Wavelets and Image Compression

Vlad Balan, Cosmin Condea

January 30, 2003

Abstract

In this paper we will examine the wavelet transform, one of the most recent mathematical tools related to signal representation and illustrate it's application in the field of image compression. The paper is divided in two main parts. In the first one we present the mathematical principles of multiresolution analysis, we illustrate them using the Haar wavelet in the one dimensional case, we present the transform algorithms and we end up discussing a number of more advanced topics. The second part starts by describing the transition to the discrete case and then presents in a step by step manner the general procedure for image compression using wavelets. There are four basic steps: applying the wavelet transform, threshold detection, quantizing and encoding the resulting data and finally applying an inverse transform. These theoretical aspects are illustrated through a MATLAB project which we developed using the Stanford University's Wavelab toolbox.

Contents

1 Wavelets

1.1 Principles of multiresolution analysis

We define a multiresolution analysis as a mathematical object consisting of the following:

- (a) A bilateral sequence of closed subspaces V_j of L^2 ordered by inclusion:

  $$\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset \ldots \subset V_j \subset V_{j-1} \subset \ldots \subset L^2 \tag{1}$$

  and obeying to the following axioms:

  $$\bigcap_j V_j = 0 \quad \text{(separation axiom)} \tag{2}$$

  $$\overline{\bigcup_j V_j} = L^2 \quad \text{(completeness axiom)} \tag{3}$$

- (b) A scaling property of the V_j subspaces:

  $$V_j = D_2(V_j) \quad \forall j \in \mathbb{Z} \quad \text{where} \tag{4}$$

  $$D_2(f) = \sum_{k=0}^{n} h_k * f(t-k) \tag{5}$$

  or: $f \in V_j \iff f(2\cdot) \in V_{j-1}$.

- (c) There exists a function $\phi \in L^2 \bigcap L^1$ such that its translates ($\phi(\cdot - k) k \in \mathbb{Z}$) form an orthonormal basis of V_0. The function ϕ is called the scaling function. We notice that the space V_0 uses one such basis vector per unit length while V_{-j} uses 2^j basis vectors per unit length.

We conclude that the functions $\phi_{j,k} \in V_j$ constitute a basis of V_j. However, we cannot form a basis of L^2 just by taking the union of these since the subspaces V_j cannot be orthogonal as a consequence of relation (a).

We can define another sequence of subspaces $W_j = V_j - V_{j-1}$. These can be proven to be pairwise orthogonal, and even more $\overline{\bigoplus_j W_j} = L^2$.

Looking at the formula $W_0 = V_0 - V_1$ and bearing in mind that V_0 uses twice as many vectors per unit length as V_1 we would be tempted to start looking for a function ψ which would constitute a basis of W_0. Such a construction is possible by going into the Fourier domain, and the reader is invited to consult [Blatter, chap. 5] The resulting functions $\psi_{j,k}$ constitute an orthonormal basis of L^2.

For our purposes it is convenient to require that:

$$\int_{-\infty}^{\infty} \phi(x)dx = 1, \int_{-\infty}^{\infty} |\phi(x)|^2 dx = 1 \tag{6}$$

$$\int_{-\infty}^{\infty} |\psi(x)|^2 dx = 1 \tag{7}$$

From axiom b) the following identity holds:

$$\phi(t) = \sqrt{(}2) \sum_{-\infty}^{\infty} h_k \phi(2t-k) \text{ for almost all } t \in \mathbb{R} \tag{8}$$

with the coefficient vector $h \in l^2(\mathbb{Z})$.
Furthermore we can prove that the wavelet function:

$$\psi(t) = \sqrt{2} \sum_{-\infty}^{\infty} g_k \phi(2t-k) \tag{9}$$

$$= \sqrt{2} \sum_{-\infty}^{\infty} (-1)^k h_{1-k} \phi(2t-k) \text{ for almost all } t \in \mathbb{R} \tag{10}$$

with the coefficient vector $g \in l^2(\mathbb{Z})$ is orthogonal to the scaling function. If one additional condition is met, the wavelet functions can be proved to be orthogonal to each other, giving us a valid basis.

We can therefore define the wavelet expansion of a function f as

$$\sum_{j,k \in \mathbb{Z}} C_{jk} 2^{j/2} \psi(2^j - k) \tag{11}$$

with the coefficients C_{jk} defined by

$$C_{jk} \;=\; \langle f, \psi_{jk} \rangle \;=\; \int_{-\infty}^{+\infty} f(x) 2^{j/2} \psi(2^j - k)\, dx \tag{12}$$

We can think of the coefficients of each the functions belonging to each V_j as representing those features of a signal that have a spread of size comparable to $2^j/\sqrt{2}$. From this point of view a multiresolution analysis process can be imagined such as the work of a common filter bank.

1.2 A simple example: The Haar Wavelet

The mathematician Alfred Haar was the first to describe, in 1910, an orthonormal system for the Hilbert space L^2 and proved it to be isomorphic to the space

$$l^2 := \{(c_k|k \in \mathbb{N})| \sum_{k=0}^{\infty} |c_k|^2 < \infty\} \tag{13}$$

Since the union of all step functions of step $2^j, j \in \mathbb{Z}$ is dense in L^2 and the other conditions are obviously satisfied we conclude that $\mathbf{1}_{[0,1)}$ constitutes a valid scale function for a multiresolution analysis.Starting from the scaling function $\phi = \mathbf{1}_{[0,1)}$ we obtain the Haar Wavelet which has the form of the following step function:

$$\psi(x) = \begin{cases} 1 & (0 \leq x < \frac{1}{2}) \\ -1 & (\frac{1}{2} \leq x < 1) \\ 0 & \text{otherwise} \end{cases}$$

We see that:

$$\psi(x) = \phi(2x) - \phi(2x-1) \tag{14}$$
$$\phi(x) = \phi(2x) - \phi(2x-1) \tag{15}$$

Intuitively, considering that at the level V_j we are left with a step function f_j of step 2^j from two neighbor steps corresponding to the intervals $[2k\dot{2}^r, (2k+1)\dot{2}^r)$ and $[(2k+1)\dot{2}^r, (2k+2)\dot{2}^r)$ we obtain the step corresponding to f_{j+1}'s interval $[2k\dot{2}^r, (2k+2)\dot{2}^r)$ as their mean value and the coefficient of the corresponding ψ_{jk} function as the difference between the mean value and the value of the function, as seen in the figure (from [Blatter, p. 24]):

1.3 Algorithms

For the analysis of a signal having as support the interval $[0,1)$ we define the scaling function ϕ and the wavelet function ψ on this interval. Considering that we have sampled the function in 2^N equally distanced points, we can approximate it by a step function. We can start applying the wavelet transform from the subspace V_{-N}.

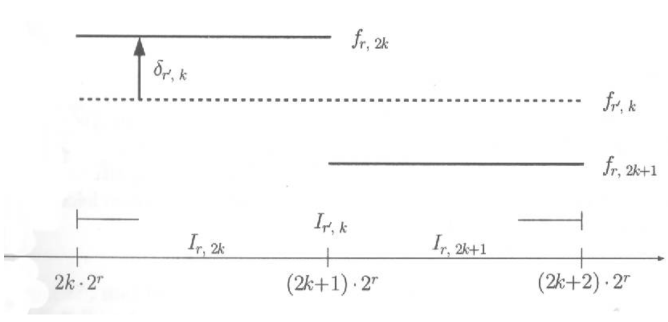

Figure 1: Haar Wavelet's action

Considering A_N as a 2^N size vector containing the coefficients a_{Nk} we can apply the following operator H_N in order to obtain a vector B_N containing the coefficients $a_{(N-1)k}$ and $C_{(N-1)k}$ in an interlaced form:

$$\begin{pmatrix}
h_0 & h_1 & h_2 & \dots & h_n & & & & & \\
g_0 & g_1 & g_2 & \dots & g_n & & & & & \\
 & & h_0 & h_1 & h_2 & \dots & h_n & & & \\
 & & g_0 & g_1 & g_2 & \dots & g_n & & & \\
 & & & & \dots & & & & & \\
 & & & & \dots & & & & & \\
 & & & & h_0 & h_1 & h_2 & \dots & h_n & \\
 & & & & g_0 & g_1 & g_2 & \dots & g_n & \\
h_2 & \dots & h_n & & & & & & h_0 & h_1 \\
g_2 & \dots & g_n & & & & & & g_0 & g_1
\end{pmatrix}$$

By applying a permutation matrix:

$$P_N[a_{N-1}1, C_{N-1}1, \dots, a_{N-1}\tfrac{N}{2}-1, C_{N-1}\tfrac{N}{2}-1]^T$$
$$= [a_{N-1}1, \dots, a_{N-1}\tfrac{N}{2}-1, \dots, C_{N-1}1, \dots, C_{N-1}\tfrac{N}{2}-1]^T$$

we move the coefficients $a_{N-1}k$ to the front, therefore we take the first half of B_N for A_{N-1}.

We conclude that we can perform a full wavelet transform by applying a series of PH operations and an inverse transform by applying a series of $H^{-1}P^{-1}$ operations. Knowing that we are multiply by the coefficients h_k in order to obtain the next sequence of as and with g_k in order to obtain the next sequence c_k we can represent the transform process by the following diagram:

$$\begin{tikzcd}
a_j \arrow[r, "h"] \arrow[dr, "g"] & a_{j-1} \arrow[r, "h"] \arrow[dr, "g"] & a_{j-2} & \dots \arrow[r, "h"] \arrow[dr, "g"] & a_0 \\
 & c_{j-1} & c_{j-2} & & c_0
\end{tikzcd}$$

while the inverse transform can be represented as:

$$\begin{tikzcd}
a_j & \rightarrow & a_{j-1} & \rightarrow & a_{j-2} & \ldots & \rightarrow & a_0 \\
 & / & & / & & & / & \\
 & & c_{j-1} & & c_{j-2} & & & c_0
\end{tikzcd}$$

Since the matrixes involved are sparse the complexity of the multiplications is $O(n)$. The complexity of the whole series is therefore $\sum_{k=1}^{N} \frac{O(n)}{2^k} = O(2n) = O(n)$.

1.4 Advanced topics

Having exposed the basic principles of wavelet analysis we proceed now to describing some properties of different common wavelets. Since our wavelet functions are centered around their set of coefficients h_k we are interested in finding the minimal set of coefficients satisfying some constraint equations. We have from (7) just by integrating both sides:

$$\int \phi dx = \sqrt{2} \sum h_k \int \phi(2x-k) d(2x-k) \tag{16}$$

$$\sum h_k = \sqrt{2} \tag{17}$$

and by integrating with respect to the L^2 norm:

$$\int \phi^2 dx = \sqrt{2} \sum (h_k \int \phi(2x-k) d(2x-k))^2 \tag{18}$$

$$\sum |h_k|^2 = 1. \tag{19}$$

Signals that are smooth present a high degree of linearity. Their Taylor series expansion around each point tends to decay very fast. We would be interested in a wavelet function whose scalar product with a given polynomial vanishes:

$$\int_{-\infty}^{\infty} x^j \psi(x) dx = 0 \text{ for} j = 0, 1, \ldots, L-1 \tag{20}$$

We say in this case that ψ has its first L moments equal to 0.
Combining equation (9),(20) we obtain

$$\sum (-1)^k h_k = 0, \quad \sum k(-1)^k h_k = 0 \tag{21}$$

and in taking for example L = 2 the minimal set of coefficients satisfying all four conditions is:

$$\begin{aligned} h_0 &= \tfrac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \tfrac{3+\sqrt{3}}{4\sqrt{2}} \\ h_2 &= \tfrac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \tfrac{1-\sqrt{3}}{4\sqrt{2}} \end{aligned} \tag{22}$$

which defines the Daubechies D_4 wavelet.
Having these coefficients we can represent the graph of the function. We start from the relation $\phi_j(x) = \sum c_k \phi_{j-1}(2x-k)$, noticing its similarity to the equation of Iterated Function Systems (fractals). We can set the box function as ϕ_0 and by iteration draw its graph while using the wavelet coefficients we represent in the same manner the wavelet function [Strang 2]:

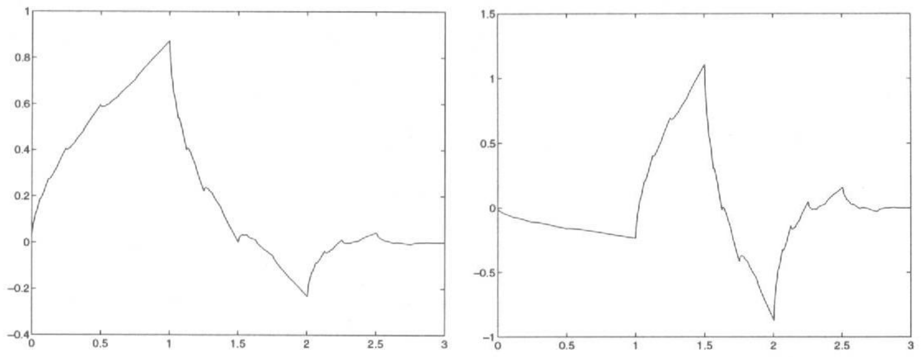

Figure 2: Daubechies D4 Scaling Function (left) and Wavelet Function (right)

An alternate method would be to compute using the same relation from the values at points $x = 2^j$ the values at points $x = 2^{j+1}$. For obtaining the initial values at the points $x = 1$ and $x = 2$ we use the fact that the function has as compact support the interval $(0,3)$ and we solve the equation:

$$\phi(1) = \sqrt{2}h_2 * \phi(1) + \sqrt{2}h_1 * \phi(2) \tag{23}$$

$$\phi(2) = \sqrt{2}h_4 * \phi(1) + \sqrt{2}h_3 * \phi(2) \tag{24}$$

which gives as $\phi(1)$ and $\phi(2)$ as eigenvectors for the eigenvalue $\lambda = 1$.
A final question that remained unanswered during our presentation is under what circumstances the construction in the formula (9) gives us a wavelet that is orthogonal to it's translates. We are not going to tackle this problem here, since it is only relevant in the construction of a wavelet function but the interested reader is can consult the article [Strang 2] or the book [Blatter].
As we can see the strong point of the wavelet transform is a good localization

both in terms of scale and position, which gives a signal good localization in time, since the frequency depends on the scale, and in space. This capacity to detect local features and features spreading over a larger distance makes the wavelet transform a suitable candidate for image compression since it is capable to retain and to evidentiate redundant information which is specific to a natural signal.
We have to bear in mind that when using wavelets such as Daubechies we are faced with a compromise between the length of the wavelet coefficients set to which the processing time is proportional and the speed of decay of the Taylor series of the processed signal. While an image signal has a slow decay due to many local irregularities (that is, there will always be trees in the background) and the filters are quite short, in audio applications where the signal is much "cleaner" they tend to be much longer.
The wavelet transform should not be seen as the universal solution for compressing discrete time signals. For example, when compressing a signal that is composed of sinusoidal functions a Fourier transform is guaranteed to give a much smaller set of meaningful coefficients.
Finally, anticipating the topic of the second part of this paper, let us say that an easy method to construct bi-dimensional wavelets is to start from an one-dimensional wavelet and take the cross products $\phi\phi, \phi\psi, \psi\phi, \psi\psi$, which give clearly orthogonal functions. Although this method can be easily used, different genuine bi-dimensional wavelets have been invented.

2 Image Compression

One area where wavelets have incontestably proven their applicability is image processing. As you know high resolution images claim a lot of disk space. In the age of information highway, the amount of data that needs to be stored or transmitted is huge. Therefore, compression greatly increases the capacity of the storage device, on one hand, and on the other, it also reduces costs.

To illustrate the use of compression take the simplest example: an image of 256 x 256, which takes approximatively 0.2 MB. On a simple floppy disk one can therefore store 7 such images. But think if this image can be compressed at a 25:1 ratio. The result is 175 images stored on the same floppy disk.

In this part of the paper, we describe the compression algorithm step by step, using the "Lenna" image (fig.3) for illustrations. Finally we will present the MATLAB code we wrote. We used the WaveLab v802 toolbox, downloaded from Stanford University's web site.

Figure 3: The Well-Known Lenna Image

2.1 Applying the transform

The compression algorithm starts by transforming the image from data space to wavelet space. This is done on several levels. We start with our data applying the bi-dimensional transform matrix and we get in the resulting image the coefficients grouped into four zones, like in the figure, where H symbolizes high frequency data and L symbolizes low frequency data, like in the figure:

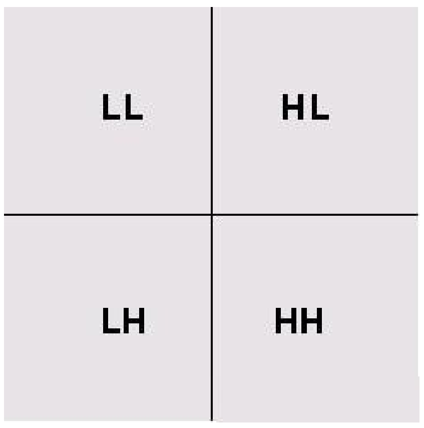

Figure 4: The Discrete Wavelet Transform Frequency Quadrants

The LL quadrant of the resulting image is the input of the next iteration. Usually for image compression purposes 4 or 5 iterations will suffice. In the next figures the result of the transform on 1 and on 5 levels.

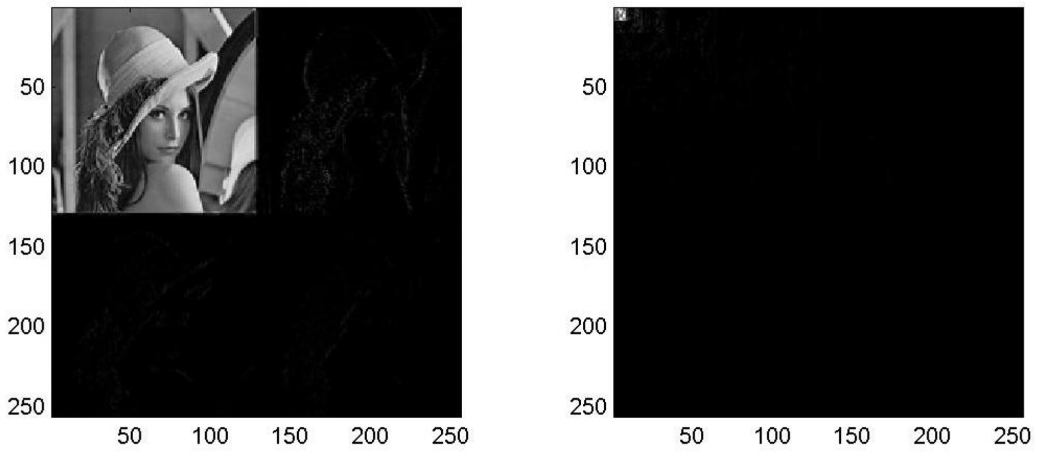

Figure 5: The Discrete Wavelet Transform (1 level and 5 levels)

The 5-level transform's data is also presented in a mesh form in order to visualize better the different intensities of the coefficients.An interesting aspect to notice is that the majority of the DWT coefficients are positioned in the upper left quadrant.

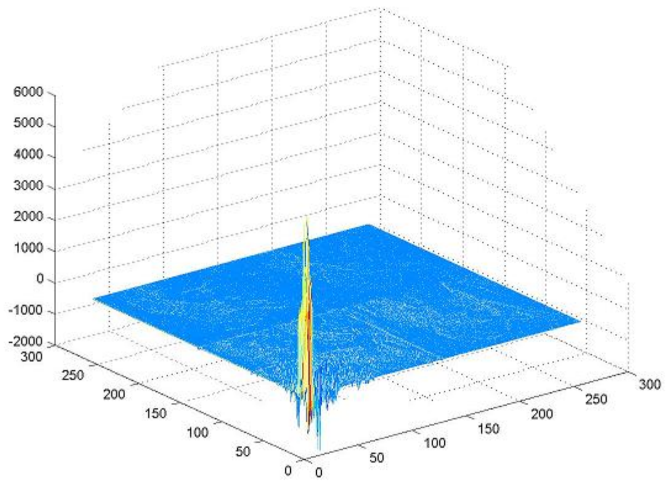

Figure 6: 3D View on the Discrete Wavelet Transform

2.2 Choosing a threshold

The next step in the algorithm is to neglect all the wavelet coefficients that fall below a certain threshold. We select our threshold in such a way as to preserve a certain percent of the total coefficients - this is known as "quantile" thresholding.

The small values of the DWT coefficients retain little detail of the picture. Therefore they can, up to a limit, be neglected. The key notion is here the "perceptual" distortion. Of course some details of the picture are consequently lost after applying the threshold but the question is to what extent the human eye can detect the difference between the original and the reconstructed image. In this direction, a human visual perception model has been created and its use in image compression has been studied. This model still remains an ongoing research project at the current time.

In what thresholding is concerned, besides the quantile one, there exist another 2 main types of thresholding:

- Hard Thresholding eliminates all the coefficients c_i that are smaller than an absolute threshold T. If we denote with $c_i^{'}$ the new coefficients:

$$c_i^{'} = \begin{cases} c_i & \text{if} \quad c_i > T \\ 0 & \text{otherwise} \end{cases}$$

- Soft Thresholding again sets an absolute limit reducing to zero all the coefficients that fall under it but at the same time it shrinks toward 0 if $c_i > T$. Keeping the notations, the relation for soft thresholding is:

$$c_i{'} = sgn(c_i) \cdot max(|c_i| - T, 0)$$

Coming back to the compression algorithm and the threshold step, in the next figure we represent the non-zero distribution in the DWT after we have chosen to use 5% of the coefficients - the greatest in absolute value (fig.7).

2.3 Compression methods

Coding We have designed a very simple compression scheme for sparse matrices in order to test the efficiency of the algorithm. We traverse the thresholded wavelet data's matrix line by line and we copy all the nonzero values to a vector. When we encounter a zero value, we start counting the

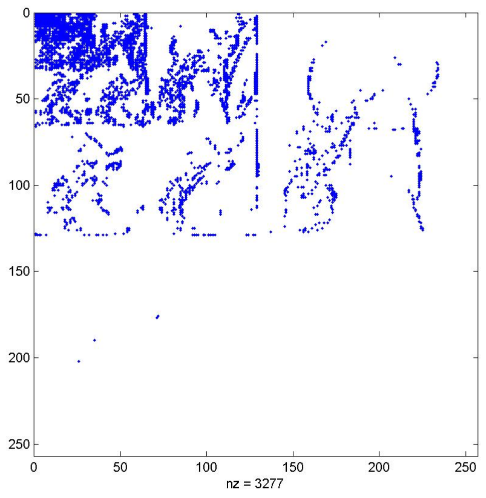

Figure 7: Distribution of Non-zero in the DWT (5% of the coefficients)

length of the sequence of zeroes to which it belongs. Every such sequence we replace with a zero value followed by it's length. After encoding our MATLAB code prints out the length of the resulting vector.

It is easy to see that the data can be reconstructed from this vector. Our compression scheme is not very efficient since we obtain about three times more data than the one belonging to the selected nonzero coefficients. This mean that when we preserve 5% of the coefficients we only compress the picture to 15% of its original data.

In order to efficiently store our data it is preferable to work with integers rather than floats. We can just round our data to the nearest integer or we can scale it first. This process is known as quantization. We did not use it in our MATLAB code since we just measure the length of the resulting vector, and consider that we could encode each value using two bytes.

More popular compression codes include Entropy coding, Huffman coding and specialized algorithms for coding wavelet transformed image data such as those created by R. Wells and J. Tian or by J.S. Walker.

In the next paragraph we will briefly discuss the entropy coder. The bottom line idea is that the coder should take advantage of long strings of 0 - which, after thresholding and quantization, they are mostly placed into the high frequency quadrants. This is done by "scanning".

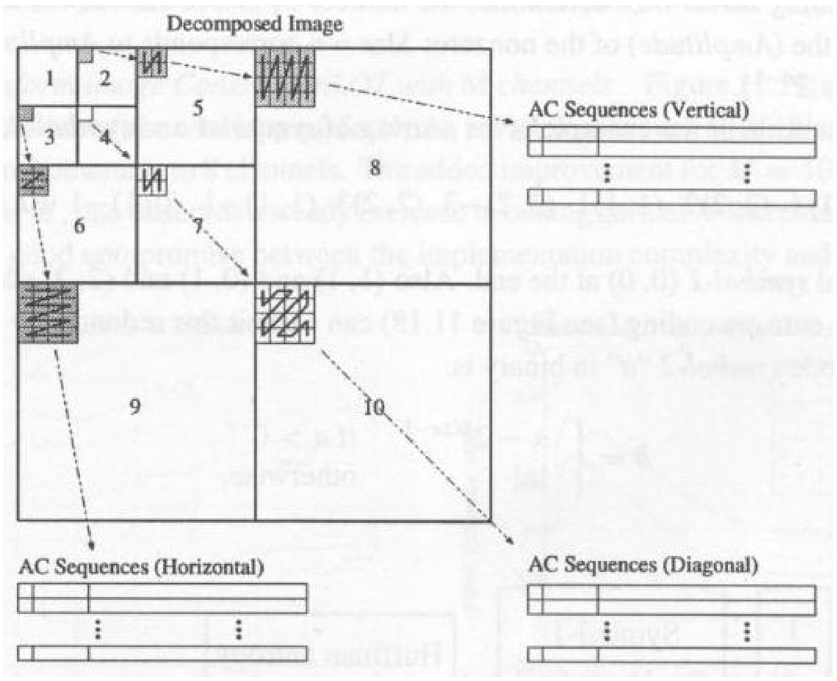

Figure 8: Scanning Method for the Entropy Coder [SN]

The 3-level DWT in the above picture illustrates how scanning is performed. If,for example, the shaded area in the 2^{nd} quadrant is found to be 0 - most likely to be so - then is can be assumed that the shaded areas in quadrants 5 and 8 also have zero coefficients. This idea can also be explained by comparing the DWT with a tree - where each parent has 4 children. If a parent is found to be 0, then all his children contain zero values.

Also to be noticed is the scanning pattern for different frequency quadrant sequences. Vertical for (2,5 and 8), horizontal for (3,6 and 9) and diagonal for (4,7 and 10). The obtained AC sequences are encoded in a standard Huffman way. The DC sequence is encoded based on the image continuity that is the differences of color are stored.

2.4 Applying the Inverse Transform

After decoding the data, the last step of the algorithm is that of applying the inverse DWT to the "doctored" image matrix. In the following we include some pictures where we set different quantile thresholds.

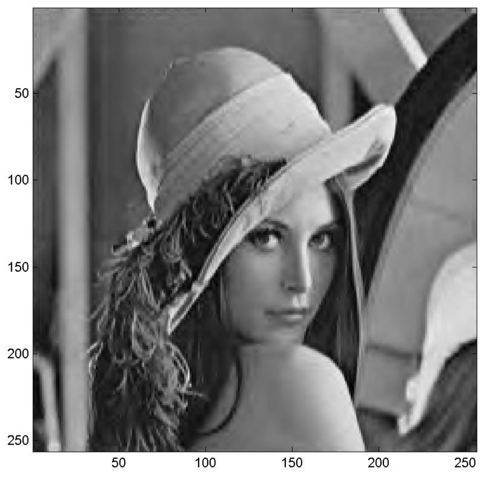

Figure 9: The Restored Lenna Image (with 10% of the coefficients)

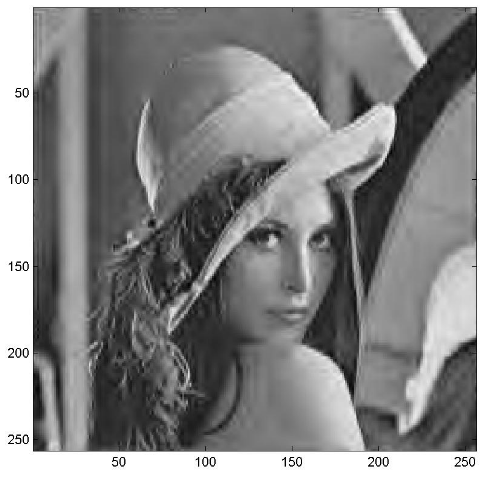

Figure 10: The Restored Lenna Image (with 5% of the coefficients)

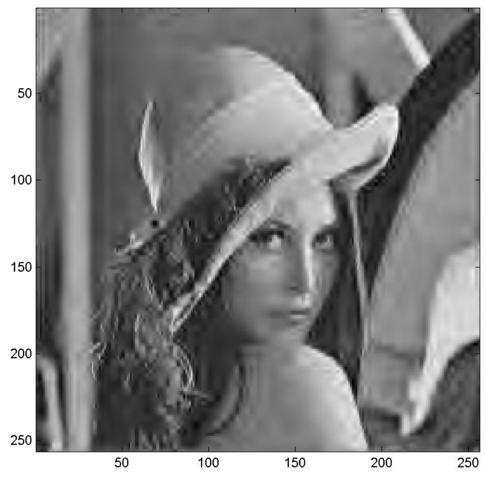

Figure 11: The Restored Lenna Image (with 3% of the coefficients)

References

[Blatter] Blatter, Christian - Wavelets, A Primer, A.K. Peters, Ltd. 1998

[Kaiser] Kaiser, Gerald - A Friendly Guide to Wavelets, Birkhauser 1994

[SN] Strang, Gilbert and Nguyen, Truong - Wavelts and Filter Banks, Wellesley-Cambridge Press 1996

[Walker] Walker, James S. - Fourier analysis and wavelet analysis. Notices of the AMS, vol. 44, No. 6, pp. 658-670, 1997

[Strang 1] Strang, Gilbert - Wavelets, American Scientist **82** (April 1994) 250 - 255

[Strang 2] Strang, Gilbert - Wavelets and Dilation Equations, Siam Review 31 (1989) 613-627

Appendix

This part contains the MATLAB code we have written for the image compression application of wavelets. We worked with the WaveLab v802 toolbox designed by the Statistics Department of Stanford University. We found it very helpful and we inspired our program from the examples it contained in this respect.
For reference, we offer the site where we downloaded it:

<http://www-stat.stanford.edu/ wavelab/>

```
ncoef= input('Enter the percentage of the DWT coefficients that
you want to keep:')
ncoef = (100-ncoef)/100;

%Presenting the image of Lenna

x=readimage('Lenna');
autoimage(x);
title('The Well-Known Lenna
Image');
uiwait;

%Presenting the image transform of Lenna

qmf=MakeONFilter('Daubechies',8);
wlenna=FWT2_PO(x,3,qmf);

wlenna_1=FWT2_PO(x,7,qmf);
y_1 = abs(wlenna_1);

y=abs(wlenna);

subplot(121);
autoimage(y_1);
title('Wavelet Transform - 1 Level')
subplot(122);
autoimage(y);
title('Wavelet Transform of Lenna - 5
Levels');
```

```
uiwait;

mesh(wlenna);
title('3D View of the Wavelet Transform');
uiwait;

%Ilustrating the non-zero elements of the WTransform matrix

coef_sort = sort(abs(wlenna(:)));

treshold = coef_sort(floor(ncoef*65536));
new_wlenna=wlenna.*(abs(wlenna)>treshold);

[i,j,v]=find(new_wlenna);
sp_lenna=sparse(i,j,v,256,256);
spy(sp_lenna);
title('Distribution of non-zero in the WT of
Lenna');
uiwait;

%Finding out how much space we need using
%a simple compression scheme

comp = zeros(1);
sz = size(x);
s = sz(1);
comp(1) = s;
n = 1;
onzero = 0;
for i=1:s
    for j=1:s
        if abs(new_wlenna(i,j)) > 0
            if onzero == 1
                comp(n) = 0;
                n = n+1;
                comp(n) = nzero;
                n = n+1;
            end
            comp(n) = new_wlenna(i,j);
            n = n+1;
        else
```

```
            if onzero == 1
                nzero = nzero+1;
            else
                onzero = 1;
                nzero = 1;
            end
        end
    end
end

disp('We compress in a vector of length');
disp(2*n);

disp('while the initial size is');
disp(2*s*s);

disp('and the compression ratio is')
disp((s*s)/n);

%Getting the image back

result = IWT2_PO(new_wlenna,3,qmf);
autoimage(result);
title('Lenna Image Restored (with 5\% of the WT coefficients)');
uiwait;
```