# Assignment #2

Due date: April 3 (Mon.)

## 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 execution m-file is executed, result images of the problems should be displayed on a single screen.** 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: **04/3 23:59**. (Refer to the delay policy in the web site)

Test images in the web site:

| | | |
|---|---|---|
| Color_baboon_256x256.raw | Color_barbara_720x576.raw | Color_lena_256x256.raw |
| Gray_baboon_256x256.raw | Gray_ barbara _720x576.raw | Gray_lena_256x256.raw |
| lenna_tr1_256x256.raw | lenna_tr2_256x256.raw | lenna_tr3_256x256.raw. |

## Notice

All the programming assignments are based on MATLAB. (Do not use any image processing function in MATLAB. But **you can use basic functions available in C++ standard library** like trigonometric, round, ceil, floor and rand, bitshift, sqrt etc. function.) 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 length would be less than 10 pages). 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)

For assignment #2, you may need to read or display color and gray raw images. For your convenient, the code to read or display 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). For a 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**, perform the convolution using the following kernels. (zero padding near the image boundaries) Show the result images and analyze them, respectively.

$$\mathbf{H}_1 = \begin{bmatrix} 0.0751 & 0.1238 & 0.0751 \\ 0.1238 & 0.2044 & 0.1238 \\ 0.0751 & 0.1238 & 0.0751 \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 0.1667 & 0.6667 & 0.1667 \\ 0.6667 & -3.3333 & 0.6667 \\ 0.1667 & 0.6667 & 0.1667 \end{bmatrix}$$

## 2. Color Coordinate Transform

2.1. **For a color image**, display color components R, G, and B, respectively. Also, 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.

2.2. For three images transformed from the original color lenna image (lenna_tr1_256x256.raw, lenna_tr2_256x256.raw, lenna_tr3_256x256.raw), obtain the values of H, S, and I, respectively. Describe the relationship between the transformed images and the original, respectively.

### 3. Geometric Transform

**For a color image**, 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**, down-sample a test image of 256x256 to an image of 128x128. And up-sample the down-sampled image to 384x384. Use **pyramid and Bell kernels,** respectively.

### 5. Quantization

5.1. **For a color image**, using the uniform quantizer, quantize an image into 2, 4 and 6 bits, respectively. Repeat the previous quantizations the compandor that is based on a compressor equation given below. Show those quantization results and provide corresponding PSNRs. **(Set PSNR variables as given in the bottom table.)**

Compressor equation: $$f(x) = 255 \cdot \frac{\int_0^x [p_u(u)]^{1/3} du}{\int_0^{255} [p_u(u)]^{1/3} du}$$

| Quantizer | 2 bits | 4 bits | 6 bits |
|---|---|---|---|
| **Uniform** | u_psnr_2 | u_psnr_4 | u_psnr_6 |
| **Compandor** | c_psnr_2 | c_psnr_4 | c_psnr_6 |

5.2. **For a gray image**, repeat the above quantization process by adding the pseudo-random noise uniform over [-16, 16]. 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 |
| **Compandor** | cn_psnr_2 | cn_psnr_4 | cn_psnr_6 |

# Results

**1.**
- H1 kernel image
- H2 kernel image

**2.1**
- R, G, B, H, S, I, Y, Cb, Cr

**2.2**
- H,S,I image of lenna tr1
- H,S,I image of lenna tr2
- H,S,I image of lenna tr3

**3.**
- Geometric transformed image

**4.**
- Down sampling image
- Up sampling image using pyramid kernel
- Up sampling image using Bell kernel

**5.1**
- 2 bit uniform quantization image
- 4bits uniform quantization image
- 6bits uniform quantization image
- 2bits compandor quantization image
- 4bits compandor quantization image
- 6bits compandor quantization image

**5.2**
- 2bits uniform quantization image after subtracting noise
- 4bits uniform quantization image after subtracting noise
- 6bits uniform quantization image after subtracting noise
- 2bits compandor quantization image after subtracting noise
- 4bits compandor quantization image after subtracting noise
- 6bits compandor quantization image after subtracting noise

# Appendix

- **A method for reading the input and displaying the output image in MATLAB.**

```
% input image(gray)
file=fopen(fullfile('F:\Test images\Gray_lenna_256x256.raw'),'rb');
lena=fread(file,fliplr([256, 256]),'*uint8')';
fclose(file);

% input image(color)
file=fopen(fullfile('F:\Test images\Color_lenna_256x256.raw'),'rb');
lena=fread(file,fliplr([256, 256*3]),'*uint8')';
fclose(file);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                    %%%%% source code %%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% output image (at once)
figure, title('Problem_1_H1_result');
imshow(uint8(H1_result));
figure, title('Problem_2_R');
imshow(uint8(lena_R));
figure, title('Problem_2_G');
imshow(uint8(lena_G));
figure, title('Problem_2_B');
imshow(uint8(lena_B));
...
```