1. Color balance , Saturation,and Demosaicing

(1)White balance

Apply the white balance on "sky.jpg" and name the output image as "WhiteBalanceImage".

$$I(x,y) = 0.299 f_R(x,y) + 0.587 f_G(x,y) + 0.114 f_B(x,y)$$

$$k_R = \frac{\bar{I}}{\bar{f}_R}, k_G = \frac{\bar{I}}{\bar{f}_G}, k_B = \frac{\bar{I}}{\bar{f}_B}$$

$$\begin{bmatrix} g_R(x,y) \\ g_G(x,y) \\ g_B(x,y) \end{bmatrix} = \begin{bmatrix} k_R & & \\ & k_G & \\ & & k_B \end{bmatrix} \begin{bmatrix} f_R(x,y) \\ f_G(x,y) \\ f_B(x,y) \end{bmatrix}$$

```
img = imread("sky.jpg");
img_R = img(:,:,1);
img_G = img(:,:,2);
img_B = img(:,:,3);
img_I(:,:) = 0.299*img_R(:,:)+0.587*img_G(:,:)+0.114*img_B(:,:);
kr = mean(img_I(:))/mean(img_R(:));
kg = mean(img_I(:))/mean(img_G(:));
kb = mean(img_I(:))/mean(img_B(:));
gr = kr*img_R;
gg = kg*img_G;
gb = kb*img_B;
RGB_img = cat(3,gr,gg,gb);
imshow(RGB_img)
title("WhiteBalanceImage")
```

(2)Adjust Saturation

Increase the saturation of "sky.jpg" by 50% ,and name the output image as "SaturationImage".

Also,You cannot use the function“rgb2hsv()or hsv2rgb()”

$$I = \frac{1}{3}(R+G+B)$$

$$S = 1 - \frac{3min(R,G,B)}{R+G+B}$$

$$H = \begin{cases} \theta . G \geq B \\ 2\pi - \theta, G < B \end{cases}$$

$$\theta = cos^{-1}[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}]$$

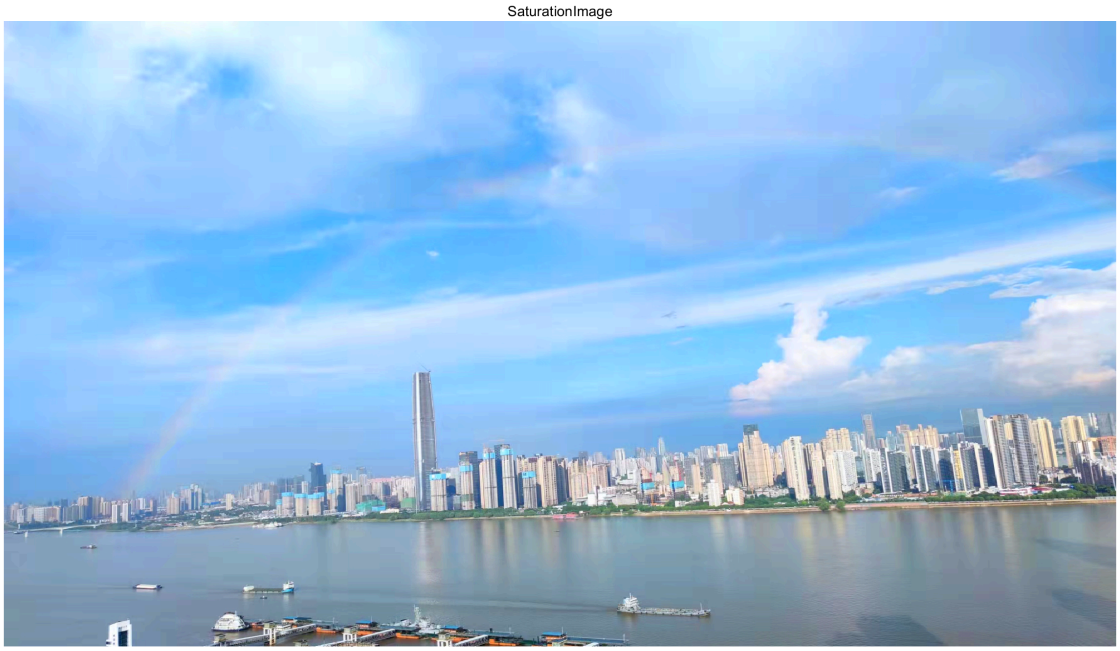
$1^\circ if\ H \in [0^\circ, 120^\circ):$
$H = H$
$\begin{cases} B = I(1-S) \\ R = I[1+\frac{S\cos H}{\cos(60^\circ - H)}] \\ G = 3I-(R+B) \end{cases}$

$2^\circ if\ H \in [120^\circ, 240^\circ):$
$H = H - 120^\circ$
$\begin{cases} R = I(1-S) \\ G = I[1+\frac{S\cos H}{\cos(60^\circ - H)}] \\ B = 3I-(R+G) \end{cases}$

$3^\circ if\ H \in [240^\circ, 360^\circ):$
$H = H - 240^\circ$
$\begin{cases} G = I(1-S) \\ B = I[1+\frac{S\cos H}{\cos(60^\circ - H)}] \\ R = 3I-(G+B) \end{cases}$

```
img_R = im2double(img_R);
img_G = im2double(img_G);
img_B = im2double(img_B);
img_I = (img_R+img_G+img_B)/3;
min_RGB = zeros(1279,2275);
for x = 1:1279
    for y = 1:2275
        min_RGB(x,y) = min([img_R(x,y),img_G(x,y),img_B(x,y)]);
        S = 1-3*min_RGB(x,y)/(img_R(x,y)+img_G(x,y)+img_B(x,y));
        S = S * 1.5;
        if (img_R(x,y)-img_G(x,y))*(img_R(x,y)-img_G(x,y))+(img_R(x,y)-img_B(x,y))*(img_G(x,y)-
            theta = pi/2;
        elseif (img_R(x,y)-img_G(x,y))*(img_R(x,y)-img_G(x,y))+(img_R(x,y)-img_B(x,y))*(img_G(x
            theta = acos(((img_R(x,y)-img_G(x,y))+(img_R(x,y)-img_B(x,y)))*(1/2)/sqrt((img_R(x,
        end
        if img_G(x,y) >= img_B(x,y)
            H = theta;
        elseif img_G(x,y) < img_B(x,y)
            H = 2*pi - theta;
        end
        if 0 <= H && H < 2*pi/3
            H = H;
            img_B(x,y) = img_I(x,y)*(1 - S);
            img_R(x,y) = img_I(x,y)*(1 + S*cos(H)/cos(pi/3 - H));
            img_G(x,y) = 3*img_I(x,y) -(img_R(x,y)+img_B(x,y));
        elseif 2*pi/3 <= H && H < 4*pi/3
            H = H - 2*pi/3;
            img_R(x,y) = img_I(x,y)*(1 - S);
            img_G(x,y) = img_I(x,y)*(1 + S*cos(H)/cos(pi/3 - H));
```

```
            img_B(x,y) = 3*img_I(x,y) -(img_R(x,y)+img_G(x,y));
        elseif 4*pi/3 <= H && H < 2*pi
            H = H - 4*pi/3;
            img_G(x,y) = img_I(x,y)*(1 - S);
            img_B(x,y) = img_I(x,y)*(1 + S*cos(H)/cos(pi/3 - H));
            img_R(x,y) = 3*img_I(x,y) -(img_G(x,y)+img_B(x,y));
        end
    end
end
img_RGB = cat(3,img_R,img_G,img_B);
imshow(img_RGB)
title("SaturationImage")
```



(3)Demosaicing

The initial image is given as "flower.mat". Please use the demosaicing algorithm to convert this single-channel image to a three-channel RGB image. Name the output image as "RGBImage". **You cannot use the function“demosaic()”**

```
load("flower.mat")
img_b = zeros(267,311);
img_r = zeros(267,311);
img_g = zeros(267,311);
a = im2double(flower);
img_b([1:2:267],[1:2:311]) = a([1:2:267],[1:2:311]);
img_r([2:2:267],[2:2:311]) = a([2:2:267],[2:2:311]);
img_g(2:2:267*311) = a(2:2:267*311);
for x = 2:2:267
    img_b(x,[1:2:311]) = (img_b(x-1,[1:2:311])+img_b(x+1,[1:2:311]))/2;
end
for x = 2:2:311
```

```
    img_b([1:267],x) = (img_b([1:267],x-1)+img_b([1:267],x+1))/2;
end
for x = 3:2:265
    img_r(x,[2:2:311]) = (img_r(x-1,[2:2:311])+img_r(x+1,[2:2:311]))/2;
end
for x = 3:2:309
    img_r([2:266],x) = (img_r([2:266],x-1)+img_r([2:266],x+1))/2;
end
img_r(:,1) = img_r(:,2);
img_r(:,311) = img_r(:,310);
img_r(1,:) = img_r(2,:);
img_r(267,:) = img_r(266,:);
img_g(1,1) = (img_g(1,2)+img_g(2,1))/2;
img_g(267,1) = (img_g(267,2)+img_g(266,1))/2;
img_g(1,311) = (img_g(1,310)+img_g(2,311))/2;
img_g(267,311) = (img_g(267,310)+img_g(266,311))/2;
img_g([3:2:265],1) = (img_g([2:2:264],1)+img_g([4:2:266],1)+img_g([3:2:265],2))/3;
img_g([3:2:265],311) = (img_g([2:2:264],311)+img_g([4:2:266],311)+img_g([3:2:265],310))/3;
img_g(1,[3:2:309]) = (img_g(1,[2:2:308])+img_g(1,[4:2:310])+img_g(2,[3:2:309]))/3;
img_g(267,[3:2:309]) = (img_g(267,[2:2:308])+img_g(267,[4:2:310])+img_g(266,[3:2:309]))/3;
img_g([2:2:266],[2:2:310]) = (img_g([1:2:265],[2:2:310])+img_g([3:2:267],[2:2:310])+img_g([2:2:
img_g([3:2:265],[3:2:309]) = (img_g([2:2:264],[3:2:309])+img_g([4:2:266],[3:2:309])+img_g([3:2:
RGBimage = cat(3,img_r,img_g,img_b);
imshow(RGBimage)
title("RGBimage")
```

2. Noise generation and degeneration

(1) ***Noise generation***

Add the following noise to the initial image separately:

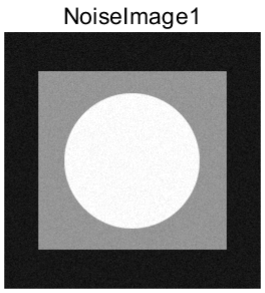
Gaussian noise: mean and variance are 25 and 25 respectively;

Salt-and-pepper noise: intensities are 0 and 255 with probabilities $P_{salt} = P_{pepper} = 0.05$;

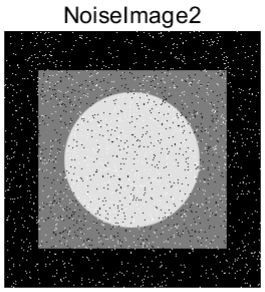
Name the output images as "NoiseImage1", "NoiseImage2" and display both images. **You are free to use the function "imnoise" or other built-in functions to generate the noise.**

```
clf;
```

```
load("init_img.mat")
img = im2double(init_img);
img1 = imnoise(img,"gaussian",25/255,25/(255^2));
imshow(img1)
title("NoiseImage1")
```



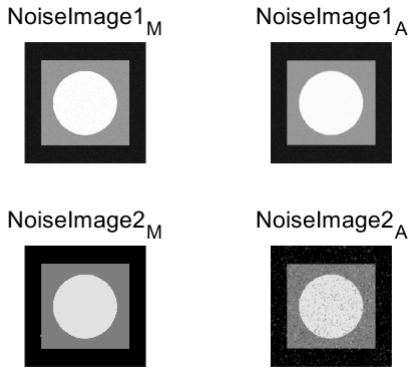

```
img2 = imnoise(img,'salt & pepper',0.05);
imshow(img2)
title("NoiseImage2")
```



(2) *Noise degeneration*

Please design a median filter and an average filter, and apply both your filters to the "NoiseImage1" and"NoiseImage2". Then display all your output image results.

```
clf;
average = fspecial('average');
img1_m = medfilt2(img1);
subplot(2,2,1),imshow(img1_m),title("NoiseImage1_M")
img1_a = imfilter(img1,average);
subplot(2,2,2),imshow(img1_a),title("NoiseImage1_A")
img2_m = medfilt2(img2);
subplot(2,2,3),imshow(img2_m),title("NoiseImage2_M")
img2_a = imfilter(img2,average);
subplot(2,2,4),imshow(img2_a),title("NoiseImage2_A")
```

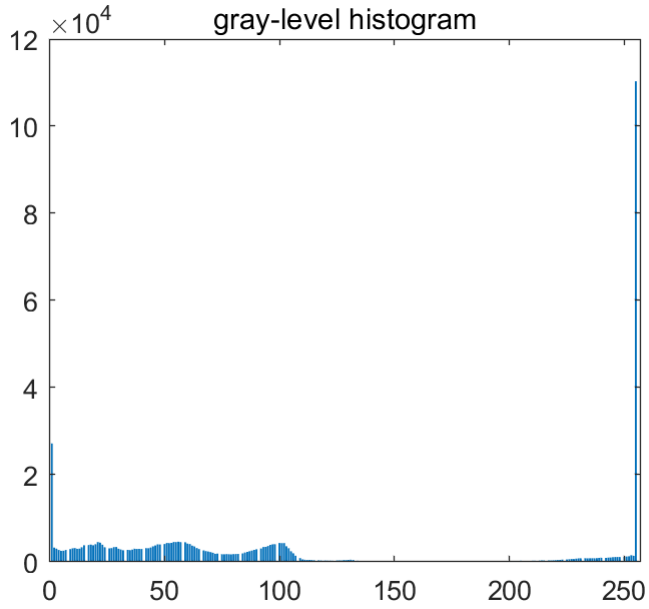


3. Log/gamma Transformation and histogram equalization and matching

(1)***Histogram***

Develop a function which computes the **gray-level histogram** of the "origin.tif". Then using this function to find the histogram and display it.

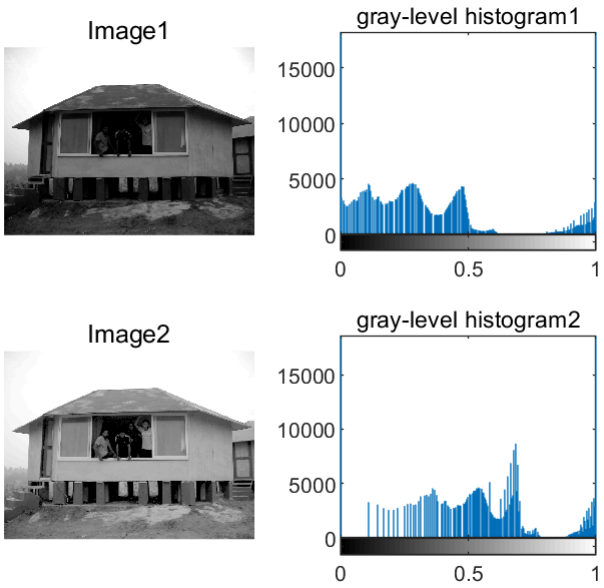
```
clf;
ori_img = imread("origin.png");
list = zeros(1,256);
for x = 1:1:600
    for y = 1:1:800
        locat = ori_img(x,y);
        list(1,locat+1) = list(1,locat+1)+1;
    end
end
bar(list)
title("gray-level histogram")
```



(2)Log Transformation and Gamma Transformation

use transform function $s = c\log(1+r)$ and $s = c \cdot r^{\gamma}$ to deal with the image.Display the output image and its gray-level histogram.

```
clf;
trs_img = im2double(ori_img);
trs_img1 = zeros(600,800);
trs_img2 = zeros(600,800);
for x = 1:1:600
    for y = 1:1:800
        trs_img1(x,y) = log2(trs_img(x,y)+1);
        trs_img2(x,y) = (trs_img(x,y))^0.4;
    end
end
subplot(2,2,1),imshow(trs_img1),title("Image1")
subplot(2,2,2),imhist(trs_img1),title("gray-level histogram1")
subplot(2,2,3),imshow(trs_img2),title("Image2")
subplot(2,2,4),imhist(trs_img2),title("gray-level histogram2")
```



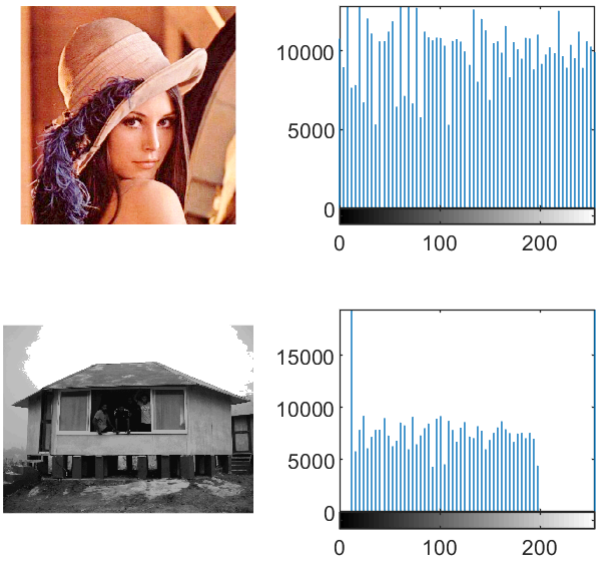
(3)*Histogram equalization*

Use the function histeq() to apply histogram equalization on the "source.jpg". Then using this function to equalize the histogram which you develop in 3.1. Display the output image and its gray-level histogram.

If you can design a function by youself instead of using function histeq() in this part, you can get a Bouns scores(5')

```
clf;
source = imread("source.jpg");
source_histeq = histeq(source);
subplot(2,2,1),imshow(source_histeq)
subplot(2,2,2),imhist(source_histeq)
```

```
img_histeq = histeq(ori_img);
subplot(2,2,3),imshow(img_histeq)
subplot(2,2,4),imhist(img_histeq)
```

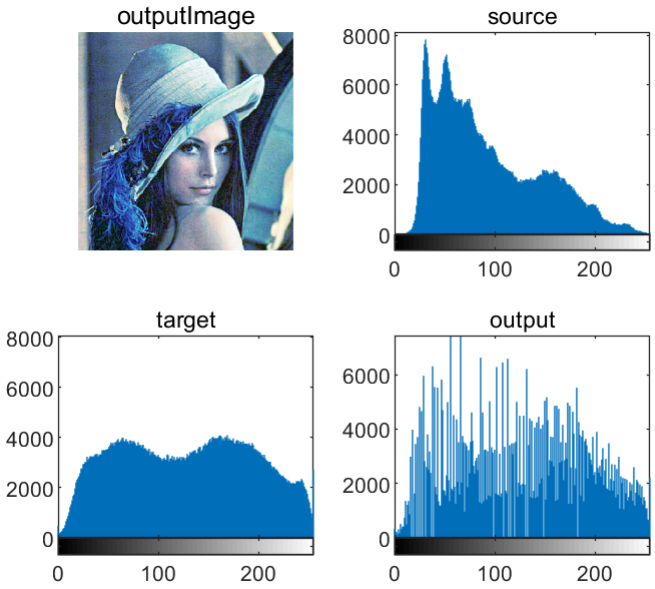


(4)*Histogram matching*

Use the function histeq() to apply the histogram matching between the source image and the target image. Use this function to display the output image. And you also need to display the gray-level histogram of the source image, the target image and the output image.

If you can design a function by youself instead of using function histeq() in this part, you can get a Bouns scores(5')

```
clf;
target = imread("target.jpg");
source_R = source(:,:,1);
source_G = source(:,:,2);
source_B = source(:,:,3);
target_R = target(:,:,1);
target_G = target(:,:,2);
target_B = target(:,:,3);
target_R_hist = imhist(target_R);
target_G_hist = imhist(target_G);
target_B_hist = imhist(target_B);
out_R = histeq(source_R,target_R_hist);
out_G = histeq(source_G,target_G_hist);
out_B = histeq(source_B,target_B_hist);
output = cat(3,out_R,out_G,out_B);
subplot(2,2,1),imshow(output),title("outputImage")
subplot(2,2,2),imhist(source),title("source")
subplot(2,2,3),imhist(target),title("target")
subplot(2,2,4),imhist(output),title("output")
```


outputImage
source
8000
6000
4000
2000
0
0
100
200
target
output