AMAT 503: Wavelets and Signal Processing

Michael P. Lamoureux

Lecture 9: February 6, 2018

Contents

0 Introduction

These notes follow the content of the course AMAT 503 at the University of Calgary. They focus on the theory of wavelets and their application to signal processing – in particular to 1D signals (such as sound) and 2D signal (such as photos). For a more in-depth analysis, please read the course textbook (which this year is Patrick van Fllet's "Discrete wavelet Transformations.")

1 Lecture 9: Daubechies Wavelet Transform

Ingrid Daubechies is a professor at Princeton. Her key idea was to worry less about the details of the filter response, and worry more about the mechanics of the computation. In particular, she found it was better to work directly with the filter coefficients that appear in the low and high pass filter of the transform.

1.1 Daubechies 4

So, for instance, in the Haar transform we have filter coefficients $(1/2, 1/2)$ and $(1/2, -1/2)$, we can instead imagine filters with four coefficients (h_0, h_1, h_2, h_3) and (g_0, g_1, g_2, g_3) and do the same construction of a transform matrix. Then ask: what conditions on the $h's$ and $g's$ give us an orthogonal matrix.

Let's do the 8x8 case. For these short, length-four convolution filters, we get a matrix like this:

$$W = \begin{bmatrix} h_3 & h_2 & h_1 & h_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_3 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_3 & h_2 & h_1 & h_0 \\ h_1 & h_0 & 0 & 0 & 0 & 0 & h_3 & h_2 \\ g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_3 & g_2 & g_1 & g_0 \\ g_1 & g_0 & 0 & 0 & 0 & 0 & g_3 & g_2 \end{bmatrix}. \tag{1}$$

Remember, like with the Haar matrix, we start with a Toeplitz matrix (constant along diagonals), and throw away every other row. That's why the h-values look like they get shifted by 2 as you go down by rows. Same for the g's. Now, half way through the matrix, at the end of the h-rows. we run out of space for the $h's$, so we wrap around back to the first to columns. Similar thing at the end of the g-rows: just wrap around.

Now, here's a neat trick. To get the h-rows perpendicular to the g-rows, just force the g's to be the same values as the h's, except in reverse order. That is, we write

$$(g_0, g_1, g_2, g_3) = (h_3, -h_2, h_1, -h_0). \tag{2}$$

For if we do that, the dot product is zero:

$$(h_0, h_1, h_2, h_3) \cdot (g_0, g_1, g_2, g_3) = (h_0, h_1, h_2, h_3) \cdot (h_3, -h_2, h_1, -h_0) = h_0h_3 - h_1h_2 + h_2h_1 - h_3h_0 = 0. \tag{3}$$

To get the first two rows orthogonal, we need that

$$(h_3, h_2, h_1, h_0, 0, 0, 0, 0) \cdot (0, 0, h_3, h_2, h_1, h_0, 0, 0) = h_0h_2 + h_1h_3 = 0. \tag{4}$$

By symmetry in the matrix, once the first two rows are orthogonal, they all are.

We also want the rows to have unit length: that is easily satisfied by requiring

$$h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1. \tag{5}$$

Now, the h's should give us a low pass filter – in particular, they need to "kill" the high frequencies, so we should require that their Fourier transform has a zero at Nyquist frequency $\omega = 1/2$. Thus we require

$$0 = H(1/2) = h_0 e^{2\pi i 0\cdot(1/2)} + h_1 e^{2\pi i 1\cdot(1/2)} + h_2 e^{2\pi i 2\cdot(1/2)} + h_3 e^{2\pi i 2\cdot(1/2)}. \tag{6}$$

Simplified, this just reads

$$0 = h_0 - h_1 + h_2 - h_3. \tag{7}$$

A careful count of the above considerations shows we have 3 equations in the 4 unknowns h_0, h_1, h_2, h_3, while the g's are completely determined by the h's. Let's add one more equation to get a solvable system: we require that the derivative of $H(\omega)$ is also zero at $\omega = 1/2$.. Thus

$$0 = H'(1/2) = h_0 \cdot 0 e^{2\pi i 0\cdot(1/2)} + h_1 \cdot 1 e^{2\pi i 1\cdot(1/2)} + h_2 \cdot 2 e^{2\pi i 2\cdot(1/2)} + h_3 \cdot e^{2\pi i 2\cdot(1/2)}. \tag{8}$$

Simplified, we have

$$0 = h_1 - 2h_2 + 3h_3. \tag{9}$$

Thus, the 4 equations in 4 unknowns are

$$\begin{aligned} h_0^2 + h_1^2 + h_2^2 + h_3^2 &= 1 && (10)\\ h_0 h_2 + h_1 h_3 &= 0 && (11)\\ h_0 - h_1 + h_2 - h_3 &= 0 && (12)\\ h_1 - 2h_2 + 3h_3 &= 0. && (13) \end{aligned}$$

. A solution is given by

$$\begin{aligned} h_0 &= \frac{1}{4\sqrt{2}}(1+\sqrt{3}) && (14)\\ h_1 &= \frac{1}{4\sqrt{2}}(3+\sqrt{3}) && (15)\\ h_2 &= \frac{1}{4\sqrt{2}}(3-\sqrt{3}) && (16)\\ h_3 &= \frac{1}{4\sqrt{2}}(1-\sqrt{3}). && (17) \end{aligned}$$

As an exercise, check that this really is a solution. If you put all the h's with opposite sign, you get another solution. Are there others??

We can plot the frequency response of the filter given by the h's, to verify it is a low pass filter. You should also plot the frequency response for the corresponding g's, to verify that it is a high pass filter. You might even like to verify that the resulting matrix W really is orthogonal. Using MATLAB, check that

$$W^* W = I. \tag{18}$$

It's not too hard to see how to generalize this to any nxn matrix, so long as n is even.

This matrix transform is called the Daubechies-4 wavelet transform.

1.2 Odd Daubcechies

Ask yourself, why didn't we use a filter of length 3? Turns out it can't work, for trivial reasons. Try it out. The general Daubechies construction only works for even length filters.

1.3 Daubechies 6

We can do the same construction with filters of length six. The 6 equations obtained are

$$\begin{aligned} h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 &= 1 \qquad (19) \\ h_0h_2 + h_1h_3 + h_2h_4 + h_3h_5 &= 0 \qquad (20) \\ h_0h_4 + h_1h_5 &= 0 \qquad (21) \\ h_0 - h_1 + h_2 - h_3 + h_4 - h_5 &= 0 \qquad (22) \\ h_1 - 2h_2 + 3h_3 - 4h_4 + 5h_5 &= 0 \qquad (23) \\ h_1 - 4h_2 + 9h_3 - 16h_4 + 25h_5 &= 0. \qquad (24) \end{aligned}$$

Where do these equations come from? Well, the first says we want rows of unit magnitude. The second says we need rows 1 and 2 to be perpendicular to each other. The third says we want rows 1 nd 3 to be perpendicular to each other. The fourth, fifth and sixth are requirements on the frequency response, namely that $H(\omega)$ and its first two derivatives should be zero at Nyquist $\omega = 1/2$. That is,

$$H(1/2) = H'(1/2) = H''(1/2) = 0. \qquad (25)$$

Six equations in six unknowns, turns out we can solve this. Done. That gives the Daubechies 6 wavelet transformation.

What about g. Well, same trick as before: g is the reverse of h, with alternating signs.

1.4 Daubechies 2N

In general, we can do this for filters of even length 2N. Get 2N equations, in 2N unknowns (the h's). These have been solved numerically for reasonable value sod N. You can find them in MATLAB, or anywhere on the web.

We plot the low pass filter response for the five different Daubechies wavelet transforms, shown in Figure 1. Notice the higher order transforms give a sharper cutoff in the frequency response.

It is worth noting that the textbook's code gives slightly different Daubechies coefficients than what the MATLAB wavelet toolkit gives. In the book, $H(0) = \sqrt{2}$, while in MATLAB, one gets

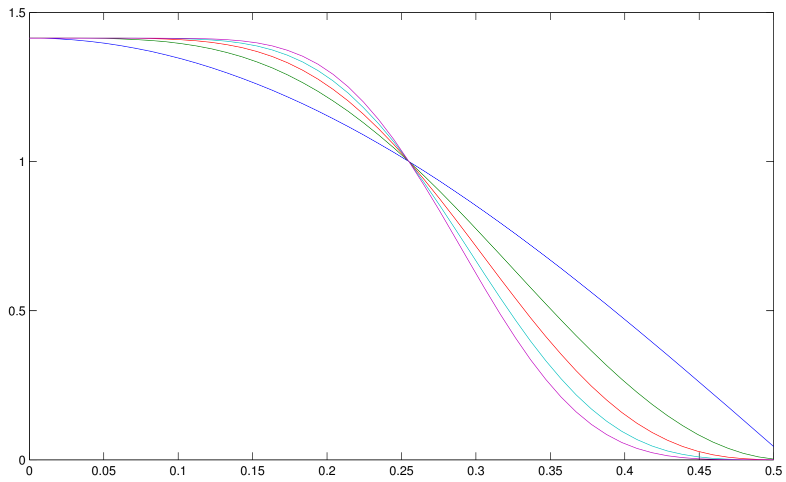

Figure 1: Low pass filter response for Daubechies transforms of length 2,4,6,8,10.

$H(0) = 1$. These are just a different normalization, but you should be aware that there is a difference.

We will now look at some code to use these transforms

2 Lecture 9+: Wavelet transforms of images

A quick and dirty way to do a 2D transform is to apply a 1D transform twice, once in the horizontal direction, and then again in the vertical direction. Not always the best thing to do, but it is a start.

For a 2D image in black and white, you can represent the image as a matrix of row and columns, with each entry in the matrix corresponding to the intensity at the aligned pixel location. To apply the Haar wavelet transform in 2D, you first apply it to every row in the matrix, and then on the resulting output, apply it to each column.

To iterate, you locate one corner of the matrix corresponding to the low pass output (in both horizontal and vertical) and apply the transform just to that corner. Wash, rinse, and repeat.

2.1 Some sample transforms

We can load in some black and white images and apply the Haar wavelet transform to both rows and columns – this give a 2D transform. Some code:

```
gry=ImageNames('ImageType','Grayscale');
file=gry{7};
A=ImageRead(file);
ImagePlot(A)
B = HWT2D1(A);
ImagePlot(B)
C = HWT2D(A,2);
ImagePlot(C)
```

The matrix A is the original image of a dog, matrix B is the first iteration of the Haar transform, and C is the 2nd iteration. These are displayed in Figure 2. It is interesting to note that in the first iteration, the lowpass filtered version fits in the upper left corner of the output, taking up 25% of the data space. The high pass outputs are in the other corners, and contain much less energy of the data (which is why it looks so black). You should try this code yourself and see what is sitting in those high pass output images – you will see some of them reveal vertical features in the original image, others reveal horizontal features. This is a very crude, but fast way of doing edge detection.

When the algorithm iterates, it only acts on the low pass "corner" of the transformed image, and leaves the other 3 corners the same. You can see the top left corner in the first iteration gets shrunk to a smaller corner in the next iteration. For the rest, it is hard to see in these images (everything is close to black), so you should try the code yourself. Or read the code provided by the textbook author.

Under the philosophy that we do transforms to concentrate energy, we can plot the cumulative energy for the image and its two transforms. This is shown in Figure 3. What you should notice is that the CE for the transformed data rises to 1 much more quickly. Indeed, in the 2nd iteration, we get 95% of the energy in just the first 5% of the coefficients. For the original image, it takes about 80% of the coefficients to meet that same level of energy. Which is to say, there is a lot more opportunity to compress in the transformed data.

2.2 Operations count

Try to Figure this out. Take a $m \times n$ matrix. Each row has n entries, so applying the 1D Haar transform to 1 row takes 2n operations. To do all the rows takes 2mn operations. Applying to the resulting columns takes another 2mn operations, so total is 4mn operations for one step of the Haar transform.

When iterating, we only apply to the corner, which is size $\frac{m}{2} \times \frac{n}{2}$. Thus we only need 4 times $mn/4$ operations to get the second iteration.

To do all the iterations takes a total of

$$4mn(1 + \frac{1}{4} + \frac{1}{16} + \cdots) = 4mn\frac{1}{1 - 1/4} = 16mn/3, \tag{26}$$

or roughly $5mn$. Anyhow, it is linear in the size of the original data, which again indicates that this is a very fast transform.

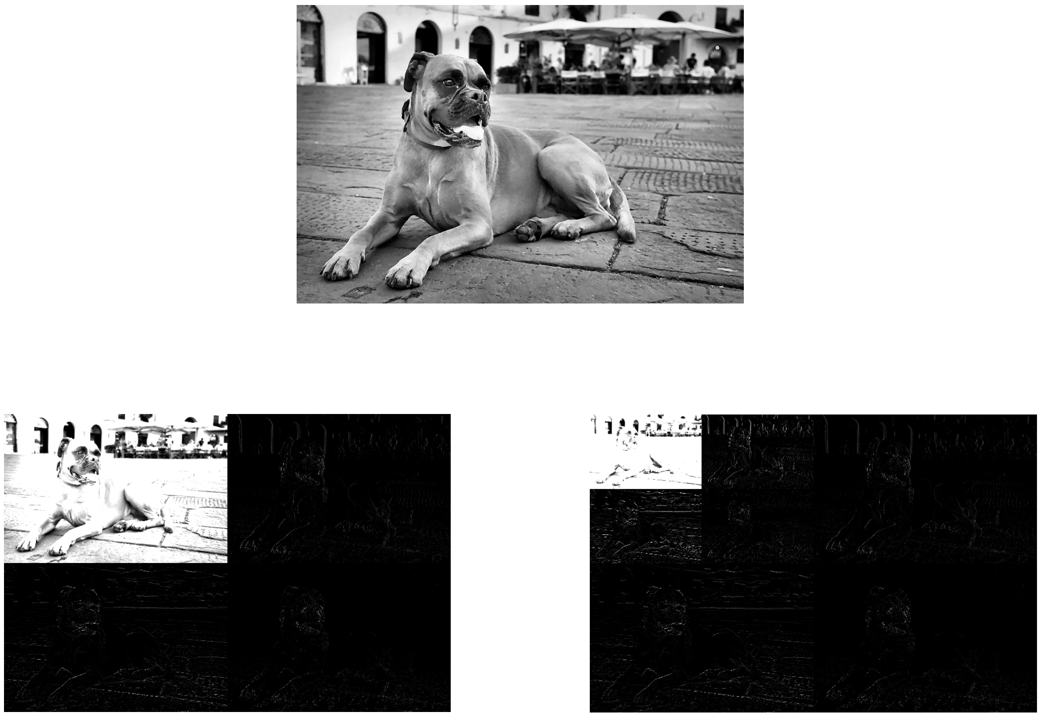

Figure 2: An image, and 2 iterations of the Haar transform in 2D.

Try to figure out how many operations are needed when using the Daubechies wavelet transform in 2D. Again, you should find that this is also a fast transform.

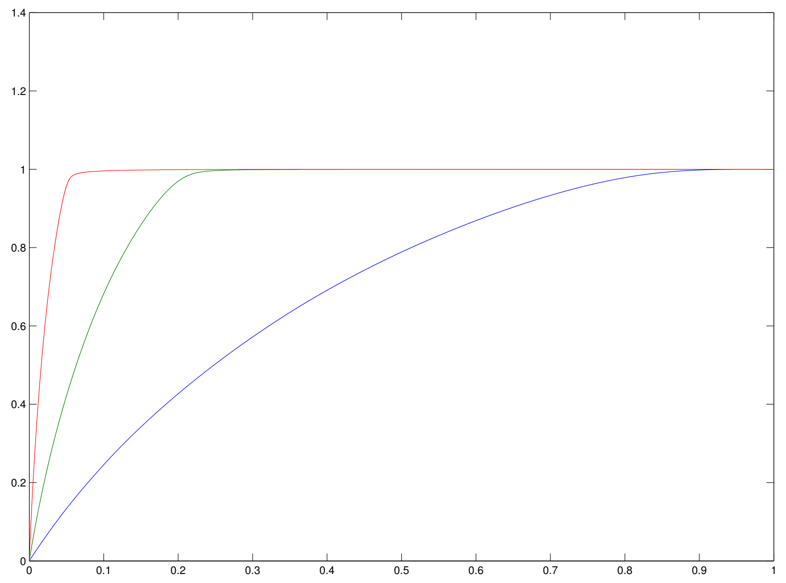

Figure 3: The cumulative energy for the image, and 2 iterations of the Haar transform.