Journal of the Chinese Institute of Engineers
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcie20

A fast convolution algorithm for biorthogonal wavelet image compression

Bing-Fei Wu [a] & Chorng-Yann Su [b]

[a] Department of Electrical and Control Engineering , National Chiao Tung University , Hsinchu, Taiwan 300, R.O.C.

[b] Department of Industrial Education , National Taiwan Normal University , Taipei, Taiwan 106, R.O.C.

Published online: 02 Mar 2011.

To cite this article: Bing-Fei Wu & Chorng-Yann Su (1999) A fast convolution algorithm for biorthogonal wavelet image compression, Journal of the Chinese Institute of Engineers, 22:2, 179-192, DOI: 10.1080/02533839.1999.9670455

To link to this article: http://dx.doi.org/10.1080/02533839.1999.9670455

Journal of the Chinese Institute of Engineers, Vol. 22, No. 2, pp. 179-192 (1999)

A FAST CONVOLUTION ALGORITHM FOR BIORTHOGONAL WAVELET IMAGE COMPRESSION†

Bing-Fei Wu*
Department of Electrical and Control Engineering
National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.

Chorng-Yann Su
Department of Industrial Education
National Taiwan Normal University
Taipei, Taiwan 106, R.O.C.

Key Words: fast convolution algorithm, wavelet image compression, symmetric extension, zerotree coding.

ABSTRACT

Symmetric filters and symmetric extension of image edges have been widely used in wavelet image compression. Since the filters are symmetric, it is possible to take advantage of the symmetric property to reduce the computational complexity for the filtering. In this paper, we present a fast convolution algorithm for the discrete wavelet transform (DWT) and the inverse DWT (IDWT) such that the transform time can be greatly reduced. Compared with regular convolution, the new algorithm can decrease the multiplication operations by nearly one half. Converted into real programming, it sped up the DWT and IDWT in our experiments by at least 12% and 55%, respectively. Incorporated with enhancing zerotree coding, the proposed algorithm results in a rapid and efficient coder. Experimental results showed that the coder is competitive with other high performance coders. The proposed convolution algorithm is also suitable for many types of wavelet-based coding, including wavelet video coding.

I. INTRODUCTION

Transform coding is a favorite technique in image compression. It has become a key component in international video communication standards (Pennebaker and Mitohell, 1993). In the past decade, the discrete cosine transform (DCT) has been the most popular transform because it provides almost optimal performance and can be implemented at an acceptable cost. However, recently the discrete wavelet transform (DWT) has received more attention because it can solve the blocking effect introduced by the DCT and it is particularly well adapted to progressive transmission (Antonini *et al.*, 1992). Furthermore, the pyramid-like multiresolution decomposition by the DWT can produce some degree of self-similarity across different scales, which is helpful in image compression.

Shapiro (1993) has introduced the embedded zerotree wavelet (EZW) algorithm, (Shapiro, 1993) which takes advantage of the features of DWT, and he has shown that the EZW is not only competitive

---

†Invited paper
*Correspondence addressee

in its performance with the most complex techniques, it is also extremely fast in its execution. In addition, the set partitioning in hierarchical trees (SPIHT) (Said and Pearlman, 1996) and the space-frequency quantization (SFQ) algorithms (Xiong *et al.*, 1997), etc., have also shown that DWT-based coders obtain performances much superior to JPEG or to most of the other DCT-based coders. Specifically, SPIHT interrupts the simultaneous progression of efficiency and complexity. It is almost the best coder in the literature no matter if the comparison concerns performance or execution time. Because of the advantages of DWT-based coders, Martucci *et. al.* (1997), have transferred DWT-based techniques from gray level image coding to video compression.

However, many of the high performance DWT-based coders, such as EZW and SPIHT, have to take most of the CPU time to perform the transformation. Thus, reducing the transform time in DWT or inverse DWT (IDWT) is the crucial issue in decreasing the total execution time of these coders. This problem has motivated researchers to develop fast convolution algorithms for DWT IDWT.

Two approaches are frequently used to improve performance in wavelet image compression. One approach selects a pair of biorthogonal linear phase filters, and the other approach uses symmetric extension of the image edges (Said and Pearlman, 1996; Xiong *et al.*, 1997). It is well-known that orthogonal wavelets have the merit of energy preservation. However, the orthogonal wavelets are highly constrained, which allows for little design flexibility. For example, it is desirable that the finite impulse response (FIR) filters used are of the linear phase, since such filters can be easily cascaded in the pyramidal filter structures without phase compensation (Antonini *et al.*, 1992) and the corresponding coder can obtain improved performance. Unfortunately, there are no orthogonal linear phase FIR filters with the exact reconstruction property except for some trivial cases (such as the Haar filter). To preserve the linear phase (corresponding to the symmetry for the wavelet) property, one can relax the orthogonality requirement by using biorthogonal bases. In this case, biorthogonal wavelets are not used for energy preservation. Some biorthogonal filters (such as 9/7-tap (Antonini *et al.*, 1992)) are designed to be "close" to orthogonal, so that the quantizers designed in the orthogonal case can be well approximated. In a two-channel filter bank, there are three forms of biorthogonal linear phase filters that can achieve perfect reconstruction (Vetterli *et al.*, 1995). In the most popular form both filters are symmetric and are of odd lengths in the analysis bank. Since the filters are symmetric, symmetric extension of the image edges can be applied. The advantage of using symmetric extension is that the number of jumps between the transformed coefficients can be reduced. These jumps are probably introduced by the circular extension of the image edges because the gray levels between the first pixel and the last are generally different in the same row or column.

Martucci (1994) used the so-called symmetric convolution to convolve symmetrically extended sequences. To use symmetric convolution to filter a sequence, one should first retrieve the right-half samples of a symmetric filter. Next, apply the appropriate symmetric extensions to the retrieved sequence and to the input sequence, and then obtain the convolution results with the following two methods. One method involves the application of a circular or skew-circular convolution to both of the symmetrically extended sequences, and then a rectangular window is used to extract their representative samples out of the base period. The other method involves taking an inverse discrete trigonometric transform (DTT) from the product of the forward DTTs of these two sequences. In this paper, we will develop a fast convolution algorithm that is similar to the former method, but our algorithm is a much more straightforward approach. The proposed algorithm is mainly applied while the symmetric filter is of odd length and the image edges are symmetrically extended. Because both the filter and the image extension are symmetric, many terms of the multiplied results will be computed repeatedly while filtering an image. Hence, if we can efficiently permute the nonrepeated terms and sum up some of the terms for each different transformed coefficient, then we can reduce the multiplication operations by nearly one half. This is the main idea proposed in this paper. The most attractive feature of our new algorithm is the simplicity of its final form. This approach provides easy implementation without additional computational complexity.

In the following sections, we will address the algorithm for the one dimensional (1-D) case, whose results can be extended to filter higher dimensional sequences by using the 1-D case for each dimension, independently. Section II describes the regular convolution in 1-D DWT and IDWT, which is often used in real programming. Section III and Section IV illustrate the frameworks of the proposed fast convolution algorithms for DWT and IDWT, respectively. The analysis concerning the computational complexity is shown in Section V. Experimental results, including the execution time, are shown in Section VI. To contrast the performance of different extensions of the image edges, we incorporate the proposed algorithm into our previous coder based on an enhancing zerotree coding (Wu and Su, 1998). Some reconstructed images at a bit rate 0.5 bits per pixel (bpp)

and the results of the peak signal to noise ratios (PSNR) at various bit rates are illustrated in this section. The conclusion of this paper is given in Section VII.

II. The 1-D DWT AND IDWT

The two-channel 1-D DWT of a sequence $a[n]$ (see Fig. 1) is defined as

$$A[k]=\sum_n h[2k-n]a[n]$$

$$B[k]=\sum_n g[2k-n]a[n], \qquad (1)$$

where $A[k]$ and $B[k]$ represent the sequences produced from the lowpass and the highpass channels, respectively. Eq. (1) describes a convolution scheme followed by a downsampler with sampling period being 2. With biortho-gonal wavelets, the corresponding IDWT to reconstruct $a[n]$ is given as

$$a[n]=a_0[n]+a_1[n]$$

$$=\sum_k \tilde{h}[2k-n]A[k]+\sum_k \tilde{g}[2k-n]B[k], \qquad (2)$$

where $a_0[n]$ and $a_1[n]$ indicate that the reconstructed sequences are contributed to by the lowpass and highpass channels, respectively. In order to have a perfect reconstruction, we set

$$g[n]=(-1)^n\tilde{h}[1-n],\ \tilde{g}[n]=(-1)^n h[1-n], \text{ and}$$

$$\sum_n h[n]\tilde{h}[n+2m]=\delta[m]. \qquad (3)$$

So far, there is nothing different from usual subband coding schemes. To distinguish the wavelet decomposition from the usual subband coding schemes, the following constraints in the filters are also imposed (Anotnini *et al.*, 1992)

$$\sum_n h[n]=2^{1/2},\ \sum_n \tilde{h}[n]=2^{1/2},$$

$$\sum_n (-1)^n h[n]=0 \text{ and } \sum_n (-1)^n \tilde{h}[n]=0. \qquad (4)$$

In the above equations, all of the filters and the sequence are infinite. However, the finiteness on filters and the sequence have to be considered in the real application. The immediate problem of Eq. (1) is how to compute the product $h[2k-n]a[n]$ if the index in the bracket is out of the defined range. For example, the filter $h[n]$ is assumed to be with only positive index terms defined at 0, ..., $L-1$, and the computation in Eq. (1) may ask for $h[-1]$ which is not defined. One possible solution for solving this finite length issue is to perform the N-point circular

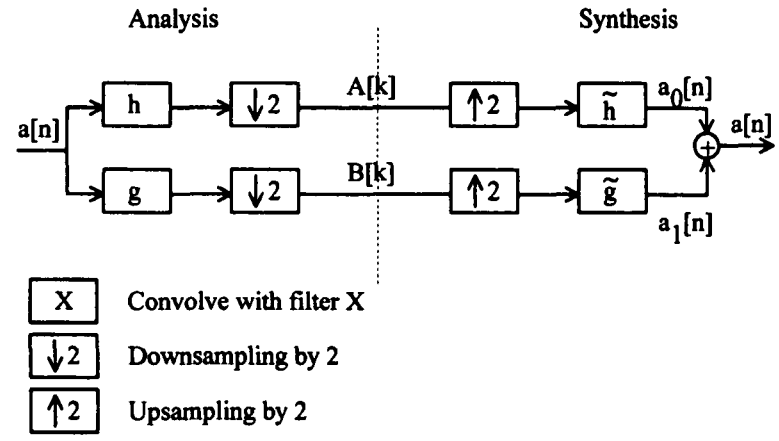


Fig. 1. Two-channel filter bank and the associated wavelets.

convolution (Oppenheim and Schafer, 1989) (assume that the sequence length is N and $N>L$). To do this, the filter $h[n]$ has to be zero-padded. Since the product of a value multiplied with a zero does not contribute anything to the convolution sum, the computation is tedious and a waste of time. A better method is to sum up only the possible nonzero terms. This can be done by computing Eq. (1) based on the filter length. Therefore, we rewrite Eq. Eq. (1) to

$$A[k]=\sum_n h[n]a[2k-n],$$

$$B[k]=\sum_n g[n]a[2k-n]. \qquad (5)$$

This is feasible in the linear and time invariant (LTI) systems. Consequently, the number of multiplication operations (NMOs) in Eq. (5) is L for each $A[k]$ or $B[k]$. It is, generally, far fewer than those required in Eq. (1) (the required NMOs is N for each $A[k]$ or $B[k]$). For the same reason, Eq. (2) is rewritten as

$$a[n]=\sum_k \tilde{h}[k]A[(k+n)/2]+\sum_k \tilde{g}[k]B[(k+n)/2],$$

$$\text{for } k+n=2l,\ l\in \mathbf{Z}. \qquad (6)$$

We refer to this type of convolution based on the filter length as the regular convolution, which is commonly used in real programming.

In Eq. (5), the same problem is how to compute the products when the index of $a[n]$ is out of the defined interval. There are two possible solutions to solve this. One solution is to extend the sequence $a[n]$ by circular extension (Fig. 2(a)), and the other is to extend it by symmetric extension (Fig. 2(b)). Circular extension and symmetric extension for the image edges are shown in Fig. 2(c) and Fig. 2(d), respectively. Circular extension is the most widely used solution in digital signal processing. This extension is suitable for all filters, but introducing possible jumps at the endpoints is its disadvantage because generally $a[N-1]$ does not equal $a[0]$. The jumps introduced into the process will lower the coding performance when using the EZW-like algorithms.

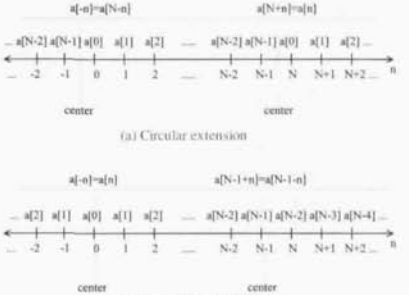


Fig. 2. Circular extension and symmetric extension. (a) circular extension in a sequence (b) symmetric extension in a sequence (c) circular extension of image edges (d) symmetric extension of image edges.

As for symmetric extension, it is only suitable for symmetric or antisymmetric filters. However, it is widely applied to image compression (Anotonini *et al*., 1992; Said and Pearlman, 1996; Shapiro, 1993; Xiong *et al*., 1997) because some symmetric filters exhibit better properties (such as regularity). For example, the favorite one is the 9/7-tap filters of (Anotonini *et al*., 1992). Note that the symmetric extension as shown in Fig. 2(b) is called whole-point symmetry (no repeat of endpoint) (Martucci *et al*., 1997), but another symmetric extension called half-point symmetry (repeat of endpoint) is not shown here.

Because the symmetric extension exhibits better performance, we have developed a fast convolution algorithm focused on the odd-length symmetric filters and the symmetric extension in the sequence ends. For simplicity, we will assume that a pair of biorthogonal wavelet filters are given and defined at

$h[n]$, $-p \le n \le p$;

$\tilde{h}[n]$, $-q \le n \le q$;

The filter lengths of these two filters are $2p+1$ and $2q+1$, respectively. Moreover, the "symmetric center" of both filters is at $n=0$. Without loss of generality, the sequence $a[n]$ is available for the time interval, $n=0, 1, \ldots, N-1$. We will assume N to be the power of two.

III. FAST CONVOLUTION ALGORITHM FOR DWT

Oversampling is undesirable in image compression. Therefore, the essential question is how to choose a method that maintains a constant number of pixels required to describe the image. That is, for an N-points input sequence we have to select exactly N nonredundant points from the lowpass and highpass channels.

Taking the finiteness of summation into account, we rewrite Eq. (5) to become

$$A[k] = \sum_{n=-p}^{p} h[n]a[2k-n],$$

$$B[k] = \sum_{n=-q+1}^{q+1} g[n]a[2k-n], \qquad (7)$$

where $g[n]$ is calculated by Eq. (3). Because the symmetric center of $g[n]$ is at $n=1$ and the filter $h[n]$ is symmetric about $n=0$, we have the following identities

$$h[n]=h[-n] \text{ and } g[1+n]=g[1-n] \qquad (8)$$

Applying the whole-point symmetric extension to both ends of the sequence $a[n]$, we get the following relations:

$$a[-n]=a[n] \text{ and } a[N-1-n]=a[N-1+n]. \quad (9)$$

Then substituting Eqs. (8) and (9) into Eq. (7) yields

$$A[-k]=A[k] \text{ and } A[N/2-1-k]=A[N/2+k] \quad (10)$$

and

$$B[-k]=B[k+1] \text{ and } B[N/2-k]=B[N/2+k]. \quad (11)$$

Rewrite Eq. (10) as a symmetric form

$$A[\alpha_L-k]=A[\alpha_L+k] \text{ and } A[\alpha_R-(k+1/2)]=A[\alpha_R+(k+1/2)], \quad (12)$$

where the left symmetric center $a_L=0$ and the right symmetric center $a_R=(N-1)/2$. Consequently, the sequence $A[k]$ exhibits a whole-point symmetry on the left side but a half-point symmetry on the right side. Because of the symmetry of the sequence $A[k]$, we only need to store the set of $N/2$-points, i.e. $A[0]$, $A[1]$, ..., $A[N/2-1]$, and then use the relations in Eq. (10) to reproduce the remains. Similarly, there are two symmetric centers in the sequence $B[k]$ as well. Rewrite Eq. (11) into a symmetric form as

$$B[\beta_L-(k+1/2)]=B[\beta_L+(k+1/2)] \text{ and } B[\beta_R-k]=B[\beta_R+k]. \quad (13)$$

From this equation the left symmetric center of the sequence $B[k]$ is $\beta_L=1/2$ and the right is $\beta_R=N/2$. Thus, the sequence $B[k]$ represents half-point symmetry on the left side and whole-point symmetry on the right side. One possible selection for storing the sequence is to record the set of $N/2$-points $B[1]$, $B[2]$, ..., $B[N/2]$. Then we can refer to the relation in Eq. (11) to get the other terms. As a result, the total number of nonredundant points stored for these two sequences, $A[k]$ and $B[k]$, is exactly equal to N. This satisfies the demands to filter an N-points sequence and then to maintain exactly N points from the lowpass and highpass channels. With no information lost in these N nonredundant points, perfect reconstruction is possible. Fig. 3 illustrates these relations.

In Fig. 3, the input sequence starts with $a[0]$ and ends at $a[N-1]$. Then we extend this sequence by continuing to take $a[-1]=a[1]$, $a[-2]=a[2]$, and so forth on the left side of this sequence. On the right side of this sequence, we take $a[N]=a[N-2]$, $a[N+1]=a[N-3]$, and so on. While evaluating a coefficient, the filter is applied to the corresponding position. For example, to calculate $A[0]$, the symmetric center of the filter $h[n]$, that is $h[0]$, is above at $a[0]$. Fig. 3(a) and Fig. 3(b) demonstrate

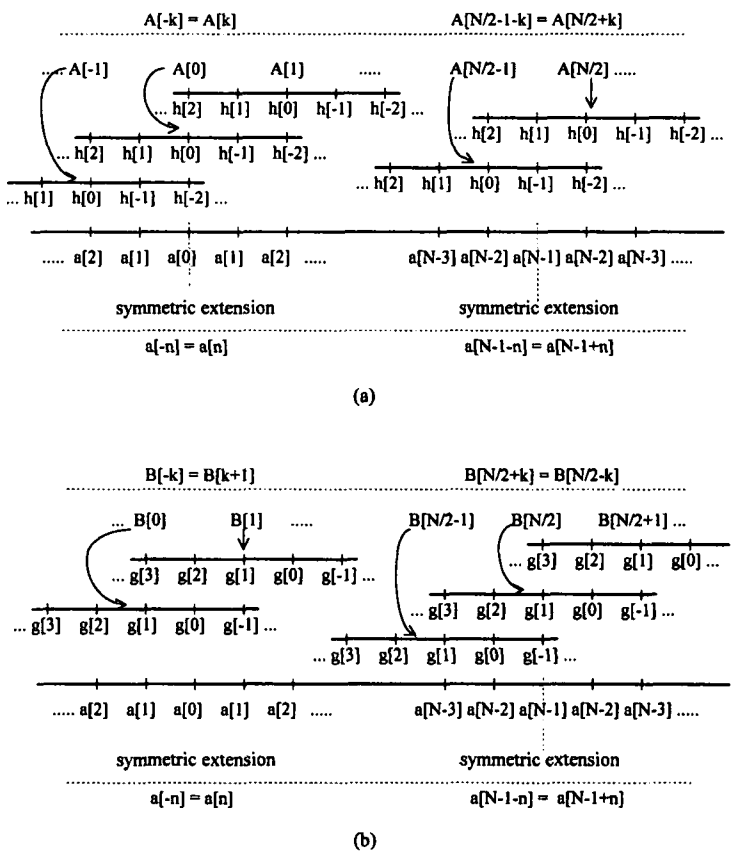


Fig. 3. The filter positions while evaluating the kth wavelet coefficient. (a) filter positions at lowpass channel (b) filter positions at highpass channel.

the filter positions for evaluating each $A[k]$ and $B[k]$, individually. The negative indexes in the filters just indicate that these filters are time-reversed in the convolution. Note that we have to shift the filter position to the right by two steps when computing the next coefficient because the downsampling is performed after the convolution. From Fig. 3, one can easily verify the relations shown in Eqs. (10) and (11).

In the above, we have addressed how to select the nonredundant coefficients of the sequence $A[k]$ and $B[k]$. Now, we will develop an efficient algorithm to calculate these coefficients. For simplicity, in the following discussion, evaluating $A[k]$ and $B[k]$ means to calculate the principal terms:

$$A[k], k=0, ..., N/2-1 \text{ and } B[k], k=1, ..., N/2. \quad (14)$$

Two skills are used in the proposed algorithm. First, to avoid the jumps between the coefficients and to create a pyramid structure, the coefficients of $A[k]$ and $B[k]$ in Eq. (14) will be stored back to the locations of $a[n]$ in the order of $A[k]$ first. Second, to reduce the NMOs, we omit the computation for the terms of the repeated products. For example, as shown in (10), the coefficient $A[0]$ can be decomposed as the sum of the $2p+1$ productsas

$$A[0]=h[-p]a[p]+...+h[-1]a[1]+h[0]a[0] +h[1]a[-1]+...+h[p]a[-p]. \quad (15)$$

Since $h[n]=h[-n]$ and $a[n]=a[-n]$, the product

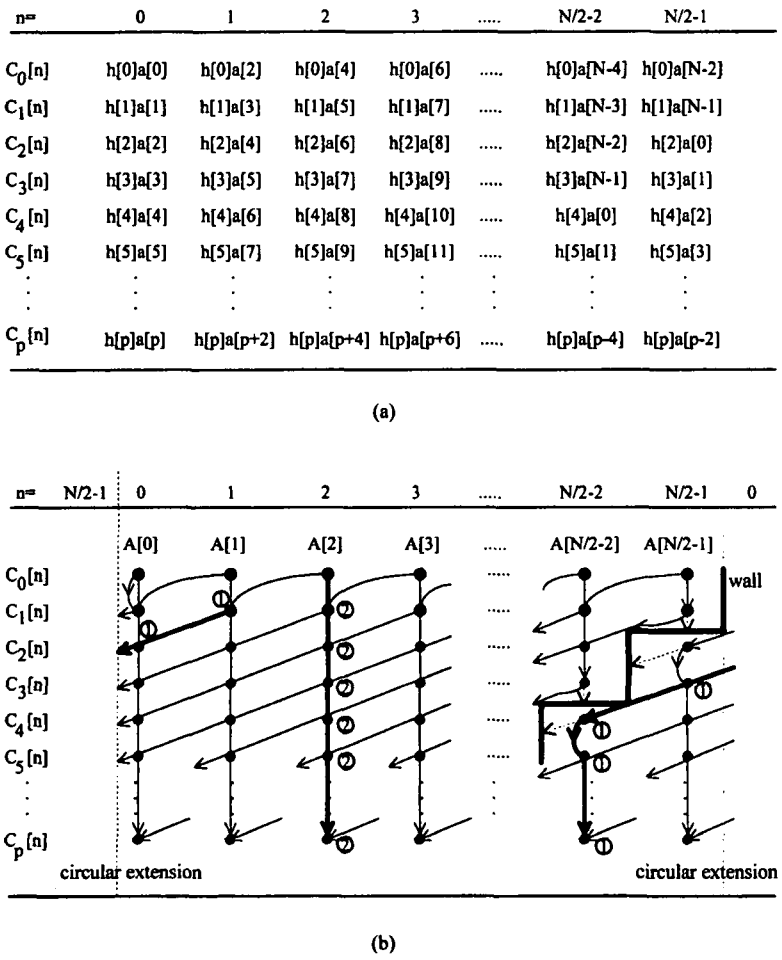


Fig. 4. Evaluation of the principal terms of $A[k]$. (a) arrangement of the products (b) corresponding points and the way to calculate each $A[k]$.

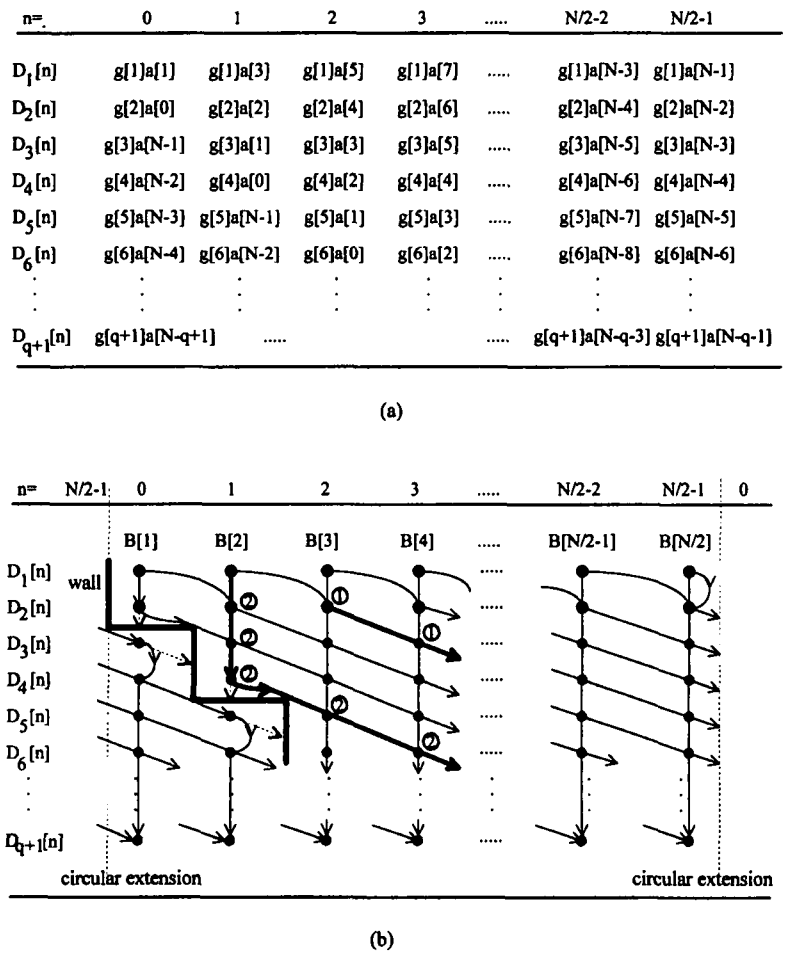


Fig. 5. Evaluation of the principal terms of $B[k]$. (a) arrangement of the products (b) corresponding points and the way to calculate each $B[k]$.

$h[1]a[-1]$ is equal to $h[-1]a[1]$. Thus, the multiplication operation on the $h[1]a[-1]$ can be discarded if $h[-1]a[1]$ is already available. Similarly, the coefficient $A[1]$ can be expressed as

$$A[1]=h[-p]a[2+p]+...+h[-1]a[3]+h[0]a[2]+h[1]a[1] +...+h[p]a[2-p]. \quad (16)$$

Because $h[1]a[1]$ is identical to $h[-1]a[1]$, we can omit the multiplication operation on this term if we sequentially calculate $A[1]$ after $A[0]$. Thus, the main idea to reduce the computational complexity is to list all the values of the possible and non-repeated products while evaluating the coefficients in Eq. (14), and then sum up some of them for each different coefficient. Fig. 4 and Fig. 5 illustrate the frameworks to calculate the sequences $A[k]$ and $B[k]$, respectively.

As shown in Fig. 4(a), we utilize a matrix C to temporarily store all the possible terms of the nonrepeated products for evaluating $A[k]$. The (i,n) element of C is defined as

$$C_i[n]=h[i]a[((2n+i))_N],\ i=0, ..., p,\ n=0, ..., N/2-1, \quad (17)$$

where the notation $((n))_N$ denotes (n modulo N). Fig. 4(b) shows the corresponding points and the method used to calculate $A[k]$. The paths, starting with the first row of C and ending at the last row, only go all the way in either southwest or south. We call the path going toward the southwest *path_1*, and the path going toward the south *path_2*. Given k, we initially set $A[k]=C_0[k]$ and sum up all the points on the two corresponding paths. Let the starting point of the two paths begin with $C_0[k]$. By this approach, $A[k]$ is now available. The next point of each path will be positioned on the next row of C. However, the column it locates on depends on the direction of the path. If the current point on the path is $C_i[m]$, the next point will be either $C_{i+1}[((m-1))_{N/2}]$ for the southwest or $C_{i+1}[m]$ for the south. After we apply the circular extension at the columns of C, a wall is defined as a boundary on the gap between $h[i]a[N-2]$ and $h[i]a[0]$, or between $h[i]a[N-1]$ and $h[i]a[1]$ for $i=0, 1, ..., p$. For a detailed explanation, consider the case $i=0$ and 1 first. The wall is along the gap between $h[0]a[N-2]$, $h[0]a[0]$ and $h[1]a[N-1]$, $h[1]a[1]$. For the case $i=2$ and 3, the wall is between $h[2]a[N-2]$, $h[2]a[0]$ and $h[3]a[N-1]$, $h[3]a[1]$. Continuing the process, we illustrate the results in Fig. 4(b). It is only when *path_1* encounters the wall that then it changes its regular direction to the south, and vice versa for *path_2*. The algorithm is summarized as follows.

Algorithm 1. Evaluate $A[k]$.

Step 1. Initialize

Assign $A[k]=C_0[k]$ and set $i=0$.

Step 2. Sum up all the points located on the two paths to $A[k]$.

Step 2.1) Start *path*_1 and *path*_2 from $C_0[k]$. Let the initial directions of *path*_1 and *path*_2 be southwest and south, respectively.

Step 2.2) Check the terminal condition of summation.
Increase i by 1. If $i>p$ then stop.

Step 2.3) Decide the direction of *path*_1.
If *path*_1 is in its current direction traveling to the next point and it is blocked by the wall, then the direction of *path*_1 will be changed to the south.

Step 2.4) Decide the direction of *path*_2.
If *path*_2 is in its current direction traveling to the next point and is blocked by the wall, then the direction of *path*_2 will be changed to the southwest.

Step 2.5) Add both of the next points on *path*_1 and *path*_2 to $A[k]$. Go back to Step 2.2).

For example, one can trace the bold solid lines indicated in Fig. 4(b) to evaluate $A[2]$. Both paths are illustrated encircling the path number. First we let $A[2]=C_0[2]$, and then continuously add all the points located on these paths together until the paths reach the pth row of matrix C. *Path*_1 travels the points, $C_1[1]$, $C_2[0]$, $C_3[N/2-1]$, $C_4[N/2-2]$, $C_5[N/2-2]$, ... , and $C_p[N/2-2]$. This shows that the direction of *path*_1 is changed after touching the point $C_4[N/2-2]$. As for the points on *path*_2, which follow the order $C_1[2]$, $C_2[2]$, $C_3[2]$, ..., and $C_p[2]$, the direction of *path*_2 is unchanged. To verify this result, substituting k with 2 into Eq. (7) and applying the relations in Eq. (8) and Eq. (9), we get

$$A[2]=h[-p]a[4+p]+...+h[-1]a[5]+h[0]a[4]+h[1]a[3]+$$
$$...+h[p]a[4-p]$$
$$=h[p]a[4+p]+...+h[1]a[5]+h[0]a[4]+h[1]a[3]$$
$$+...+h[p]a[4-p]$$
$$=\text{sum of } path\_2+h[0]a[4]+\text{ sum of } path\_1. \quad (18)$$

That is the exact result obtained by Algorithm 1.

Note that the direction checking in Algorithm 1 (step 2.3 and step 2.4) is only necessary for a few terms of $A[k]$. For most of the terms of $A[k]$, the paths will follow the initial direction all the way until the end. For $A[0]$, ... , $A[p-1]$, we just have to check the direction of *path*_1. As for the last $Trunc(p/2)$ terms of $A[k]$, (the $Trunc(x)$ function returns an integer-type value that is the value x rounded toward zero), only the direction checking for *path*_2 is necessary. Once the direction is changed, the path will follow the new direction and there will be no additional modifications.

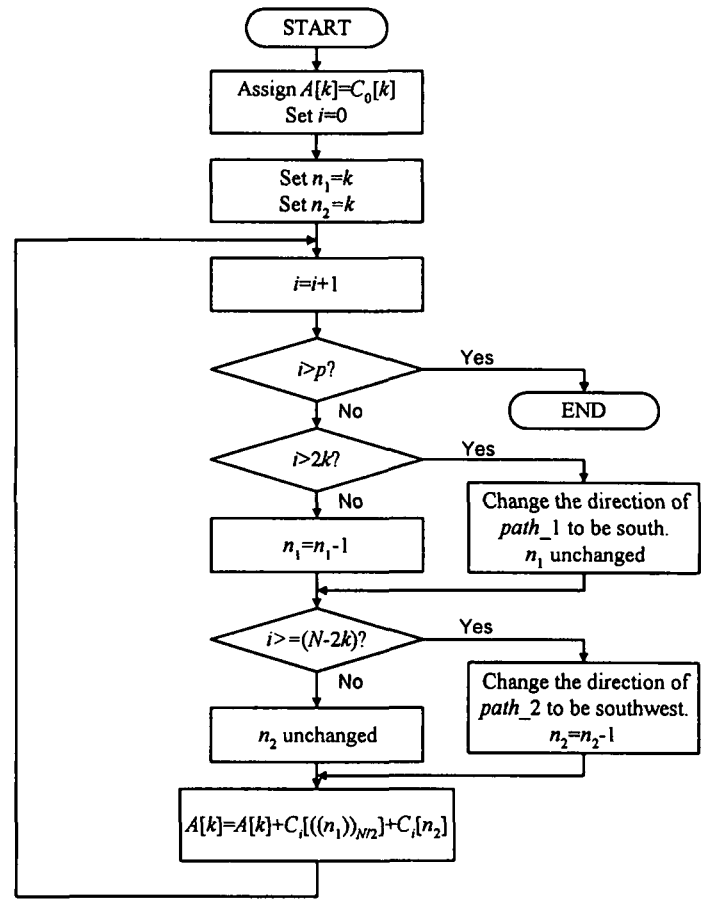


Fig. 6. Flowchart for evaluating $A[k]$.

Similarly, as shown in Fig. 5(a), a matrix D is used to temporarily record all the possible terms of the nonrepeated products for evaluating $B[k]$. The (i,n) element in this matrix is defined as

$$D_i[n]=g[i]a[((2(n+1)-i))_N],$$
$$i=1, ..., q+1, n=0, ..., N/2-1. \quad (19)$$

To obtain $B[k]$, we follow the steps stated in Algorithm 1 and replace $A[k]$ by $B[k]$, $C_0[k]$ by $D_1[k-1]$, p by $q+1$, southwest by southeast, and set the initial value of i to 1, respectively. The next point of $D_i[m]$ on a path is either $D_{i+1}[((m+1))_{N/2}]$ in the southeast direction or $D_{i+1}[m]$ in the south direction here. As with the first example, it is only necessary to check the directions of the paths for just a few coefficients of $B[k]$. Fig. 5(b) shows the corresponding points and the method to calculate each $B[k]$. The bold solid lines denote the paths for $B[2]$. One can decompose $B[2]$ by Eq. (7) into several product terms, and use the same approach mentioned in Eq. (18) to verify this result.

Flowcharts to calculate each $A[k]$ and $B[k]$ are individually shown in Fig. 6 and Fig. 7, respectively.

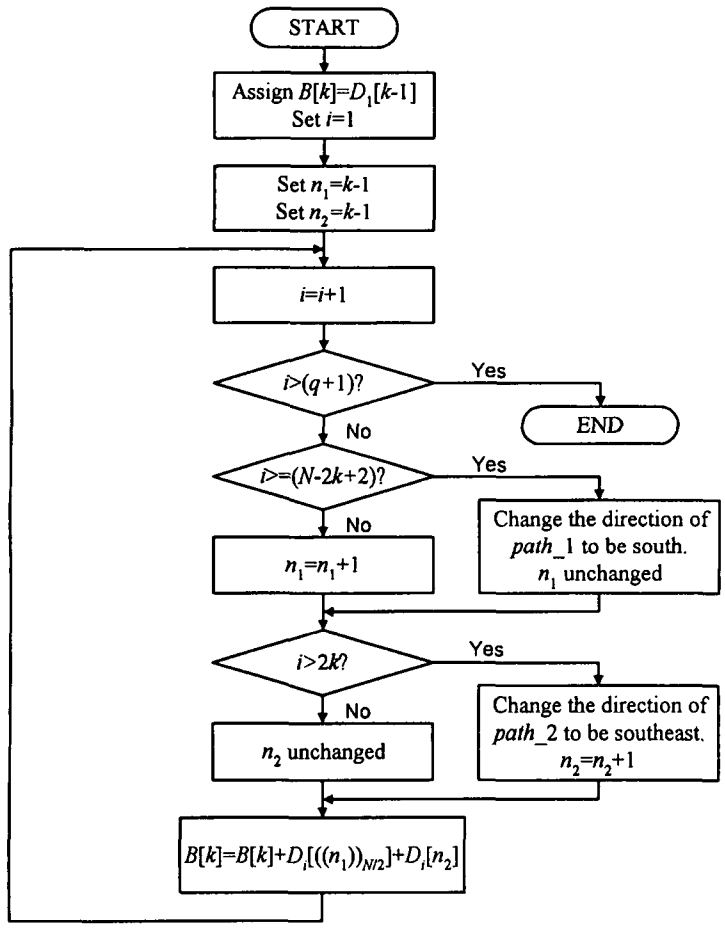


Fig. 7. Flowchart for evaluating $B[k]$.

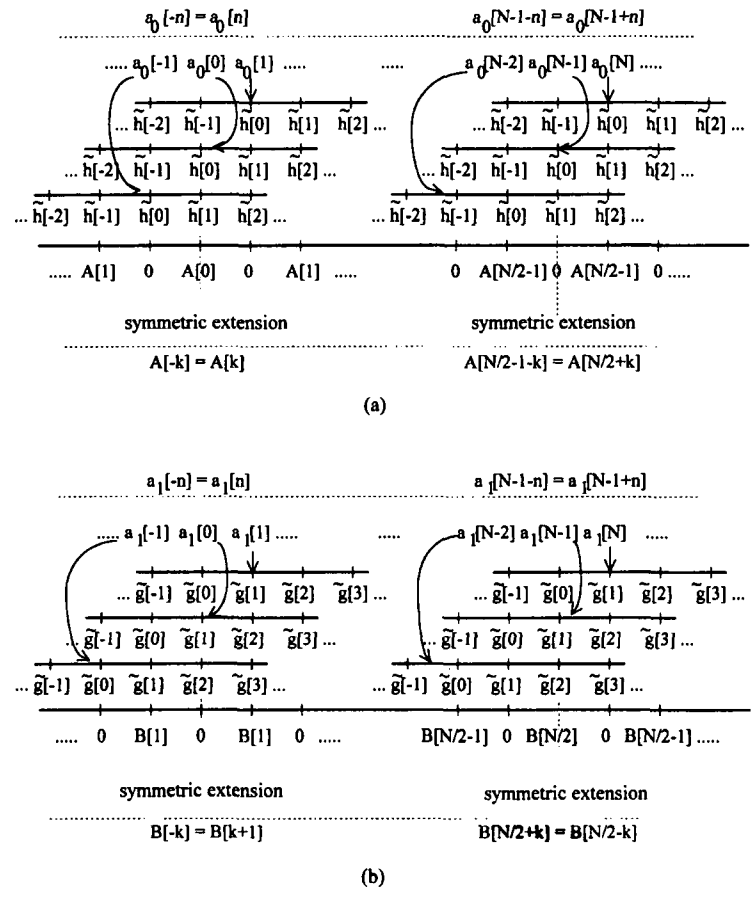


Fig. 8. Filter positions while reconstructing the nth coefficient. (a) filter positions at lowpass channel (b) filter positions at highpass channel.

We treat both n_1 and n_2 in these figures as the columns on *path*_1 and *path*_2, respectively.

IV. FAST CONVOLUTION ALGORITHM FOR IDWT

We will first discuss how to reconstruct the sequence $a[n]$ from the stored principal terms of $A[k]$ and $B[k]$ by a method similar to the method in Section III. Later, we will efficiently list all the possible nonrepeated product terms and sum up some of them for each $a[n]$.

Considering the finiteness of the summation of Eq. (6), we can rewrite it to

$$a[n] \equiv a_0[n]+a_1[n] = \sum_{k=-q}^{q} \tilde{h}[k]A[(k+n)/2] + \sum_{k=-p+1}^{p+1} \tilde{g}[k]B[(k+n)/2], \quad \text{for } k+n=2l,\ l \in \mathbf{Z}, \tag{20}$$

where $\tilde{g}[k]$ is calculated from Eq. (3). The symmetric centers of $\tilde{g}[k]$ and $\tilde{h}[k]$ are at $k=1$ and at $k=0$, respectively. Thus, the constraints on these two filters are

$$\tilde{h}[k]=\tilde{h}[-k] \text{ and } \tilde{g}[1+k]=\tilde{g}[1-k]. \tag{21}$$

To calculate $a[n]$ as shown in Eq. (20), first we individually extend the sequences $A[k]$ and $B[k]$ by Eqs. (10) and (11), respectively, and then upsample them by interpolating zeros. Next, applying the filters as Eq. (21) to these two sequences, we can obtain the relationship as shown in Eq. (9).

Figure 8(a) denotes the reconstructed sequence from the lowpass channel. According to Eq. (10), the sequence $A[k]$ is a whole-point symmetric extension on the left side, but a half-point symmetric extension on the right side. After the extension, we upsample this sequence by interpolating zeros. Then using the filter, we get

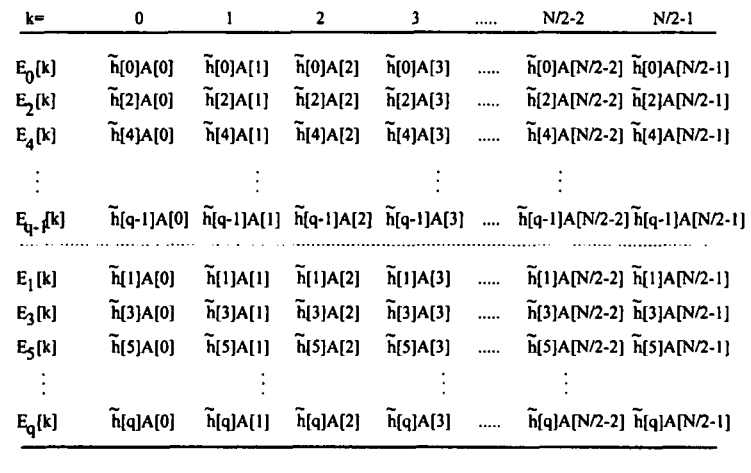
$$a_0[-n]=a_0[n] \text{ and } a_0[N-1-n]=a_0[N-1+n]. \tag{22}$$

Similarly, Fig. 8(b) shows the reconstructed sequence from the highpass channel. The sequence $B[k]$ is a half-point symmetry on the left side, and a whole-point symmetry on the right side. Substituting the symmetry filter as shown in Eq. (21) into Eq. (20), we have

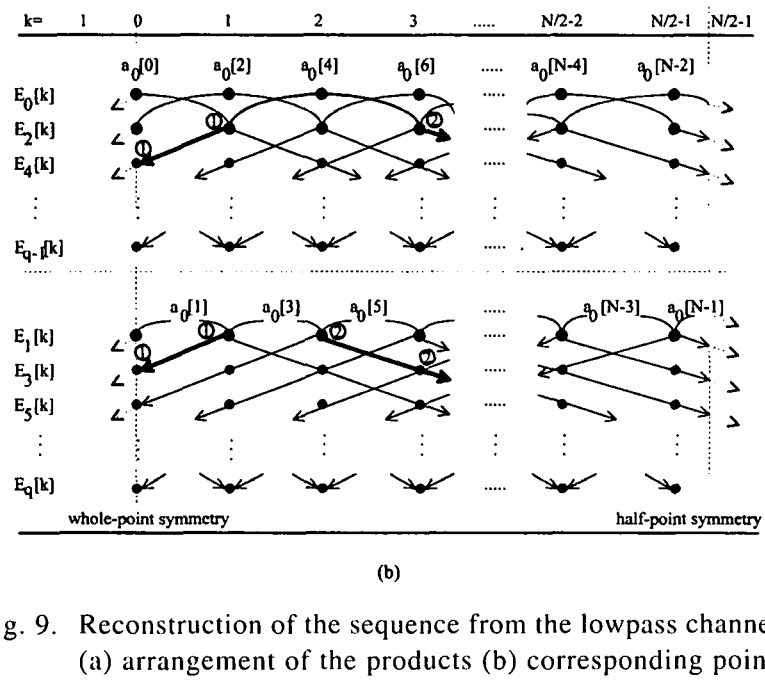
$$a_1[-n]=a_1[n] \text{ and } a_1[N-1-n]=a_1[N-1+n]. \tag{23}$$

With the symmetric property of these two sequences, what we need to do is to calculate the index terms of $a_0[n]$ and $a_1[n]$ at $n=0, \ldots, N-1$, and sum them up point by point to obtain the reconstructed sequence $a[n]$.

To efficiently convolve $A[k]$ and $B[k]$ with filters , we use two matrices E and F to temporarily record all the possible terms of the nonrepeated products to reconstruct $a[n]$. The matrix E stores the products reconstructed from the lowpass channel, and

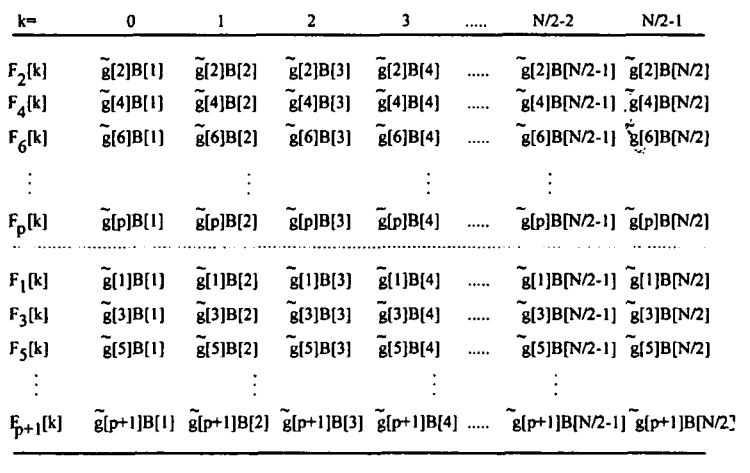


(a)

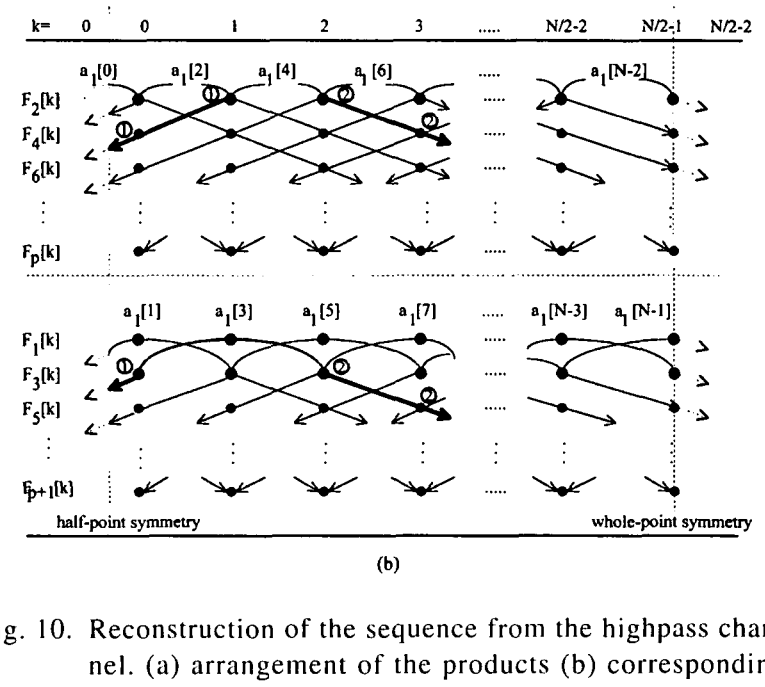


(b)

Fig. 9. Reconstruction of the sequence from the lowpass channel. (a) arrangement of the products (b) corresponding points and the way to calculate each value.



(a)



(b)

Fig. 10. Reconstruction of the sequence from the highpass channel. (a) arrangement of the products (b) corresponding points and the way to calculate each value.

the matrix F keeps the products reconstructed from the highpass channel. The (i,k) element of these two matrices is individually defined as:

$$E_i[k]=\tilde{h}[i]A[k],\ i=0,\ ...,\ q,\ k=0,\ ...,\ N/2-1, \quad (24)$$

and

$$F_i[k]=\tilde{g}[i]B[k+1],\ i=0,\ ...,\ p+1,\ k=0,\ ...,\ N/2-1, \quad (25)$$

Figures 9(a) and 10(a) list the elements in these two matrices, respectively. Fig. 9(b) and Fig. 10(b) show the corresponding points and the way to calculate each reconstructed value. We assume that p is even and q is odd in these figures for simple illustration. However, there are no constraints on p and q in our design. To calculate the even terms of $a[n]$, we will unite the even terms of $a_0[n]$ and $a_1[n]$. The sequence $a_0[n]$ is obtained by summing up the elements in the matrix E while the sequence $a_1[n]$ is given from the matrix F. To evaluate the even terms of $a_0[n]$, we assign $E_0[n]$ to $a_0[2n]$ and then add together all the points located on the two paths to $a_0[2n]$. These paths travel in the even-index rows of matrix E. As shown in Fig. 9(b), these two paths independently travel in the southwest and southeast directions. The elements in matrix E are extended by the whole-point symmetry on the left side, and by the half-point symmetry on the right side. As a result, one can easily add all the points on the paths without changing directions. For simplicity, these paths start with the same point $E_0[n]$ and will terminate at the largest even-index row of matrix E. The initial point, $E_0[n]$, does not need to be added to $a_0[2n]$ again. To calculate the even terms of $a_1[n]$, we have to sum up all the points on the two corresponding paths that travel in the even-index rows of matrix F. As shown in Fig. 10(b), these two paths start with $F_2[n-1]$ and $F_2[n]$ for $a_1[2n]$, then go southwest and southeast, respectively. To maintain the directions of the paths, the elements in matrix F are extended by the half-point symmetry and by the whole-point symmetry on the left side and on the right side, individually. Figure 11 illustrates the complete flowchart to calculate the even terms of $a[n]$.

Similarly, to evaluate the odd terms of $a[n]$, we have to add the odd terms of $a_0[n]$ and $a_1[n]$ together. The odd terms of $a_0[n]$ are obtained from the odd-index rows of matrix E while the odd terms of $a_1[n]$ are found in the odd-index rows of matrix F. One can follow an approach similar to the one in the last paragraph to acquire the odd terms of $a[n]$. The

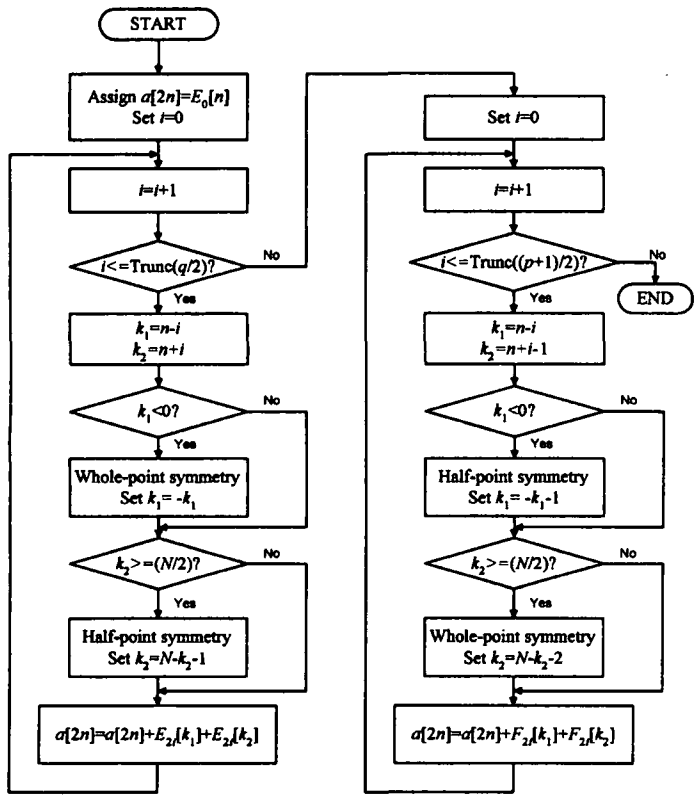


Fig. 11. Flowchart of evaluating the even terms of the reconstructed sequence $a[n]$

flowchart is shown in Fig. 12.

Note that k_1 and k_2 in Fig. 11 and Fig. 12 are used to denote the columns on *path*_1 and *path*_2, respectively.

V. COMPUTATIONAL COMPLEXITY

Assume that the size of an image is N by N. We perform a one-dimensional DWT on the rows and columns of the image. Thus, we have to execute the DWT up to $2N$ times. For an N-point circular convolution, it requires N multiplication operations to evaluate each transformed coefficient. As a result, the NMOs taken in a row will be equal to N^2. Thus, the total NMOs are $2N^3$ for a one scale DWT. For the next scale, the value of N is reduced by one half. The NMOs turn out to be $2N^3+2(N/2)^3$ for a 2-scale DWT. Consequently, for s-scale DWT, the total NMOs are up to

$$\sum_{i=1}^{s} 2(N/2^{i-1})^3 = 16(1-2^{-3s})N^3/7 . \quad (26)$$

For regular convolution, it takes $2p+1$ and $2q+1$ multiplication operations to calculate each $A[k]$ and $B[k]$, respectively. Then the NMOs for a row equal to be $(2p+1)N/2+(2q+1)N/2=(p+q+1)N$. Thus, the regular convolution will demand $2(p+q+1)N^2$ multiplication steps for a one scale DWT. Advancing to the next scale, N is reduced by one half but the NMOs for each $A[k]$ and $B[k]$ are unchanged. So the number of multiplication operations for an s-scale DWT will be

$$\sum_{i=1}^{s} 2(p+q+1)(N/2^{i-1})^2 = 8(p+q+1)(1-2^{-2s})N^2/3 . \quad (27)$$

Table 1. The comparison of computational compolexity among N-point circular convolution, regular convolution and the proposed fast convolution with image size N by N (Pixels), s-Scale DWT,

Convolution	The number of multiplication operations
N-point circular convolution	$16(1-2^{-3s})N^3/7$
Regular convolution	$8(p+q+1)(1-2^{-2s})N^2/3$
Fast convolution	$4(p+q+2)(1-2^{-2s})N^2/3$

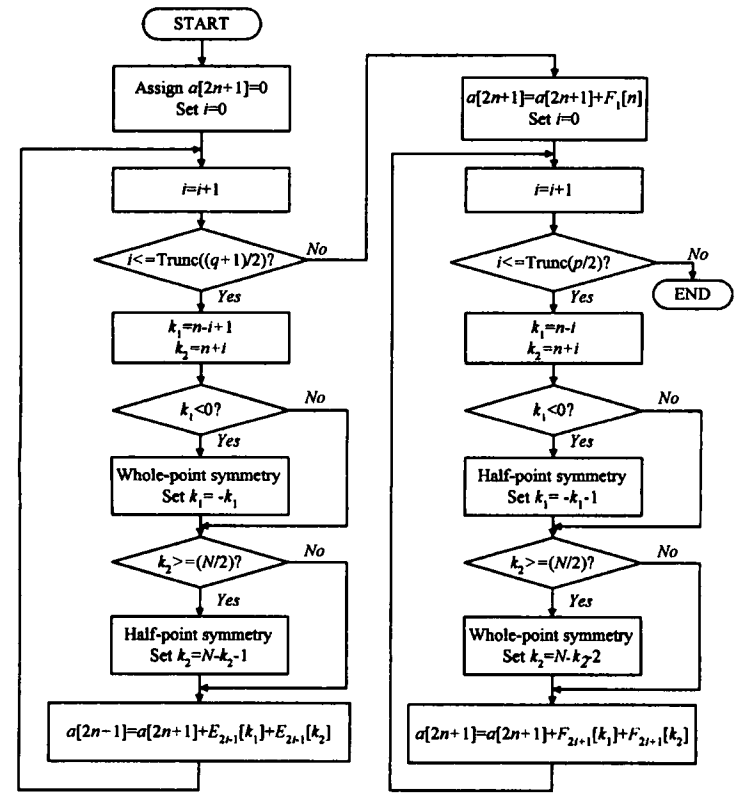


Fig. 12. Flowchart of evaluating the odd terms of the reconstructed sequence $a[n]$

Since our fast convolution algorithm omits the computations on the repeated products, the NMOs of the convolution are the sum of the total elements in matrices C and D for a row of an image. That is, $(p+1)N/2+(q+1)N/2=(p+q+2)N/2$. For the next scale, N is reduced by one half but the NMOs for each $A[k]$ and $B[k]$ are unchanged. Thus, this new convolution will take

$$\sum_{i=1}^{s} (p+q+2)(N/2^{i-1})^2 = 4(p+q+2)(1-2^{-2s})N^2/3 \quad (28)$$

multiplication operations for an s-scale DWT. Comparing Eq. (27) with Eq. (28), we see that the NMOs have been reduced nearly to one half. Table 1 lists the comparison of computational complexity among these three convolutions.

VI. NUMERICAL RESULTS

To compare the performance of symmetric and

Table 2. The Performance comparison of the symmetric extension and the circular extension in the image edges using the EZC coder.

Extension	Images	PSNR(dB)		
		Bit rate=0.2 bpp	Bit rate=0.5 bpp	Bit rate=1 bpp
Symmetric extension	Barbara	27.88	32.65	37.77
	Goldhill	30.00	33.30	36.57
	Lena	33.31	37.20	40.39
	Mandrill	22.75	25.65	29.24
Circular extension	Barbara	27.70	32.47	37.64
	Goldhill	29.88	33.17	36.48
	Lena	33.02	37.02	40.25
	Mandrill	22.73	25.61	29.19

circular extensions of an image, we used our previous study, the enchancing zerotree coding (EZC) (Wu and Su, 1998), to test four images, namely Barbara, Goldhill, Lena, and Mandrill. The EZC coder achieved fairly good results for image compression due to the following three reasons. First, the EZC uses a content-dependent scheme to decompose subbands such that the transformed coefficients get closer to zero. Second, the EZC applies a flag at each subband to denote whether all the coefficients in this subband should be encoded corresponding to the current quantizer. Finally, the optimal compensations on the retrieved coefficients are performed in the EZC coder. All the test images used in this section were 8 bpp, and 512×512. These images can be obtained at the URL, http://links.uwaterloo.ca/greyset2.base.html site.

Table 2 shows the coding results from applying the EZC coder to these test images at various bit rates. Both the symmetric extension and the circular extension are reported for comparison. We utilized the 9/7-tap filter (Antonini *et al.*, 1992) for this coder. As shown in Table II, the performance of the symmetric extension is superior to the circular extension. The former can improve the performance, achieving 0.1 to 0.29 dB for most images. Some reconstructed images for these two extensions are shown in Fig. 13. Although there is no perceptible difference between these two extensions, one still can use the symmetric extension to avoid possible jumps at the transformed coefficients. Note that the reported bit rates are calculated from the actual compressed files and the PSNRs are from the reconstructed images given by the decoding algorithm. This coder is available by using an anonymous ftp to cc.nctu.edu.tw under the directory pub/EZC.

To exhibit the improvement in CPU time, both the regular convolution and our new convolution algorithms were tested on two IBM compatible personal computers (PC). One PC was equipped with an AMD K6-166 MHz CPU and the other was equipped with an Intel Pentium-133 MHz CPU. The results of the average CPU time are listed in Table III. Three pairs of filters 9/7-tap, 9/3-tap (Antonini *et al.*, 1992), and 13/11-tap (Villasenor *et al.*, 1995) were used in our experiments. For the PC with the AMD CPU, the fast convolution algorithm can increase the execution time up to at least 12% and 55% while executing DWT and IDWT, respectively. However, it can achieve at least a 17% improvement for DWT and an 86% improvement for IDWT using the PC equipped with the Intel CPU. All of the execution time results point out that the proposed algorithm can reduce the transform time. The test programs were written in object-Pascal language Delphi 3.0. Of course, one can further reduce the execution time by the assembly language.

VI. CONCLUSION

In this paper, a fast convolution algorithm for DWT/IDWT was proposed. This algorithm focused on the odd length symmetric filters and the symmetric extension of an image. The aim of this paper was to reduce the transform time, which is the essential problem for many wavelet-based image processing algorithms. The proposed algorithm can decrease the number of multiplication operations by nearly one half according to the mathematical analysis on the computational complexity. Converting it to a real program, we can speed up both the DWT and IDWT to at least 12% and 55%, respectively. The most attractive feature of the new algorithm is its simplicity in its final form. The proposed structure can be easily implemented in the hardware design for image or video compression systems.

ACKNOWLEDGMENT

This work was supported by National Science

Table 3. The comparison of average CPU time (sec) of regular convolution and the proposed fast convolution with image size 512 by 512, 5-scale decomposition, and the fillter length (9,7), (9,3) and (13,11).

Convolution	AMD K6-166MHz CPU						Intel Pentium 133MHz CPU					
	(9,7)		(9,3)		(13,11)		(9,7)		(9,3)		(13,11)	
	DWT	IDWT	DWT	IDWT	DWT	IDWT	DWT	IDWT	DWT	IDWT	DWT	IDWT
Regular convolution	1.59	1.94	1.33	1.57	2.13	2.64	1.86	2.98	1.52	2.40	2.61	4.30
Fast convolution	1.38	1.20	1.19	1.01	1.82	1.64	1.43	1.60	1.16	1.26	2.23	2.30
% of reduced time	15%	62%	12%	55%	17%	61%	30%	86%	31%	90%	17%	87%

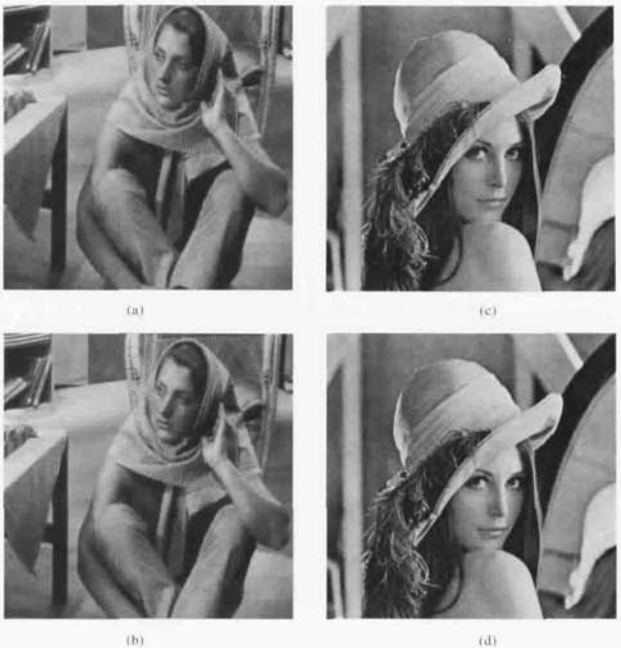


Fig. 13. The comparison of the reconstructed images of the symmetric extension and the circular extension at bit rate=0.5 bpp. (a) Barbara, symmetric extension, and PSNR=32.65 dB. (b) Barbara, circular extension, and PSNR=32.47 dB. (c) Lena, symmetric extension, and PSNR=37.20 dB. (d) Lena, circular extension, and PSNR=37.02 dB.

Council of the ROC under Grant NSC 87-2213-E-009-043.

NOMENCLATURE

$a[n]$	input sequence
$a_0[n]$	reconstructed sequence at lowpass channel
$a_1[n]$	reconstructed sequence at highpass channel
$A[k]$	output transformed sequence at lowpass channel

$B[k]$	output transformed sequence at highpass channel
$C_i[n]$	temporary matrix used to store all the possible product terms for evaluating $A[k]$
$D_i[n]$	temporary matrix used to store all the possible product terms for evaluating $B[k]$
$E_i[k]$	temporary matrix used to store all the possible product terms for evaluating $a_0[n]$
$F_i[k]$	temporary matrix used to store all the possible product terms for evaluating $a_1[n]$
$g[n]$	analysis highpass filter
$\tilde{g}[n]$	synthesis highpass filter
$h[n]$	analysis lowpass filter
$\tilde{h}[n]$	synthesis lowpass fitler
$((n))_N$	n modulo N
$Trunc(x)$	round x near zero

REFERENCES

1. Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I., 1992, "Image Coding Using Wavelet Transform," *IEEE Trans. Image Processing*, Vol. 1, pp. 205-220.
2. **Sullivan, G.,** 1988, "Draft Text of Recommendation H.263 Version 2 ("H.263+") for Decision".
3. Martucci, S.A., Sodagar, I., Chiang, T., and Zhang, Y.-Q., 1997, "A Zerotree Wavelet Video Coder," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 7, No. 1, pp. 109-118.
4. Martucci, S.A., 1994 "Symmetric Convolution and the Discrete Sine and Cosine Transforms," *IEEE Trans. Signal Processing*, Vol. 42, No. 5, pp. 1038-1051.
5. Oppenheim, A.V. and Schafer, R.W., 1989, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice Hall.
6. Pennebaker, W.B. and Mitchell, J.L., 1993, *JPEG-Still Image Data Compression Standard*. New York, NY: Van Nostrand Reinhold .
7. Said, A. and Pearlman, W.A., 1996, "A new, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 6, no. 3, pp.243-250.
8. Shapiro, J.M., 1993, "Embedded Image Coding Using Zerotrees of Wavelets Coefficients," *IEEE Trans. Signal Processing*, Vol. 41, pp. 3445-3462.
9. Strang, G. and Nguyen, T., 1996, *Wavelets and Filter Banks*. Wellesley MA.
10. Vetterli, M. and Kovacevic, J., 1995, *Wavelets and Subband Coding*. Englewood Cliffs, NJ: Prentice Hall.
11. Villasenor, J.D., Belzer, B., and Liao, J., 1995, "Wavelet Filter Evaluation for Image Compression", *IEEE Trans. Image Processing*, Vol. 4, No. 8, pp. 1053-1060.
12. Wu, B.-F. and Su, C.-Y., 1998, "Low Computational Complexity Enhancing Zerotree Coding for Wavelet-based Image Compression," submitted to *Signal Processing: Image Communication.*
13. Xiong, Z., Ramchandran, K., and Orchard, M.T., 1997, "Space-frequency Quantization for Wavelet Image Coding," *IEEE Trans. Image Processing*, Vol. 6, no. 5, pp. 677-693.

Discussions of this paper may appear in the discussion section of a future issue. All discussions should be submitted to the Editor-in-Chief.

Manuscript Received: Aug. 19, 1998
and Accepted: Sep. 22, 1998

針對雙正交小波影像壓縮所提出來的一個快速摺積演算法

吳炳飛

國立交通大學電機系

蘇崇彥

國立臺灣師範大學工業教育系

摘 要

對稱性濾波器和在影像邊緣之對稱性擴展已經被廣泛地使用在小波影像壓縮上。因爲濾波器是對稱的，所以我們有可能利用其對稱性來減少濾波時的計算量。在此篇文章中，我們即是針對這點提出一個快速的摺積演算法給離散時間的小波轉換 (DWT) 和反轉換 (IDWT) 用，使得轉換的時間可以被大大地降低。與正常的摺積來比較，新的演算法可以降低乘法的運算量接近一半。轉換成實際的程式，它可以加快 DWT 和 IDWT 的執行達到至少 12% 和 55% 的程度。將它併入一個增強式的零樹編碼中，可以產生一個相當快且有效的編碼器。實驗結果顯示此編碼器與其他高性能的編碼器相比有相當地競爭性。值得一提的是所提出來的演算法適用於許多以小波爲主的編碼上包括小波視訊編碼。

關鍵詞：快速摺積演算法、小波影像壓縮、對稱性擴展、零樹編碼。