Revised 1 February 2017

Project #5: Wavelet Analysis of Images

In this lab you will use MATLAB to study the following topics:

- Two-Dimensional Discrete Wavelet Transform
- Multiscale Analysis of Images
- Fast Two-Dimensional Wavelet Transform
- Denoising and Compressing Images by Wavelet Methods

Preliminaries

Reading from Textbook: Before beginning your MATLAB work, read Sections 3.6.

Uvi_Wave toolbox: Many of the MATLAB m-files that are used in this assignment are in the public domain *Uvi_Wave* toolbox. In connection with Project #4 you should have already downloaded this package of files from the course web page (or the web page of the textbook), made it a subdirectory of your own MATLAB directory for this course, and set the Path for MATLAB (see Question 4 of Project #4 for more details).

Caution: The image files in the *Uvi_Wave* `wdemo` folder have the extension `*.mat`. This file extension has recently been appropriated for a completely different purpose by the ever-helpful folks at Microsoft. If you cannot open these files and are running Microsoft Windows on your computer, go to the subdirectory *control panel/programs/default programs* and click on "Associate a file type with a program". Scroll down the list of file types until you get to `*.mat` and change the default program for this type to MATLAB. Also check the *path* setting in MATLAB and make sure that it includes the *Uvi_Wave* directory and all of its subdirectories.

Diary File: You will need to record the results of your MATLAB session to generate your lab report. Use the `Current Directory` field in the desktop toolbar to change to your directory. Then type

```
diary lab5.txt
```

followed by the `Enter` key.

Lab Write-up: Now that your diary file is open, type the comment line

```
% Math 357 MATLAB Project #5 -- Wavelet Analysis of Images
%
```

at the MATLAB prompt. Type

```
format compact
```

so that your diary file will not have unnecessary spaces.

Random Seed: Initialize the random number generator by typing

```
rand('seed', abcd)
```

where *abcd* are the last four digits of your Student ID number. This will ensure that you generate your own particular random vectors and matrices.

BE SURE TO INCLUDE THIS LINE IN YOUR LAB WRITE-UP

Labels and Comments: Put labels to mark the beginning of your work on each part of each question, so that your edited lab write-up has the format

```
% Question 1 (a) ...
⋮
% Question 1 (b) ...
```

and so on. Be sure to answer all the questions in the lab assignment. Insert comments in your diary file as you work through the assignment.

The lab report that you hand in must be your own work. The following problems use randomly generated matrices and vectors, so the matrices, vectors, and graphs in your lab report will not be the same as those of other students doing the lab. Sharing of lab report files is not allowed in this course.

Question 1. Two-Dimensional Discrete Wavelet Transform

You will explore the use of wavelet transforms for image processing (see Section 3.6 of the textbook). Set the `path` so that MATLAB can find the UVI_WAVE directories and subdirectories.

(a) Images as Matrices: Type `gnimgdmo` at the MATLAB prompt to run the image-generating demo from the UVI_WAVE `wdemo` directory. Use the MATLAB text editor to open the m-file `genimg.m` in the `wdemo` directory. Notice how few lines of MATLAB codes are needed to generate these complex images.

Now generate a random 8-bit integer matrix $\mathbf{X}$ of size 8×8 and display it as an image by the commands

```
X = floor(256*rand(8,8));

figure, show(X), colormap(gray)
```

Notice that the $(1, 1)$ entry in $\mathbf{X}$ determines the grayscale of the upper left square in the figure, while the $(8, 1)$ entry determines the grayscale of the lower left square (see Section 3.6.1). Now modify your matrix $\mathbf{X}$ to make row 4 black and column 2 white; then make the value of the $(4, 2)$ square 128 (half-way between black and white) and leave all the other squares unchanged. Redisplay the matrix as a figure. Put the title Fig. 1: Random Squares with Black Row and White Column on the figure. Save and print the MATLAB figure.

(b) One-scale Wavelet Transform of Images: Let $\mathbf{W_a}$ be the $N \times N$ analysis matrix for a one-scale wavelet transform, where N is even. Then $\mathbf{W_a}$ can be written in 2×1 block form as

$$\mathbf{W_a} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are of size $N/2 \times N$ (see Section 3.6.2 of the textbook). The rows of $\mathbf{A}$ are the *trend rows*, whereas the rows of $\mathbf{B}$ are the *detail rows.*

Generate and display the matrix $\mathbf{W_a}$ and the matrices $\mathbf{A}$ and $\mathbf{B}$ for the CDF(2,2) analysis transform with $N = 8$ using the m-file `cdfamat.m` from Assignment #4 (use the colon operator to extract $\mathbf{A}$ and $\mathbf{B}$ from $\mathbf{W_a}$). Observe how each row of $\mathbf{A}$ and $\mathbf{B}$ is obtained from the row above by shifting to the right two places (with wraparound).

Now generate the matrix $\mathbf{W_a}$ and the matrices $\mathbf{A}$ and $\mathbf{B}$ for the CDF(2,2) analysis transform with $N = 64$ by the same method. *Be sure to put ; at the end of each command when* $N = 64$ *so that the large matrices* $\mathbf{W_a}$*,* $\mathbf{A}$*, and* $\mathbf{B}$ *will NOT appear on screen. If you forget to do this, remove these displayed matrices from your diary file before printing.*

Generate and display a 64×64 test image $\mathbf{X}$ by

```
X = genimg(0,64,64);
figure, show(X, -1), colormap(gray), axis off
```

(The m-files `show.m` and `colormap.m` are from the UVI_WAVE toolbox). Put the title Fig. 2: 64×64 Synthetic Test Image on the figure. Save and print the figure.

The one-scale wavelet transform of the image encoded by $\mathbf{X}$ is the matrix

$$\mathbf{Y} = \mathbf{W_a} \mathbf{X} \mathbf{W_a}^{\mathrm{T}}.$$

The block decomposition of $\mathbf{W_a}$ gives a decomposition of $\mathbf{Y}$ into four $N/2 \times N/2$ blocks:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{X} \begin{bmatrix} \mathbf{A}^{\mathrm{T}} & \mathbf{B}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{AXA}^{\mathrm{T}} & \mathbf{AXB}^{\mathrm{T}} \\ \mathbf{BXA}^{\mathrm{T}} & \mathbf{BXB}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{Y_{ss}} & \mathbf{Y_{sd}} \\ \mathbf{Y_{ds}} & \mathbf{Y_{dd}} \end{bmatrix}$$

(see Section 3.6 of the textbook for an interpretation of each block in terms of trend/detail of rows and columns of $\mathbf{X}$). Calculate and display these matrices for the test image $\mathbf{X}$ generated above:

```
Yss = A*X*A'; Ysd = A*X*B'; Yds = B*X*A'; Ydd = B*X*B';
figure
subplot(2,2,1), show(Yss, -1), colormap(gray), axis off, hold on
subplot(2,2,2), show(Ysd, -1), colormap(gray), axis off
subplot(2,2,3), show(Yds, -1), colormap(gray), axis off
subplot(2,2,4), show(Ydd, -1), colormap(gray), axis off
```

Here the *Uvi_Wave* m-file `show.m` automatically adjusts the gray scale to get maximum contrast. Put the title Fig. 3: One-Scale CDF(2, 2) Transform of Synthetic Test Image on the MATLAB figure. Label the subplots as Trend, Vertical Detail, Horizontal Detail, and Diagonal Detail (see Section 3.6.2 of the textbook for an explanation of this terminology). Save and print the figure.

(c) Multiresolution Representation of Images: The synthesis matrix $\mathbf{W_s}$ is the inverse of the analysis matrix. It can be written in 1×2 block form as

$$\mathbf{W_s} = \begin{bmatrix} \mathbf{C} & \mathbf{D} \end{bmatrix}$$

where the matrices $\mathbf{C}$ and $\mathbf{D}$ are of size $N \times N/2$. The columns of $\mathbf{C}$ are the *trend vectors*, whereas the columns of $\mathbf{D}$ are the *detail vectors*.

Generate and display the matrix $\mathbf{W_s}$ and the matrices $\mathbf{C}$ and $\mathbf{D}$ for the CDF(2, 2) transform with $N = 8$ using the m-file `cdfsmat.m` from Assignment #4 (use the colon operator to extract $\mathbf{C}$ and $\mathbf{D}$ from $\mathbf{W_s}$). Observe how each column of $\mathbf{C}$ and $\mathbf{D}$ is obtained from the column to the left by shifting down two places (with wraparound).

Now generate the matrix $\mathbf{W_s}$ and the matrices $\mathbf{C}$ and $\mathbf{D}$ for the CDF(2, 2) synthesis transform with $N = 64$ by the same method. *Be sure to put ; at the end of each command when $N = 64$ so that the large matrices $\mathbf{W_s}$, $\mathbf{C}$, and $\mathbf{D}$ will NOT appear on screen. If you forget to do this, remove these displayed matrices from your diary file before printing.*

The one-scale inverse wavelet transform of $\mathbf{Y}$ is the matrix

$$\mathbf{X} = \mathbf{W_s}\mathbf{Y}\mathbf{W_s}^{\mathrm{T}}.$$

To obtain the *multiresolution representation* of the image $\mathbf{X}$, use the block decomposition of $\mathbf{Y}$ from part **(b)** and the trend and detail matrices $\mathbf{C}$ and $\mathbf{D}$:

$$\begin{aligned} \mathbf{X_{ss}} &= \mathbf{C}\mathbf{Y_{ss}}\mathbf{C}^{\mathrm{T}}, \quad \mathbf{X_{sd}} = \mathbf{C}\mathbf{Y_{sd}}\mathbf{D}^{\mathrm{T}}, \\ \mathbf{X_{ds}} &= \mathbf{D}\mathbf{Y_{ds}}\mathbf{C}^{\mathrm{T}}, \quad \mathbf{X_{dd}} = \mathbf{D}\mathbf{Y_{dd}}\mathbf{D}^{\mathrm{T}}. \end{aligned}$$

Calculate and display the multiresolution representation for the test image $\mathbf{X}$ generated above (see Section 3.6.5):

```
Xss = C*Yss*C'; Xsd = C*Ysd*D'; Xds = D*Yds*C'; Xdd = D*Ydd*D';
figure
subplot(2,2,1), show(Xss, -1), colormap(gray), axis off, hold on
subplot(2,2,2), show(Xsd, -1), colormap(gray), axis off
subplot(2,2,3), show(Xds, -1), colormap(gray), axis off
subplot(2,2,4), show(Xdd, -1), colormap(gray), axis off
```

Put the title Fig. 4: One-Scale CDF(2, 2) Multiresolution Representation of Synthetic Test Image on the figure. Label the subplots as Trend, Vertical Detail, Horizontal Detail, and Diagonal Detail. Save and print the figure.

Question 2. Image Compression and Multiscale Analysis

You will perform a wavelet analysis of a famous photograph: the *Lena* image in the UVI_WAVE `wdemo` directory. This image has been used since the 1970's as a test image for comparing image-processing algorithms; go to *Wikipedia* for more information about its history. At the MATLAB prompt, type `wvt2ddmo`. This is the two-dimensional wavelet transform demo in UVI_WAVE that includes the *Lena* image. Position the figure window so that you can still read the demo text in the main MATLAB window as the demo proceeds. The *Lena* image has a much more complex combination of textures and contours than the synthetic images.

(a) To load and display the image type

```
load lena, X = lena;
size(X)
figure, show(X), colormap(gray), axis off
```

(you don't need to print the image). The size of the matrix $\mathbf{X}$ is 256×256. Perform a one-scale CDF(2,2) wavelet transform of $\mathbf{X}$ and display the result:

```
Wa = cdfamat(256); Y = Wa*X*Wa' ;
Ybr = bandadj(Y,1);
figure, show(Ybr), colormap(gray)
```

Here, as explained in the demo, you use the *Uvi_Wave* m-file `bandadj.m` to adjust the brightness levels of the four parts of the wavelet transform, as in the `wvt2ddmo` demo that you saw at the beginning of the project (without this adjustment, only the $\mathbf{Y}_{\mathbf{ss}}$ part would be visible). The 128×128 trend matrix $\mathbf{Y}_{\mathbf{ss}}$ in the upper-left corner gives a rougher version of the original image, whereas the other 128×128 matrices in the picture give the horizontal, vertical and diagonal details. Insert the title Fig. 5: One-Scale CDF(2,2) Transform of Lena Image. Save and print the figure.

To determine how much information about the image is contained in the three detail portions, set the trend block $\mathbf{Y}_{\mathbf{ss}}$ in wavelet transform to zero and take the inverse transform:

```
Yd = Y; Yd(1:128, 1:128) = 0;
Ws = cdfsmat(256); Xd = Ws*Yd*Ws' ;
figure, show(Xd), colormap(gray), axis off
```

You should get figure that you saw in the `wtdemo`; it shows the *edges* of the picture (you don't need to print this figure).

(b) Image Compression: One of the most useful features of wavelet methods in image processing is the compression of images without loss of perceptible information. This is done by compressing the detail matrix `Yd`. Fix a threshold level of 30 and set to zero all entries in `Yd` whose absolute value is 30 or less (the value of 30 for this image gives good results; in general, experimentation is needed to find a good threshold for an image). Then create a *compressed* transformed matrix `Yc` by replacing the 128×128 upper left-hand block of zeros in `Yc` by the trend `Yss`, and take the inverse transform:

```
Ydc = threshold(Yd, 30) ;
Yc = Ydc; Yss = Y(1:128, 1:128);
Yc(1:128, 1:128) = Yss ;
Xc = Ws*Yc*Ws' ;
figure, show(Xc), colormap(gray), axis off
```

(Use the m-file `threshold.m` that you created in Project #3.) Notice that the image displayed looks very much like the original. Insert the title Fig. 6: Compressed Lena Image, but don't print the image yet.

To determine the number of nonzero entries in the compressed transform `Yc` and to find the compression ratio, calculate

```
numY = ones(256, 256).*(abs(Y)>0);
numYc = ones(256, 256).*(abs(Yc)>0);
```

```
sumY = sum(numY*ones(256, 1))
sumYc = sum(numYc*ones(256, 1))
compress = sumY/sumYc
```

Be careful to use the element-by-element multiplication $.*$ (note the period mark before the asterisk) in the first two lines. This ensures that the matrices `numY` and `numYc` have entries that are either 1 or 0, depending on whether the corresponding entries of `Y` and `Yc` are nonzero or zero. Thus `sumY` and `sumYc` counts the number of nonzero entries in each matrix. Insert the label compression ratio $= r$ (where r is the numerical value of `compress` that you have calculated) below the image in the MATLAB figure, but don't print the image yet.

(c) Error measures in image processing: The difference between two eight-bit image matrices `X1` and `X2` (both assumed to be of size 256×256 and with integer entries between 0 and 255) can be measured by the *Mean Square Error* (MSE):

```
MSE = (norm(X1 - X2, 'fro')^2)/2^16
```

Here the command `norm(X, 'fro')^2` calculates the sum of the squares of the entries in the matrix X, and the denominator $2^{16} = (256)^2$ is the total number of entries in the image matrices. With this normalization, the MSE between the 256×256 matrix with entries all 1 and the zero matrix is 1. A large MSE corresponds a big difference in the images. The largest possible value for MSE is $(255)^2$, since the differences between the individual entries of `X1` and `X2` are at most 255.

An equivalent measurement of image difference that corresponds more closely to how the human brain responds to light intensity is the *Peak Signal to Noise Ratio* (PSNR), which is defined on a logarithmic scale in *decibels* (dB) by

```
PSNR = 10*log10(255^2/MSE)
```

Here `log10` is the logarithm to base 10 in MATLAB. The ratio `255^2/MSE` is bigger than 1, so PSNR is a positive number. *Small* values of MSE correspond to *large* values of PSNR. As a *rule of thumb*, if the PSNR exceeds 40 dB, then the two images are perceived as the same.

Calculate the values of the MSE between the matrices `X` and `Xc` and the associated PSNR using these formulas with `X1 = X` and `X2 = Xc`. Insert the values of MSE and PSNR in Figure 6 below the image. Save and print Figure 6. What is your opinion about validity of the 40 dB rule of thumb criterion concerning the compressed image in this case?

(d) Two-Scale Analysis of Image: To make a two-scale analysis of the *Lena* image, repeat the one-scale analysis on the trend matrix $\mathbf{Y_{ss}}$:

```
Yss = Y(1:128, 1:128);
W2a = cdfamat(128);
Y2 = Y; Y2(1:128, 1:128) = W2a*Yss*W2a';
```

Note that you are using the 128×128 CDF$(2,2)$ analysis matrix on $\mathbf{Y_{ss}}$ and then inserting the transformed matrix into the upper left-hand block of $\mathbf{Y}$. This gives the *two-scale* CDF$(2,2)$ transform $\mathbf{Y}^{(2)}$. To display the result, brighten the image using the `bandadj.m` utility:

```
Y2br = bandadj(Y2,2);

figure, show(Y2br), colormap(gray)
```

Insert the title Fig. 7: CDF$(2,2)$ transform over 2 scales in the MATLAB figure. Save and print Figure 7.

Question 3. Fast Two-Dimensional Wavelet Transform

The matrix formulation of the two-dimensional wavelet transform and inverse transform in Questions #1 and #2 of the project is helpful in understanding the theory of this transform. For numerical calculation, however, the matrix formulation is impractical, since an $N \times N$ image will require the order of N^3 arithmetic operations to calculate the product of three $N \times N$ matrices. Just as in the case of the one-dimensional wavelet transform, there is a fast implementation, either using lifting or using the two-channel filter bank (convolution) method. In both methods the one-dimensional wavelet transform is performed on the rows and columns of the image matrix.

We will use the *symlets* wavelet transform. This is an orthogonal transform that has the most symmetry of the Daubechies family of transforms.

(a) Multiscale Analysis of Image: To calculate the symlets(24) wavelet transform of an image, use the *filter bank* implementation in the *Uvi_Wave* toolbox.

```
[h,g,rh,rg] = symlets(24);
```

Here the vector **h** gives the 24 nonzero coefficients of the *lowpass filter* and the vector **g** gives the 24 nonzero coefficients of the *highpass filter* for the one-dimensional symlets(24) analysis transform. Plot these coefficients by

```
figure, subplot(2,2,1), plot([-11:12], h), hold on
subplot(2,2,2), plot([-11:12], g)
```

The vector **rh** gives the 24 nonzero coefficients of the *lowpass filter* and the vector **rg** gives the 24 nonzero coefficients of the *highpass filter* for the one-dimensional symlets(24) inverse wavelet transform. Plot these coefficients by

```
subplot(2,2,3), plot([-12:11], rh)
subplot(2,2,4), plot([-12:11], rg)
```

Answer the following questions:

1. How are the graphs of **h** and **rh** related?
2. How are the graphs of **g** and **rg** related?

See Definition 4.6 in Section 4.9 of the textbook, where the notation is changed to $\mathbf{h}_0 = \mathbf{h}$ (lowpass) and $\mathbf{h}_1 = \mathbf{g}$ (highpass) for the analysis filters, and $\mathbf{g}_0 = \mathbf{rh}$ (lowpass) and $\mathbf{g}_1 = \mathbf{rg}$ (highpass) for the synthesis filters.

Label the four graphs lowpass analysis filter, highpass analysis filter, lowpass synthesis filter, and highpass synthesis filter. Insert the title Fig. 8: Symlets(24) Wavelet Filters. Save and print the figure.

Make a fast one-scale symlets(24) transform of the *Lena* image and then adjust the brightness levels for printing:

```
load lena, X = lena;
Y = wt2d(X,h,g,1);
Ybr = bandadj(Y,1);
figure, show(Ybr), colormap(gray), axis off
```

Calculate `norm(X, 'fro')` and `norm(Y, 'fro')`. Explain why your answers are predicted by orthogonal property of the symlets(24) transform.

Insert the title Fig. 9: One-scale Symlets(24) Transform of Lena Image. Save and print the graph. Notice that the four subregions of the transformed image show more fine-scale edge detail than the CDF$(2,2)$ transform in Question #2 (this is due to the longer lengths of the filters). The trade-off is that the more computational time is needed to obtain the transform.

(b) Multiresolution Representation: Carry out a two-scale symlets(24) multiresolution representation of the *Lena* image using the fast transform/inverse transform:

```
Trend = mres2d(X,h,rh,g,rg,2,0);
Details = mres2d(X,h,rh,g,rg,2,4);
Image = Trend + Details;
figure, show(Trend), colormap(gray), axis off
figure, show(Details), colormap(gray), axis off
figure, show(Image), colormap(gray), axis off
```

Put titles Fig. 10: Trend of Two-Scale Multiresolution Representation, Fig. 11: Sum of Details of Two-Scale Multiresolution Representation, Fig. 12: Sum of Trend and Details of Two-Scale Multiresolution Representation at the tops of the figures. Calculate the values of the MSE between the matrices `X` and `Image` and the associated PSNR using the formulas in Question **2(c)** with `X1 = X` and `X2 = Image`. Insert these values in Figure 12 below the image. The Image plot is not quite as sharp as the original image; this is confirmed by the value less than 40 Db of PSNR. Save and print the figures.

Question 4. Denoising and Compressing Images

Adapting audio terminology, we use the term *noise* to describe distortions of images due to numerical or physical imperfections. Often images are produced under less than ideal conditions of lighting or signal transmission, and consequently are contaminated by significant amounts of noise. This is the case, for example, with many medical images or with digitized images of old photographs. One of the most important applications of wavelet techniques in image processing is to remove noise. You can do this by compression of the wavelet transform of the noisy image.

(a) Adding Noise to an Image: Generate a matrix of random integers (the integer parts of independent normal random variables with mean zero and standard deviation 50) and add it to the *Lena* image:

```
Noise = floor(50*randn(256,256));
Xn = X + Noise;
figure, show(Xn), colormap(gray), axis off
```

You can see that the image is considerably degraded by the noise. Calculate the Mean Square Error (MSE) between the original image `X` and the noisy image `Xn`, and the associated Peak Signal to Noise Ratio (PSNR), as described Question **2(b)** with `X1 = X` and `X2 = Xn`. Put the title Fig. 13: Noisy Lena Image above the image, and the values of the MSE and PSNR that you have calculated under the figure. Save and print the figure.

(b) Removing Noise by Wavelet Techniques: Take a four-scale symlets(24) wavelet transform of the noisy image:

```
Yn = wt2d(Xn,h,g,4);
```

The trend portion of the transformed matrix `Yn` is in the upper-left 16×16 block, whereas the first-scale detail is in the lower-right 128×128 block. Calculate the ratios of the largest element in the trend to the largest element in the detail block:

```
Yss = Yn(1:16,1:16); Ydd = Yn(129:256, 129:256);
a = max(max(abs(Yss)))
b = max(max(abs(Ydd)))
ratio = a/b
```

The ratio should be quite large. This indicates that a large proportion of the detail entries can be set to zero without changing the image substantially. Doing this will compress the image file and remove some of the noise. For this particular image and wavelet transform, some experimentation shows that setting 95% of the coefficients to zero (20:1 compression) is a good choice for maximizing the PSNR. The *Uvi_Wave* utility `elmin` will do this:

```
Ync95 = elmin(Yn,95);
Xnc95 = iwt2d(Ync95,rh,rg,4);
figure, show(Xnc95), colormap(gray), axis off
```

Put the title Fig. 14: 20:1 Compressed and Denoised Lena Image above the figure. Calculate the Mean Square Error and the Peak Signal to Noise Ratio between the original image matrix `X` and the matrix `Xnc95` using the formulas in Question **2(c)**. Put these values under the figure. The MSE should be smaller and the PSNR larger than the MSE and PSNR for the noisy image (although the PSNR will still be below 40 Db). Save and print the figure.

Remark: More refined image-processing methods (such as are found in the MATLAB Wavelet toolbox) can be applied to the compressed image to improve its appearance.

Final Editing of Lab Write-up: After you have worked through all the parts of the lab assignment, you will need to edit your diary file. Remove all errors and other material that is not directly related to the questions. Your write-up should only contain the keyboard input and the MATLAB output (including Figures 1-14), together with the answers to the questions that you have written.

Preview the document before printing and remove unnecessary page breaks and blank space. Put your name and four-digit ID number on each page. (If you have difficulty doing this using your text editor, you can write this information by hand after printing the report.)