Purpose: Learn how to apply wavelet transform techniques to grayscale image data.

Procedure: For *each* MATLAB m-file you write, be sure to comment the code sufficiently, and include initial comments as a crude help system. For this project, **you are free to use any programs you like in the Image Processing Toolbox, the Wavelet Toolbox, or the Signal Processing Toolbox**. To see what is available in these Toolboxes, start MATLAB and type `help images`, or `help wavelet`, or `help signal`, respectively. However, be aware that many such Toolbox programs often perform unexpected steps and conversions silently. Unless you read the help files for the programs *carefully*, you can end up with subtle (and some not-so-subtle) errors in your results. Consider yourself warned!

You will be performing various wavelet transform techniques on two different monochrome images. One image, of size 256×256, is called `iptest256a` which is stored in the MATLAB MAT file called `iptest_im.mat`. The other image, of size 512×512, is called `B512` which is stored in the MATLAB MAT file called `lena_noise.mat`. Both are 8-bit grayscale images. The first image is an artificially created test image, and the second one is a moderately noisy version of the ubiquitous Lena image. You can find both of these MATLAB MAT files in the Files→Images directory of the course web site.

In most cases, you will be relatively free to choose which wavelets to use, but to keep a reasonable scope on this project let's restrict the choices to the following. For **orthonormal** wavelets: **Haar**, any of the classic **Daubechies** family (called `db1` to `db10` in MATLAB), and any of the **symlet** family (called `sym2` to `sym8` in MATLAB); for **biorthogonal** wavelets, use the standard choices that come with the MATLAB Wavelet Toolbox (called `bior1.1` to `bior6.8` in MATLAB).[1] This range of choices will be sufficient, I assure you. Note that in the Wavelet Toolbox the names `haar`, `db1`, and `bior1.1` all refer to exactly the same thing.

There will be three main activities for this project. First, you'll familiarize yourself with the various wavelets and how to decompose and reconstruct an image. Second, you'll implement some very basic edge detection using wavelets. Third, you'll try to "clean up" a noisy image using wavelet denoising techniques. The first two activities must be accomplished primarily by your own MATLAB code. While you can use a similar approach to example code I've shown you, and you can make calls in your code to Toolbox commands, such "outside help" cannot constitute the primary part of your program(s). *You must write your own code.* For the third activity (denoising), you will try some very basic denoising using your own code, then compare the results to what you can obtain using the interactive denoising options of the MATLAB Wavelet Toolbox.

- **Familiarization**

  - Start the MATLAB Wavelet Toolbox using the `wavemenu` command.[2] Click the Wavelet Display button and use the associated GUI to become familiar with the Haar, Daubechies, symlet, and biorthogonal wavelets that are available. Write a MATLAB program to display the filter coefficients and the associated frequency domain magnitude and phase of specified wavelet filters. You may want to take some ideas from the code provided in the examples `subband_filt_ex??.m` or `wavelet_FWT_ex??.m`, but *don't* just duplicate any of that code for your own programs (yes,

[1] The most widely used biorthogonal wavelets, CDF 9/7, actually have several minor variations listed in the literature. One variation of CDF 9/7 is in the MATLAB Wavelet Toolbox as `bior4.4`. The MathWorks has free m-files on their website which implement other variations of CDF 9/7 if you ever need to do that.

[2] The `wavemenu` command is being replaced by the `waveletAnalyzer` command, but the former still works for now.

I recognize it immediately). You may also want to explore the `wfilters` command to directly obtain the exact values of the filter coefficients. Include figures in your report that show all four sets of coefficients (the lowpass and highpass decomposition filter coefficients, and the lowpass and highpass reconstruction filter coefficients), and their associated frequency domain magnitude and phase, for each wavelet family. Do this, in particular, for the following wavelets: `haar`, `db4`, `sym4`, and `bior4.4`. In your report, **comment on what you observe** about how the magnitude and phase characteristics compare for these particular wavelets. When labeling the FIR filter coefficients, use the common wavelet labels ($h_\phi(-n)$, $h_\psi(-n)$, $h_\phi(n)$, $h_\psi(n)$) rather than the equivalent common subband coding labels (h_0, h_1, g_0, and g_1). Be consistent with this throughout your report.

- ➪ Write a second MATLAB program that performs a *three-level* wavelet decomposition/reconstruction on the `iptest256a` image from the `iptest_im.mat` file. The program should generate figures that show the original image, the decomposition *in a tiled format* similar to your text, the reconstructed image, and a scaled difference image comparing the original to the reconstruction. Experiment with using various wavelets for this and comment intelligently and critically on what you observe about the results. Include as a minimum figures in your report showing the three-level decomposition of this figure using `haar` wavelets and `sym4` wavelets. Results with other wavelets can be shown if desired.

- **Edge Detection**

  - ➪ Using a similar approach to the very simple wavelet-based edge detection technique described in your text (Example 6.23 on pp. 512–514), write a MATLAB program to find **all** the edges of the `iptest256a` image, using at least two levels of decomposition. Include figures in your report that each show the modified decomposition and the resulting reconstruction (see Figure 6.32), using `haar` wavelets and and least one other wavelet of your choice (chosen from the wavelets that you investigated in the "Familiarization" step above). In your report, **comment on the edge detecting abilities** of the Haar wavelets compared to the other wavelet(s).
  - ➪ Repeat the step above, but in a way that emphasizes primarily the **horizontal** edges of the `iptest256a` image.

- **Denoising**

  - ➪ Following the general method described in slides of `chap6part8`, write a MATLAB program to apply wavelet-based denoising to the `B512` image from the `lena_noise.mat` file. Try a small variety of wavelets and levels of decomposition to get a feel for the process. For the thresholding step, you can choose to just zero out entire detail coefficient arrays (which is easy, but rather crude), apply a hard threshold, and/or apply a soft threshold. Include in your report a figure that shows what you consider your best denoising results (on a full page, show the original noisy image on top with the denoised image on the bottom) using your own program, and in the text be sure to specify the procedure (such as which wavelet, how many levels of decomposition, type of thresholding and at which level of decomposition, etc.) that you used to obtain the results. ***Take note:*** this step is *not* intended to be an infinite time sink! Try a a few approaches then move on to the next step. Some people find it easier to go to the next step before trying this step, then come back to this one after getting a better idea about denoising. I'll leave that up to you.

- ➪ Use the interactive GUI of the MATLAB Wavelet Toolbox to denoise the image. First, be sure to load `lena_noise.mat` into your MATLAB workspace. The `B512` image should now be a 512×512 matrix of `uint8` pixels in your workspace. Start the MATLAB Wavelet Toolbox using the `wavemenu` or `waveletAnalyzer` command. Click the Wavelet 2-D button. Select File, Import from Workspace, Import Image, and double click on `B512`. The image is now loaded into the memory space of the GUI reserved for variables. You may want to go to the lower right corner and change the Colormap to gray (unless you like the default pink color!). Go to the upper right corner and select a wavelet (name *and* number) and select the number of decomposition levels (Level) that you desire. Click on the Analyze button to perform the decomposition. Click on the De-noise button. You'll be presented with a GUI showing the histograms of all the various detail coefficient matrices, and marker lines showing where the thresholds are. You can choose various options such as soft or hard thresholding. Leave the noise structure setting on "unscaled white noise." Just below the noise structure selection, you can select which (H, V, or D) of the various detail coefficient thresholds you want to modify. Note you can modify any of the levels independently using the various slider bars. Observe how the marker lines move on the histograms as you adjust the thresholds. When you have all the thresholds set where you want them, click the De-noise button and the result will appear to the right of the noisy original image, above the histograms. You can iterate the cycle of adjusting the thresholds and clicking the De-noise button until you get the results you want. If you want to change the wavelet used or change the number of decomposition levels available to you, you'll need to back up to the appropriate previous step. When you're satisfied with your denoising results, click the Close button, and click Yes to update the synthesized image (which is the modified reconstruction). To save your results as a workspace matrix, you can select File, Export to Workspace, Export Image, and type in an appropriate variable name (it will be saved as a matrix of `double` rather than `uint8`). If instead you wish to save your results as a MAT file, you can select File, Save, Synthesized Image, and type in an appropriate file name. Include in your report a figure that shows what you consider your best denoising results (on a full page, show the original noisy image on top with the denoised image on the bottom) using the MATLAB Wavelet Toolbox GUI, and in the text be sure to specify all the settings that you used in the GUI to obtain the results.
- ➪ In your report, briefly compare the advantages/disadvantages of using your own MATLAB program to using the MATLAB Wavelet Toolbox GUI for denoising. Don't just write something trivial; think about it for a bit before you write this part of your report.

Questions/Discussion: The write-up for this project report should be concise, using wording and formatting suitable for an IEEE technical journal.[3] Include a short background of wavelet analysis and how it compares to Fourier analysis, including the concept of joint time-frequency analysis (JTFA). Explain succinctly but clearly those points specifically mentioned in the various steps above (along with the figures required); specify which, if any, Toolbox program you used; and include any pertinent lessons you learned in the process of completing this project. Remember, this report might be something you read in five years to help you remember how to perform wavelet analysis, when you need to use it on the job.

Turn in: Turn in your project report in PDF file format, along with any original m-files you wrote as part of this project as separate files, via e-mail attachments. Be sure that the name of each team

[3]Your team's project report must follow the format referenced on the course web site.

member appears on the first page of the report (below the title of the report). Name your PDF file "`Last1_Last2_proj02.pdf`" please, where "Last1" is the last name of team member 1, and "Last2" is the last name of team member 2.

Don't wait until the due date approaches to start this project! It really needs "sink-in" time in your brain (i.e., try it, walk away, come back later and try it again) for it to have the most benefit...

❄❄❄ Let the fun begin! ❄❄❄