Assignment #4

Due date: May 1 (Mon.)

Submission

E-mail a zip file including the source codes, a report and test images to TA. **You should submit function m-files per problems like Appendix at the end of this document. If you don't follow the code structure in Appendix, the score will be deducted.** The filename should be named as student idn_name.zip (e.g. 20173000_kdhong.zip).
TA's e-mail address is **wschoi@issserver.kaist.ac.kr**

Due date: **05/01 23:59**. (Refer to the delay policy in the web site)

Test images in the web site:
Gray_baby_512x512.raw Color_baby_512x512.raw

Notice

All the programming assignments are based on MATLAB. (Do not use any function in MATLAB. But **you can use basic functions available in C++ standard library** like round, ceil, floor, rand, bitshift, sqrt, sum, exponential, log, trigonometric and abs etc. functions **and vectorization**.) All source codes for submission should include comments.
Describe your work and analyze the corresponding results in the report. A proper length of the report is 10 pages of A4 size with figures. Report exceeding the recommended length will get a penalty. The report should include the followings.

1. Simple theoretical backgrounds & programming strategies
2. Result images
3. Analysis of the results

If a **copy version** is found, the score will be **zero** point without any exception.

Scoring policy: implementation (60), processing time (10), and report (30)

The criteria of the scoring deduction applied to common problems:

1. Using MATLAB functions
   - Related to 'Language Fundamental' except to basic functions available in C++ standard library: -2 points per problem.
     'Language Fundamental':
     https://kr.mathworks.com/help/matlab/language-fundamentals.html?lang=en
   - the other MATLAB functions: -50% per problem.
2. Inexecutable code: -50% per problem.

1. **Image Transform**

(for Gray_baby_512x512.raw)

1. DFT

a. Apply the DFT using the fast algorithm (so-called FFT) to the given image and **display the magnitude in log scale and the phase of the DFT coefficients,** respectively. All processes should be finished within 3 seconds.
b. Apply the inverse FFT to the transformed result and examine the **reconstructed image.** All processes should be finished within 3 seconds.

2. DCT

a. Perform the DCT for the given image and **display the magnitude in log scale of the DCT coefficients. Perform the inverse DCT and verify the reconstructed image**. Each process should be finished within 15 seconds.
b. Divide the given image into 16x16 blocks and **apply 16x16 block DCT to each block**, and then **display the DCT coefficients in log scale**. **Perform the inverse DCT and verify the reconstructed image**. Each process should be finished within 3 seconds.
c. Compare the transformed results of a and b.

3. Discrete Hadamard transform (DHT)

Apply the DHT to the given image and **display magnitude in log scale of the DHT coefficients. Perform the inverse DHT and verify the reconstructed image**.
Examine the reconstructed image and explain the efficiency of energy compaction.

4. Compression Efficiency of DFT, DCT, and DHT

Explain which transform can provide a better performance from the viewpoint of compression rate if we try to compress the transformed images.

5. Wavelet Transform

a. Perform 3-level wavelet transform for the given image.
b. Perform the inverse wavelet transform and examine the reconstructed image.
c. Explain the characteristics at each band.

2. Image Enhancement

(for Gray_baby_512x512.raw and Color_baby_512x512.raw)

1. Point operations

a. Histogram equalization

Implement the histogram equalization program and apply it to the **given gray image.**

In addition, implement histogram equalization programs for the RGB, HSI, and CMY coordinates, respectively, and apply them to the **given color image.**

b. Histogram modification

Apply the histogram modification to the given gray image by using the Rayleigh distribution. (v_{min}=0)

(I request you set input parameter like bottom table at the beginning of the code.)

alpha	`alpha`

2. Spatial Domain Filtering

(for Gray_baby_512x512.raw)

a. Gamma correction

Implement a Gamma correction algorithm.

User can select the value of γ.

(Hint: Intensity range of output image is bounded by [0, 255].)☐

(I request you set input parameter like bottom table at the beginning of the code.)

γ	`gamma`

b. Spatial filtering

Add a salt & pepper noise and a Gaussian noise to **a gray image**. And apply the Mean filtering, median filtering, and directional smoothing to noisy images with kernel sizes of 3 and 7, respectively.

Compare the PSNR of the noisy image with those of the filtered images.

(I request you set output parameter like bottom table to check the PSNR value.)

PSNR	**Kernel 3**		
	Mean filtering	**Median filtering**	**Directional smoothing**
Salt&pepper noise	SP_3_mean	SP_3_med	SP_3_dir
Gaussian noise	Gau_3_mean	Gau_3_med	Gau_3_dir

PSNR	**Kernel 7**		
	Mean filtering	**Median filtering**	**Directional smoothing**
Salt&pepper noise	SP_7_mean	SP_7_med	SP_7_dir
Gaussian noise	Gau_7_mean	Gau_7_med	Gau_7_dir

c. Weak edge enhancement

Apply a weak edge enhancement algorithm for the given **gray image** by using an inverse contrast ratio with local window sizes of 3 and 7, respectively.

3. Transform Domain Filtering

Add salt & pepper noise and a Gaussian noise to the given gray image. And apply LPF and HPF in the DFT domain and the DCT domain, respectively. Analyze the inverse transformed images from each domain.

Results

1.1 a. -FFT: magnitude,
-FFT: phase,
1.1 b. -FFT: reconstructed image
1.2 a. -DCT: magnitude,
-DCT: reconstructed image
1.2 b. -DCT 16x16: magnitude,
-DCT 16x16: reconstructed image
1.3 -DHT: magnitude
-DHT: reconstructed image
1.5 a.-DWT: 3 level DWT
1.5 b.-DWT: reconstructed image

2.1 a. Histogram equalization
-Gray,
-RGB,
-HSI,
-CMY,

2.1 b. -the result image after input alpha

2.2 a. -the result image after input γ.

2.2 b.
-Image with salt & pepper noise,
-Image with Gaussian noise

-Mean filtered image (3x3 kernel) (salt & pepper noise)
-Mean filtered image (7x7 kernel) (salt & pepper noise)
-Mean filtered image (3x3 kernel) (Gaussian noise)
-Mean filtered image (7x7 kernel) (Gaussian noise)

-Median filtered image (3x3 kernel) (salt & pepper noise)
-Median filtered image (7x7 kernel) (salt & pepper noise)
-Median filtered image (3x3 kernel) (Gaussian noise)
-Median filtered image (7x7 kernel) (Gaussian noise)

-Directional filtered image (3x3 kernel) (salt & pepper noise)
-Directional filtered image (7x7 kernel) (salt & pepper noise)
-Directional filtered image (3x3 kernel) (Gaussian noise)
-Directional filtered image (7x7 kernel) (Gaussian noise)

2.2 c.
-Weak edge enhancement (window size: 3x3)
-Weak edge enhancement (window size: 7x7)

3.
-Apply **LPF** to the image which did**n't add noise** in the **DFT** domain.
-Apply **HPF** to the image which did**n't add noise** in the **DFT** domain.
-Apply **LPF** to the image which add **salt & pepper noise** in the **DFT** domain.
-Apply **HPF** to the image which add **salt & pepper noise** in the **DFT** domain.
-Apply **LPF** to the image which add **Gaussian noise** in the **DFT** domain.
-Apply **HPF** to the image which add **Gaussian noise** in the **DFT** domain.

-Apply **LPF** to the image which did**n't add noise** in the **DCT** domain.
-Apply **HPF** to the image which did**n't add noise** in the **DCT** domain.
-Apply **LPF** to the image which add **salt & pepper noise** in the **DCT** domain.
-Apply **HPF** to the image which add **salt & pepper noise** in the **DCT** domain.
-Apply **LPF** to the image which add **Gaussian noise** in the **DCT** domain.
-Apply **HPF** to the image which add **Gaussian noise** in the **DCT** domain.

Appendix (Important)

- You should **make function m-file for each problem as shown in the bottom example.**
- Please display output images **at each figure** when executing execution m-file as shown in the bottom example.

'Problem_1.m'

```
Clear all

function Problem_1()
   % Template for EE535 Digial Image Processing
   % Insert the code in the designated area below

   %% Loading directory for image files
   imgdir = uigetdir('Image Directory');
   file = fopen(fullfile(imgdir,'\Gray_Baby_512x512.raw'),'rb');
   gray_image = fread(file,fliplr([512,512]),'*uint8')';
   fclose(file);

   file = fopen(fullfile(imgdir,'\Color_Baby_512x512.raw'),'rb');
   color_image = fread(file,fliplr([512,512*3]),'*uint8')';
   fclose(file);

   %%
   %%---------------------Insert code below ----------------------%%
   output_image = InnerFunction(gray_image); % Sample code - delete
this

   %% Displaying figures (Edit this part as needed)
   figure; imshow(output_image,[]); title('Problem_1.1');
   figure; imshow(output_image2,[]); title('Problem_1.2');
   ...

   %%---------------------------------------------------------------%%

end

function OUTPUT = InnerFunction(INPUT)
%%-----------------------------------------------------------------%%
                          %function code%
%%-----------------------------------------------------------------%%
end
```