Averaging and Differencing

For simplicity to describe the Averaging and differencing process we take only the first row of an 8*8 matrix. This row is shown below. Because our matrix is 8*8 the process will involve three steps (2^3 =8)

[**3** 5 **4** 8 **13** 7 **5** 3]

Step 1

For the first step we take the average of each pair of components in our original string and place the results in the first four positions of our new string. The remain four numbers are differences of the first element in each pair and its corresponding average e.g. 3-4=-1, 4-6=-2, these numbers are called detail coefficients. Our result of the first step therefore contains four averages and four detail coefficients (bold) as shown

[4 6 10 4 **-1 -2 3 1]**

Step 2

We then apply this same method to the first four components of our new string resulting in two new averages and their corresponding details coefficients. The remain four detail coefficients are simply carried directly down from our previous step. And the result for step two is as follow.

[5 7 **-1 3 -1 -2 3 1]**

Step 3

Performing the same averaging and differencing to the remaining pair of averages completes step three. The last six components have again been carried down from the previous step. We know have as our string. One row average in the first position followed by seven detail coefficient

[6 **-1 -1 3 -1 -2 3 1**]

Haar Wavelet Transform model

The easiest of all discrete wavelet transformations is the Discrete Haar Wavelet Transformation (HWT). Analysis of the Two-Dimensional HWT You can see why the wavelet transformation is well-suited for image compression. The two-dimensional HWT of the image has most of the energy conserved in the upper left-hand corner of the transform - the remaining three-quarters of the HWT consists primarily of values that are zero or near zero. The transformation is *local* as well - it turns out any element of the HWT is constructed from only four elements of the original input image. If we look at the HWT as a block matrix product, we can gain further insight about the transformation.

Suppose that the input image is square so we will drop the subscripts that indicate the dimension of the HWT matrix. If we use *H* to denote the top block of the HWT matrix and *G* to denote the bottom block of the HWT, we can express the transformation as:

$$B = WAW^T = \begin{bmatrix} H \\ G \end{bmatrix} A \begin{bmatrix} H \\ G \end{bmatrix}^T = \begin{bmatrix} H \\ G \end{bmatrix} A \begin{bmatrix} H^T & G^T \end{bmatrix} = \begin{bmatrix} HA \\ GA \end{bmatrix} \begin{bmatrix} H^T & G^T \end{bmatrix} = \begin{bmatrix} HAH^T & HAG^T \\ GAH^T & GAG^T \end{bmatrix}$$

We now see why there are four blocks in the wavelet transform. Let's look at each block individually. Note that the matrix H is constructed from the lowpass Haar filter and computes weighted averages while G computes weighted differences. The upper

left-hand block is HAHT - HA averages columns of A and the rows of this product are averaged by multiplication with HT. Thus the upper left-hand corner is an approximation of the entire image. In fact, it can be shown that elements in the upper left-hand corner of the HWT can be constructed by computing weighted averages of each 2 x 2 block of the input matrix. Mathematically, the mapping is

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow 2 \cdot (a + b + c + d)/4$$

The upper right-hand block is HAGT - HA averages columns of A and the rows of this product are differenced by multiplication with GT. Thus the upper right-hand corner holds information about vertical in the image - large values indicate a large vertical change as we move across the image and small values indicate little vertical change. Mathematically, the mapping is

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow 2 \cdot (b + d - a - c)/4$$

The lower left-hand block is GAHT - GA differences columns of A and the rows of this product are averaged by multiplication with HT. Thus the lower left-hand corner holds information about horizontal in the image - large values indicate a large horizontal change as we move down the image and small values indicate little horizontal change. Mathematically, the mapping is

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow 2 \cdot (c + d - a - b)/4$$

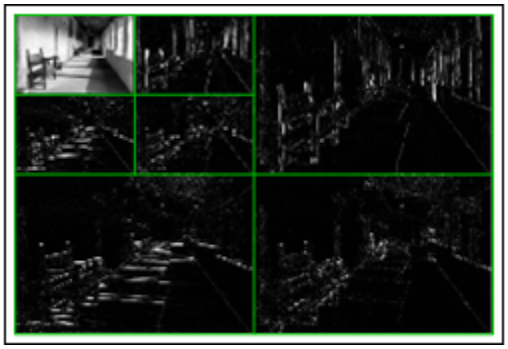
The lower right-hand block is differences across both columns and rows and the result is a bit harder to see. It turns out that this product measures changes along $\pm$ 45-degree lines. This is diagonal differences. Mathematically, the mapping is

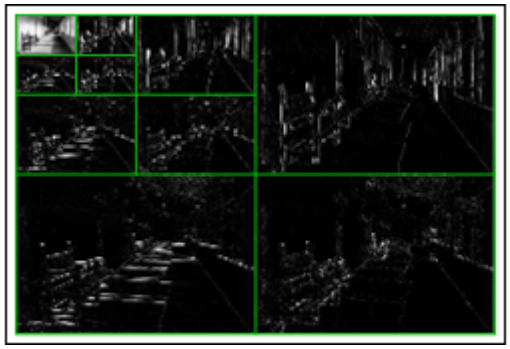
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow 2 \cdot (b + c - a - d)/4$$

To summarize, the HWT of a digital image produces four blocks. The upper-left hand corner is an approximation or blur of the original image. The upper-right, lower-left, and lower-right blocks measure the differences in the vertical, horizontal, and diagonal directions, respectively.

Iterating the Process

If there is not much change in the image, the difference blocks are comprised of (near) zero values. If we apply quantization and convert near-zero values to zero, then the HWT of the image can be effectively coded and the storage space for the image can be drastically reduced. We can iterate the HWT and produce an even better result to pass to the coder. Suppose we compute the HWT of a digital image. Most of the high intensities are contained in the blur portion of the transformation. We can iterate and apply the HWT to the blur portion of the transform. So in the composite transformation, we replace the blur by itstransformation! The process is completely invertible - we apply the inverse HWT to the transform of the blur to obtain the blur. Then we apply the inverse HWT to obtain the original image. We can continue this process as often as we desire (and provided the dimensions of the data are divisible by suitable powers of two). The illustrations below show two iterations and three iterations of the HWT.





$$x = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$x = {}^{1}/_{\sqrt{2}} \begin{bmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{bmatrix}$$

Top left: **a+b+c+d** = 4-point average or 2-D low pass (L0-L0) filter.

Top right : **a-b+c-d** = average horizontal gradient or horizontal highpass and vertical lowpass (Hi-L0) filter.

Lower left : **a+b-c-d** = Average vertical gradient or horizontal lowpass and vertical high pass (L0-Hi) filter.

Lower right a**-b-c+d** =diagonal curvature or 2-D highpass (Hi-Hi) filter

To apply this transform to a complete image, we group the pixels into 2*2 blocks and apply (3) to each block. The result (after reordering)is shown in figure 1(b). to view the result sensibly we have grouped all the top left sub image in figure 1(b) and done the same for the components in the other 3 positions to form the corresponding other 3 sub images.

E.g. (1).

$$x = \begin{bmatrix} 12 & -2 \\ -2 & 0 \end{bmatrix}$$

$$x' = \begin{bmatrix} 4 & 6 \\ 6 & 8 \end{bmatrix}$$

E.g. (2).

$$x = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

$$x' = \begin{bmatrix} 7 & -1 \\ -2 & 0 \end{bmatrix}$$