View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Debrecen Electronic Archive

Recent Innovations in Mechatronics (RIiM) Vol. 2. (2015). No. 1-2.
DOI: 10.17667/riim.2015.1-2/16.

Practical Implementation of Second Generation Wavelet Transformation on 2D Images by Transformation Matrixes

Sandor J. PIROS
Electrical Engineering and Mechatronics Department
University of Debrecen, Faculty of Engineering
Debrecen, Hungary
piros@eng.unideb.hu

Peter KORONDI
Dept. of Mechatronics, Optics and Engineering Informatics
Budapest University of Technology and Economics
Budapest, Hungary

***Abstract*—To perform second generation wavelet transformation in digital signal processing is a common operation. Most obvious way to carry out this task is by program cycle management. Computational software usually executes direct matrix algebraic operations much faster, compared to cycles. Avoiding organizing cycles, can increase the processing speed by magnitudes. This paper introduces transformation matrixes to execute the key steps of wavelet transformation: filtering and downsampling. By the employment of the described transformational matrixes, wavelet transformation and filtering of two dimensional images or discrete datasets could be easier and faster.**

***Index Terms*— Signal processing algorithm, Digital signal processing, Wavelet transformation, Discrete wavelet transform, Matrix operations**

I. INTRODUCTION

TRADITIONAL signal processing transformations like Fourier transformation and even first generation wavelet transformations work well for infinite or periodical signals. Images and most of the datasets are bonde[1]d in size and typically are not periodic and especially when they are discrete, but usually they have a positive property, that neighboring elements are highly correlated. They are not the best candidates for the previously mentioned transformations; discrete wavelet transformation can cope with them better.

General wavelet transformation comprises mainly two steps, filtering (to compare the signal to the kernel) followed by a downsampling process. In the case of second generation wavelet transformation there are three steps, first is splitting the data into two parts, for example to even and odd elements, predict odds using even part, thus generating the detail coefficients and finally update even part using detail coefficients [1]. Inverse transformation has similar steps in opposite order.

Discrete wavelet transformation does the same (Figure 1), decompose the image with a high pass and a low pass filter; ***h(n)*** high-pass filter provides the detail coefficients, ***g(n)*** filter provides the approximation coefficients. After downsampling the image high frequency coefficients we get the first level coefficients and downsampling the low frequency coefficients we could get the further (second and so on) level coefficients [2].

II. WAVELET TRANSFORMATION

A. Second Generation Wavelet Transformation of 1D (one-dimensional) Signal

The simplest use of this wavelet transformation is for 1D signal.

Figure 1 General DWT representation using cascading filter bank Decomposition into low and high frequencies by convolution, followed by downsampling (***g(n)*** low pass, ***h(n)*** high pass filters).

Manuscript received August 31, 2014. The research of S. Piros was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4. A/2-11-1-2012-0001 'National Excellence Program'.

Sandor J. Piros is with University of Debrecen Engineering faculty, Electrical Engineering and Mechatronics Department, (phone:+36-20-511-3429, e-mail: piros@eng.unideb.hu).

Peter Korondi is with Budapest University of Technology and Economics, Dept. of Mechatronics, Optics and Engineering Informatics, Budapest (email: korondi@mogi.bme.hu).

Restriction on ***h*** and ***g*** filters: they must complement each other so, that we should be able to reconstruct the original signal.

Filtering:

In a 1D system we have an input signal ***f***, and the transformed signal is ***g***, than the linear constant coefficient difference transformation of the signal looks like:

$$\sum_{(k)\in\mathcal{R}_a} a_k\, g(m-k) = \sum_{(k)\in\mathcal{R}_b} b_k\, f(m-k) \quad \textbf{(1)}$$

The region of the filters are and , a_k and b_k are real numbers. By rearranging this difference equation we get, that if $a_0 \neq 0$ and $(0) \in$ the transformed signal would be:

$$g(m) = -\sum_{(k)\in\mathcal{R}_a} a_k{}'\, g(m-k) + \sum_{(k)\in\mathcal{R}_b} b_k{}'\, f(m- \quad \textbf{(2)}$$

for example let the direction of recursion be from left to right.

B. Spatial 2D (two-dimensional) System

How to carry out wavelet transformation in higher dimensional systems? In a general 2D spatial system the two variables are ***m*** and ***n***, the linear constant coefficient difference equations could be written as:

$$\sum_{(k,l)\in\mathcal{R}_a} a_{k,l}\, g(m-k,n-l) = \\ = \sum_{(k,l)\in\mathcal{R}_b} b_{k,l}\, f(m-k,n-l) \quad \textbf{(3)}$$

Whereas ***f*** is the input image, ***g*** is the transformed image, $b_{k,l}$ are the feedforward coefficients, $a_{k,l}$ are the feedback coefficients [3]. The solution for ***y***:

$$g(m,n) = \sum_{(k,l)\in\mathcal{R}_a} a'_{k,l}\, g(m-k,n-l) + \\ + \sum_{(k,l)\in\mathcal{R}_b} b'_{k,l}\, f(m-k,n-l) \quad \textbf{(4)}$$

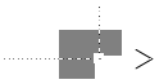

Figure 2 Filtering progression in a 2 Dimensional system

The progression could be e.g. horizontally left to right, vertically up to down by each row, Figure 2 gives an example.

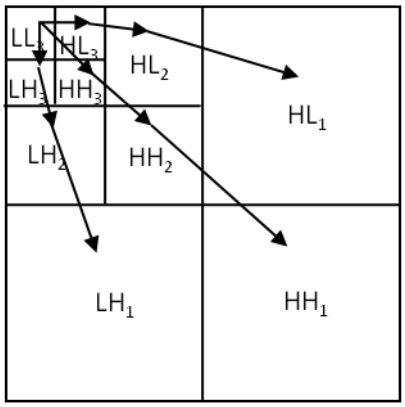

Figure 3 An example of a hierarchical image transformation in 2D system; ***L*** is low frequency component, ***H*** is the high frequency component of the signal

Wavelet transformations are hierarchical transformations. Figure 3 example shows the transformation of a square shape image into fragments of similar shapes [4].

III. MATRIX OPERATIONS

A. First Step of the Transformation: Downsampling

One important step of wavelet transformation is downsampling. When the signal is a 1D, it is very simple to take every other element apart, select even and odd elements. What to do, how to proceed with 2D images? To answer this question is not so self-explanatory. For example in each cycle of the transformation we can transform the picture first along ***x*** axis, then ***y, x, y*** and so on (Figure 4).

The most important observation is, that the shape of the pixels, so the shape of the transformed image becomes distorted after every other transformation step (H_{11}, H_{21}, H_{31} in Figure 4). The size of the original image in this example was 2^nx2^m pixels, so H_{11} should be 2^nx2^{m-1}, H_{12} 2^{n-1}x2^{m-1} pixels size and so on.

The purpose of the transformation is to replace dataset elements by the extent of deviation from their predicted values. We have assumed that the elements are correlated to each other, so the elements of the transform should be near to zero values.

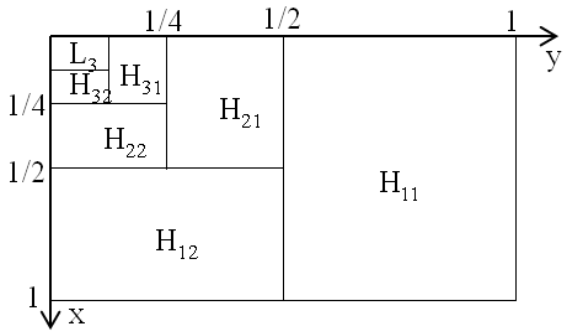

Figure 4 Image transformation in 2D

The practical implementation of discrete wavelet transformation was described above. The undesirable or objectionable feature of that method is to perform instruction cycles. Numerical programs like Matlab execute matrix operations and manipulations much faster. We can create transformation matrix ***T***, which is capable to perform the downsampling and approximation in one run.

$$\mathbf{T}=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ \vdots & & & & & \dots & 0 & 0 \\ \dots & \dots & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ \vdots & & & & & \dots & 0 & 0 \\ \dots & \dots & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{5}$$

$$\mathbf{T}^{\mathbf{T}}=\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{6}$$

P is a picture or dataset to be transformed, the resulting matrix consists of two submatrixes, ***P'*** should contain the approximation coefficients and ***D*** submatrix for the detail coefficients.

$$\boldsymbol{T} \cdot \boldsymbol{P} = \begin{bmatrix} \boldsymbol{P'} \\ \boldsymbol{D} \end{bmatrix} \tag{7}$$

$$\boldsymbol{P} \cdot \boldsymbol{T}^{\boldsymbol{T}} = [\boldsymbol{P'}, \boldsymbol{D}] \tag{8}$$

Equation (7) works in ***x*** direction and equation (8) for ***y*** direction.

For example in ***x*** direction:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{1,1}-a_{2,1} & a_{1,2}-a_{2,2} & a_{1,3}-a_{2,3} & a_{1,4}-a_{2,4} \\ a_{3,1}-a_{4,1} & a_{3,2}-a_{4,2} & a_{3,3}-a_{4,3} & a_{3,4}-a_{4,4} \end{bmatrix} \tag{9}$$

and in ***y*** direction:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} =$$

$$= \begin{bmatrix} a_{1,1} & a_{1,3} & a_{1,1}-a_{1,2} & a_{1,3}-a_{1,4} \\ a_{2,1} & a_{2,3} & a_{2,1}-a_{2,2} & a_{2,3}-a_{2,4} \\ a_{3,1} & a_{3,3} & a_{3,1}-a_{3,2} & a_{3,3}-a_{3,4} \\ a_{4,1} & a_{4,3} & a_{4,1}-a_{4.2} & a_{4,3}-a_{4.4} \end{bmatrix} \tag{10}$$

B. Inverse-Transformation

The previously described transformation is completely lossless, but it could be modified for lossy transformation, if compression is more important than fidelity [5].

Following this process reversely we can get back the original information, dataset or image, which is the inverse-transformation.

During restoration of the original image we have to use the inverse of the transformation matrix, i.e.:

$$\boldsymbol{P} = \boldsymbol{T}^{-1} \cdot \begin{bmatrix} \boldsymbol{P'} \\ \boldsymbol{D} \end{bmatrix} \tag{11}$$

$$\boldsymbol{P} = [\boldsymbol{P'}, \boldsymbol{D}] \cdot (\boldsymbol{T}^{\boldsymbol{T}})^{-1} \tag{12}$$

Equation (11) works in ***x*** direction and equation (12) for ***y*** direction.

For example in ***x*** direction:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \cdot$$

$$\cdot \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{1,1}-a_{2.1} & a_{1,2}-a_{2.2} & a_{1,3}-a_{2,3} & a_{1,4}-a_{2,4} \\ a_{3,1}-a_{4.1} & a_{3,2}-a_{4.2} & a_{3,3}-a_{4.3} & a_{3,4}-a_{4.4} \end{bmatrix}$$

(13)

and in ***y*** direction:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{1,1} & a_{1,3} & a_{1,1}-a_{1,2} & a_{1,3}-a_{1,4} \\ a_{2,1} & a_{2,3} & a_{2,1}-a_{2,2} & a_{2,3}-a_{2,4} \\ a_{3,1} & a_{3,3} & a_{3,1}-a_{3,2} & a_{3,3}-a_{3,4} \\ a_{4,1} & a_{4,3} & a_{4,1}-a_{4.2} & a_{4,3}-a_{4.4} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

(14)

In case we want to filter the signal, the signal filtering can be addressed also with matrix operations. The simplest case is, when a pixel value is estimated or predicted by the average of its two neighbors (½ 0 ½) like in paragraph A. The filtration is carried out by using the matrix ***F*** in ***x*** direction (equation (16) and its transpose in the ***y*** direction (equation (17).

$$\mathbf{F}= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{½} & \mathbf{0} & \mathbf{½} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{½} & \mathbf{0} & \mathbf{½} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \mathbf{½} & \mathbf{0} & \mathbf{½} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(15)

$$\boldsymbol{F \cdot P = P_F}$$

(16)

$$\boldsymbol{P \cdot F^T = P_F}$$

(17)

where $\boldsymbol{P_F}$ is the filtered image. We can notice that the pixels of the last row or column have neighbor only at one side vertically or horizontally, obviously it is not possible to figure out average value.

For example in ***x*** direction:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ ½ & 0 & ½ & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ ½(a_{1,1}+a_{3.1}) & ½(a_{1,2}+a_{3.2}) & ½(a_{1,3}+a_{3,3}) & ½(a_{1,4}+a_{3,4}) \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \end{bmatrix}$$

(18)

and in ***y*** direction:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \cdot \begin{bmatrix} 1 & ½ & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & ½ & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} a_{1,1} & ½(a_{1,1}-a_{1,3}) & a_{1,3} & a_{1,3} \\ a_{2,1} & ½(a_{2,1}-a_{2,3}) & a_{2,3} & a_{2,3} \\ a_{3,1} & ½(a_{3,1}-a_{3,3}) & a_{3,3} & a_{3,3} \\ a_{4,1} & ½(a_{4,1}-a_{4.3}) & a_{4,3} & a_{4,3} \end{bmatrix}$$

(19)

Equation (20 shows an example of a higher order filter using quadratic spline interpolation ($-^1/_{16}$ 0 $^9/_{16}$ 0 $^9/_{16}$ 0 $-^1/_{16}$). Because the size of this example filter is small (for an image of 8x8 pixel size only) and the filter is distorted near the edges, so only the third line contains the complete kernel of this filter.

$$F= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ^1/_2 & 0 & ^9/_{16} & 0 & -^1/_{16} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{-^1/_{16}} & \mathbf{0} & \mathbf{^9/_{16}} & \mathbf{0} & \mathbf{^9/_{16}} & \mathbf{0} & \mathbf{-^1/_{16}} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -^1/_{16} & 0 & ^9/_{16} & 0 & ^1/_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -^1/_8 & 0 & ^9/_8 & 0 \end{bmatrix}$$

(20)

To solve the problem when the filter is overlapping the edge of the picture, there are different methods. For example we can adjoin black or white pixels outside the boundary or

mirroring the last rows or columns like it was done above.

C. Practical Implementation of the Transformation

The best way to demonstrate this algorithm is through the example of an image transformation. Figure 5 is a 256 by 256 pixels size 8 bit gray scale image, after transformation we can get the transformed image: Figure 6. For generating a predicted value we have used the average value of two neighbors of a_{2k}: $p_{2k}=a_{2k-1}- a_{2k+1}$.

Figure 5 Example picture 256x256 pixels

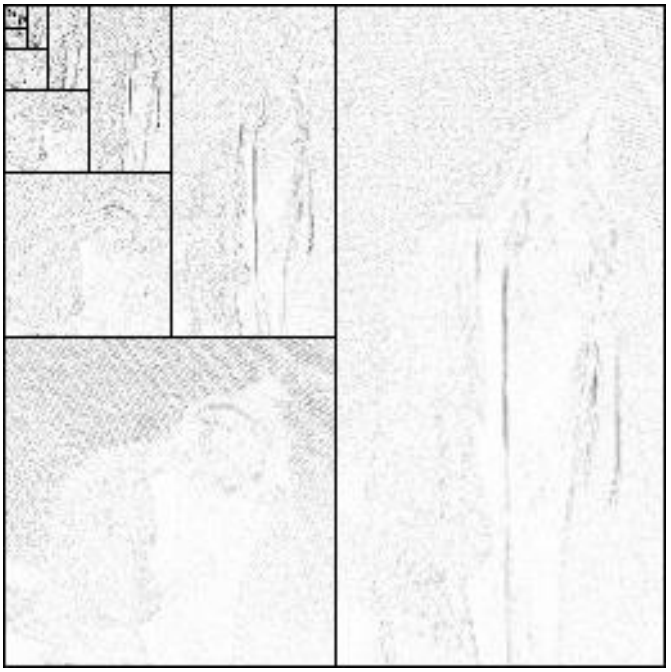

Figure 6 Detail coefficients of the wavelet transform of Figure 5 image.

By the help of the transformed image we can restore the original image by inverse transformation as it is shown step by step on Figure 7, starting from the one pixel stage unto the next to the last stage (from 1 st to (n-1)th step 256x128 pixels image).

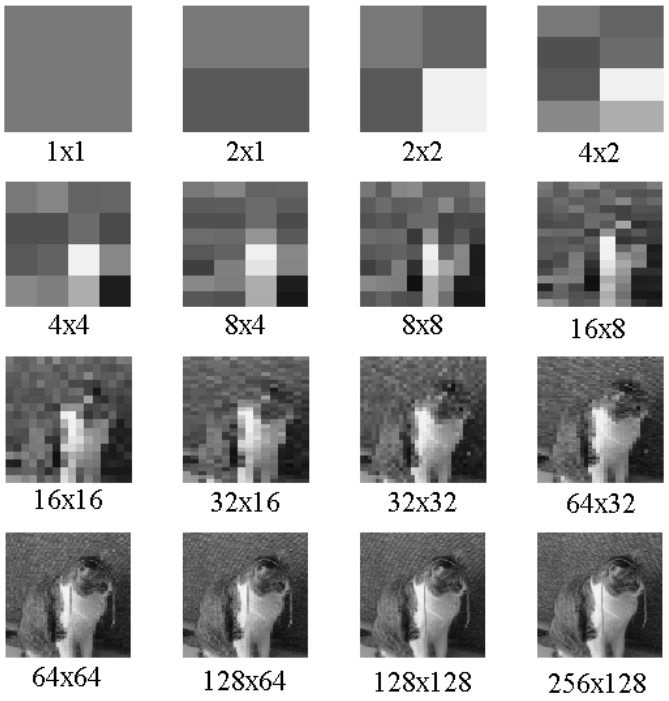

Figure 7 Series of approximation coefficients of the wavelet transform

IV. CONCLUSION

This article is about an algorithm, how to transform and compress discrete dataset for storage and transmission purposes. This algorithm could provide an economical way for handling a correlated or highly correlated discrete or quantized analog signal, image or any multidimensional data set. To prepare a corresponding wavelet transformation method we used transformation matrixes. Using matrix operations instead of calculating in cycles can positively influence processing the speed of digital signal processing.

REFERENCES

[1] Fadil Santosa, "Second Generation Wavelets: Theory and Application," University of Minnesota, Institute for Mathematics and its Applications, 06 Oct. 2011. [Online]. Available: http://www.ima.umn.edu/industrial/97_98/sweldens/fourth.html. [Accessed 16 Apr. 2013].

[2] R. Polikar, "The Wavelet Tutorial," Rowan University, College of Engineering Web Servers, 12 1 2001. [Online]. Available: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html.

[3] J. W. Woods, Multidimensional Signal, Image, and Video Processing and Coding, Waltham: Elsevir Inc., 2012.

[4] J. Mukhopadhyay, Image and Video Processing in the Compressed Domain, Boca Raton: CRC Press, 2011.

[5] K. F. S. L. Nick D. Panagiotacopulos, "Lossy Image Compression Using Wavelets," *Journal of Intelligent and Robotic Systems,* vol. 28, no. 1-2, pp. 39-59, 2000.

.