ECE 732
Project 3
Wavelets, Compression, and Denoising

Due: 5pm, 20 December 2003

This project investigates the discrete wavelet transform (DWT) and its application to signal and image compression and denoising. First we will investigate the 1-d DWT, then move on to multiscale image analysis using the 2-d DWT.

The 1-D DWT and Signal Denoising

1. Consider three length-4 scaling filters parameterized by $\alpha = 0.1\pi,\ \pi/3,\ 0.9\pi$ (see class notes for the filter expression in terms of α).

a. Verify that all three filters satisfy the sufficient conditions of Cohen's theorem.

b. Using the method of successive approximations, plot the scaling function associated with each filter. Describe the behavior of each function and assess the potential of each.

c. Propose your own length-4 scaling filter that meets the requirements in Cohen's theorem, and compare it to the other three.

d. The basic recursive scaling equation allows us to express the wavelet function in terms of the scaling function and the filter $h_1(n) = (-1)^{(1-n)}h(1-n)$. Using this relationship and the approximate scaling functions obtained above, plot the wavelet function associated with each scaling function.

2. Compare the performance of the four scaling filters for 1-d signal denoising.

a. Generate the test signal 'Doppler' (x) using `makesig.m`.

b. Generate a white Gaussian noise sequence w of standard deviation $\sigma = 0.05$ using built-in Matlab function `randn.m`.

c. Add noise sequence to test signal sequence, $y = x + w$.

d. For each scaling filter, compute the DWT of the noisy signal using Matlab's Wavelet Toolbox or using `mdwt.m` from the Rice Wavelet Toolbox.

d. Threshold the wavelet coefficients of y; set coefficient's whose magnitude is less than 3σ to zero, leave other coefficients unaltered.

e. Compute inverse DWT of thresholded wavelet coefficients using Matlab's Wavelet Toolbox or `midwt.m` from the Rice Wavelet Toolbox to obtain a signal estimate $\widehat{x}$.

(continued on next page)

The 2-D DWT and Multiscale Image Analysis

3. Develop a Matlab function for computing the J-level, $J = 1, 2, \ldots, \log_2 N$, Haar wavelet transform of an $N \times N$ image (assume N is a power of 2). Your function can repeatedly call the function `haar.m` that I wrote (downloadable from the course website) or you can start from scratch. Your function should take two arguments: an input image and the number of levels J you wish to compute. Your function should output an array of $N \times N$ wavelet coefficients (in the arrangement discussed in class).

4. Develop a Matlab function for computing the inverse J-level, $J = 1, 2, \ldots, \log_2 N$, Haar wavelet transform of an $N \times N$ array of Haar wavelet coefficients. Your function could repeatedly call the function `ihaar.m` which reverses the process of `haar.m`, for example. Your function should take two arguments: an input array of Haar wavelet coefficients and the number of levels J in that array. Your function should output an image reconstructed from the input coefficients.

5. Test your forward and inverse Haar transform code by applying it to the `camera` image (downloadable from the course website). You can test it with other (grayscale) images if you want, too.

(continued on next page)

Experiments with the 2-D DWT in Compression, Denoising and Deconvolution

6. *Image Compression:* Compare the compression performance of the Haar wavelet transform with the Fourier transform using the test image(s). Compute the transforms, set all coefficients to zero except for the largest (in magnitude) 25%, 10%, 5%, and 1% and reconstruct an approximation to the original image by applying the corresponding inverse transform. This simulates the process of compressing by factors of 1/4, 1/10, 1/20, and 1/100. Display the resulting images and comment on the relative quality of wavelet vs. Fourier results.

7. *Image Denoising:* Compare the denoising performance of the Haar wavelet transform with the Fourier transform (`fft2`) using the test image(s). Add a small amount of Gaussian white noise (with variance σ^2) to each image, compute the transforms, set all coefficients to zero except those whose magnitude is larger than 3σ (you can also try factors other than 3), and reconstruct an estimate of the original image by applying the corresponding inverse transform. Note that we have defined the Haar wavelet transform to be orthonormal, so the noise variance in the wavelet coefficients is σ^2 (the same as in each pixel). However, Matlab's `fft2` is not normalized, and you should verify the proper normalization level (I think you should use a threshold of $3N\sigma$ to account for the lack of normalization). Display results and comment on the difference between the wavelet and Fourier methods.

8. *Denoising vs. Wiener Filtering:* Construct a `fft2` based Wiener filter and compare its performance to the Haar wavelet and `fft2` denoising schemes investigated above. Display results and comment on the difference between the denoising and Wiener methods.

9. *Wavelet-based Image Restoration:* Develop a wavelet-based approach to image restoration using a two-step procedure: First, deconvolve the image using an "aggressive" Wiener filter (this should result in a sharp, but very noisy, image); second, devise an appropriate wavelet denoising operation to reduce the noise that was amplified by Wiener filtering. Note that the noise will be colored (non-white) after the first Wiener filtering step. Wavelet coefficients at fine scales will contain a larger level of noise than those at coarse scales. The precise amount of noise in each wavelet subband depends on the Wiener filter's frequency response and the frequency support of each wavelet. Derive an *exact* expression for the noise variance in each wavelet subband (assuming the original blurred image was contaminated with Gaussian white noise with variance σ^2), and propose an appropriate threshold level for the coefficients in each subband based on the noise variance (e.g., three times the standard deviation of the noise in each coefficient). Test your method on the deblurring problem studied in Part (e) of Homework 4, and compare your wavelet-based approach to Wiener filtering alone. Experiment with different noise levels, different blurring functions, and different images.