1 Class 2-D Wavelet: Complete Form

Many concepts are evident in the convolution method wavelet transforms. Why? It is true that a simple matrix multiply can also provide means of transformation. However, a convolution provides a simpler concept. One goal of these experiments is to measure computational strength of each method.

A transform must take a given matrix, and produce a result which contains four components: average, vertical, horizontal and diagonal difference. The can be done by either over complete or complete one dimensional wavelet transform means.

The over complete is addressed in the next class. A complete transform method returns a result matrix which is the same size as the source matrix. The result contains the four compoents. Each compoent resides on 4 corners of the matrix. Given a matrix B, the transform is to yield the following form:

$$B \Rightarrow \begin{matrix} H & D \\ A & V \end{matrix}$$

where A is the average matrix, H is the horizontal matrix, V is the vertical compoent, and D is the Difference Matrix. Why could this transform have the form:

$$B \Rightarrow \begin{matrix} A & V \\ H & D \end{matrix}$$

In truth, it is not obvious. In the first transform has the A compoent as the row and column average. So it is in the second. In both cases, V is the result of the difference row wise, and average column wise. Like wise, H is the average of the rows, and difference of columns. Finally, D is the difference of rows and difference of the columns. Furthermore, each case involve piping the results of row operation into the column operations or vise-versa.

This set of classes uses the first. Only reason for choosing such a method at the time of this classes construction is convention. Such a convention allows a standard comparision for accuracy, on a known solution. Further classes are to explore results via non-conventional means. Also, it is hoped that properties of each method will be revealed.

This class provides three functions. The column wavelet provides a transform solely on the columns. The row wavelet likewise, provides a wavelet transform on each of the rows. The 2-D wavelet transform is simple a row wavelet followed by a column wavelet transform.

It should be noted that these wavelet transforms are Haar Wavelet transforms.. Multiple resolutions functions can be made, but again it is still the Haar Wavelet at the core. Other class clones may be produced to use other wavelet basis such as Daubachie or Coeflet.

Another support class required for support of the two dimensional wavelet is the image reader/ writer. Such a class of functions provide means of acquiring data for a matrix to be computed. Granted this matrix could have been populated by other means. Generating such a class consumed some time on this project. The reason is that there a few mechanisms for doing this. One is to use raw image types. These types are known as by their extensions: pgm and ppm. Another popular mechanism is Image Magick. The reason for Image Magick is that it is available on most UNIX platforms (Linux, IRIX, Solaris, and Mac OSX). Otherwise, schemes such as Apple's Quicktime would be used. The other reason for the use of other software to acquire the image is that the objective of this class is not produce image translators for each image type imaginable.

1.1 Method: Column Wavelet Transform

The column wavelet transform is provides a source matrix, and returns a result. In order to yield this result, each column is extracted into a vector and that vector is fed into a one-dimensional transform. The result of the one dimensional transform is placed the corresponding row of the result matrix. Mathematically, this algorithm is as follows:

$\forall j \in col$
  n = 0
  $\forall i \in row$ in reverse order
    S_{n++} =source$_{i,j}$
  $S \stackrel{W}{\Rightarrow} R$
  n = 0
  $\forall i \in row$ in reverse order
    result$_{i,j}$ $= R_{n++}$

Note that the need for the n index is keep the order straight for this wavelets convention.

1.2 Method: Row Wavelet Transform

Like the column wavelet transform, the row wavelet transform takes a source matrix and returns a resulting matrix. The only two significant differences are first the items being operated on (rows not columns). Second, the lack of need for the n index.

$\forall i \in row$
  $\forall j \in col$
    S_j =source$_{i,j}$
  $S \stackrel{W}{\Rightarrow} R$
  n = 0

$\forall j \in col$ in reverse order
$\text{result}_{i,j} = R_j$

1.3 Method:Wavelet Transform

As stated above, the two dimensional wavelet transform is simply row wavelet transform followed by a column wavelet transform. It could have been done in reverse order. However, the result would be the same.

Just of note, this class lacks for the moment an inverse transform function. Not that such a thing does not exist mathematically. It simply was not implemented at the time of this document.

2 Class 2-D Wavelet Over Complete

The point of the over complete wavelet transform is to preserve the mathematical properties that are produced by the transform itself. Since the original signal can be recovered from the complete wavelet transform, either method can be used without energy loss. Also the over complete version takes up more space the original.

Once again, the reason for the over complete is to preserve the mathematical properties of the transform. Also, the over complete form allows for individual compoents to be computed without bothering with the remaining components being computed.

The functions included in the Over Complete 2-D Wavelet Transform class include computing the row average, row difference, column average and column difference. Also include are functions to compute the four 2-D wavelet matrix components (averages, vertical, horizontal, and diagonal difference). Some of these functions are simply wrappers around other class functions.

3 April 5, 2003

3.1 Multi-Resolution 2-D Wavelet Transform

The algorithm is as follows:

$\forall s \in resolution$

1. Let k = $\frac{rows}{2^s}$
2. Let l = $\frac{columns}{2^s}$
3. $\forall i \in k$ Row Transform on ith row up to column l.
4. $\forall j \in l$ Column Transform on the jth column up to the kth row.

The inverse transform is as follows:
$\forall s \in resolution$ in reverse order

1. Let k = $\frac{rows}{2^s}$
2. Let l = $\frac{columns}{2^s}$
3. $\forall i \in k$ Column Inverse Transform on jth column up to the kth row.
4. $\forall j \in l$ Column Transform on the ith row up to the lth column.

Sounds simple. What is needed for all of the xform and inverse xform calls to work are:

1. A Filter for Haar average
2. A filter for Haar difference
3. source matrix
4. result matrix

Inverse Transform components:
Column Inverse Xform:

1. let k = row
2. let k2 = $\frac{row}{2}$
3. $\forall i \in [0, k2)$
   - $sY[2i] = (T[i][j] - T[k2 + i][j])\sqrt{1/2}$
   - $sY[2i + 1] = (T[i][j] + T[k2 + i][j])\sqrt{1/2}$
4. $\forall i \in [0, k) T[i][j] = sY[i]$

Row Inverse Xform:

1. let k = column
2. let k2 = $\frac{column}{2}$
3. $\forall j \in [0, k2)$
   - $sX[2i] = (T[i][j] - T[i][k2 + j])\sqrt{1/2}$
   - $sX[2i + 1] = (T[i][j] + T[i][k2 + i])\sqrt{1/2}$
4. $\forall i \in [0, k) T[i][j] = sY[i]$

Note: sY and sX belong to the class and are allocated and deallocated in the wavelet inverse xform method.
Row X Form

1. s = column
2. s2 = column/2
3. $\forall k \in [0, s) xA[k] = xD[k] = 0.0$
4. $\forall k \in [0, s) \forall l \in [0, l)$
   - n = k -l
   - if ($n \in [0, s)$)
     - $xA[k] += W[i][n] * hA[l]$
     - $xD[k] += W[i][n] * hD[l]$
5. $\forall k \in [0, s2)$
   - $W[i][k] = xA[2k + 1]$
   - $W[i][k + s2] = xD[2k + 1]$

Column X Form

1. s = row
2. s2 = row/2
3. $\forall k \in [0, s) yA[k] = yD[k] = 0.0$
4. $\forall k \in [0, s) \forall l \in [0, l)$
   - n = k -l
   - if ($n \in [0, s)$)
     - $yA[k] += W[n][j] * hA[l]$
     - $yD[k] += W[n][j] * hD[l]$
5. $\forall k \in [0, s2)$
   - $W[k][j] = xA[2k + 1]$
   - $W[k + s2][j] = xD[2k + 1]$

The multi-resolution wavelet transform and inverse multi-resolution transform resembles the vector-matrix version. Convolution is built into the transform. However, there are structural changes.

The wavelet transform (multiresolution) uses private members of the class (hA, hD, xD/yD, xA/yA). Both Haar filters are maintained this way. Also both row and column transforms have average and difference myVector classes for temporary storage. All of these members are allocated and destroyed from the wavelet transform method itself. The simplified algorithm of the row transform is:

1. initialize xA and xD to zero

2. $\forall k \in columns \forall l \in filter$

   - n = k - l
   - if ($n \in columns$)
     $xA_k = W_{i,n} * hA_l$
     $xD_k = W_{i,n} * hD_l$

3. Transfer back to W
   $W_i = xA|xD$

1. initialize yA and yD to zero

2. $\forall k \in rows \forall l \in filter$

   - n = k - l
   - if ($n \in columns$)
     $yA_k = W_{i,n} * hA_l$
     $yD_k = W_{i,n} * hD_l$

3. Transfer back to W
   $W_j = yA|yD$

Note: W_i are the row vectors and W_j are the column vectors, and $W_{i,j}$ is the element from the ith row and jth column.

3.1.1 Computational Cost:

The cost of this algorithm is computed by the per row and per column cost, and then for the matrix and then for the multi-resolution steps. Per row the cost is 3k, where k is the number of columns. Per column the cost is 3l, where l is the number rows. For the whole matrix, one resolution costs 6kl operations to compute the wavelet transform. Per resolution, the rows and columns shrink by 2^i for each resolution, i, performed. The limit of this cost comes up to $12kl$ operations. Thus the cost is linear.

3.2 Expected Results

The expected result is a picture within a picture. Each average component has a further transform on it. The three resolution transform has the form:

$$W_3 = \begin{array}{cccc} A_3 & V_3 & V_2 & \\ H_3 & D_3 & & V_1 \\ H_2 & & D_2 & \\ H_1 & & & D_1 \end{array}$$

To obtain the inverse, an exact reverse procedure is necessary, otherwise the distortion is hideous.