COMPUTING SYMMETRIC WAVELET TRANSFORMS USING DCT AND DST

Mingui Sun, Ching-Chung Li, and Robert J. Sclabassi

Departments of Neurosurgery/Electrical Engineering
University of Pittsburgh, Pittsburgh, PA 15213

Abstract

In this paper the computational issues for symmetric wavelet transforms are investigated. We present a novel frequency-domain algorithm using the discrete cosine transforms (DCTs) and discrete sine transforms (DSTs). A high efficiency is achieved when this algorithm is applied to signals of finite duration, especially images.

1 Introduction

Recently, special attention has been paid to the wavelet transform (WT) where the wavelet is symmetric with respect to a zero or non-zero axis[1]. This type of WT is important in signal processing since a symmetric wavelet corresponds to a linear filter which does not produce pattern distortions in the processed signal. It is well-known that an orthogonal wavelet cannot be symmetric if its support is finite and the same function is used for analysis and synthesis. In practice, however, a filter of infinite length may not cause any significant problem since the filter can be truncated as long as the filter coefficients attenuate sufficiently fast[1]. The error due to truncation decreases as the filter size increases, but the computational cost increases rapidly when the WT is computed directly in the time (or spatial) domain.

This paper presents a filter-size-independent computational approach where symmetric WTs are computed in the frequency domain by the discrete sine and cosine transforms (DSTs and DCTs). The DCTs and DSTs[2] have been extensively studied. Well-designed algorithms have been incorporated into various software packages and hardware devices. Our work connects symmetric WTs to the DCTs and DSTs, enabling us to take the advantage of these efficient algorithms for fast processing of signals and images.

2 Rationales

It is well known that convolution between a pair of signals can be computed by taking the DFT on these signals, multiplying the results, and then computing the inverse DFT. The total operation count for the whole computational process is roughly $\eta N \log N$, where N is the size of the longer signal and η is an algorithm-dependent factor. The DFT is a complex-valued transformation where each complex multiplication requires six floating-point arithmetic operations. As a result, the value of η may be very large, suggesting a low computational efficiency. We approach this problem by converting the DFT to the DCTs and DSTs which require fewer floating-point operations. This conversion is possible because of the symmetry present in the data. We first mirror-extend the input signal at the borders and then transform the extended signal to the Fourier domain using the DCT. The linear-phase filters corresponding to the wavelet and scaling function are also transformed to the Fourier domain (via the DST and DCT) where the convolutional operations in the conventional algorithm[3] are replaced by much simpler operations of multiplication and duplication. Once the desired wavelet scale level is reached, a DCT is applied again to obtain the time (or spatial) domain result.

3 DCT/DST Formulation

Let the input signal, $x(n)$, have $N+1$ discrete samples indexed by $n = 0, 1, \cdots, N-1, N$. We make a mirror-extension which duplicates the signal at each border by setting $x(1) = x(2N-1), x(2) = x(2N-2), \cdots, x(N-1) = x(N+1), x(N-2) = x(N+2), \cdots$. Appendix 2 shows that the DFT of $x(n)$, $X(k)$, can be written as

$$X(k) = 2 \cdot \mathrm{DCT}_N^1[r_n x(n)] \tag{1}$$

where

$$r_n = \begin{cases} \frac{1}{2}, & n = 0, N \\ 1, & \text{otherwise.} \end{cases}$$

and DCT_N^1 is the Type I DCT (Appendix 1).

Let $h(n)$ and $g(n)$ be, respectively, the low-pass and high-pass filters corresponding to the wavelet and scaling function. We have previously shown[1] that $h(n)$ can only possess an even-symmetry, while $g(n)$ can be either even- or anti-symmetric. We have also shown that the axis of symmetry is either an integer (usually zero) or a half-integer (usually ± 0.5). In order to compute the WT, we first pad zeros beyond the last non-zero sample on either side of each filter forming a sequence of length $2N$, and then convert the DFTs of $h(n)$ and $g(n)$ to DCTs and DSTs.

3.1 Conversion of $h(n)$

Let $h(n)$ be even-symmetric with respect to β. When $\beta = 0$, we have[1] $h(n) = h(-n)$, and the DCT conversion is similar to that in Eq. (1), i.e.,

$$H(k) = 2 \cdot \mathrm{DCT}^1_N[r_n h(n)] \tag{2}$$

Alternatively, when $\beta = 1/2 + r$, where r is an integer, we have, by Appendix 2,

$$H(k) = 2e^{-i\pi k(r+1/2)/N}\mathrm{DCT}^2_N[h(n+r+1)] \tag{3}$$

where $\mathrm{DCT}^2_N[h(n)]$ is the Type II DCT (Appendix 1).

3.2 Conversion of $g(n)$

When $g(n)$ is even-symmetric with respect to zero, the result is identical to that given in Eq. (2) except that symbols h and H are replaced by g and G. On the other hand, when $g(n)$ is anti-symmetric with respect to α, the following results can be shown (Appendices 1 and 2):

If $\alpha = 0$, we have $g(n) = -g(-n)$, and

$$G(k) = -2ie^{-i\pi kr/N}\mathrm{DST}^1_N[g(n+r)]$$

On the other hand, if $\beta = 1/2 + r$, where r is an integer, we have $g(n) = -g(2r+1-n)$ and

$$G(k) = -2ie^{-i\pi k(2r+1)/(2N)}\,\mathrm{DST}^2_N[g(n+r)].$$

4 Fourier Domain Computation

The Fourier domain equivalence of the conventional algorithm[3] is given by

$$W_{2^{j+1}}(k) = S_{2^j}(k)G_{2^j}(k) \quad \text{and} \tag{4}$$

$$S_{2^{j+1}}(k) = S_{2^j}(k)H_{2^j}(k). \tag{5}$$

This algorithm is initialized by $S_{2^0}(k) = X(k)$, $H_{2^0}(k) = H(k)$ and $G_{2^0} = G(k)$. The "standard form" of the forward WT involves a down-sampling process (also called decimation) between scale levels. Thus, the total number of wavelet coefficients obtained is the same as the input data. In some image processing applications (e.g., edge detection), however, the decimation process is often purposely eliminated and the number of coefficients is kept the same in all scale levels. Due to the presence or absence of this decimation process, there are two different ways to implement (4) and (5).

WT without Decimation In this case, the number of samples in each of $S_{2^j}(k)$, $W_{2^j}(k)$, $H_{2^j}(k)$ and $G_{2^j}(k)$ remains the same for all values of j. Since filter $h_{2^j}(n)$ is formed by adding a zero to every adjacent samples of $h_{2^{j-1}}(n)$, we have $H_{2^j}(k) = H_{2^{j-1}}(2k)$ which implies that $H_{2^j}(k)$ can be obtained without any arithmetic operations. The same argument applies to $G_{2^j}(k)$. Figure 1 illustrates this process where the curve on each disk represents $|H_{2^j}(k)|$. Any angular area given by $\theta = 2^{-j}\pi$ contains a set of $2^{-j}N$ independent values of $H_{2^j}(k)$.

WT with Decimation In this case, the sizes of $S_{2^j}(k)$ and $W_{2^j}(k)$ decrease by one-half for each increment of j. Let $\tilde{z}(2n) = z(n)$, where $\tilde{z}(n)$ and $z(n)$ are, respectively, any $2N'$ and N' periodic sequences. Then, it can be verified, by using the identity $\sum_{n=0}^{N'-1}\tilde{z}(2n) = (1/2)\sum_{n=0}^{2N'-1}\tilde{z}(n)[1+(-1)^n]$, the DFTs of $z(n)$ and $\tilde{z}(n)$ are related by

$$Z(k) = \frac{1}{2}[\tilde{Z}(k) + \tilde{Z}(k+N')]. \tag{6}$$

Eq. (6) indicates that $H_{2^j}(k)$ and $G_{2^j}(k)$ can be generated from $H_{2^{j-1}}(k)$ and $G_{2^{j-1}}(k)$ by an averaging process as illustrated in Figure 2.

When the inverse WT is desired, the frequency domain algorithm is given by

$$S_{2^{j-1}}(k) = W_{2^j}(k)\overline{G_{2^{j-1}}(k)} + S_{2^j}(k)\overline{H_{2^{j-1}}(k)} \tag{7}$$

where the top bar denotes complex conjugate. The algorithm starts from $j = J$ and ends with $j = 1$.

5 Conversion to the Time Domain

It has been shown[1] that the values of α and β (axes of symmetry) cannot be zero at the same time. Therefore, the results of the WT must contain displacements which may be considerably large if the WT is computed without decimation. By using (4) and (5) repeatedly, it can be shown that $S_{2^j}(k) = Y_s(k)e^{i\pi\gamma_s/N}$ and $W_{2^j}(k) = -iY_w(k)e^{i\pi\gamma_w/N}$, where $Y_s(k)$ and $Y_w(k)$ are real-valued periodic functions, and $\gamma_s = -\beta(2^j - 1)$ and $\gamma_w = \beta - (\beta + \alpha)2^{j-1}$, respectively, are the displacements in $s_{2^j}(n)$ and $w_{2^j}(n)$ with respect to the input signal $x(n)$. Using these results, the time domain signals $s_{2^j}(n)$ can be written into two types of DCTs:

Case 1. γ_s is an integer. We have $Y_s(k) = Y_s(-k)$ and

$$s_{2^j}(n - \gamma_s) = \frac{1}{N}\mathrm{DCT}^1_N[r_k Y_s(k)]. \tag{8}$$

[1] The use the negative index with respect to the modulo of the signal's period. In this case, $h(n) = h(-n)$ is equivalent to $h(n) = h(-n) \bmod 2N$

Case 2. γ_s is a half-integer. We have $Y_s(k) = -Y_s(-k)$ and

$$s_{2^j}(n - \lfloor\gamma_s\rfloor) = \frac{1}{N}\mathrm{DCT}_N^3[r_k Y_s(k)] \quad (9)$$

where DCT_N^3 is given in Appendix 1 and $\lfloor\gamma_s\rfloor = \gamma_s - 1/2$.

The case of $W_{2^j}(k)$ is similar to that of $S_{2^j}(k)$ except that γ_w is the control factor and that the DFT is converted to the DSTs. There also exist two cases:

Case 1. γ_w is an integer. We have $Y_w(k) = Y_w(-k)$ and

$$w_{2^j}(n-\gamma_w) = \frac{1}{2N}\sum_{k=0}^{2N-1}(-i)Y_w(k)e^{-i\pi(n+\gamma_w)k/N} \quad (10)$$

$$= \frac{1}{N}\mathrm{DST}_N^1[Y_w(k)]. \quad (11)$$

Case 2. γ_w is a half-integer. We have $Y_w(k) = -Y_w(-k)$ and

$$w_{2^j}(n - \lceil\gamma_w\rceil) = \frac{1}{N}\mathrm{DST}_N^3[r_k Y_w(k)] \quad (12)$$

where DST_N^3 is given in Appendix 1 and $\lceil\gamma_w\rceil = \gamma_w + 1/2$.

6 Comparison of Efficiency

We demonstrate the computational efficiency of the DCT/DST approach through an example. We assume $h(n) = h(-n)$ and $g(n) = g(-2-n)$ and require that the 2-D spatial-domain results $s_{2^j}(n,m)$ and $w_{2^j}(n,m)$, for $j = 1,2,3,4$, be computed (the worst case for the DCT/DST approach). The input is a 512×512 image which is first fully decomposed and then reconstructed from the spatial-domain wavelet coefficients. For simplicity, we exclude the arithmetic operations that are common to all algorithms.

Three algorithms are compared: 1) conventional WT[3], 2) DFT, and 3) DCT/DST. In the second algorithm, we simply use the DFT (implemented by the FFT algorithm) to compute convolutions. In the third algorithm, we apply the FFTP algorithm[4] to compute the DCT_N^1. The number of arithmetic operations O_1, O_3, and O_3, respectively, corresponding to the three algorithms are given by

$$O_1 = 32 \times (4M+1)N^2 \approx 8.389 \times 10^6(4M+1) \quad (13)$$

$$O_2 = 208N^2\log_2 2N + 56N^2 \approx 5.599 \times 10^8, \text{ and} \quad (14)$$

$$O_3 = 52N^2\log_2 2N - 22N^2 \approx 1.305 \times 10^8 \quad (15)$$

where M is the size of the truncated filter. By comparing among O_1, O_2 and O_3, it is clear that the conventional WT is the most efficient algorithm if the filter size is extremely short, as in the case of the Haar wavelet. For $3 < M \le 16$, the DCT/DST becomes the more efficient algorithm. When $M = 20$, the computational efficiency of the DCT/DST is approximately 5.2 times of the conventional WT and 4.3 times of the DFT. Furthermore, it is noticed that the number of operations required for the DCT/DST can be further reduced when the spatial domain $s_{2^j}(n,m)$ and $w_{2^j}(n,m)$ do not have to be evaluated for every j.

7 Conclusion

We have developed an efficient approach to the computation of symmetric WTs using the DCTs and DSTs. This approach allows us to apply sophisticated DCT/DST algorithms and permits the use of long filters without affecting computational efficiency.

Appendix 1 Six Types of DCTs and DSTs

We define six types of the discrete cosine transforms (DCTs) and discrete sine transforms (DSTs) as follows:

$$\mathrm{DCT}_N^1[x(n)] = \sum_{n=0}^{N} x(n)\cos\left(\frac{\pi nk}{N}\right) \quad (16)$$

$$\mathrm{DCT}_N^2[x(n)] = \sum_{n=0}^{N-1} x(n)\cos\left(\frac{\pi(n+0.5)k}{N}\right) \quad (17)$$

$$\mathrm{DCT}_N^3[X(k)] = \sum_{k=0}^{N-1} X(k)\cos\left(\frac{\pi(n+0.5)k}{N}\right) \quad (18)$$

$$\mathrm{DST}_N^1[x(n)] = \sum_{n=1}^{N-1} x(n)\sin\left(\frac{\pi nk}{N}\right) \quad (19)$$

$$\mathrm{DST}_N^2[x(n)] = \sum_{n=1}^{N} x(n)\sin\left(\frac{\pi(n-0.5)k}{N}\right) \quad (20)$$

$$\mathrm{DST}_N^3[X(k)] = \sum_{k=1}^{N} X(k)\sin\left(\frac{\pi(n-0.5)k}{N}\right). \quad (21)$$

These definitions may be slightly different from those found in the literature[2]. We point out that a unification of definitions can be made by normalizing and scaling the inputs.

Appendix 2 Conversion from DFT to DCT/DST

In this appendix, we present six cases of conversion between the DFT and the DCT/DST.

Case 1 $y_1(n) = y_1(-n)$: The signal is an even function and its DFT is given by

$$Y_1(k) = \sum_{n=0}^{N} y_1(n)e^{-i\pi nk/N} + \sum_{n=N+1}^{2N-1} y_1(n)e^{-i\pi nk/N} \quad (22)$$

Changing summation variable yields

$$Y_1(k) = 2\sum_{n=0}^{N} r_n y_1(n)\cos(\pi nk/N) = 2\mathrm{DCT}_N^1[r_n y_1(n)] \quad (23)$$

with $r_n = \begin{cases} \frac{1}{2}, & n = 0, N \\ 1, & \text{otherwise} \end{cases}$

Case 2 $y_2(n) = -y_2(-n)$: The signal is an odd function implying $y_2(0) = y_2(N) = 0$. The DFT of $y_2(n)$ can be derived in a similar fashion as in Case 1, the result is

$$Y_2(k) = -2i\mathrm{DST}_N^1[y_2(n)]. \quad (24)$$

Case 3 $y_3(n) = y_3(2\beta - n)$: y_3 is an even-symmetric signal with respect to an integer β. Since $y_3(n) = y_1(n - \beta)$, the circular shift property of the DFT yields

$$Y_3(k) = 2e^{-i\pi k\beta/N}\mathrm{DCT}_N^1[r_n y_3(n + \beta)]. \quad (25)$$

Case 4 $y_4(n) = -y_4(2\alpha - n)$: y_4 is an anti-symmetric signal with respect to an integer α. A procedure similar to case 3 shows

$$Y_4(k) = -2ie^{-i\pi k\alpha/N}\mathrm{DST}_N^1[y_4(n + \alpha)]. \quad (26)$$

Case 5 $y_5(n) = y_5(2r + 1 - n)$: It can be checked that the center of symmetry of $y_5(n)$ is located at $\beta = (2r + 1)/2$ which is a half-integer. Taking the DFT of $y_5(n)$ and manipulating indices, it can be shown that

$$Y_5(k) = 2e^{-i\pi k(2r+1)/2N)}\mathrm{DCT}_N^2[y_5(n + r + 1)]. \quad (27)$$

Case 6 $y_6(n) = -y_6(2r+1-n)$: $y_6(n)$ is an antisymmetric signal with respect to a half-integer $\alpha = (2r + 1)/2$. A derivation similar to case 5 yields

$$Y_6(k) = -2ie^{-i\pi k(2r+1)/(2N)}\mathrm{DST}_N^2[y_5(n + r)]. \quad (28)$$

References

[1] M. Sun, C-C. Li, H. Szu, Y. Zhang, and R. J. Sclabassi, "Symmetrical wavelet transforms for edge localization," *Optical Engineering - The Journal of SPIE*, Vol. 33, No. 7, 1994, pp. 2272-2281.

[2] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEET Trans. ASSP*, Vol. ASSP-32, 1984, pp. 803-816.

[3] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Signal Processing*, Vol. 14, No. 7, pp. 710-732, 1992.

[4] M. Sun, C. C. Li, L. N. Sekhar and R. J. Sclabassi, "Efficient computation of discrete pseudo Wigner distribution," *IEEE Trans. ASSP*, Vol. ASSP-37, 1989, pp. 1135-1142.

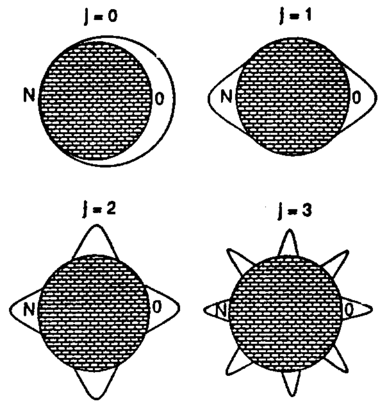

Figure 1 The periodic nature of $H_{2^j}(k)$ is illustrated by the hatched disks. The curve on each disk represents $|H_{2^j}(k)|$. Since $H_{2^j}(k) = H_{2^{j-1}}(2k)$, the curves duplicate and narrow themselves as j increases. However, the number of slots in each of the disks is equal to $2N$ regardless of the values of j.

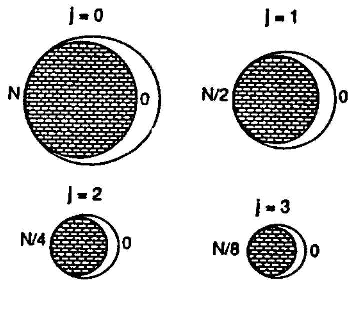

Figure 2 Due to the decimation process, the number slots in $H_{2^j}(k)$ reduces by half with each increment of j; however, the spectral profile of $H_{2^j}(k)$ keeps unchanged.