}_{n=0}^{M_1-1}$ are its decomposition parts at the first level. Here

$$N_1 = \begin{cases} \frac{N}{2} & \text{for } N \text{ is an even number} \\ \frac{N+1}{2} & \text{for } N \text{ is an odd number} \end{cases}$$

and

$$M - 1 = N - N_1.$$

There are two issues dealing with boundary filtering when we use the *Lifting scheme* or the *Correction method* to generate the integer wavelets. The first

is how to process the boundaries that occur in the start-up wavelet transformation if there is any. The second is how to deal with the boundaries in the deductive formula. If the boundaries in the start-up wavelet transform have already been established, then those in the upgrading formula are easy to establish. In fact, for the *Lifting scheme* of the biorthogonal filters, the boundaries in both steps should be processed in the same way. While, for the *Correction method*, it is easy to see from (5.3)-(5.4) that one has more choices to process boundaries in the second step. Therefore, the only thing we need to discuss here is the process by which the boundaries in the start up wavelet transformations are established. Assume we begin with compact supported biorthogonal wavelets. We prefer to take following symmetric extension, which allows the boundary filtering scheme keeps the same as the one for internal [7]:

(1). If current biorthogonal filters have even length, we extend the boundaries of the signal as

$$\begin{cases} c^0_{-k} = c^0_{k-1}, & k = 1, 2, \cdots \\ c^0_{N+k} = c^0_{N-k-1}, & k = 0, 1, \cdots. \end{cases}$$

(2). If the filters have odd length, we do the extension as

$$\begin{cases} c^0_{-k} = c^0_k, & k = 1, 2 \cdots \\ c^0_{N+k} = c^0_{N-k-2}, & k = 0, 1, \cdots \end{cases}$$

For example, if we apply the forward wavelet transform of the Example 3, to a signal $\{c^0_n\}_{n=0}^{N-1}$, we can still use the same formula for both the internal and boundary conditions of the signal. With the symmetric extension above in the boundary, the specific decomposition algorithm will be:

Step 1: Compute the high-pass components

$$d^{1,0}_k = c^0_{2k} - c^0_{2k+1},\ k = 0, 1, \cdots, M_1 - 1$$

Step 2: Compute the low-pass components

$$c^1_k = \text{Int}\left(\frac{d^{1,0}_k}{2}\right) + c^0_{2k+1}\ k = 0, 1, \cdots, N_1 - 2$$

$$c^1_{N_1-1} = \begin{cases} \text{Int}\left(\frac{d^{1,0}_{M_1-1}}{2}\right) + c^0_{N-1} & \text{for } N \text{ is an even number} \\ c^0_{N-1} & \text{for } N \text{ is an odd number} \end{cases}$$

Step 3: Update the high-pass components

$$\begin{cases} d^1_0 = d^{1,0}_0 - \text{Int}\left(\frac{c^1_0 - c^1_1}{4}\right) \\ d^1_k = d^{1,0}_k - \text{Int}\left(\frac{c^1_{k-1} - c^1_{k+1}}{4}\right),\ k = 0, 1, \cdots, M_1 - 2, \end{cases}$$

and then,

$$d^1_{M_1-1} = \begin{cases} d^{1,0}_{M_1-1} - \text{Int}\left(c^1_{N_1-2} - c^1_{N_1-1}\right), & \text{for } N \text{ is even;} \\ d^{1,0}_{M_1-1} - \text{Int}\left(c^1_{N_1-3} - c^1_{N_1-1}\right), & \text{for } N \text{ is odd.} \end{cases}$$

Obviously, if we make the same symmetric extension to the sequence $\{c^1_k\}_{k=0}^{N_1-1}$ and use the same algorithm in Step 3 of the *Example* 3 for both the internal and boundary conditions of the signal, the result would be the same as the above.

As for the reconstruction, it just follows the above in reverse.

§7. Some Applications and discussion

The above sections have shown the processes necessary to obtain an integer reversible wavelet transform suitable for signal or image processing, particularly lossless compression. Such a transform can be obtained either using the Lifting method, or the Correction method. If we have a pair of given biorthogonal wavelet filters, and try to get an integer version without losing any beneficial features of the filters, we suggest using the *lifting* technique. For example, all interpolation wavelets [1] can be modified to be corresponding integer wavelets without loosing any properties of the original wavelets. On the other hand, if we want to create some new integer transforms which have to match some expected features, then the *correction method* is a good choice.

Before talking about any applications of the integer wavelet transform described above, we first prove a useful property of precision preservation *(PPP)* which holds for both the *Lifting* and *Correction* upgrading technique. This is the same property as mentioned at the end of Section 2. This fact implies that the precision of the transform coefficients can remain identical to the precision of the data, thus reducing the need for additional computer memory during the transform computation. This is an extremely powerful technique when the target data are large images, or the requirements establish a need for speed.

Lemma 7.1. *Suppose that the integer wavelet transform starts with a pair of biorthogonal filters with the PPP feature discussed in Section 2, that is, the forward/inverse transform (4.3)/(4.6) possesses this feature. Then, the same property will be preserved in the whole algorithm if we adopt the Lifting scheme to be the upgrading formula.*

In other words, Lemma 7.1 states if we only use the working units with the same precision as the original signal or image to calculate the wavelet transform developed in Section 4, the equations (4.8) and (4.6) are still the backward operations of the equations (4.3) and (4.7).

Proof. Assume that every entry of the original images or signals is represented by q bits, that is, the range of the pixel values is within the interval $[-2^{q-1}, 2^{q-1} - 1]$. According to the hypothesis of the lemma, the equation

(4.3) and its inverse (4.6) have the *PPP* feature. Therefore, what we have to verify here is that the equation (4.7) and its inverse (4.8) can preserve the same feature. We rewrite (4.7) and (4.8) as follows:

$$c_k^1 = c_k^{1,0} + b_k \tag{7.1}$$

and the inverse

$$c_k^{1,0} = c_k^1 - b_k \tag{7.2}$$

Here,

$$b_k = \text{Int}\left(\frac{\alpha_c}{\alpha_d}\sum_l s_l d_{k-l}^1\right).$$

In fact, the quantity b_k would have the same value in both (4.7) and (4.8) if we calculate it in the same way. On the other hand, if the working unit for b_k is q bits, the machine might give b_k another value, say $\tilde{b}_k$ ($-2^{q-1} \le \tilde{b}_k < 2^{q-1}$). $\tilde{b}_k$ is not equal to b_k in the sense of mathematics if the value of b_k is beyond the interval $[-2^{q-1}, 2^{q-1}-1]$. However, $\tilde{b}_k$ should be the same in both (4.7) and (4.8). Therefore, the machine will automatically implement (7.1) and (7.2) as

$$c_k^1 = \begin{cases} c_k^{1,0} + \tilde{b}_k & \text{for } -2^{q-1} \le c_k^{1,0} + \tilde{b}_k < 2^{q-1} \\ c_k^{1,0} + \tilde{b}_k - 2^q & \text{for } c^{1,0} + \tilde{b}_k \ge 2^{q-1} \\ c_k^{1,0} + \tilde{b}_k + 2^q & \text{for } c_k^{1,0} + \tilde{b}_k < -2^{q-1} \end{cases} \tag{7.1m}$$

and

$$c_k^{1,0} = \begin{cases} c_k^1 - \tilde{b}_k & \text{for } -2^{q-1} \le c_k^1 + \tilde{b}_k < 2^{q-1} \\ c_k^1 - \tilde{b}_k - 2^q & \text{for } c_k^1 - \tilde{b}_k \ge 2^{q-1} \\ c_k^1 - \tilde{b}_k + 2^q & \text{for } c_k^1 \tilde{b}_k < -2^{q-1} \end{cases} \tag{7.2m}$$

It is easy to see that (7.2m) is just the backward operation of (7.1m), which provides the evidence that the conclusion of this lemma is correct.

A similar argument can show the same *PPP* feature will hold for the integer wavelet transforms generated by the *Correction method* in Section 5.

As we have seen above, when we take advantage of the *PPP* feature of the wavelets developed in this paper, the transform (we call it *PPP transform*) is different from the transform given mathematically by (4.3) and (4.7) (we call it *RI transform*) because of the different computation manners. Both of the transforms can be used in lossless compression due to their reversible ability. However, only the latter, which does not limit the bit depth to represent the coefficients–without using the *PPP* feature, can be used wherever ordinary wavelets are used, especially in lossy image compression, since it is similar to the corresponding (biorthogonal) wavelet except for the scaling factors and the rounding error. The advantages of using integer wavelets are obvious: perfect reconstruction without any rounding errors, faster calculation and less memory usage than floating calculations.

Following are some applications by utilizing these types of transforms.

Application 1. *Lossless image compression*

As pointed out at the beginning of this paper, the integer wavelet transformation established by the techniques described in this paper can always be used for lossless image compression because of the reversible ability. Especially, the *PPP* feature discussed in Lemma 7.1 can be utilized when necessary, which allows us to save computer resources during the computation. However, some of the coefficients obtained from the *PPP transform* will have different output values from the corresponding coefficients of the *RI transform*, although the transforms have the same filters. We have to evaluate the effect on the compression efficiency. In other word, we have to consider the coefficient entropy after transform if we take advantage of *PPP*. The entropy gives the measurement of efficiency for lossless compression. As we already know, the entropy of a sequence $\{c_n\}_{n=0}^{N-1}$ is given as

$$H(C) = -\sum_j P(\xi_j) \log P(\xi_j),$$

where

$$P(\xi_j) = \frac{1}{N}(|n : c_n = \xi_j|).$$

We are going to indicate that for a pair of proper analysis filters, *PPP transform* and *RI transform* are of nearly equal efficiency on lossless compression, which means the entropies after these two transforms are almost the same.

Image	**Ex.3 (R)**	**Ex.3 (P)**	**Ex.5 (R)**	**Ex.5 (P)**
air1_g	5.6843	5.6823	5.6145	5.6116
camera	4.8890	4.8811	4.7684	4.7694
cmpnd1	2.8279	2.8279	2.7559	2.7435
cmpnd2	3.0335	3.0032	2.8521	2.8143
ct	1.8707	1.8707	1.7819	1.8766
cr	3.3255	3.3236	3.2616	3.2666
faxballs	1.1965	1.1965	1.1345	1.1345
finger	6.7989	6.7916	6.7024	6.7020
goldy	4.8032	4.8032	4.6829	4.6828
hotely	4.8273	4.8273	4.7113	4.7097
lenna	4.1962	4.1963	4.1292	4.1291
mountain	6.8121	6.7009	6.7918	6.7817
mri	3.3328	3.3328	3.2051	3.4085
peppers	4.6376	4.6342	4.5817	4.6115
us	3.8308	3.7716	3.7149	3.7214
x − ray	2.4892	2.4841	2.3510	2.3544
Average	4.0347	4.0215	3.9400	3.9573

Table 1. Entropy Comparison

Table 1 gives the entropy comparison between *RI transform* and *PPP transform*. The images listed in the table come from JPEG test image set and

Waterloo BragZone. We have made some modifications to those images, such as dimensions, which might be a little different from the originals. Column 2 gives the entropies by directly using the wavelet in Example 3 (*RI transform*), and column 3 gives the entropies by using the same wavelet but with *PPP* feature (*PPP transform*). Column 4 and column 5 give the same comparison for the wavelet in Example 4. The way we calculate the entropy after l-level ($l = 5$ in the table) wavelet transform is as follows: Suppose C^l represents the lowest-frequency image after l level transformation. HD^j, VD^j and DD^l ($j = 1, \cdots, l$) represent the high frequency parts related to horizontal, vertical and diagonal details in the jth level. The entropy for Table 1 is from:

$$\frac{1}{4^l}H(C^l) + \sum_{j=1}^{l} \frac{1}{4^j}\left(H(HD^j) + H(VD^j) + H(DD^j)\right).$$

We have following observations. Suppose that the value range of original image is within $[-2^{q-1}, 2^{q-1} - 1]$. Let $\{c^1_{P,j}\}$ and $\{d^1_{P,j}\}$ represent the coefficients from *PPP transform*, and $\{c^1_{R,j}\}$ and $\{d^1_{R,j}\}$ be those from *RI transform*. According to the formula (7.1m), after one level of wavelet transformation, $d^1_{P,j} \neq d^1_{R,j}$ only if $d^1_{R,j} \geq 2^{q-1}$ or $d^1_{R,j} < -2^{q-1}$. In general, for those images with certain smoothness, the coefficients $\{d^1_{R,j}\}$ of high-frequency bands should be distributed around zero. This implies that most coefficients related to the high frequency would keep the same value in both transforms. The entropy of these coefficients would not change a lot. According to our experiments, the entropy calculated in these bands goes down, for most nature images, after only one lever transform. On the other hand, if we properly choose the scaling factors α_c and α_d, for example, to make the summation of all entries in the low-pass filter is equal to or less than 1, as we have done for the examples in this paper, most coefficients in the set $\{c^1_{P,j}\}$ will be the same as those in $\{c^1_{R,j}\}$, and the major continuity of the *RI transform* coefficients $\{c^1_{R,j}\}$ will be kept. Intuitively, the entropies for both transformations won't have big difference. The results given in Table 1 have shown our conclusion is correct.

Application 2. *Large scale medical image compression*

Here, we are talking about lossy compression. As we mentioned before, the *PPP* feature is not applicable to this kind of compression. However, there are still some advantages for using the wavelets described above besides the integer computation. Usually, 12 bits are used to represent one pixel in medical images. In this situation, the value of the pixels vary from 0 to 4095. Such images require careful treatment when a transform coding method is applied for compression. If we use ordinary biorthogonal wavelets, the range of the transform coefficients will expand to $[-2^{16}, 2^{16}]$ when five levels of transform are used. Therefore, a longer working unit has to be assigned, which consumes significant computer resources. However, the integer wavelet technique developed in this paper will solve this problem. For example, if

we use the transforms given in Example 3, 4 and 5, the values of transform coefficients will be limited to the range of $[-2^{13}, 2^{13}]$. Even if we do not use the *PPP* property for these wavelets, 16 bits for the working unit is sufficient for all computations.

Although this paper establishes the structure for creating a wavelet-like integer transform, we do not imply the examples in this paper are necessarily the best wavelets for any particular application. However, we do claim if one is going to use such a technique, the ideas suggested in this paper will provide the best implementation for the effort involved.

Note. After completing this research, we found a similar result to the one described in Section 4 was independently provided by Calderbank -Daubechies - Sweldens -Yeo [11] and Dewitte -Cornelis [12]. In [11], the authors also provide another approach based on the LTF precoder. However, the fixed precision feature in this paper is quite unique and we found it to be extremely useful.

Acknowledgments. This work has been partially supported by the US Navy under SBIR Contract N00039-94-C-0013. The first author has been partially supported by the National Science Foundation of P. R. China and the Science Foundation of Zhongshan University.

References

1. Wim Sweldens, *The lifting scheme: A coustom-design construction of biorthogonal wavelets*, Applied and Computational Harmonic Analysis, Vol. 3, No. 2, April 1996.
2. A. Zandi, J. Allen, E. Schwartz and M. Boliek, *CREW: Compression with recersible embedded wavelets*, in IEEE Data Conference, (Snowbird, Utah) March 1995, 212–221
3. A. Said, *An image multiresolution representation for lossless and lossy imagecompression*, Sumitted to the IEEE Transactions on Image Processing.
4. J. Villasenor, B. Belzer, and J. Liao, *Wavelet filter evaluation for image compression*, IEEE Trans. Image processing, Vol. 4, 1053–1060.
5. G. R. Kuduvalli and R. M. Rangayyan, *Perfomance analysis of reversible image compression techniques for high-resolution digital teleradiology*, IEEE Trans. Medical Imaging, Vol. 11, 430–445, Sept. 1992.
6. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes: the art of science programming*, Cambrige University Press, Cambrige, New York, 1986.
7. G. Strang and Truong Nguyen, *Wavelets and filter banks*, Wellesley-Cambridge Press, 1996.

8. X. Wu, *Lossless compression of continuous-tone images via contex selection, quantization, and modeling*, IEEE Trans. Image Processing, Vol. 6, no. 5, May 1997.
9. M.J. Weinberger, J.J. Rissanen, and R.B. Aprs, *Applications of universal context modeling to lossless compression of gray-scale images*, IEEE trans. Imge Processing,vol. 5, no. 4, Apr. 1996.
10. I. Daubechies and W. Sweldens, *Factoring wavelet transforms into lifting step*, Preprint, (http://cm.bell-labs.com/who/wim/).
11. A. Calderbank, I. Daubechies, W. Sweldens, and Boon-Lock Yeo, *Wavelet transforms that map integers to integers*, Applied and Computational Harmonic Analysis (ACHA), to appear.
12. S. Dewitte and J. Cornelis, *Lossless Integer Wavelet Transform*, IEEE signal processing letters, Vol. XX, No.Y, June 1997.

Hongyang Chao
Dept. of Computer Science
Zhongshan University
P.R.China
AND
CIS Inc.
3401 University, Suite 104
Denton, TX 76208, USA
chao@compsci.com

Paul Fisher AND Zeyi Hua
Computer and Information Sciences Inc.,
3401 University, Suite 104
Denton, TX 76208, USA
fisher@compsci.com, hua@compsci.com