Assignment #2

Due date: April 3 (Tue.)

Submission

E-mail a zip file including the source codes, and a report to TA. An execution m-file should be included in the source codes. **If the submitted execution m-file is executed, the result images of the problems should be displayed on a single screen.** The filename should be named as student idn_name.zip (e.g. 20183000_kdhong.zip).
TA's e-mail address is **ltk010203@kaist.ac.kr**

Due date: **04/03 23:59**. (Refer to the delay policy in the web site)

Test images in the web site:
Color_monarch_512x512.raw Gray_monarch_512x512.raw
Gray_monarch_noisy_512x512.raw
Color_baboon_256x256.raw Color_lena_256x256.raw

Notice

All the programming assignments are based on MATLAB. (Do not use any image processing function like FFT, conv, etc. in MATLAB.) 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 5 pages of A4 size without figures (if you include figures, the total length would be less than 10 pages). Report with a length exceeding the recommended one will get a penalty score. 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)

To perform Assignment #2, you may need to read or display color and gray raw images. For the sake of convenience, the code for reading and displaying images is given in the Appendix.

You can get test images in the website. Each pixel of a color image consists of three channel components, R (red), G (green), and B (blue). In the gray image, each pixel consists of a single channel component.

Data format of provided color images (H: image height, W: image width)

R_{11}	G_{11}	B_{11}	R_{12}	G_{12}	B_{12}	…	…	$R_{1(W)}$	$G_{1(W)}$	$B_{1(W)}$
…	…	…	…	…	…	…	…	…	…	…
$R_{(H)1}$	$G_{(H)1}$	$B_{(H)1}$	$R_{(H)2}$	$G_{(H)2}$	$B_{(H)2}$	…	…	$R_{(H)(W)}$	$G_{(H)(W)}$	$B_{(H)(W)}$

1. Math preliminary

For a gray image with noise（Gray_monarch_noisy_512x512.raw), perform the convolution using the following kernels. (zero padding near the image boundaries) Show the result images and analyze them, respectively.

$$\mathbf{H}_1 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad \mathbf{H}_2 = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

2. Color Coordinate Transform

For a color image (Color_monarch_512x512.raw), display color components R, G, and B, respectively. In addition, transform color coordinates RGB to HSI and YC_bC_r, respectively, and display each component of the HSI coordinates, and each component of the YC_bC_r coordinates, respectively. From the displayed results, compare the characteristics of the coordinates HSI, YC_bC_r, and RGB.

3. Geometric Transform

For a color image (Color_baboon_256x256.raw), implement the affine transform of an image using homogeneous coordinates. The implementation should work for **any floating point parameter** values and any user can change the input parameters. By changing the parameters of the affine transform, perform the translation, rotation and scaling of the image respectively. **(Set the input parameters as in the bottom tables at the beginning of the code.)**

direction	translation	scaling
x	`Tr_x`	`Sc_x`
y	`Tr_y`	`Sc_y`

rotation
Ang

4. Image down and up sampling

For a color image (Color_lena_256x256.raw),

1. Down-sample a test image of 256x256 to an image of 64x64, and determine and display its up-sampled version of 256x256, using a Peg kernel.
2. Up-sample the down-sampled image using the other convolution kernels such as pyramid and cubic B-spline. Compare the performance of those kernels based on PSNR and subjective quality of the up-sampled images. Also, discuss the results based on the frequency characteristics of the convolution kernels.

5. Quantization

For a gray image (Gray_monarch_512x512.raw), by adding the pseudo-random noise uniform over [-16, 16], quantize the noisy image using the uniform quantizer into 2, 4 and 6 bits, respectively. Repeat the previous quantizations using the Lloyd-Max quantizer.
After subtracting the added noise from the quantization result, show and discuss your results. **(Set PSNR variables as given in the bottom table.)**

Quantizer	2 bits	4 bits	6 bits
Uniform	`un_psnr_2`	`un_psnr_4`	`un_psnr_6`
Lloyd-Max	`ln_psnr_2`	`ln_psnr_4`	`ln_psnr_6`

Results

1.
- H1 kernel image
- H2 kernel image

2.
- R, G, B, H, S, I, Y, Cb, Cr

3.
- Geometric transformed image

4.1
- Down sampling image using Peg kernel
- Up sampling image using Peg kernel

4.2
- Down sampling image using pyramid kernel
- Up sampling image using pyramid kernel
- Down sampling image using cubic B-spline kernel
- Up sampling image using cubic B-spline kernel

5.
- 2bits uniform quantization image after subtracting noise
- 4bits uniform quantization image after subtracting noise
- 6bits uniform quantization image after subtracting noise
- 2bits Lloyd-Max quantization image after subtracting noise
- 4bits Lloyd-Max quantization image after subtracting noise
- 6bits Lloyd-Max quantization image after subtracting noise

Appendix (Important)

- You should **make function m-files 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 variables

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
```