Al-khwarizmi Engineering Journal

Al-Khwarizmi Engineering Journal, vol.1, no.1,pp 26-37, (2005)

The Determination of Critical-Sampling Scheme of Preprocessing for Multiwavelets Decomposition as 1st and 2nd Orders of Approximations.

W. A. Mahmoud* **Z. J. M. Saleh**** **N. K. Wafi***

Electrical Engineering Dept / College of Engineering /University of Baghdad.*

*Information Engineering Dept ./ Al-Khwarizmi Engineering College / University of Baghdad***

Abstract

One of the important differences between multiwavelets and scalar wavelets is that each channel in the filter bank has a vector-valued input and a vector-valued output. A scalar-valued input signal must somehow be converted into a suitable vector-valued signal. This conversion is called preprocessing. Preprocessing is a mapping process which is done by a prefilter. A postfilter just does the opposite.

The most obvious way to get two input rows from a given signal is to repeat the signal. Two rows go into the multifilter bank. This procedure is called "Repeated Row" which introduces oversampling of the data by a factor of 2.

For data compression, where one is trying to find compact transform representations for a dataset, it is imperative to find critically sampled multiwavelet transforms schemes which this paper focuses on finding a simple and easy to follow algorithm for its computation.

One famous multiwavelet filter used here is the GHM filter proposed by Geronimo, Hardian, and Massopust. The GHM basis offers a combination of orthogonality, symmetry, and compact support, which can not be achieved by any scalar wavelet basis. Using a computer program for the proposed method, an example test on Lena image is verified which shows image properties after a single level decomposition and the reconstructed image after reconstruction.

Keyword: Discrete Multiwavelete Transform (DMWT), Inverse Discrete Multiwavelete Transform (IDMWT), Critical-Sampling, Schema of Processing.

1. Introduction

As multiwavelet filter banks require a vector-valued input signal, there are a number of ways to produce such a signal from 2-D signal image data. Perhaps the most obvious method is to use adjacent rows and columns of the image data [6]. However, this approach does not work well for general multiwavelets and leads to reconstruction artifacts in the lowpass data after coefficient quantization [6]. This problem can be avoided by constructing "constrained" multiwavelets, which possess certain key properties. Unfortunately, the extra constraints are somewhat restrictive; image compression tests show that constrained multiwavelets do not perform as well as some other multifilters [2]. Another approach is to first split each row or column into two half-length signals, and then use these two half signals as the channel inputs into the multifilter. A naive approach, as Strela points out [6], is to simply take the odd samples for one signal and the even samples for the second signal

. 2. Multiwavelets Theory

As in the scalar wavelet case, the theory of multiwvelets is based on the idea of multiresolution analysis (MRA), analyzing the signal at different scales or resolutions. The difference is that multiwavelets have several scaling functions. The standard multiresolution has one scaling function $\phi(t)$ [2].

For notational convenience, the set of scaling functions can be written using the vector notation $\Phi(t)\equiv[\phi_1(t), \phi_2(t)\ldots \phi_r(t)]^T$,where $\Phi(t)$ is called the multiscaling function. Likewise, the multiwavelet function is defined from the set of wavelet functions as $\Psi(t)\equiv[\psi_1(t), \psi_2(t)\ldots \psi_r(t)]^T$. When r=1, $\Psi(t)$ is called a *scalar* wavelet, or simply wavelet. While in principle r can be arbitrarily large. The multiwavelets studied to date are primarily for r=2 [7].

The multiwavelet two-scale equations resemble those for scalar wavelets:

$$\Phi(t) = \sqrt{2}\sum_{k=-\infty}^{\infty} H_k \Phi(2t-k) \qquad (1)$$

$$\Psi(t) = \sqrt{2}\sum_{k=-\infty}^{\infty} G_k \Phi(2t-k) \qquad (2)$$

Note, however, that $\{H_k\}$ and $\{G_k\}$ are *matrix* filters, i.e., H_k and G_k are $r\times r$ matrices for each integer k. The matrix elements in these filters provide more degrees of freedom than a traditional scalar wavelet. These extra degrees of freedom can be used to incorporate useful properties into the multiwavelet filters, such as orthogonality, symmetry, and high order of approximation. The key, then, is to figure out how to make the best use of these extra degrees of freedom. Multifilter construction methods are already being developed to exploit them. However, the multi-channel nature of multiwavelets also means that the sub-band structure resulting from passing a signal through a multifilter bank is different. Sufficiently different, in fact, so that established quantization methods do not perform as well with multiwavelets as they do with wavelets [2].

One famous multiwavelet filter is the GHM filter proposed by Geronimo, Hardian, and Massopust [3]. The GHM basis offers a combination of orthogonality, symmetry, and compact support, which can not be achieved by any scalar wavelet. Eqs.(1) and (2) th[illegible] wavelet functions satisfy the following two-scale dilation equations:

$$\begin{bmatrix}\phi_1(t)\\ \phi_2(t)\end{bmatrix} = \sqrt{2}\sum_k H_k \begin{bmatrix}\phi_1(2t-k)\\ \phi_2(2t-k)\end{bmatrix} \qquad (3)$$

$$\begin{bmatrix}\psi_1(t)\\ \psi_2(t)\end{bmatrix} = \sqrt{2}\sum_k G_k \begin{bmatrix}\phi_1(2t-k)\\ \phi_2(2t-k)\end{bmatrix} \qquad (4)$$

where H_k for GHM system are four scaling matrices H_0, H_1, H_2, and H_3, [9],

$$H_0 = \begin{bmatrix}\frac{3}{5\sqrt{2}} & \frac{4}{5}\\ -\frac{1}{20} & -\frac{3}{10\sqrt{2}}\end{bmatrix},\quad H_1 = \begin{bmatrix}\frac{3}{5\sqrt{2}} & 0\\ \frac{9}{20} & \frac{1}{\sqrt{2}}\end{bmatrix},$$
$$H_2 = \begin{bmatrix}0 & 0\\ \frac{9}{20} & -\frac{3}{10\sqrt{2}}\end{bmatrix},\quad H_3 = \begin{bmatrix}0 & 0\\ -\frac{1}{20} & 0\end{bmatrix} \qquad (5)$$

also, G_k for GHM system are four wavelet matrices G_0, G_1, G_2, and G_3, [9],

$$G_0 = \begin{bmatrix}-\frac{1}{20} & -\frac{3}{10\sqrt{2}}\\ \frac{1}{10\sqrt{2}} & \frac{3}{10}\end{bmatrix},\quad G_1 = \begin{bmatrix}\frac{9}{20} & -\frac{1}{\sqrt{2}}\\ -\frac{9}{10\sqrt{2}} & 0\end{bmatrix}$$
$$G_2 = \begin{bmatrix}\frac{9}{20} & -\frac{3}{10\sqrt{2}}\\ \frac{9}{10\sqrt{2}} & -\frac{3}{10}\end{bmatrix},\quad G_3 = \begin{bmatrix}-\frac{1}{20} & 0\\ -\frac{1}{10\sqrt{2}} & 0\end{bmatrix} \qquad (6)$$

There are four remarkable properties of the Geronomo-Hardin-Massopust scaling functions, as follows [2]:

- They each have short support (the intervals [0,1] and [0,2]).
- Both scaling functions are symmetric, and the wavelets form a symmetric/antisymmetric pair.
- All integers translates of the scaling functions are orthogonal.
- The system has second order of approximation.

While the very first multiwavelet literature goes back further [7], some of the earliest developed multiresolution theory of multiwavelets can be found in a paper by Goodman et al. [10]. Strela's in his Ph.D. thesis [6] extends the theory of multiwavelets even further and presents it in terms of PR multifilter banks in both the time and frequency domains.

The 2×2 matrix filters in our [illegible] require vector inputs. [illegible]ast be transformed

into two 1-D signals. This transformation is called pre-processing. For some multiwavelets, the pre-processing must be accompanied by an appropriate pre-filtering operation that depends on the spectral characteristics of the multiwavelet filters [11]. However, some multiwavelets obviate the pre-filtering (and the pre-processing) operation due to certain desirable properties of their basis functions; these multiwavelets are called balanced multiwavelets [1].

3. A Critically-Sampled Scheme of Preprocessing: Approximation-Based Preprocessing

A different way to get input rows for the multiwavelet filter bank is to preprocess the given scalar signal f [n]. For data compression, where one is trying to find compact transform representations for a dataset, it is imperative to find critically sampled multiwavelet transforms schemes [3].

A preprocessing algorithm based on the approximation properties of the continuous-time multiwavelets, which yields a critically sampled signal representation suggested by J. Geronimo and developed by V. Strela, P. Niels, and G. Strang [2].

Let the continuous-time function $f(t)$ belong to the scale-limited subspace V_0 generated by translates of the GHM scaling functions. This means that $f(t)$ is a linear combination of translates of those functions [2]:

$$f(t)=\sum_n v_{1,n}^{(0)}\phi_1(t-n)+v_{2,n}^{(0)}\phi_2(t-n) \quad (7)$$

Suppose that the input sequence f [n] contains samples of $f(t)$ at half-integers:

$$f[2n]=f(n), \qquad f[2n+1]=f(n+1/2). \quad (8)$$

$\phi_1(t)$ vanishes at all integer points. $\phi_2(t)$ is nonzero only at the integer 1. Sampling the eq. (8) at integers and half-integers gives:

$$f[2n]=\phi_2(1)v_{2,n-1}^{(0)} \quad (9)$$

$$f[2n+1]=\phi_2(3/2)v_{2,n-1}^{(0)}+\phi_1(1/2)v_{1,n}^{(0)}+\phi_2(1/2)v_{2,n}^{(0)}$$

The coefficients $v_{1,n}^{(0)}, v_{2,n}^{(0)}$ can be easily found from (9):

$$v_{1,n}^{(0)}=\frac{\phi_2(1)\,f[2n+1]-\phi_2(1/2)\,f[2n+2]-\phi_2(3/2)\,f[2n]}{\phi_2(1)\phi_1(1/2)}$$

$$v_{2,n}^{(0)}=\frac{f[2n+2]}{\phi_2(1)}$$

Taking into account t

$$v_{1,n}^{(0)}=\frac{\phi_2(1)\,f[2n+1]-\phi_2(1/2)(f[2n+2]+f[2n])}{\phi_2(1)\phi_1(1/2)}$$

$$v_{2,n}^{(0)}=\frac{f[2n+2]}{\phi_2(1)} \quad (11)$$

The eq. (11) give a natural way to get two input rows $v_{1,n}^{(0)}, v_{2,n}^{(0)}$ starting from a given signal f [n]. To synthesize the signal on output, invert eq.(11) and recover eq.(9) [2].

4. Multiwavelets Transform Computation: Basic Principles

For computing Discrete Multiwavelet Transform, a transform matrix can be written as follows [12]:

$$\begin{bmatrix} H_0 & H_1 & H_2 & H_3 & 0 & 0 & \dots \\ G_0 & G_1 & G_2 & G_3 & 0 & 0 & \dots \\ 0 & 0 & H_0 & H_1 & H_2 & H_3 & \dots \\ 0 & 0 & G_0 & G_1 & G_2 & G_3 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (12)$$

where H_i and G_i are the low- and high-pass filter impulse responses. They are 2-by-2 matrices which can be written as follows:

$$\begin{bmatrix} H_{0_{0,0}} & H_{0_{0,1}} & H_{1_{0,0}} & H_{1_{0,1}} & \dots & \dots & \dots & \dots & \dots \\ H_{0_{1,0}} & H_{0_{1,1}} & H_{1_{1,0}} & H_{1_{1,1}} & \dots & \dots & \dots & \dots & \dots \\ G_{0_{0,0}} & G_{0_{0,1}} & G_{1_{0,0}} & G_{1_{0,1}} & \dots & \dots & \dots & \dots & \dots \\ G_{0_{1,0}} & G_{0_{1,1}} & G_{1_{1,0}} & G_{1_{1,1}} & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & H_{0_{0,0}} & H_{0_{0,1}} & H_{1_{0,0}} & H_{1_{0,1}} & \dots \\ 0 & 0 & 0 & 0 & H_{0_{1,0}} & H_{0_{1,1}} & H_{1_{1,0}} & H_{1_{1,1}} & \dots \\ 0 & 0 & 0 & 0 & G_{0_{0,0}} & G_{0_{0,1}} & G_{1_{0,0}} & G_{1_{0,1}} & \dots \\ 0 & 0 & 0 & 0 & G_{0_{1,0}} & G_{0_{1,1}} & G_{1_{1,0}} & G_{1_{1,1}} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (13)$$

By examining the transform matrices of the DAUB4 scalar wavelet [13] and the corresponding one of multiwavelets as shown in (12), one can see that in the multiwavelets transform domain there are first and second low-pass coefficients followed by first and second high pass filter coefficients rather than one low-pass coefficient followed by one highpass coefficient. Therefore, if we separate these four coefficients, there are four subbands in the transform domain [14].

5. A General Procedure for Computing DMWT Using a Critically-Sampled Scheme of Preprocessing

A general procedure can be made for computing a single-level 2-D discrete multiwavelets transform using GHM four multifilters and using a critically-sampled scheme of preprocessing (approximation-based scheme of preprocessing) described in sec.3.

By using a critically-sampled scheme of preprocessing (approximation-based scheme of preprocessing), the DMWT matrix has the same dimensions of the input which should be a square matrix N×N where N must be power of 2. Transformation matrix dimensions which should be equal to image dimensions after preprocessing will be N×N for a critical-sampled scheme of preprocessing.

There are two orders of approximation types of critically-sampled preprocessing 1st order and 2nd order approximations. For the eq.(10) and using GHM scaling function graph (Fig. 1a), values for $\phi_1(1/2)$, $\phi_2(1/2)$, $\phi_2(1)$ and $\phi_2(3/2)$ should be found for first order approximation. For any N×N image matrix and using the eq. (10), 1st order approximation-based preprocessing can be summarized as follows where every two rows generate two new rows:

a- For any odd row,

$$\text{new odd - row} = \frac{1}{\phi_2(1)\phi_1(1/2)}(\phi_2(1)[\text{same odd - row}] - \phi_2(1/2)[\text{next even - row}] - \phi_2(3/2)[\text{previous even - row}]) \quad (14)$$

b-For any even-row,

$$\text{new even - row} = \frac{\text{same even - row}}{\phi_2(1)} \quad (15)$$

It can be seen from Fig. 1a that the values of $\phi_1(t)$ and $\phi_2(t)$ are non-zero for t values of [0, 2]. Since these functions are generated from a 256 sample then:

1. $\phi_1(1/2)$ = the 64th value in the iterated vector of ϕ_1,
2. $\phi_2(1/2)$ = the 64th value in the iterated vector of $\phi_2 = \phi_2(3/2)$,
3. $\phi_2(1)$ = the 128th value in the iterated vector of ϕ_2.

substituting values of $\phi_1(1/2)$, $\phi_2(1)$ and $\phi_2(1/2)$ in Eqs.(14) and (15) for 1st order approximation results,

$$\text{new odd - row} = (0.373615)[\text{same odd - row}] + (0.1108619)[\text{next even - row}] + (0.1108619)[\text{previous even - row}] \quad (16)$$

$$\text{new even - row} = (\sqrt{2} - 1)[\text{same even - row}] \quad (17)$$

for 2nd order approximation, Eqs. (16) and (17) become,

$$\text{new odd - row} = (10/8\sqrt{2})[\text{same odd - row}] + (3/8\sqrt{8})[\text{next even - row}] + (3/8\sqrt{2})[\text{previous even - row}] \quad (18)$$

$$\text{new even - row} = [\text{same even - row}] \quad (19)$$

It should be noted that when computing the first odd row, the previous even-row in eq. (16) is equals to zero. In the same manner, when computing the last odd row, the next even-row in Eq. (16) is equals to zero. The same thing is valid for eq. (18).

It is obvious now why the dimension of the resulting matrix after approximation-based preprocessing has the same dimension as before preprocessing.

The following procedure for computing DMWT using approximation-based preprocessing is valid for both 1st and 2nd order of approximation with one exception of using Eqs.(16) and (17) for 1st order approximation preprocessing step and Eqs. (18) and (19) for 2nd order approximations preprocessing step:

1. *Checking image dimensions*: Image matrix should be a square matrix, N×N matrix, where N must be power of 2. So checking input image dimensions is the first step of the transform procedure. If the image is not a square matrix some operation must be done to the image like resizing the image or adding rows or column of zeros to get a square matrix.
2. *Constructing a transformation matrix*: Using the transformation matrix (12) format, an N/2×N/2 transformation matrix should be constructed using GHM low- and high-pass filters matrices given in (5) and (6) substituting GHM values as given by

(13), an N×N transformation matrix results with same dimensions of input image dimensions after preprocessing.

3.*Preprocessing rows*: Approximation-based row preprocessing can be computed by applying Eqs. (16) and (17) to the odd- and even-rows of the input N×N matrix respectively for the 1st order approximation preprocessing. For 2nd order approximation preprocessing, Eqs. (16) and (17) are replaced with Eqs. (18) and (19) for preprocessing odd- and even-rows of the input N×N matrix respectively. Input matrix dimensions after row preprocessing is the same N×N.

4. *Transformation of image rows*:

i. Apply matrix multiplication to the N×N constructed transformation matrix by the N×N row preprocessed input image matrix.

ii. Permute the resulting N×N matrix rows by arranging the row pairs 1,2 and 5,6 …, N–3, N–2 after each other at the upper half of the resulting matrix rows, then the row pairs 3,4 and 7,8,…, N–1,N below them at the next lower half.

5.*Preprocess columns*: to repeat the same procedure used in preprocessing rows,

i. Transpose the row transformed N×N matrix resulting from step 4.

ii. Repeat step 3 to the N×N matrix (transpose of the row transformed N×N matrix) which results in N×N column preprocessed matrix.

6.*Transformation of image columns* : transformation of image columns is applied next to N×N column preprocessed matrix as follows:

i. Apply matrix multiplication to the N×N constructed transformation matrix by the N×N column preprocessed matrix.

ii. Permute the resulting N×N matrix rows by arranging the row pairs 1,2 and 5,6 …, N–3, N–2 after each other at the upper half of the resulting matrix rows, then the row pairs 3,4 and 7,8,…, N–1, N below them at the next lower half.

7.*The Final Transformed Matrix*: to get the final transformed matrix:

i. Transpose the resulting matrix from column transformation step.

ii. Apply coefficients permutation [15] to the resulting transpose matrix. The final DMWT matrix using approximation-based preprocessing has the same dimensions, N×N, of the original image matrix.

6. A General Example for Computing DMWT Using a Critically-Sampled Scheme of Preprocessing

To verify the general procedure for computing single-level DMWT using critically-sampled scheme of preprocessing, let's take a general 2-D signal, for example any 8×8 matrix, and apply the following steps:

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

$$X=\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} & x_{0,5} & x_{0,6} & x_{0,7} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} & x_{1,7} \\ x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} & x_{2,7} \\ x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} & x_{3,6} & x_{3,7} \\ x_{4,0} & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} & x_{4,6} & x_{4,7} \\ x_{5,0} & x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} & x_{5,6} & x_{5,7} \\ x_{6,0} & x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} & x_{6,6} & x_{6,7} \\ x_{7,0} & x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} & x_{7,6} & x_{7,7} \end{bmatrix} \quad (20)$$

2. For an 8×8 matrix input 2-D signal, X, construct a 4×4 (N/2×N/2) transformation matrix ,W_2, using GHM low- and high-pass filters,

$$W_2=\begin{bmatrix} H_0 & H_1 & H_2 & H_3 \\ G_0 & G_1 & G_2 & G_3 \\ H_2 & H_3 & H_0 & H_1 \\ G_2 & G_3 & G_0 & G_1 \end{bmatrix} \quad (21)$$

As GHM filters, H's and G's, are 2×2 matrices, the transformation matrix, W_2, dimension after substituting filters coefficients values will be 8×8 (N×N) matrix with same dimension of the input matrix after preprocessing.

3. Apply row preprocessing to the input 2-D matrix, X, using approximation-based preprocessing which results in a matrix,

$$X=\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} & x_{0,5} & x_{0,6} & x_{0,7} \\ x_{1,0} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} & x_{1,7} \\ x_{2,0} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} & x_{2,7} \\ x_{3,0} & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} & x_{3,6} & x_{3,7} \\ x_{4,0} & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} & x_{4,6} & x_{4,7} \\ x_{5,0} & x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} & x_{5,6} & x_{5,7} \\ x_{6,0} & x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} & x_{6,6} & x_{6,7} \\ x_{7,0} & x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} & x_{7,6} & x_{7,7} \end{bmatrix} \begin{matrix} \text{odd} \\ \text{even} \\ \text{odd} \\ \text{even} \\ \text{odd} \\ \text{even} \\ \text{odd} \\ \text{even} \end{matrix} \xrightarrow{\text{preprocess rows}} a=\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4} & a_{0,5} & a_{0,6} & a_{0,7} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} \\ a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} \\ a_{5,0} & a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} \\ a_{6,0} & a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} \\ a_{7,0} & a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} \end{bmatrix} \begin{matrix} \text{odd} \\ \text{even} \\ \text{odd} \\ \text{even} \\ \text{odd} \\ \text{even} \\ \text{odd} \\ \text{even} \end{matrix} \quad (22)$$

i- *1st order approximation-based preprocessing*, any odd row in the approximation preprocessed [a] matrix can be found from its corresponding odd row of [X] matrix with the even row previous and next to it in the [X] matrix. In the same manner any even row in [a] matrix can be found from its corresponding even row of [X] matrix. Using Eqs.,(16) and (17), this can be done as follows,

$$a_{\text{odd - row}} = (0.373615)[X_{\text{same odd - row}}] + (0.11086198)[X_{\text{next even - row}}] + (0.11086198)[X_{\text{previous even - row}}] \quad (23)$$

$$a_{\text{even-row}} = (\sqrt{2}-1)[X_{\text{same even-row}}] \quad (24)$$

ii- *2nd order approximation-based preprocessing*, any odd row in the approximation preprocessed [a] matrix can be found from its corresponding odd row of [X] matrix with the even row previous and next to it in the [X] matrix. In the same manner any even row in [a] matrix can be found from its corresponding even row of [X] matrix. Using Eqs.,(18) and (19), this can be done as follows,

$$a_{\text{odd-row}} = (10/8\sqrt{2})[X_{\text{same odd-row}}] + (3/8\sqrt{2})[X_{\text{next even-row}}] + (3/8\sqrt{2})[X_{\text{previous even-row}}] \quad (25)$$

$$a_{\text{even-row}} = [X_{\text{same even-row}}] \quad (26)$$

4. Row transformation is performed as follows,

i. let

ii. pe

$$z=\begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} & z_{0,3} & z_{0,4} & z_{0,5} & z_{0,6} & z_{0,7} \\ z_{1,0} & z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} & z_{1,5} & z_{1,6} & z_{1,7} \\ z_{2,0} & z_{2,1} & z_{2,2} & z_{2,3} & z_{2,4} & z_{2,5} & z_{2,6} & z_{2,7} \\ z_{3,0} & z_{3,1} & z_{3,2} & z_{3,3} & z_{3,4} & z_{3,5} & z_{3,6} & z_{3,7} \\ z_{4,0} & z_{4,1} & z_{4,2} & z_{4,3} & z_{4,4} & z_{4,5} & z_{4,6} & z_{4,7} \\ z_{5,0} & z_{5,1} & z_{5,2} & z_{5,3} & z_{5,4} & z_{5,5} & z_{5,6} & z_{5,7} \\ z_{6,0} & z_{6,1} & z_{6,2} & z_{6,3} & z_{6,4} & z_{6,5} & z_{6,6} & z_{6,7} \\ z_{7,0} & z_{7,1} & z_{7,2} & z_{7,3} & z_{7,4} & z_{7,5} & z_{7,6} & z_{7,7} \end{bmatrix} \xrightarrow{\text{Permute}} p=\begin{bmatrix} z_{0,0} & z_{0,1} & z_{0,2} & z_{0,3} & z_{0,4} & z_{0,5} & z_{0,6} & z_{0,7} \\ z_{1,0} & z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} & z_{1,5} & z_{1,6} & z_{1,7} \\ z_{4,0} & z_{4,1} & z_{4,2} & z_{4,3} & z_{4,4} & z_{4,5} & z_{4,6} & z_{4,7} \\ z_{5,0} & z_{5,1} & z_{5,2} & z_{5,3} & z_{5,4} & z_{5,5} & z_{5,6} & z_{5,7} \\ z_{2,0} & z_{2,1} & z_{2,2} & z_{2,3} & z_{2,4} & z_{2,5} & z_{2,6} & z_{2,7} \\ z_{3,0} & z_{3,1} & z_{3,2} & z_{3,3} & z_{3,4} & z_{3,5} & z_{3,6} & z_{3,7} \\ z_{6,0} & z_{6,1} & z_{6,2} & z_{6,3} & z_{6,4} & z_{6,5} & z_{6,6} & z_{6,7} \\ z_{7,0} & z_{7,1} & z_{7,2} & z_{7,3} & z_{7,4} & z_{7,5} & z_{7,6} & z_{7,7} \end{bmatrix} \quad (28)$$

5. Apply column transformation,

i. transpose [p] matrix.

$$p^{t}=\begin{bmatrix} z_{0,0} & z_{1,0} & z_{4,0} & z_{5,0} & z_{2,0} & z_{3,0} & z_{6,0} & z_{7,0} \\ z_{0,1} & z_{1,1} & z_{4,1} & z_{5,1} & z_{2,1} & z_{3,1} & z_{6,1} & z_{7,1} \\ z_{0,2} & z_{1,2} & z_{4,2} & z_{5,2} & z_{2,2} & z_{3,2} & z_{6,2} & z_{7,2} \\ z_{0,3} & z_{1,3} & z_{4,3} & z_{5,3} & z_{2,3} & z_{3,3} & z_{6,3} & z_{7,3} \\ z_{0,4} & z_{1,4} & z_{4,4} & z_{5,4} & z_{2,4} & z_{3,4} & z_{6,4} & z_{7,4} \\ z_{0,5} & z_{1,5} & z_{4,5} & z_{5,5} & z_{2,5} & z_{3,5} & z_{6,5} & z_{7,5} \\ z_{0,6} & z_{1,6} & z_{4,6} & z_{5,6} & z_{2,6} & z_{3,6} & z_{6,6} & z_{7,6} \\ z_{0,7} & z_{1,7} & z_{4,7} & z_{5,7} & z_{2,7} & z_{3,7} & z_{6,7} & z_{7,7} \end{bmatrix} \quad (29)$$

ii. preprocess $[p]^{t}$ in the same manner of preprocessing described in step 3 (***i*** and ***ii***) above to get [P] matrix.

iii. let $[b] = [W_2]\times[P]$ (30)

iv. permute [b] to get [B] matrix which is 8×8 matrix also.

$$b=\begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} & b_{0,4} & b_{0,5} & b_{0,6} & b_{0,7} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} & b_{1,6} & b_{1,7} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} & b_{2,6} & b_{2,7} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} & b_{3,6} & b_{3,7} \\ b_{4,0} & b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} & b_{4,5} & b_{4,6} & b_{4,7} \\ b_{5,0} & b_{5,1} & b_{5,2} & b_{5,3} & b_{5,4} & b_{5,5} & b_{5,6} & b_{5,7} \\ b_{6,0} & b_{6,1} & b_{6,2} & b_{6,3} & b_{6,4} & b_{6,5} & b_{6,6} & b_{6,7} \\ b_{7,0} & b_{7,1} & b_{7,2} & b_{7,3} & b_{7,4} & b_{7,5} & b_{7,6} & b_{7,7} \end{bmatrix} \xrightarrow{\text{Permute}} B=\begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} & b_{0,4} & b_{0,5} & b_{0,6} & b_{0,7} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} & b_{1,6} & b_{1,7} \\ b_{4,0} & b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} & b_{4,5} & b_{4,6} & b_{4,7} \\ b_{5,0} & b_{5,1} & b_{5,2} & b_{5,3} & b_{5,4} & b_{5,5} & b_{5,6} & b_{5,7} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} & b_{2,6} & b_{2,7} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} & b_{3,6} & b_{3,7} \\ b_{6,0} & b_{6,1} & b_{6,2} & b_{6,3} & b_{6,4} & b_{6,5} & b_{6,6} & b_{6,7} \\ b_{7,0} & b_{7,1} & b_{7,2} & b_{7,3} & b_{7,4} & b_{7,5} & b_{7,6} & b_{7,7} \end{bmatrix} \quad (31)$$

6. The final DMWT matrix [Y]results from apply the following:

i. transpose [B] matrix to get [y] matrix

ii. apply coefficients permutation to each of the four basic subbands of matrix [y] to get the final DMWT matrix [Y].

$$[y]=\left[\begin{array}{cccc|cccc} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,3} & y_{0,4} & y_{0,5} & y_{0,6} & y_{0,7} \\ y_{1,0} & y_{1,1} & y_{1,2} & y_{1,3} & y_{1,4} & y_{1,5} & y_{1,6} & y_{1,7} \\ y_{2,0} & y_{2,1} & y_{2,2} & y_{2,3} & y_{2,4} & y_{2,5} & y_{2,6} & y_{2,7} \\ y_{3,0} & y_{3,1} & y_{3,2} & y_{3,3} & y_{3,4} & y_{3,5} & y_{3,6} & y_{3,7} \\ \hline y_{4,0} & y_{4,1} & y_{4,2} & y_{4,3} & y_{4,4} & y_{4,5} & y_{4,6} & y_{4,7} \\ y_{5,0} & y_{5,1} & y_{5,2} & y_{5,3} & y_{5,4} & y_{5,5} & y_{5,6} & y_{5,7} \\ & & & & & & y_{6,5} & y_{6,6} & y_{6,7} \\ & & & & & & y_{7,5} & y_{7,6} & y_{7,7} \end{array}\right]$$

Coff. Perm.

$$[Y]=\left[\begin{array}{cccc|cccc} y_{0,0} & y_{0,2} & y_{0,1} & y_{0,3} & y_{0,4} & y_{0,6} & y_{0,5} & y_{0,7} \\ y_{2,0} & y_{2,2} & y_{2,1} & y_{2,3} & y_{2,4} & y_{2,6} & y_{2,5} & y_{2,7} \\ y_{1,0} & y_{1,2} & y_{1,1} & y_{1,3} & y_{1,4} & y_{1,6} & y_{1,5} & y_{1,7} \\ y_{3,0} & y_{3,2} & y_{3,1} & y_{3,3} & y_{3,4} & y_{3,6} & y_{3,5} & y_{3,7} \\ \hline y_{4,0} & y_{4,2} & y_{4,1} & y_{4,3} & y_{4,4} & y_{4,6} & y_{4,5} & y_{4,7} \\ y_{6,0} & y_{6,2} & y_{6,1} & y_{6,3} & y_{6,4} & y_{6,6} & y_{6,5} & y_{6,7} \\ y_{5,0} & y_{5,2} & y_{5,1} & y_{5,3} & y_{5,4} & y_{5,6} & y_{5,5} & y_{5,7} \\ y_{7,0} & y_{7,2} & y_{7,1} & y_{7,3} & y_{7,4} & y_{7,6} & y_{7,5} & y_{7,7} \end{array}\right] \quad (32)$$

7. A Computer Test

Two general computer programs computing a single-level DMWT using a critical-sampled scheme of preprocessing (1st and 2nd order approximation) are written using MATLAB v.6.5 for a general N×N 2-D signal (or image). An example test is applied to "Lena" image by using this computer program of the proposed method for computing discrete multiwavelets transform using a critical-sampled scheme of preprocessing and the results are shown in Fig.2 and Fig.3for the 1st and 2nd order approximations respectively.

As shown in both Figs.2 and 3, the original "Lena" image, Fig. 2a, and 3a dimensions are 512×512 (N×N). After a single-level of multiwavelets decomposition using a critical-sampled scheme of preprocessing, image dimensions will be a matrix of 512×512 (N×N) as shown in Figs. 2b and 3b. The upper-left most, L_1L_1, subband of 128×128 dimension, is zoomed in as in Figs. 2c and 3c.

8. A General Procedure for Computing Inverse DMWT Using a Critically-Sampled Scheme of Postprocessing

To reconstruct the original 2-D signal (N×N matrix) from the discrete multiwavelets transformed 2-D signal the Inverse Discrete Multiwavelets Transform (IDMWT) should be used.

A general procedure can be followed for computing a single-level 2-D discrete multiwavelets inverse transform using GHM four multifilters and using a critically-sampled scheme of postprocessing (approximation-based scheme of postprocessing).

By using a critically-sampled scheme of preprocessing (appr… preprocessing), the I… dimensions of the input which should be a square matrix N×N where N must be power of 2. So, to reconstruct the original N×N matrix, a reconstruction matrix, which is the inverse (or transpose) of transformation matrix given (12), dimensions should be equal to critical-sampled preprocessed DMWT N×N matrix dimensions.

As there are two orders of approximation types of critically-sampled preprocessing, 1st order and 2nd order approximations, there are correspondingly two types of critically-sampled postprocessing methods that should be followed; one for each order of approximations.

To compute a single-level 2-D Inverse Discrete Multiwavelets Transform using critically-sampled scheme of postprocessing, the next steps should be followed:

1. *Coefficients Shuffling* [15], which is applied to the DMWT N×N matrix four basic subbands individually. For each subband, coefficients shuffling, shuffles columns first then rows.
2. *Column reconstruction*,
   - ***i.*** Transpose the coefficients shuffled N×N matrix.
   - ***ii.*** Apply shuffling by arranging the row pairs 1,2 and 3,4,…,(N/2)−1,N/2 of the coefficients shuffled N×N matrix transpose to be the row pairs 1,2 and 5,6,…, N−3, N−2 of the resulting matrix and arranging the row pairs (N/2)+1,(N/2)+2 and (N/2)+3,(N/2)+4,…, N−1,N of the coefficients shuffled 2N×2N matrix transpose to be the row pairs 3,4 and 7,8,…, N−1, N of the resulting matrix.
   - ***iii.*** Multiply an N×N reconstruction matrix (N×N transformation matrix (12) transpose) with the resulting N×N shuffled matrix from ***ii***.
3. *Postprocessing*, a critical-sampled scheme of postprocessing can be computed as follows:
   - ***i.*** 1st order approximation postprocessing: can be computed by applying the

$$\text{odd-row} = [[\text{same odd-row}] - (0.11086198)[\text{next even-row}] - (0.11086198)[\text{previous even-row}]]/(0.373615) \quad (33)$$

$$\text{even-row} = [\text{same even-row}]/(\sqrt{2}-1) \quad (34)$$

to the odd- and even-rows of the column reconstructed N×N matrix respectively.

ii. 2nd order approximation postprocessing: can be computed by applying the equations:

$$\text{odd-row} = [[\text{same odd-row}] - (3/8\sqrt{8})[\text{next even-row}] - (3/8\sqrt{2})[\text{previous even-row}]]/(10/8\sqrt{2}) \quad (35)$$

$$\text{even-row} = [\text{same even-row}] \quad (36)$$

to the odd- and even-rows of the column reconstructed N×N matrix respectively.

4. *Row reconstruction*

i. Transpose the postprocessed N×N resultant matrix.

ii. Apply shuffling by arranging the row pairs 1,2 and 3,4,…,(N/2)−1,N/2 of the N×N postprocessed resultant matrix transpose to be the row pairs 1,2 and 5,6,…, N−3, N−2 of the resulting matrix and arranging the row pairs (N/2)+1, (N/2)+2 and (N/2)+3, (N/2)+4,…, N−1, N of the N×N postprocessed resultant matrix transpose to be the row pairs 3,4 and 7,8,…, N−1, N of the resulting matrix.

iii. Multiply a N×N reconstruction matrix (N×N transformation matrix (12) transpose) with the resulting N×N shuffled matrix from ***ii***

5. *Postprocessing*, a critical-sampled scheme of postprocessing can be done by the same process of step 3 above which results in the N×N original reconstructed 2-D signal matrix.

9. A General Example for Computing Inverse DMWT Using a Critically-Sampled Scheme of Postprocessing

To verify IDMWT procedure in the previous section, apply it to the 8×8 matrix, [*Y*], given in (32) as the N×N critically-sampled preprocessed DMWT matrix to reconstruct 8×8 matrix, [*X*], given in (20) as the N×N original 2-D signal matrix as follows:

1. Apply coefficients shuffling to each subband of [*Y*] matrix of (32) which results in [*y*] matrix of (32).

2. Column reconstruction applied now to [*y*] matrix of (32),

i. transpose [*y*] to get [*B*] matrix given in (31).

ii. apply shuffling to [*B*] matrix given in (31) to have [*b*] matrix of (31) as a result of shuffling.

iii. using [W_2] matrix given in (21),

$$[P] = [W_2]^t \times [b] \quad (37)$$

3. Postprocessing [*P*] matrix results in $[p]^t$ given in (29).

4. Row reconstruction applied on [*p*]′ matrix,

i. transpose $[p]^t$ matrix of (29) which results in [*p*] matrix of (28).

ii. apply shuffling to [*p*] matrix given in (28) to get [*z*] matrix of (28) as a result of shuffling.

iii. using [W_2] matrix given in (21),

$$[a] = [W_2]^t \times [z] \quad (38)$$

[*a*] is given in (22).

5. Critically-sampled postprocessing [*a*] matrix results in [*X*] of (20) which is the original reconstructed 2-D signal.

10. A Computer Test

Two general computer programs computing a single-level IDMWT using a critical-sampled scheme of postprocessing (1st and 2nd postprocessing) are written using MATLAB v.6.5 for a general 2N×2N 2-D decomposed image.

An example test is applied to the decomposed Lena image shown in Figs.2b and 3b to a" image by using of the proposed

method of computing inverse discrete multiwavelets transform using 1st and 2nd order approximations postprocessing respectively and the results is shown in Fig.4.

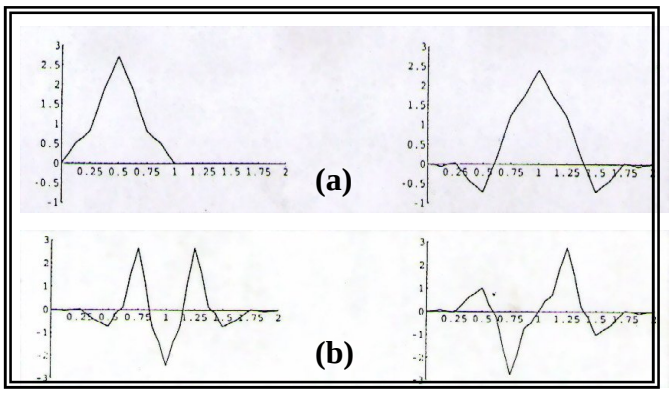


Fig. 1: ***GHM Pair of , (a) Scaling Functions, (b) Multiwavelets.***

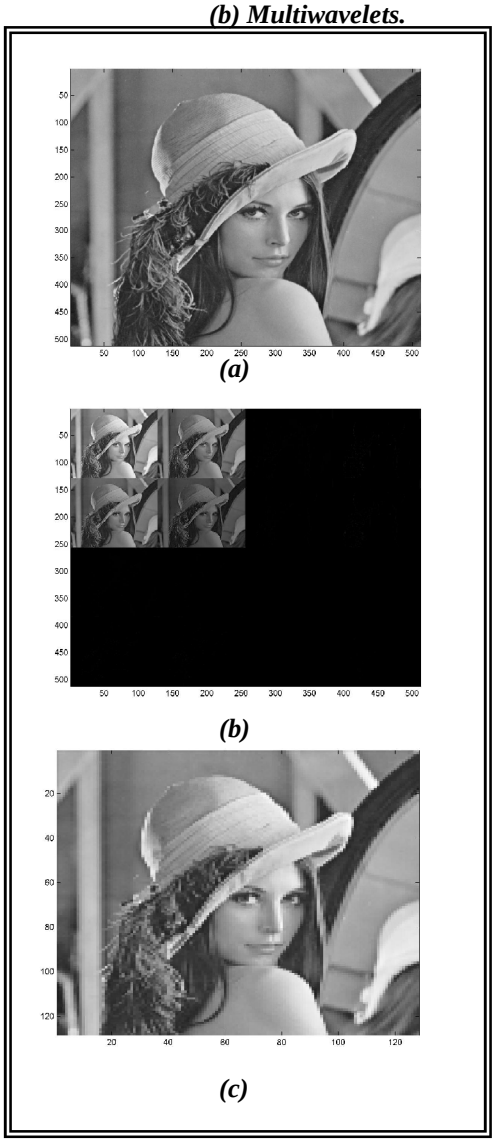


Fig.2: ***Lena Image, (a) Original, (b) After Single-Level of DMWT Using a Critical-sampled Scheme of Preprocessing(1st Order Approx.) and, (c) Zoomed In Upper-Left Most, L_1L_1 , Subband of (b).***

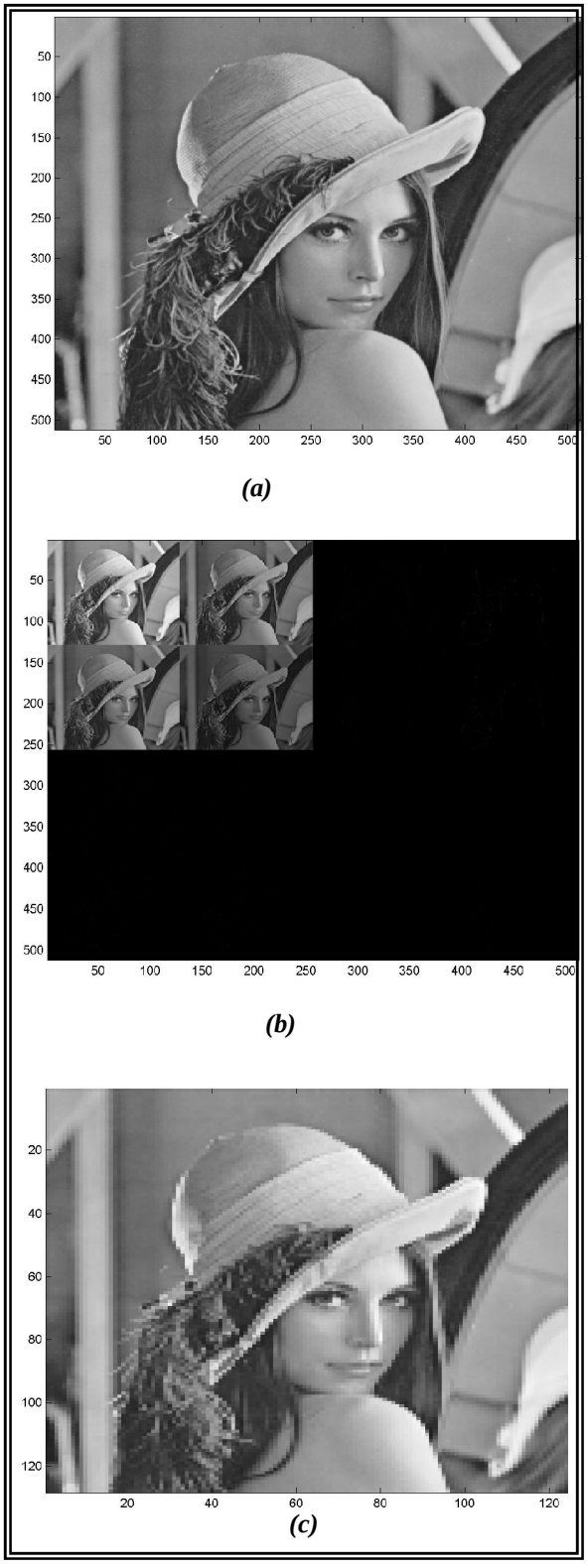
11. Conclusion

Multiwavelets filter banks require a vector-valued input signal. This is another issue which is addressed when multiwavelets are used in the transform process. A scalar-valued input signal must somehow be converted into a suitable vector-valued signal. This conversion is called *preprocessing*.

The most obvious way to get two input rows from a given signal is to repeat the signal using repeated row preprocessing (Over-sampled scheme of preprocessing). An approximation-based preprocessing algorithms have been also used as a critical-sampled scheme of preprocessing the signal.

Using a critical-sampled scheme of preprocessing (Approximation-based preprocessing) ensures the same original image dimensions while using an over-sampled scheme of preprocessing (repeated row preprocessing) introduces an oversampling of data by a factor of 2 which doubles the original image dimensions. In the same time, the upper-left most subband (L_1L_1) of the decomposed image using critical-sampling scheme of preprocessing, which usually the 2nd, 3rd,… levels of decompositions are applied to it, has quarter dimensions of the original while the upper-left most subband (L_1L_1) of the decomposed image using an over-sampled scheme of preprocessing has half dimensions of the original. So that critical-sampled representation of the signal minimizes the redundancy for data compression applications.

It should be mentioned here also that Discrete Multiwavelets Transform computation algorithm using a critical-sampled scheme of preprocessing (approximation-based preprocessing) should be applied to a matrix with a size at least equal to 8×8. Also matrix approximation scheme is the scheme that is chosen for multiwavelet compression. Such a scheme will keep the be transformed after



(a)

(b)

(c)

Fig.3: ***Lena Image, (a) Original, (b) After Single-Level of DMWT Using a Critical-sampled Scheme of Prepro.(1st Order Approx.) and, (c) Af… Critical-sampled Sch…***

(a)

(b)

(c)

Fig.4: ***(a) Reconstructed Lena Image Using IDMWT and 1st Order Approx. Postprocessing, (b) Reconstructed …WT and 2nd Order Approx. …c)Original Lena Image***

12. References

1. Michael B. Martin, *Applications of Multiwavelets to Image Compression*, M.Sc. Thesis in Electrical Engineering, Virginia Polytechnic Institute and State University(Virginia Tech),Blacksburg, Virginia, June, 1999.
2. Vasily Strela, Peter Niels Heller, Gilbert Strang, Pankaj Topiwala, and Christopher Heil, *The Application of Multiwavelet Filterbank to Image Processing*, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 8, No. 4, pp(548-563),April 1999.
3. J. Geronimo, D. Hardin, and P. R. Massopust, *Fractal Function and Wavelet Expansions Based on Several Functions*, J. Approx. Theory, vol.78, pp(373-401), 1994).
4. Yun Q. Shi, Huifang Sun: *Image and Video Compression for Multimedia Engineering, Fundamentals, Algorithms, and standards*, CRC Press LLC, 2000.
5. Panrong Xiao: *Image Compression By Wavelet Transform*, M.Sc. Thesis, Dept. of Computer and Information Science, East Tennessee State University, August, 2001.
6. Vasily Strela, *Multiwavelets: Theory and Applications*, Ph.D. thesis, Massachusetts Institute of Technology, 1996.
7. Michael B. Martin and Amy E. Bell, Member, IEEE, *New Image Compression Techniques Using Multiwavelets and Multiwavelet Packets*, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 10, NO. 4, APRIL 2001.
8. Daubechies, *Ten Lectures on Wavelets*, pp.(251-254), SIAM, 1992.
9. V. Strela and A. T. Walden, *Orthogonal and biorthogonal multiwavelets for signal denoising and image compression*, Proc. SPIE, 3391:96-107, 1998.
10. T. N. T. Goodman and S. L. Lee, *Wavelets of multiplicity r*, Trans. of the Amer. Math. Society, 342(1):307-324, March 1994.
11. P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, Englewood Cli_s, NJ, first edition, 1993.
12. S. Dunn: *Digital Color*, http://davis.wpi.edu/~matt/courses/color.
13. K. Sayood, and Morgan Kaufmann, *Introduction to Data Compression*, lecture 01-2003 Spring Text Book, Second edition, Academic Press, http:// www.cs.sunysb.edu/~amohr/cse391/200 3-spring/ lectures/cse391-lecture01.pdf.
14. J. Ziv, A. Lempel: *A Universal Algorithm for Sequential Data, Compression*, IEEE Trans. Inform. Theory, 1977, vol. 23, no. 3, pp. 337-343.
15. W. A. Mahmoud, Z. J. M. Saleh, and N. K. Wafi, *A Simple and Easy to Verify Algorithm For Computing GHM Multiwavelets Transform and Inverse Transform Using an Over-Sampled Scheme of Preprocessing and Postprocessing Respectively***,** Journal of Engineering, College of Engineering, University of Baghdad, accepted for publication, EEN/68, 2004.

DMWT

DMWT