Development of a 2-D Wavelet Transform based on Kronecker Product

Waleed A. Mahmoud*, Ahmed S. Hadi* and Talib M. Jawad**
*University of Baghdad, **University of Al-Nahrain.

Abstract

Advances in wavelet transform have produces algorithms capable of surpassing the existing digital signal processing. This paper presents a new wavelet transform computation method that verifies the potential benefits of the kronecker product and gains much improvement in terms of low computational complexity.

A fast algorithm for computing 2-D wavelet transform based on a modified orthogonal matrix is developed using kronecker product. The algorithm has several promising features which make it suitable for 2-D signal processing applications.

Keywords: 2-D DWT and Kronecker Product.

Introduction

The discrete wavelet transform (DWT) is found to be an efficient and useful tool for signal processing applications [1, 2]. It has become a powerful tool in various digital processing such as audio, image and multimedia [1-3]. Fast computation methods introduced recently applied in several approaches for directional representations of image data, each one with the intent of efficiently representing image features. Among these examples include Ridglet [4], Curvelets [5], Contourlets [6], and Shearlets [7].

As the choice of transform used depends in particular, on computational complexity which is measured in terms of the number of multiplications and additions required for the implementation of the transform.

This paper facilitates the computation of discrete 2-D wavelet transform involving a computation procedure consisting mainly of basic arithmetic operations like matrix multiplication, permutations, shuffling and other easy to verify operations.

A simple way to perform wavelet decomposition on 2-D signal is to alternate between operations on the rows and columns. First, wavelet decomposition is performed on each row of the 2-D signal matrix. Then, wavelet decomposition is performed to each column of the previous result. The process is repeated to perform the complete wavelet decomposition. The basic idea developed in this paper is that of solving the problem of cascaded two steps multiplication of 2-D DWT computation into only one step of multiplication.

A particular set of wavelets is specified by a particular set of numbers, called wavelet filter coefficients. For simplicity we will restrict to wavelet filters in a class discovered by Daubechies. The most localized embers often used are Haar and Daubechies 4 (Db4) coefficients. For easy of notation we will use the notation h(0) and h(1) for Haar coefficients and h(0), h(1), h(2), and h(3) for Daubechies 4 coefficients.

In the 2-D Haar bases wavelet, the matrix form will be as

$$T=\begin{bmatrix} h(0) & h(1) & 0 & 0 \\ 0 & 0 & h(0) & h(1) \\ h(1) & -h(0) & 0 & 0 \\ 0 & 0 & h(1) & -h(0) \end{bmatrix}_{N\times N}$$

$$=\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad \text{......................}(1)$$

The Daubechies wavelet transform are defined in the same way as Haar wavelet transform by computing the running averages and differences via scalar products with scaling signals and wavelets the only difference between them consists in how these scaling signals and wavelets are defined [8].

The simplest and most localized number is the 2-D DWT Db4 which has four coefficient h(0), h(1), h(2), and h(3) where

h(0)=$\frac{(1+\sqrt{3})}{4\sqrt{2}}$0.4830, h(1)= $\frac{(3+\sqrt{3})}{4\sqrt{2}}$=0.8365, h(2)= $\frac{(3-\sqrt{3})}{4\sqrt{2}}$=0.224, and h(3)= $\frac{(1-\sqrt{3})}{4\sqrt{2}}=-0.1294$

The Daubechies scaling function and the formed matrix is [9],

$$T=\begin{bmatrix} h(0) & h(1) & h(2) & h(3) \\ h(2) & h(3) & h(0) & h(1) \\ h(3) & -h(2) & h(1) & -h(0) \\ h(1) & -h(0) & h(3) & -h(2) \end{bmatrix}_{N\times N}$$

$$=\begin{bmatrix} 0.4830 & 0.8365 & 0.2241 & -0.1294 \\ -0.1294 & -0.2241 & 0.8365 & -0.4830 \\ 0.2241 & -0.1294 & 0.4830 & 0.8365 \\ 0.8365 & -0.4830 & -0.1294 & -0.2241 \end{bmatrix} \quad \text{...} (2)$$

Fast Algorithm for Computing 2-D DWT:

Any good and successful transform should have the following categories available, low complexity and efficient implementation. The typical approach is to process each of the rows inorder and then process each column of the result [10]. Computation complexity is measured in terms of the number of multiplications and additions required for the implementation of the transform. Efficient implementation is a measure of how well the transformation can be realized using parallel processing. This result is a simple and easy algorithm to use for fast and overall one packet computation.

Mahmoud et al [11] suggested a fast algorithm for computing wavelet transform of 2-D signal matrix; such method provides a mixed orthogonal matrix that offers benefits over the conventional methods in terms of reduce the computation and simplify the implementation.

To compute a single level is orthogonal-based fast discrete wavelet transform (FDWT) for 2-D signal, the following steps should be followed:

1. Signal matrix “X” should be square (N×N), where N must be power of 2.
2. Construct a transform matrix (T) using the desired wavelet bases functions.
3. Apply transformation by multiplying the transform matrix, T, by the input signal matrix and next multiplying their result by the transpose of T, thus

$$Y = T.X.T^t \quad \text{...} (3)$$

Where Y is the final N×N discrete wavelet transform matrix of the, N×N input signal matrix, X.

Let’s take a general 2-D signal, for example any 4×4 matrix, and apply the following steps to compute 2-D FDWT using the fast seperable method:

1. Let X be the input 2-D signal,

$$X_{4.4}=\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

2. For an 4×4 matrix input 2-D signal, X, construct a 4×4 transformation matrix, T, using Haar coefficients filter, or using Db4 coefficients filter:

   a. Apply row transformation by:

[Z]=[T][X] .... (4)

By using Haar:

$$Z(Haar)_{4.4}=$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} . \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$=\begin{bmatrix} 4.2426 & 5.6569 & 7.0711 & 8.4853 \\ 15.5563 & 16.9706 & 18.3848 & 19.7990 \\ -5.6569 & -5.6569 & -5.6569 & -5.6569 \\ -2.8284 & -2.8284 & -2.8284 & -2.8284 \end{bmatrix}$$

   b. By using Db4:

$$Z(Db4)_{4.4}$$

$$=\begin{bmatrix} 0.4830 & 0.8365 & 0.2241 & -0.1294 \\ -0.1294 & -0.2241 & 0.8365 & -0.4830 \\ 0.2241 & -0.1294 & 0.4830 & 0.8365 \\ 0.8365 & -0.4830 & -0.1294 & -0.2241 \end{bmatrix}$$

$$. \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$=\begin{bmatrix} 5.0002 & 6.4144 & 7.8286 & 9.2428 \\ -0.0004 & -0.0004 & -0.0004 & -0.0004 \\ 14.7986 & 16.2128 & 17.6270 & 19.0412 \\ -5.6564 & -5.6564 & -5.6564 & -5.6564 \end{bmatrix}$$

3. Transpose the Z matrix,

   a. For Haar:

$$Z^t(Haar)_{4\times 4}$$

$$=\begin{bmatrix} 4.2426 & 15.5563 & -5.6569 & -2.8284 \\ 5.6569 & 16.9706 & -5.6569 & -2.8284 \\ 7.0711 & 18.3848 & -5.6569 & -2.8284 \\ 8.4853 & 19.7990 & -5.6569 & -2.8284 \end{bmatrix}$$

b. For Db4:

$$Z^t(Db4)_{4.4} = \begin{bmatrix} 5.0002 & -0.0004 & 14.7986 & -5.6564 \\ 6.4144 & -0.0004 & 16.2128 & -5.6564 \\ 7.8286 & -0.0004 & 17.6270 & -5.6564 \\ 9.2428 & -0.0004 & 19.0412 & -5.6564 \end{bmatrix}$$

4.LetB]=[T][Z]^t .... (5)

a. For Haar:

$$B(Haar)_{4\times 4} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}.$$

$$\begin{bmatrix} 4.2426 & 15.5563 & -5.6569 & -2.8284 \\ 5.6569 & 16.9706 & -5.6569 & -2.8284 \\ 7.0711 & 18.3848 & -5.6569 & -2.8284 \\ 8.4853 & 19.7990 & -5.6569 & -2.8284 \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 23 & -8 & -4 \\ 11 & 27 & -8 & -4 \\ -2 & -2 & 0 & 0 \\ -1 & -1 & 0 & 0 \end{bmatrix}$$

b. For Db4:

$$B(Db4)_{4\times 4} = \begin{bmatrix} 0.4830 & 0.8365 & 0.2241 & -0.1294 \\ -0.1294 & -0.2241 & 0.8365 & -0.4830 \\ 0.2241 & -0.1294 & 0.4830 & 0.8365 \\ 0.8365 & -0.4830 & -0.1294 & -0.2241 \end{bmatrix}. \begin{bmatrix} 5.0002 & -0.0004 & 14.7986 & -5.6564 \\ 6.4144 & -0.0004 & 16.2128 & -5.6564 \\ 7.8286 & -0.0004 & 17.6270 & -5.6564 \\ 9.2428 & -0.0004 & 19.0412 & -5.6564 \end{bmatrix}$$

$$= \begin{bmatrix} 8.3391 & -0.0006 & 22.1960 & -7.9993 \\ -0.0001 & 0 & -0.0001 & 0 \\ 11.8033 & -0.0006 & 25.6602 & -7.9993 \\ -1.9998 & 0 & -1.9998 & 0 \end{bmatrix}$$

5.The final DWT matrix [Y] is equal to the transpose of [B] matrix.

a. For Haar:

$$Y_{4.4} = \begin{bmatrix} 7 & 11 & -2 & -1 \\ 23 & 27 & -2 & -1 \\ -8 & -8 & 0 & 0 \\ -4 & -4 & 0 & 0 \end{bmatrix}$$

b. For Db4:

$$Y_{4.4} = \begin{bmatrix} 8.3391 & -0.0001 & 11.8033 & -1.9998 \\ -0.0006 & 0 & -0.0006 & 0 \\ 22.1960 & -0.0001 & 25.6602 & -1.9998 \\ -7.9993 & 0 & -7.9993 & 0 \end{bmatrix}$$

Let's take a general 2-D signal, for example any 4×4 matrix, and apply the following steps to compute 2-D FDWT using fast method:

1.Let X be the input 2-D signal,

$$X_{4.4} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

2.For a 4×4 matrix input 2-D signal, X, construct a 4×4 transformation matrix, T.

3.Apply row transformation by [Y]=[T][X][T]^t

a. For Haar:

$$Y_{4.4} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} . \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 11 & -2 & -1 \\ 23 & 27 & -2 & -1 \\ -8 & -8 & 0 & 0 \\ -4 & -4 & 0 & 0 \end{bmatrix}$$

b. For Db4:

$$Y_{4.4} = \begin{bmatrix} 0.4830 & 0.8365 & 0.2241 & -0.1294 \\ -0.1294 & -0.2241 & 0.8365 & -0.4830 \\ 0.2241 & -0.1294 & 0.4830 & 0.8365 \\ 0.8365 & -0.4830 & -0.1294 & -0.2241 \end{bmatrix}. \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}. \begin{bmatrix} 0.4830 & -0.1294 & 0.2241 & 0.8365 \\ 0.8365 & -0.2241 & -0.1294 & -0.4830 \\ 0.2241 & 0.8365 & 0.4830 & -0.1294 \\ -0.1294 & -0.4830 & 0.8365 & -0.2241 \end{bmatrix}$$

$$= \begin{bmatrix} 8.3391 & -0.0001 & 11.8033 & -1.9998 \\ -0.0006 & 0 & -0.0006 & 0 \\ 22.1960 & -0.0001 & 25.6602 & -1.9998 \\ -7.9993 & 0 & -7.9993 & 0 \end{bmatrix}$$

Proposed Kronecker Product Method for Computing 2-D Discrete Wavelet Transform:

As mentioned in the previous section Fast Mahmoud [11] method of computation of 2-D Wavelet transform consists of applying twice the matrix multiplication and in cascade. First, wavelet decomposition is performed on each row of the 2-D signal matrix. Then, wavelet decomposition is performed to each column of the previous result.

As an alternative method to Mahmoud method an investigation of replacing the two cascade steps matrix multiplication by only one step computation. This was achieved through the construction of transform matrix using Kronecker product.

Given an N×N matrix A=[a_{ij}] and an M×N matrix B, then the Kronecker product $A \otimes B$ generate C matrix of NM×NM elements defined as follows:

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1M}B \\ a_{21}B & a_{22}B & \dots & a_{2M}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{N1}B & a_{N2}B & \dots & a_{NM}B \end{bmatrix} \quad \dots\dots (6)$$

Note that $A \otimes B \neq B \otimes A$, where $\otimes$ stand for the Kronecker product [12].

In the proposed method, we will consider the 2-D DWT by using 1-D computation instead of the conventional 2-D computation of 2-D DWT.

Equation (3) requires T.X to be computed first, and then the result will be multiply by T^t. While in our proposed method this can be done by one shot, as shown below:

$$Y_{N^2.1} = T_{N^2.N^2}.X_{N^2.1} \quad \dots\dots (7)$$

Where $T_{N^2.N^2} = T_{N.N} \otimes T_{N.N}$ $\quad \dots\dots (8)$

$X_{N^2.1}$ is the 1-D signal of $X_{N.N}$ after reshape by taking the row wise, and $Y_{N^2.1}$ is the 1-D signal of $Y_{N.N}$.

So, $Y_{N^2.1}$ must be reshaped into $Y_{N.N}$ to get the 2-D DWT.

For example,

Let, $X_{4.4} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{bmatrix}$

Reshape it to $X_{16,1}$=[$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,4}$ $x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,4}$ $x_{4,1}$ $x_{4,2}$ $x_{4,3}$ $x_{4,4}$]^T=[x_0 . . . x_{15}]^T.

Let T be the transformation matrix for DWT type Haar then,

$$T_{16.16} = T_{4.4} \otimes T_{4.4}$$

$Y_{16.1} = T_{16.16}.X_{16.1}$. Finaly, reshape $Y_{16.1}$ to get $Y_{4.4}$.

Illustrative Example:

1. Reshape $X_{4.4}$ to $X_{16,1}$=[1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16]^T

2. Reshape $T_{4.4}$ to $T_{16.16}$, by using the kronecker product:

   a. For Haar:

$$T_{16.16} = T_{4.4} \otimes T_{4.4} =$$

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdots$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & -1 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \end{bmatrix}$$

   b. For Db4:

$$T_{16.16} = T_{4.4} \otimes T_{4.4} =$$

$$
\left[\begin{matrix}
0.2333 & 0.4040 & 0.1082 & -0.0625 \\
-0.0625 & -0.1082 & 0.4040 & -0.2333 \\
0.1082 & -0.0625 & 0.2333 & 0.4040 \\
0.4040 & -0.2333 & -0.0625 & -0.1082 \\
-0.0625 & -0.1082 & -0.0290 & 0.0167 \\
0.0167 & 0.0290 & -0.1082 & 0.0625 \\
-0.0290 & 0.0167 & -0.0625 & -0.1082 \\
-0.1082 & 0.0625 & 0.0167 & 0.0290 \\
0.1082 & 0.1875 & 0.0502 & -0.0290 \\
-0.02900 & -0.0502 & 0.1875 & -0.1082 \\
0.0502 & -0.0290 & 0.1082 & 0.1875 \\
0.1875 & -0.1082 & -0.0290 & -0.0502 \\
0.4040 & 0.6997 & 0.1875 & -0.1082 \\
-0.1082 & -0.1875 & 0.6997 & -0.4040 \\
0.1875 & -0.1082 & 0.4040 & 0.6997 \\
0.6997 & -0.4040 & -0.1082 & -0.1875
\end{matrix}\right.
$$

$$
\begin{matrix}
0.4040 & 0.6997 & 0.1875 & -0.1082 \\
-0.1082 & -0.1875 & 0.6997 & -0.4040 \\
0.1875 & -0.1082 & 0.4040 & 0.6997 \\
0.6997 & -0.4040 & -0.1082 & -0.1875 \\
-0.1082 & -0.1875 & -0.0502 & 0.0290 \\
0.0290 & 0.0502 & -0.1875 & 0.1082 \\
-0.0502 & 0.0290 & -0.1082 & -0.1875 \\
-0.1875 & 0.1082 & 0.0290 & 0.0502 \\
-0.0625 & -0.1082 & -0.0290 & 0.0167 \\
0.0167 & 0.0290 & -0.1082 & 0.0625 \\
-0.0290 & 0.0167 & -0.0625 & -0.1082 \\
-0.1082 & 0.0625 & 0.0167 & 0.0290 \\
-0.2333 & -0.4040 & -0.1082 & 0.0625 \\
0.0625 & 0.1082 & -0.4040 & 0.2333 \\
-0.1082 & 0.0625 & -0.2333 & -0.4040 \\
-0.4040 & 0.2333 & 0.0625 & 0.1082
\end{matrix}
$$

$$
\begin{matrix}
0.1082 & 0.1875 & 0.0502 & -0.0290 \\
-0.0290 & -0.0502 & 0.1875 & -0.1082 \\
0.0502 & -0.0290 & 0.1082 & 0.1875 \\
0.1875 & -0.1082 & -0.0290 & -0.0502 \\
0.4040 & 0.6997 & 0.1875 & -0.1082 \\
-0.1082 & -0.1875 & 0.6997 & -0.4040 \\
0.1875 & -0.1082 & 0.4040 & 0.6997 \\
0.6997 & -0.4040 & -0.1082 & -0.1875 \\
0.2333 & 0.4040 & 0.1082 & -0.0625 \\
-0.0625 & -0.1082 & 0.4040 & -0.2333 \\
0.1082 & -0.0625 & 0.2333 & 0.4040 \\
0.4040 & -0.2333 & -0.0625 & -0.1082 \\
-0.0625 & -0.1082 & -0.0290 & 0.0167 \\
0.0167 & 0.0290 & -0.1082 & 0.0625 \\
-0.0290 & 0.0167 & -0.0625 & -0.1082 \\
-0.1082 & 0.0625 & 0.0167 & 0.0290
\end{matrix}
$$

$$
\left.\begin{matrix}
-0.0625 & -0.1082 & -0.0290 & 0.0167 \\
0.0167 & 0.0290 & -0.1082 & 0.0625 \\
-0.0290 & 0.0167 & -0.0625 & -0.1082 \\
-0.1082 & 0.0625 & 0.0167 & 0.0290 \\
-0.2333 & -0.4040 & -0.1082 & 0.0625 \\
0.0625 & 0.1082 & -0.4040 & 0.2333 \\
-0.1082 & 0.0625 & -0.2333 & -0.4040 \\
-0.4040 & 0.2333 & 0.0625 & 0.1082 \\
0.4040 & 0.6997 & 0.1875 & -0.1082 \\
-0.1082 & -0.1875 & 0.6997 & -0.4040 \\
0.1875 & -0.1082 & 0.4040 & 0.6997 \\
0.6997 & -0.4040 & -0.1082 & -0.1875 \\
-0.1082 & -0.1875 & -0.0502 & 0.0290 \\
0.0290 & 0.0502 & -0.1875 & 0.1082 \\
-0.0502 & 0.0290 & -0.1082 & -0.1875 \\
-0.1875 & 0.1082 & 0.0290 & 0.0502
\end{matrix}\right]
$$

3. $Y_{16.1} = T_{16.16}.X_{16.1}$

a. For Haar:

$$Y_{16.1} = [7\ \ 23\ -4\ -4\ \ 11\ \ 27\ -4\ -4\ -1\ -1\ \ 0\ \ 0\ -1\ -1\ \ 0\ \ 0]^{T}$$

Reshape it we will get:

$$Y_{4.4} = \begin{bmatrix} 7 & 11 & -1 & -1 \\ 23 & 27 & -1 & -1 \\ -4 & -4 & 0 & 0 \\ -4 & -4 & 0 & 0 \end{bmatrix}$$

b. For Db4:

$$Y_{16.1} = [8.3391 - .0006\ 22.1960 - 7.9993\ -0.0001\ 0 - 0.0001\ 0\ 11.8033\ -0.0006\ 25.6602 - 7.9993 - 1.9998\ 0\ -1.9998\ 0]^{T}$$

Reshape it we will get:

$$Y_{4.4} = \begin{bmatrix} 8.3391 & -0.0001 & 11.8033 & -1.9998 \\ -0.0006 & 0 & -0.0006 & 0 \\ 22.1960 & -0.0001 & 25.6602 & -1.9998 \\ -7.9993 & 0 & -7.9993 & 0 \end{bmatrix}$$

Conclusions

This paper facilitates the computation of 2-D DWT and involving a computation procedure consisting mainly of basic arithmetic operation of matrix multiplication and other simple and easy operation like permutations and shuffling.

Some flashing remarks can be concluded after studying the proposed direct computation method which based on Kronecker product:

1. It offers benefits over methods in terms of savings in computation. It is basically the same previous matrix in which only the transformation coefficients are expanded to satisfy the arrangement of the input matrix into a vector. The reason why we have

chosen such an arrangement is that this scattering of the data tends to be less sensitive to coefficient reduction and thus gives better quality of the image. Basically all matrix coefficients have the same local correction and statistical prosperities that the conventional matrix have.

2. The single level multiplication of the Kronecker product is equivalent to two cascaded levels of the conventional method.
3. The new method required a vector-valued input signal, which is a new issue addressed for 2-D transform computation. This conversation is called preprocessing.
4. The transform matrix dimension used in computing 2-D DMWT algorithm should equal $N^2 \times N^2$ for a matrix of $N \times N$ dimension. Having this in mind; the 2-D DWT implementation has to be adapted to work on 16×16 for 4×4 input matrix.

It is possible to generalize this idea to other transforms like FFT, Multiwavelet, Slantlet, Ridglelet, Curvelet and Shearlet.

References

[1] G. Strong, V. Strela, “Short Wavelet s and matrix Dilation Equations”, IEEE Transactions on Acoustic, Speech and Signal Processing, No. 43, PP. 108-115, 1995.

[2] Nath, V.K., Hazarika, D. and Mahauta, A., “A 3-D Block Transform Based Approach To Color Image Compression”, IEEE Transactions on Signal Processing, 2003.

[3] Chang, Y. and Li, Y., “Design of Highly Efficient VLSI Architectures for 2-D DWT and 2-D IDWT”, IEEE, PP. 133-140, 2001.

[4] Vetterli, M. and Minh, D.N., “Orthogonal Finite Ridgelet Transform for Image Compression”, Proc. of IEEE International Conference on Image Processing, 2000.

[5] Gandes, E., Deanet, L., Donoho, D. and Ying, L., “Fast Discrete Curvelet Transforms and Multiscale Model”, Simul. Vol.5, PP. 861-899, 2006.

[6] Do, M.N. and Vetterli, M.,”TheContourlet Transform and efficient Directional Multi Resolution Image Representation”, IEEE Trans. Image Processing Vol.14, PP. 2091-2106, 2005.

[7] Easley, G., Labate, D. and Lim, W., “Sparse Directional Image representations Using the Discrete Shearlet Transform”, Appl. Comput. Harmon. Aal.,Vol 25, PP.25-46, 2008.

[8] M. Mahmoud and M. Dessouky, “Comparison between Haar and Daubechies Wavelet Transformations on FPGA Technology”, World Academy of Science, Engineering, and Technology 26, pp. 68-72, 2007.

[9] K. Wahid and V. Dimitrov, “An Algebraic Integer Based Encoding Scheme for Implementing Daubechies Discrete Wavelet Transforms“, IEEE, pp. 967-971, 2002.

[10] Burrus, S., Gopinath, R. and Guo, H., “Introduction to Wavelets and Wavelet Transforms”, Prentice-Hall, Inc., 1998.

[11] Mahmoud, W. A., Taai, H. and Shnazi, M., “New Fast Method for Computing Wavelet Coefficients from 1-D up to 3-D”, Proceeding of 1st International Conference on Digital Communication and Computer Applications, PP. 313-323, Jordan, 2007.

[12] Y. Jin and H. Koga, “Basic Properties of Complete Complementary Codes Using The DFT Matrices and The Kronecker Product”, International Symposium on Information Theory and its Applications, ISITA, 7-10 December 2008.

الخلاصة

التقدم في تحويل المويجات قد انتج خوارزميات قادرة على تجاوز معالجات الاشارات الرقمية الحالية. في هذا البحث تم تقديم طريقة حساب تحويل مويجات جديدة اعتمدت على طريقة كرونيكير والتي حسنت كثيراً تقليل التعقيد الحسابي.

تم تطوير تحويل سريع للمويجات ثنائي البعد يعتمد على تحسين المصفوفة المتعامدة باستخدام الضرب كرونيكير. الخوارزمية المستخدمة لها ميزات عدة واعدة التي تجعلها مناسبة لتطبيقات معالجة الإشارات ثنائية البعد. هذه المصفوفة حسنت من تنفيذ تحويل المويجات ثنائي البعد بالمقارنة مع الطرق التقليدية.