An Analysis of 2D Discrete Wavelet Transforms

Nitish A. Gupta
Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, Florida 32816, USA
nitish.gupta@knights.ucf.edu

***Abstract*—Frequency analysis of images in Fourier domain has wide range of applications, but it fails to localize those frequency components in spatial domain. This is where wavelets come into picture since they can localize the frequency components in the spatial domain. In this paper we will discuss the implementation of Haar and Daubechies wavelet analysis on an image. Later we will discuss their applications in image compression and compare wavelet compression with Fourier compression.**

I. Introduction

A. Haar Wavelet Decomposition

From the theory of Quadrature Mirror filter bank, we know that for a perfect reconstruction of a signal, we must have the aliasing term as zero, which in turn gives us the relation that the input and output filters, (essentially low pass and high pass filters) must be orthogonal to each other. Following this the discrete Haar wavelet function can be written as,

$$h_0[n] = \begin{cases} \frac{1}{\sqrt{2}}, & n = 0, -1 \\ 0, & \text{otherwise} \end{cases} \qquad h_1[n] = \begin{cases} \frac{1}{\sqrt{2}}, & n = 0 \\ -\frac{1}{\sqrt{2}}, & n = -1 \\ 0, & \text{otherwise} \end{cases}$$

Each and every filter used in wavelet analysis is nothing but a function of original low-pass filter H0. In case of 2D discrete signal like images, the decomposition can be carried out by first applying the low pass and high pass filter row-wise followed by down-sampling and then repeating the same process on the two outputs column-wise. This eventually gives us four different components, diagonal (high-high), vertical (high-low), horizontal (low-high) and approximation (low-low). The approximation component is the averaged and down sampled version of original image.

For full or complete decomposition, we iterate this process on the approximation component until we get a single valued approximation coefficient which cannot be decomposed further.

B. Reconstruction

For a perfect reconstruction, the above process has to be carried out in reverse order i.e. the filters are replaced by their corresponding orthogonal pair and the down-sampling is replaced by up-sampling by the same factor.

C. Daubechies Wavelet Analysis

The Daubechies Wavelet is a family of wavelets with different orders of wavelet functions. Haar wavelet is a special case of this family, also known as db1, with two-valued wavelet signal. In this paper, we will use the 2nd order Daubechies wavelet given by,

$$L = \frac{1}{4\sqrt{2}}\left[1+\sqrt{3} \quad 3+\sqrt{3} \quad 3-\sqrt{3} \quad 1-\sqrt{3}\right],$$

$$H = \frac{1}{4\sqrt{2}}\left[1-\sqrt{3} \quad \sqrt{3}-3 \quad \sqrt{3}+3 \quad -1-\sqrt{3}\right]$$

Since Haar is a two valued function, it is unable to address a large change in image intensity value. For this reason, we use a higher order member from this family to get a better representation. The decomposition and reconstruction process is similar in case of every wavelet transform.

II. Implementation

A. Decomposition

The first step in implementing any wavelet transform is designing the low pass filter h_0. Once we have selected the low pass filter, all other filters are function of h_0. Haar decomposition of an image *f*, is obtained by first passing it through a low pass filter and a high pass filter followed by down-sampling and then repeating the same procedure with the resulting two outputs column-wise. This will give us 4 components of first level decomposition as mentioned in section I(A) which are *a, h, v* and *d*. To decompose it further, we take only the 'a' part of previous decomposition. An image of size N x N, where N = 2^p, can only be decomposed *p* times. The down-sampling by two implies taking every alternate sample of the filtered signal. If the signal is $\left[a \ b \ c \ d\right]$, the down-sampling will return $\left[a \ c\right]$ and $\left[b \ d\right]$.

In case of Daubechies wavelet (generalization of Haar in a way), the only thing that changes from the Haar decomposition are the low pass filter coefficients. In this implementation, we use the db2 coefficients as mentioned in section I(C) as our low pass filter. The low pass filter can be obtained by shifting and translating the low pass filter. Once we have designed the filters, the decomposition is same as the Haar decomposition explained above.

B. Thresholding and Compression

We have seen that decomposition gives us four components, namely, a, h, v and d. We know that 'a' is the only low pass filtered component and we use it for further decomposition. The other three, h, v and d contain information above the changes in the image. An image consists of a very few edges as compared to the constant or non-varying regions. We can use this intuition to filter out the changes or small variations that comes due to varying light intensity or noise. This will not only improve the quality of our image but also results in compression. Although many well-known compression techniques have been developed using this intuition, here, we will simply use the process of thresholding for compression. After decomposition, we apply a threshold filter to the h, v and d component so that we only retain the actual edges in the image while getting rid of false variation and noise. We take the ratio of number of non-zero intensity elements in the original image to the number of non-zero intensity elements in the decomposed image and call it the **compression ratio**. Increasing the compression ratio leads to lower size and resolution. We can repeat the thresholding process at each level and observe the effect on the resulting image which is discussed in detail in the Results section.

C. Reconstruction

Before reconstructing a decomposed image, we design the inverse filters based on h_0. For Haar transform, the inverse filters are same as the forward filters. We first start with up-sampling each component column-wise from the lowest decomposition level. Up-sampling by a factor of two implies, padding zeros after every alternate row/column. For example, a matrix like this $[a\ c]$ gets up-sampled by two as $[a\ 0\ c\ 0]$ (column-wise up-sampling). Once all four components have been up-sampled, we then pass them through a corresponding inverse filter of the forward filter (If forward was Low pass filter, Inverse is High pass filter). We then add the filtered signals pairwise, $a + h$ and $v + d$ to get two signals. These two signals are again up-sampled and inversed to give a single output which is nothing but the reconstructed image of this particular level. This image is now treated as 'a' of the next reconstruction level and the above process is repeated till we get the final reconstructed image of the original image.

D. Complexity

The computational complexity of Fourier transform is $O(N^2)$ and of fast Fourier transform is $O(N \log N)$. However for wavelet transform of N x N image f(N), consists of two convolution i.e. with low pass and high pass filter each with cost 2N i.e. $O(N)$. We then split the signal into two part of size N/2 each and then recursively follow the same convolution and splitting which gives us n(N/2) where n represents the number of decompositions. Thus we get the total cost is 2N + n (N/2) which is nothing but $O(N)$. This shows that wavelet transform is computationally more efficient than Fourier transform.

III. Results

The above wavelet transform methods where evaluated several times with different combination of decomposition level and threshold values at each level. The Mean square error is calculated using the mean of squared difference between the pixels of original image and transformed image. The graph in figure 2 shows that for different compression ratios, the wavelet transform always performs better than Fourier transform. As can be seen from the images, the quality of images after compression on wavelet transform is a lot better than the quality of images after compression on Fourier transform. One thing to my surprise here is that the Haar wavelet is performing better then Daubechies wavelet. This might be due the wrapping of the last two terms in the db_2 matrix at the beginning of the same row/column so as to treat it as periodic.

$$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 & & & & & & \\ c_3 & -c_2 & c_1 & -c_0 & & & & & & \\ & & c_0 & c_1 & c_2 & c_3 & & & & \\ & & c_3 & -c_2 & c_1 & -c_0 & & & & \\ \vdots & \vdots & & & & & \ddots & & & \\ & & & & & & c_0 & c_1 & c_2 & c_3 \\ & & & & & & c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & & & & & & & c_0 & c_1 \\ c_1 & -c_0 & & & & & & & c_3 & -c_2 \end{bmatrix}$$

Fig. 1. Periodicity in db matrix

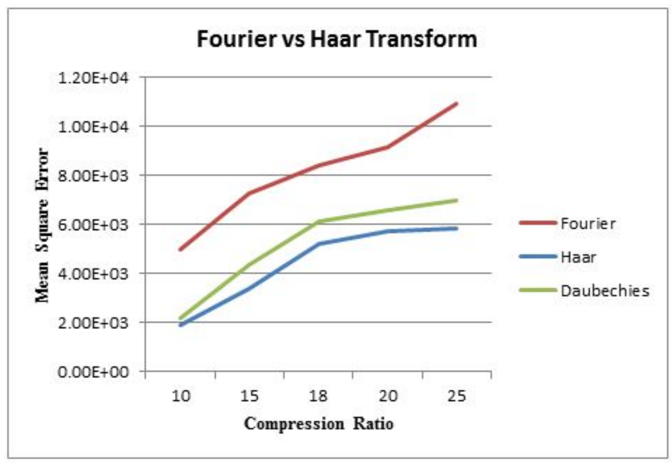

Fig. 2. Comparison of Fourier and Wavelet Transform

Fig. 3. (a) Fourier Transform, (b) 2-level Haar Transform with CR:10

IV. Conclusion

Wavelet transform is an excellent tool for visualizing frequencies in the spatial domain. Its applications in image compression have been widely studied and several efficient image compression using wavelet transform have been developed. We have seen that even simple threshold compression technique using wavelet transform out-performs Fourier transform.

Fig. 4. (a) Fourier Transform, (b) 2-level Haar Transform with CR:20

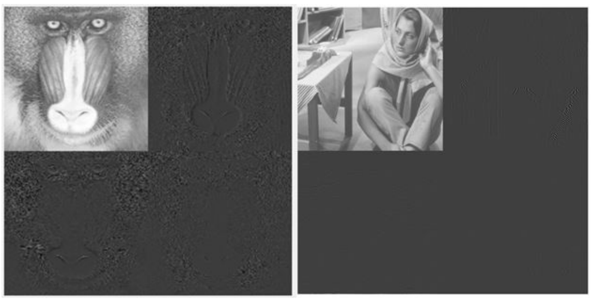

Fig. 5. a, h, v and d components of 1-level decomposition (a) Haar (b) Daubechies Transform

Fig. 6. a, h, v and d components of 2-level Daubechies Transform

In addition, the wavelet based transform is computationally more efficient than Fourier transform. In conclusion, wavelet transform proves to be a powerful tool in image processing.

References

[1] Haar Wavelet Image Compression https://people.math.osu.edu/husen.1/teaching/572/image_comp.pdf

[2] The Daubechies D4 Wavelet Transform http://www.bearcave.com/software/java/wavelets/daubechies/index.html