Wavelets
CS 430
©Denbigh Starkey

Background

The discrete Fourier transform dominated image processing for over 40 years before beginning to get competition from wavelets. The major concepts of wavelet theory, with their corresponding Fourier concepts are shown side by side below:

Wavelet series expansion	Fourier series expansion
Discrete wavelet transform	Discrete Fourier transform
Fast wavelet transform (FWT)	Fast Fourier transform (FFT)
Continuous wavelet transform	Integral Fourier Transform

A number of different functions can be used as what is called the mother function in wavelet theory. I'll keep life simple and just look at the Haar approach.

First I'll look at a couple of approaches for computing 1D wavelets and then extend into 2D.

1D Wavelets – Approach 1

I'll be using the H_2 and H_4 Haar matrices that I developed in the previous set of notes:

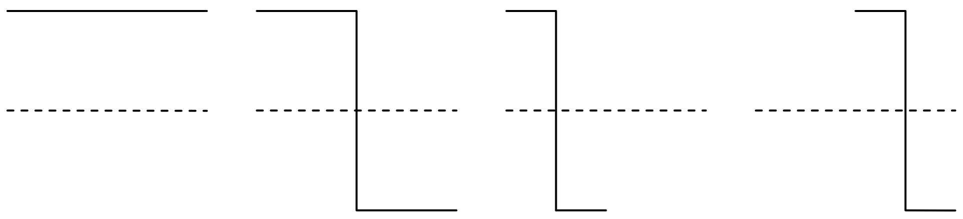
$$H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

$$H_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}.$$

As I discussed there, these are orthonormal matrices, and so the rows are all orthogonal (their dot products are 0) and have unit length, and so just as we can use the unit x, y, and z vectors, (1, 0, 0), (0, 1, 0), and (0, 0, 1) to define any point in three-space, we can use these basis function vectors to define any point in two- or four-space.

To develop these vectors, we can either use the formulae given in the notes on the Haar transform, or we can use a process called dilation and shifting. Take the 4×4 case and start with (1, 1, 1, 1) and (1, 1, -1, -1). To dilate the second one we take the two 1's and reduce them to a single 1, and do the same for the two -1's. We then fill with 0's to get (1, -1, 0, 0). To get the fourth Haar basis vector we shift this to get (0, 0, 1, -1). Normalizing these four vectors then gives the four rows of H_4. H_2 is easier because the first step gives us the two rows without needing to dilate or shift.

If we draw the four H_4 row vectors (not normalized) we get the shapes below:



The shapes of these curves give rise to the name wavelets.

Say that we want the 1D Haar transform of the message (3, 2, 2, -1) using H_4, and so we want to describe this as a linear combination of the four orthonormal basis vectors. To do this we take the dot product of the message to be encoded with each basis vector.

$$(3, 2, 2, -1) \,.\, \tfrac{1}{2}(1, 1, 1, 1) = \tfrac{1}{2}(3 + 2 + 2 - 1) = 3$$
$$(3, 2, 2, -1) \,.\, \tfrac{1}{2}(1, 1, -1, -1) = \tfrac{1}{2}(3 + 2 - 2 + 1) = 2$$
$$(3, 2, 2, -1) \,.\, \tfrac{1}{2}(\sqrt{2}, -\sqrt{2}, 0, 0) = \tfrac{1}{2}\sqrt{2}(3 - 2) = \tfrac{1}{2}\sqrt{2}$$
$$(3, 2, 2, -1) \,.\, \tfrac{1}{2}(0, 0, \sqrt{2}, -\sqrt{2}) = \tfrac{1}{2}\sqrt{2}(2 + 1) = \tfrac{3}{2}\sqrt{2}$$

So the 1D transform using these basis vectors is $(3, 2, \frac{\sqrt{2}}{2}, \frac{3\sqrt{2}}{2})$. To restore the original multiply the values by the vectors to get:

$$3\times \tfrac{1}{2}(1, 1, 1, 1) + 2\times \tfrac{1}{2}(1, 1, -1, -1) + \frac{\sqrt{2}}{2}\times \tfrac{1}{2}(\sqrt{2}, -\sqrt{2}, 0, 0)$$
$$+ \frac{3\sqrt{2}}{2}\times \tfrac{1}{2}(0, 0, \sqrt{2}, -\sqrt{2}) = (3, 2, 2, -1).$$

1D Wavelets – Approach 2

An approach that I prefer is to recursively encode the 1D vector. This is most easily shown by example. Say that we want to encode the 4-element vector (10, 6, 3, 5). We can average the first half of the vector to get 8, and the second half to get 4, and just store (8, 4) knowing that we can later expand this to get an approximation of the original vector, (8, 8, 4, 4). However obviously we've lost detail, but we can save that by also storing the differences from the averages. 8 is 2 less than 10 and 2 greater than 6, and 4 is -1 greater than three and -1 less than 5, and so we can encode the original as (8, 4, 2, -1) without any loss of information.

We can take this one level further by also averaging the 8 and the 4 to get 6, and will also need a difference element of 2 to be able to restore 6 back to 8 and 4.

Putting this all together, our vector (10, 6, 3, 5) has the encoding (6, 2, 2, -1). Note that the first element is an average of all of the original elements, just as the top left corner of the tiled image shown in the Haar notes was also an average image.

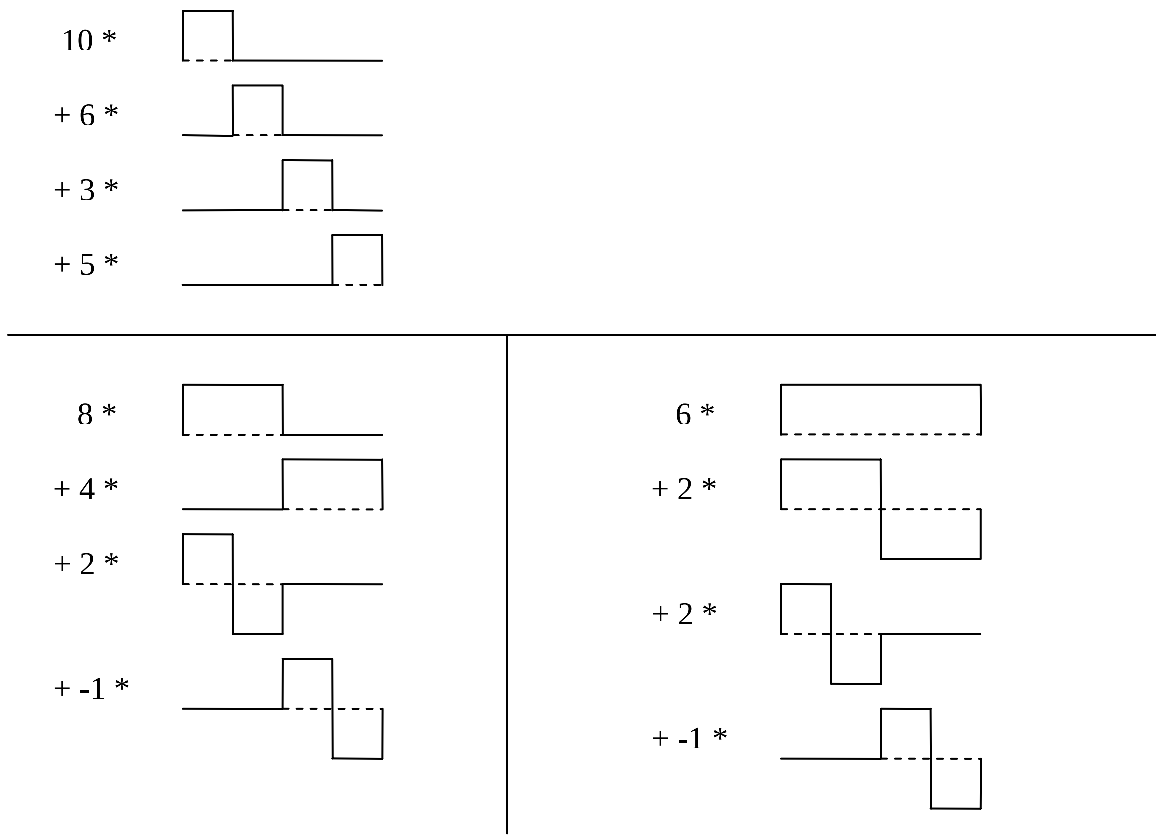
Restoring is easy. The first two elements of (6, 2, 2, -1) give (8, 4, 2, -1), and then the second half differences give (10, 6, 3, 5), our original vector.

Now let's look at this more graphically. The original vector can be represented as a waveform equation as shown in the top left of the diagram below (next page). Compare this to the waveform in the bottom left of the diagram and you will see that it has the same values at each point in the waveform. Finally compare it against the lower right, and again the summed values are the same.

Another thing to look at is the waveforms on the lower right represented as vectors. They are:

(1, 1, 1, 1)
(1, 1, -1, -1)
(1, -1, 0, 0)
(0, 0, 1, -1)

which are the same as the rows of the Haar matrix, H_4, not normalized.



If we do the same for the fist two waveforms on the lower right, resizing for 2×2, we get the vectors

(1, 1)
(1, -1)

which are the rows of H2, again not normalized.

In this example I've kept things simple by using numbers that lead to integer averages. However it is easy to double the numbers at each level to avoid fractions, so I'll ignore this detail.

One might ask what one has gained with the 1D transform. We started with a length 4 vector, and the encoded form is also length 4. In general a length N vector will be encoded as a length N vector. The advantages are the same as they were for the Haar image transform. The difference values tend to be very small, except in areas where there are dramatic changes in intensity in the original image, and so there are significant opportunities for compression since the histogram will be centered tightly around zero. We can also, as we could for the Haar images, change many of the values to zero with minimal loss of quality, while improving the compression even more.

Second Example – Length 8 vector: I'll show the recursive approach applied to the vector (6, 6, 8, 4, 2, 4, 5, 5). Taking pairwise averaging the algorithm will go through the following steps:

(6, 6, 8, 4, 2, 4, 5, 5)	-- original
(6, 6, 3, 5, 0, 2, -1, 0)	-- pairwise averages and differences
(6, 4, 0, -1, 0, 2, -1, 0)	-- 1st half pairwise averages and differences
(5, 1, 0, -1, 0, 2, -1, 0)	-- 1st pair average and difference

The last vector is the result of the encoding. Note how, apart from the first averaged element, the values are clustered around zero.

Say we were to now go to lossy compression by changing any 1 or -1 to zero. This will give the vector (5, 0, 0, 0, 0, 2, 0, 0), which is obviously very compressible. Restoring this to the original will give:

(5, 0, 0, 0, 0, 2, 0, 0)
(5, 5, 0, 0, 0, 2, 0, 0)
(5, 5, 5, 5, 0, 2, 0, 0)
(5, 5, 7, 3, 5, 5, 5, 5)

Which is close to the original, and has the same total intensity, but obviously not perfect. However the errors will become much less extreme as we go to a wider range of intensity values.

2D Fast Wavelet Transform using the 1D Transform

To get a 2D image transform we can, of course, use the matrix multiplication approach that I described in the Haar notes. A faster way to do the transform is to use a 1D transform on the rows and columns. Specifically, first do a 1D transform on each row, and then do a 1D transform on each column.