EE3731C SIGNAL PROCESSING METHODS: PROJECT

(Percentage of final mark: 20%)

In this ***individual*** project, you will investigates the discrete wavelet transform (DWT) and its application to image denoising and image compression, as well as, signal filtering.

In case you do not have access to Matlab's Wavelet Toolbox, you can download the Rice Wavelet Toolbox (https://github.com/ricedsp/rwt). Before you start, you should download the relevant files (**camera.mat, lena512.mat, haar.m**, **ecg_hfn.dat,** and **ecg_hfn.m**) from the the IVLE workbin, and get familiar with the following Matlab functions: **dwt2, idwt2, wavedec2, waverec2,** and **filter**.

1. Image Denoising

(a) Load the **camera** image and display it:

```
>> load camera.mat; imagesc(im); colormap 'gray'; axis image;
```

(b) Develop a Matlab function **haar_dec.m** for computing the J-level, $J = 1, 2, \ldots, \log_2 N$, Haar wavelet transform of an $N \times N$ image, where N is a power of 2. Your function can repeatedly call the function **haar.m** (downloadable from the IVLE workbin) which computes 1-level of the 2-D Haar wavelet transform. Your function should take two arguments: an input image *im* and the number of levels J, and output an array of $N \times N$ wavelet coefficients C in the arrangement as illustrated in Figure 1. Test your Matlab function **haar_dec.m** on the **camera** image (i.e., im) for $J = 3$. [Hint: you can use **wavedec2** to verify your results.]

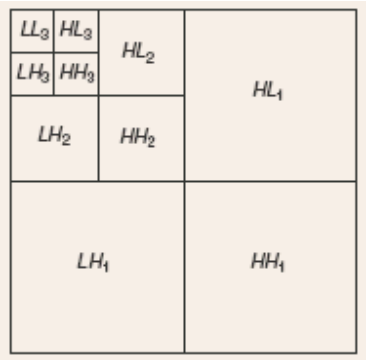

Figure 1: Standard 2-D DWT decomposition.

(c) Develop a Matlab function **haar_rec.m** 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 a function **ihaar.m** (you need to write it yourself) which reverses the process of **haar.m**. Your function should take two arguments: an input array of Haar wavelet coefficients C and the number of levels J in that array. Your function should output an image reconstructed from the input coefficients. Test your Matlab functions on the **camera** image for $J = 3$. [Hint: you can use **waverec2** to verify your results.]

(d) Add a small amount of Gaussian white noise (with variance σ^2, e.g., σ=10) to the image:

```
>> sigma = 10; im1 = im + randn(size(im))*sigma;
```

(e) Display the noisy image and report its peak signal-to-noise ratio (PSNR). (Look up for the definition of PSNR if you are not familiar with the term.)

(f) Compute the 4-level (i.e., $J = 4$) Haar wavelet transform of the *noisy* image using your **haar_dec.m**. Threshold the coefficients by setting the small detail coefficients to zero, i.e., change the coefficients whose *magnitude* is less than 3σ to zero and keep the other coefficients. Display the thresholded coefficients in 2D, and discuss your observation.

(g) Reconstruct the denoised image by computing the inverse transform of thresholded wavelet coefficients using **haar_rec.m**. Display the denoised image and report the PSNR.

(h) Repeat steps (d)-(g) to perform image denoising for σ = 10, 20, 30, 40, 50, 60, 70, and 80. Plot PSNR versus σ, and discuss your observation.

(i) Instead of using hard-thresholding as in step (f), design and implement a better thresholding scheme. For σ = 20, display the denoised image and report the PSNR.

2. Image Compression

(a) Load the **Lena** image and display it;

(b) Compute the 5-level (i.e., $J = 5$) Haar wavelet transform of the **Lena** image;

(c) Set all coefficients to zero except for the largest (in magnitude) 20%, 10%, 5%, and 1%; reconstruct an approximation to the original image by applying the corresponding inverse transform. This simulates image compression by factors of 1/5, 1/10, 1/20, and 1/100. [Hint: You can use Matlab function **sort** to sort array elements in descending order.]

(d) Display the reconstructed images and report their PSNR values. Comment on the visual quality of the reconstructed images.

3. Signal Filtering

In MATLAB, the output y(n) of the filter can be found for an input x(n) by using the **filter** command. (Note: Pay attention to the significance of z, z^{-1}, and the notation of the filter coefficient array in MATLAB.)
The command y = filter(b, a, x) filters the data in the array x with the filter described by the arrays a and b to create the filtered vector y. The filter is a "Direct Form II Transposed" implementation of the standard difference equation:

y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)- a(2)*y(n-1) - ... - a(na+1)*y(n-na),

where na is the feedback filter order, and nb is the feedforward filter order, and a(1) = 1.

A Noisy ECG signal is provided in the file **ecg_hfn.dat** (available in the IVLE workbin; **ecg_hfn.m** can be used to view the signal.). The sampling rate of this signal is 1000Hz.

(a) Apply the Hanning filter whose transfer function is given by

$$H(z) = \frac{1}{4}[1 + 2z^{-1} + z^{-2}]$$

Specify the filter in terms of the a and b arrays via the **filter** command in MATLAB, and plot the ECG signal before and after filtering.

(b) Plot the power spectrum of the ECG signal before and after filtering, using the **spectrum** object and the **psd** command. Discuss your observations.

What to Submit

1. A well-written, concise project report.
2. Your source code.

Project Due: 11:59pm, Friday, Nov 15, 2013

Online Submission Checklist

You must submit the following **electronic** files in **one zipped folder** using **IVLE** tools.

- The pdf file of the project report;
- Source code and, if necessary, a readme file containing instructions to run your code.