CS5680/CS6680 – Fall Semester 2017
Assignment 4 – Filter Techniques in the Frequency Domain
Due: 11:59 p.m. Saturday, October 21, 2017
Total Points: 50 points

General Assignment Instructions: The same as the previous assignments.

For all the Fourier transform related questions, make sure that the center of the image in the frequency domain corresponds to the lowest frequency of (0, 0).

Problem I: Exercises on Low-pass and High-pass Filters in the Frequency Domain [Total: 8 points]
1. [4 points]
Design a Gaussian **low-pass** filter with the standard deviation σ_1 of 25 at *x* direction (row) and the standard deviation σ_2 of 50 at *y* direction (column) in the frequency domain (Refer to slide 63 of Ch3.2.DIPBasicFreq.pdf for the equation when $\sigma_1=\sigma_2$). Obtain the filtered image by filtering the original image ***Sample*** with the designed Gaussian filter (i.e., perform the Gaussian low-pass filtering operation in the Fourier frequency domain). Display the original image, the Gaussian low-pass filter **(treat it as an image)**, and the filtered image in figure 1 with appropriate titles.

2. [4 points]
Design a Butterworth **high-pass** filter of order 2 with a cutoff frequency of 30 in the frequency domain (Refer to slide 73 of Ch3.2.DIPBasicFreq.pdf for the equation). Obtain the filtered image by filtering the original image ***Sample*** with the designed Butterworth filter (i.e., perform the Butterworth high-pass filtering operation in the Fourier frequency domain). Display the original image, the Butterworth high-pass filter **(treat it as an image)**, and the filtered image in figure 2 with appropriate titles.

Problem II: Noise Modeling and Basic Restoration Techniques [Total: 8 points]
1. [4 points]
Given the ***City*** image, apply the noise modeling formula *G(u,v) = F(u,v)H(u,v)* **(This operation is a pixel-wise multiplication)** to create a severely turbulent image, ***BlurCity.bmp***. Here, *F(u,v)* is the original image in the centered Fourier domain and $H(u,v) = e^{-kD(u,v)^{5/3}}$, where $D(u, v)$ is the distance from any point (u, v) to the center (origin) in the Fourier transform domain and $k = 0.0025$. Display the filter H **(treat it as an image)** and the generated severely turbulent image in figure 3 with appropriate titles.

2. [4 points]
Read in ***BlurCity.bmp*** and restore it by using the **Wiener** filter with **an appropriately determined constant *g***. The Wiener filter is defined as follows:

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + g}\right]G'(u,v)$$

Here, all the operations are pixel-wise algorithmic operations. *H(u, v)* is the same as the one defined in problem II.1. *G'(u,v)* is the severely turbulent image generated in Problem II.1 in the centered Fourier domain. $\hat{F}(u,v)$ is the restored image in the centered Fourier domain. Display the restored image in figure 4 with the appropriate title. (Note: Please do not call Matlab built-in function deconvwnr or other equivalent built-in functions to solve the problem.)

Problem III: Exercise on Certain Operations in the Frequency Domain [Total: 8 points]
1. [4 points]
Apply the Fourier transform on two images ***Sample*** and ***Capitol***, respectively. Display the magnitude and the phase of the two Fourier transformed images in figure 5 with appropriate titles (Refer to slide 11 of

Ch3.2.DIPBasicFreq.pdf for the equations**). Note: Appropriate scaling operations are needed for proper display due to the large dynamic range of images in the frequency domain. For example, the log transformation needs to be applied to the magnitude for proper display.**

2. [4 points]
Exchange the phase components of the two Fourier transformed images and take an inverse Fourier transform. That is, use the phase of ***Capitol*** and the magnitude of ***Sample*** to reconstruct the new ***Capitol*** image. Use the phase of ***Sample*** and the magnitude of ***Capitol*** to reconstruct the new ***Sample*** image. Display the two corresponding reconstructed images in figure 6 with appropriate titles.

Problem IV: Remove Additive Cosine Noise [Total: 8 points]
The noisy image ***boy_noisy.gif*** has been generated by adding some noise in the form of a cosine function. Your goal is to remove the cosine interference. This can be done as follows:

1. **[1 point]** Compute the centered DFT (Discrete Fourier Transform) of the noisy image.
2. **[3 points]** Compute the magnitude and find the frequencies corresponding to the ***four largest distinct magnitudes*** (do not consider the magnitude at the center – a very large value).
3. **[2 points]** Replace the value of each of the four frequencies, which correspond to the four largest distinct magnitudes found in step 2), by the average of its 8 neighbors. Note: Make sure you will perform this change on their symmetric points (i.e., their complex conjugate counterpart) to ensure the proper reconstruction.
4. **[1 point]** Take the inverse DFT transform and display the original image and the resultant image side-by-side in figure 7 with appropriate titles.
5. **[1 point]** Explain why the four largest distinct values of the magnitude were chosen to do the processing on the Matlab console.

Problem V: Preliminary Wavelet Transform [Total: 8 points]
1. [3 points]
Call a Matlab built-in function to compute the maximum decomposition level for image ***Lena***. Apply a maximum-level "db2" wavelet decomposition on ***Lena*** by using appropriate Matlab function(s). Apply the inverse wavelet transform to restore the image. Use "if-else" statement to compare your restored image with the original image so the appropriate message indicating the equality or inequality between these two images is displayed. (Note: The original and the restored images must be the same to receive the full credit.)

2. [5 points]
Apply an *x*-level "db2" wavelet decomposition on *Lena* by using appropriate Matlab function(s), where *x* = floor(maximum decomposition level / 2) and the maximum decomposition level is computed in the previous problem.

Independently perform the inverse wavelet transform after each of the following operations:

a) **[2 points]** Set all the approximation coefficients as 0's.
b) **[2 points]** Set the second level vertical detail coefficients (e.g., LH2) as 0's.

Note: For the above two "set" operations, please do not call Matlab built-in functions.
Display the two reconstructed image side-by-side in figure 8.

[1 point] On the Matlab console, summarize the reasons for the appearance of the reconstructed images after performing the above two independent operations.

Problem VI: A Simple yet Famous Solution to a Practical Problem [10 points]

1. Load in ***Lena*** and call imnoise function to add Gaussian white noise of zero mean and 0.01 variance to it.

2. Apply a 3-level "db2" wavelet decomposition on the noisy *Lena*.
3. Estimate the noise variance at the 1st-level subband (e.g., LH1, HL1, HH1) by $\sigma^2 = [\,median\ (|\ f_{ij}\ |) / 0.6745\ ]^2$, where f_{ij} is the wavelet coefficient in the LH1, HL1, and HH1 subband.
4. Compute the adaptive threshold *t* of the 1st-level subband as follows: $t = \sigma\sqrt{2\log\ M}$ where *M* is the number of elements in the 1st-level subband (LH1, HL1, and HH1).
5. Modify the wavelet coefficient f_{ij} in the LH1, HL1, and HH1 subband using the soft thresholding:

$$\hat{f}_{ij} = \begin{cases} f_{ij} - t & if \quad f_{ij} \geq t \\ f_{ij} + t & if \quad f_{ij} \leq -t \\ 0 & if \quad |\ f_{ij}\ | < t \end{cases}$$

6. Apply steps 3, 4, and 5 on the 2nd-level subband (e.g., LH2, HL2, HH2) by using the information in the 2nd-level subband.
7. Apply steps 3, 4, and 5 on the 3rd-level subband (e.g., LH3, HL3, HH3) by using the information in the 3rd-level subband.
8. Take the inverse wavelet transform to get the denoised image.
9. Display the noisy image and the denoised image side-by-side in figure 9.