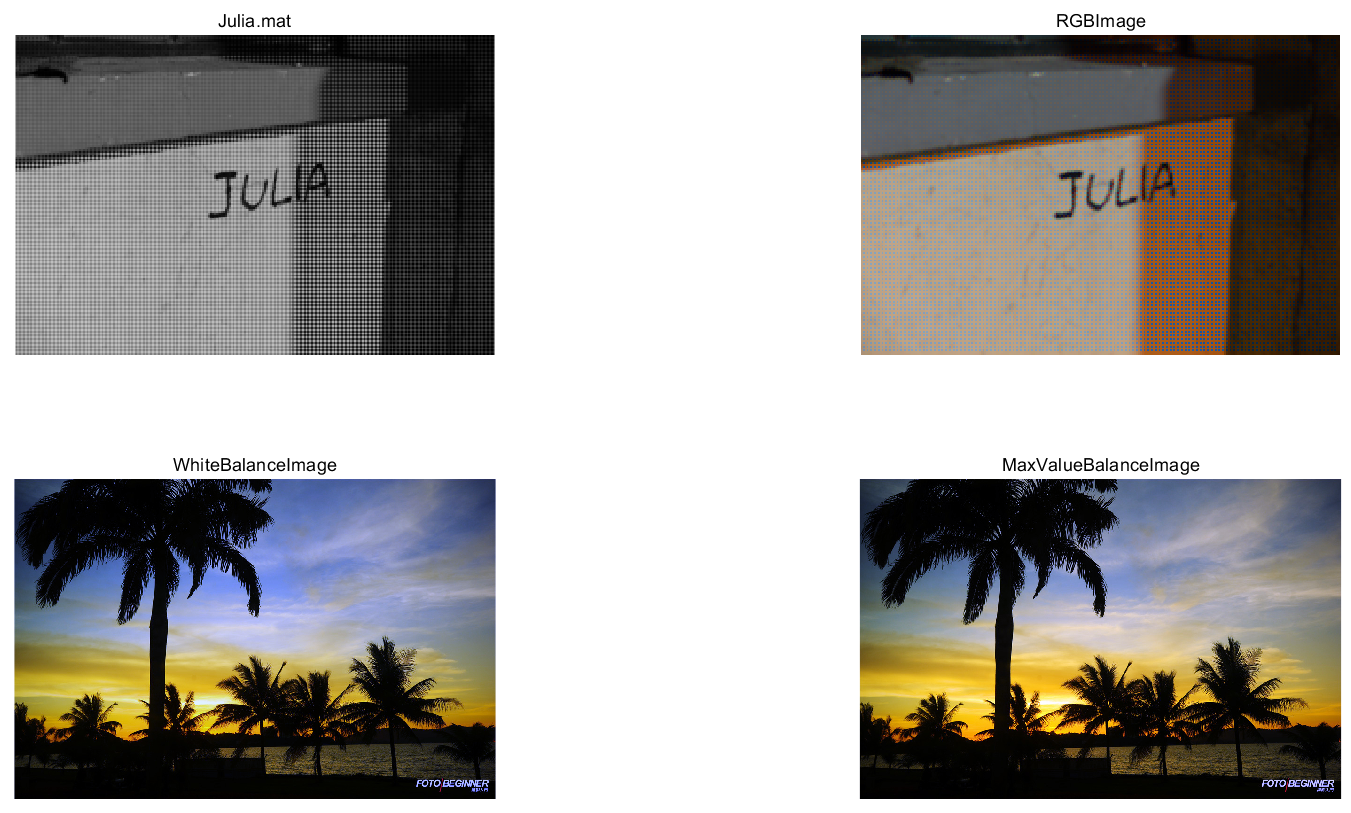
***1.******Demosaicing and color balance***



clear all;

clf;

load 'Q1\Julia.mat';

[n,m]=size(Julia);

a=double(Julia);

img\_B=uint8(zeros(n,m));

img\_G=uint8(zeros(n,m));

img\_R=uint8(zeros(n,m));

for i=[2:n-1]

for j=[2:m-1]

x1=a(i-1,j)+a(i+1,j)+a(i,j-1)+a(i,j+1);%相邻4个

x1=uint8(x1/4);

x2=a(i-1,j-1)+a(i-1,j+1)+a(i+1,j-1)+a(i+1,j+1);%角上4个

x2=uint8(x2/4);

x3=a(i,j-1)+a(i,j+1);%左右2个

x3=uint8(x3/2);

x4=a(i-1,j)+a(i+1,j);%上下2个

x4=uint8(x4/2);

if mod(i,2)==1%奇数行

if mod(j,2)==1%蓝色

img\_B(i,j)=uint8(a(i,j));

img\_G(i,j)=x1;

img\_R(i,j)=x2;

else%绿色

img\_B(i,j)=x4;

img\_G(i,j)=uint8(a(i,j));

img\_R(i,j)=x3;

end

else

if mod(j,2)==1%绿色

img\_B(i,j)=x4;

img\_G(i,j)=uint8(a(i,j));

img\_R(i,j)=x3;

else%红色

img\_B(i,j)=x2;

img\_G(i,j)=x1;

img\_R(i,j)=uint8(a(i,j));

end

end

end

end

img\_B(1,1)=uint8(a(1,1));

img\_G(1,1)=uint8((a(2,1)+a(1,2))/2);

img\_R(1,1)=uint8(a(2,2));

img\_B(n,1)=uint8(a(n,1));

img\_G(n,1)=uint8((a(n-1,1)+a(n,2))/2);

img\_R(n,1)=uint8(a(n-1,2));

img\_B(1,m)=uint8(a(1,m-1));

img\_G(1,m)=uint8(a(1,m));

img\_R(1,m)=uint8(a(2,m));

img\_B(n,m)=uint8(a(n,m-1));

img\_G(n,m)=uint8(a(n,m));

img\_R(n,m)=uint8(a(n-1,m));

for i=[2:n-1]

if mod(i,2)==1

img\_B(i,1)=uint8(a(i,1));

img\_G(i,1)=uint8((a(i-1,1)+a(i,2)+a(i+1,1))/3);

img\_R(i,1)=uint8((a(i-1,2)+a(i+1,2))/2);

img\_B(i,m)=uint8(a(i,m-1));

img\_G(i,m)=uint8(a(i,m));

img\_R(i,m)=uint8((a(i-1,m)+a(i+1,m))/2);

else

img\_B(i,1)=uint8((a(i-1,1)+a(i+1,1))/2);

img\_G(i,1)=uint8(a(i,1));

img\_R(i,1)=uint8(a(i,2));

img\_B(i,m)=uint8((a(i-1,m-1)+a(i+1,m-1))/2);

img\_G(i,m)=uint8((a(i-1,m)+a(i,m-1)+a(i+1,m))/3);

img\_R(i,m)=uint8(a(i,m));

end

end

for i=[2:m-1]

if mod(i,2)==1

img\_B(1,i)=uint8(a(1,i));

img\_G(1,i)=uint8((a(1,i-1)+a(2,i)+a(1,i+1))/3);

img\_R(1,i)=uint8((a(2,i-1)+a(2,i+1))/2);

img\_B(n,i)=uint8(a(n,i));

img\_G(n,i)=uint8((a(n,i-1)+a(n-1,i)+a(n,i+1))/3);

img\_R(n,i)=uint8((a(n-1,i-1)+a(n-1,i+1))/2);

else

img\_B(1,i)=uint8((a(1,i-1)+a(1,i+1))/2);

img\_G(1,i)=uint8(a(1,i));

img\_R(1,i)=uint8(a(2,i));

img\_B(n,i)=uint8((a(n,i-1)+a(n,i+1))/2);

img\_G(n,i)=uint8(a(n,i));

img\_R(n,i)=uint8(a(n-1,i));

end

end

result=cat(3,img\_R,img\_G,img\_B);

sky=imread('Q1\sky.jpg');

fr=sky(:,:,1);

fg=sky(:,:,2);

fb=sky(:,:,3);

i=0.299\*fr+0.587\*fg+0.114\*fb;

imean=sum(i(:));

frmean=sum(fr(:));fgmean=sum(fg(:));fbmean=sum(fb(:));

kr=imean/frmean;kg=imean/fgmean;kb=imean/fbmean;

gr=kr\*fr;gg=kg\*fg;gb=kb\*fb;

whitebalance=cat(3,gr,gg,gb);

[m,n]=size(fr);

rmax=max(fr(:));gmax=max(fg(:));bmax=max(fb(:));

srgb=min([rmax,gmax,bmax]);

nr=sum(sum(fr>srgb));ng=sum(sum(fg>srgb));nb=sum(sum(fb>srgb));

nmax=max([nr,ng,nb]);

r=sort(fr(:),'descend');tr=r(nmax);

g=sort(fg(:),'descend');tg=g(nmax);

b=sort(fb(:),'descend');tb=b(nmax);

kr=srgb/tr;kg=srgb/tg;kb=srgb/tb;

gr=kr\*fr;gg=kg\*fg;gb=kb\*fb;

colorbalance=cat(3,gr,gg,gb);

subplot(2,2,1);imshow(uint8(Julia));title('Julia.mat');

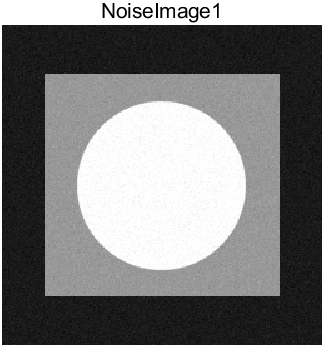
subplot(2,2,2);imshow(result);title('RGBImage');

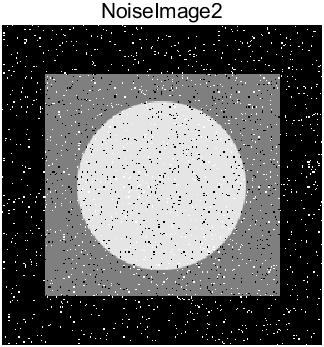
subplot(2,2,3);imshow(whitebalance);title('WhiteBalanceImage')

subplot(2,2,4);imshow(colorbalance);title('MaxValueBalanceImage');

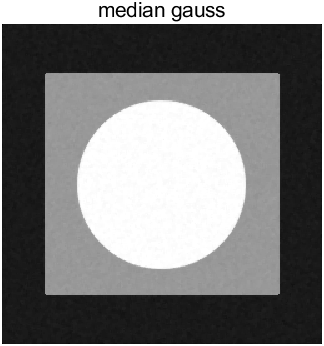
***2.******Noise generation and degeneration***

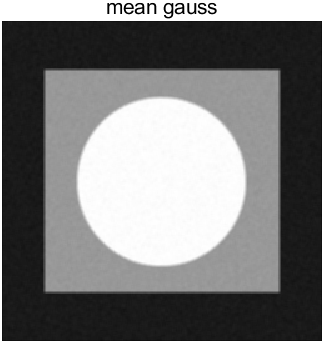
(1) *Noise generation*

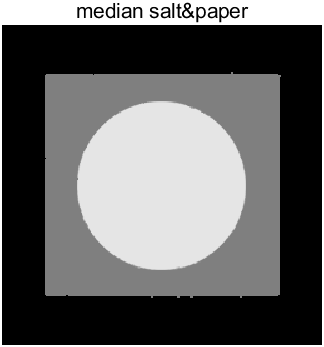


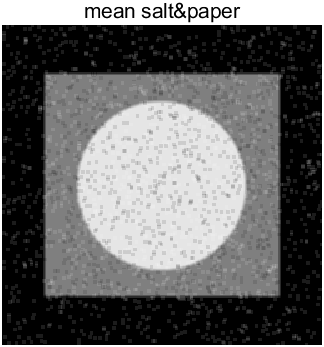


(2) *Noise degeneration*









load Q2\init\_img.mat

a=init\_img;

imshow(a);title("init img");

b=imnoise(a,'gaussian',25/255,25/255/255);

c=imnoise(a,'salt & pepper',0.05);

imshow(b);title("NoiseImage1");

imshow(c);title("NoiseImage2");

median\_gauss=medfilt2(b);

median\_saltpaper=medfilt2(c);

x=fspecial('average');

mean\_gauss=imfilter(b,x);

mean\_saltpaper=imfilter(c,x);

imshow(median\_gauss);title("median gauss");

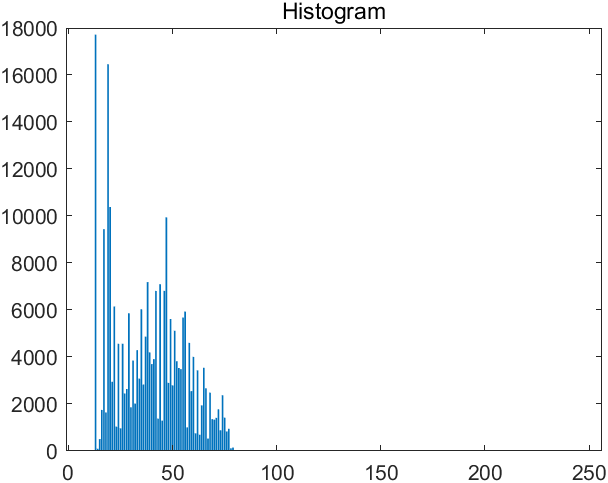
imshow(mean\_gauss);title("mean gauss");

imshow(median\_saltpaper);title("median salt&paper");

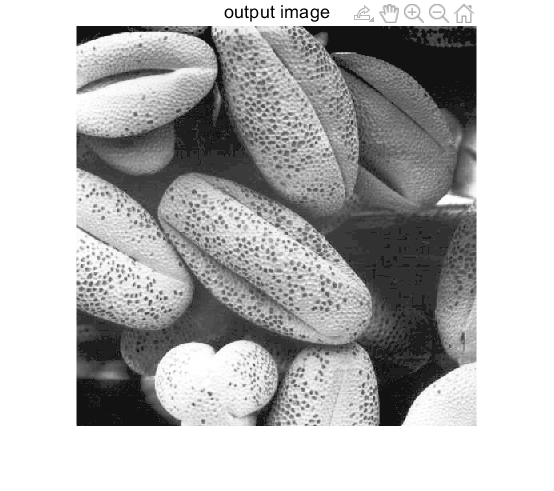
imshow(mean\_saltpaper);title("mean salt&paper");

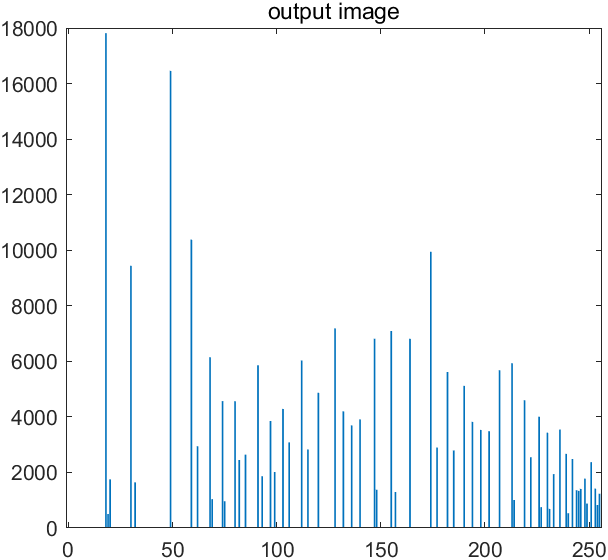
***3.******Histogram equalization and histogram matching***

(1)*Histogram*



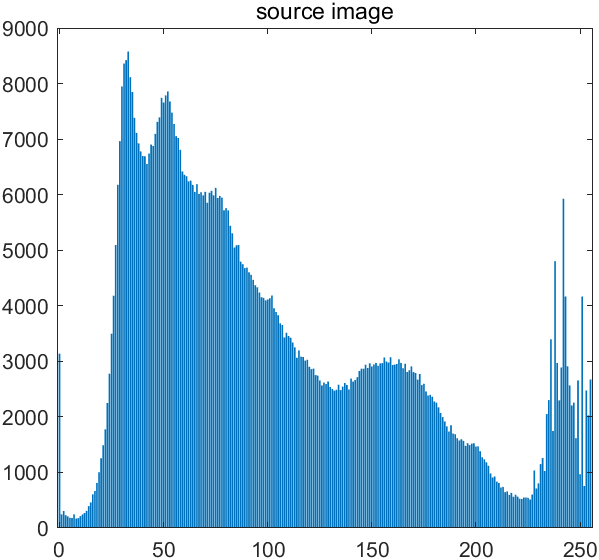
(2)*Histogram equalization*

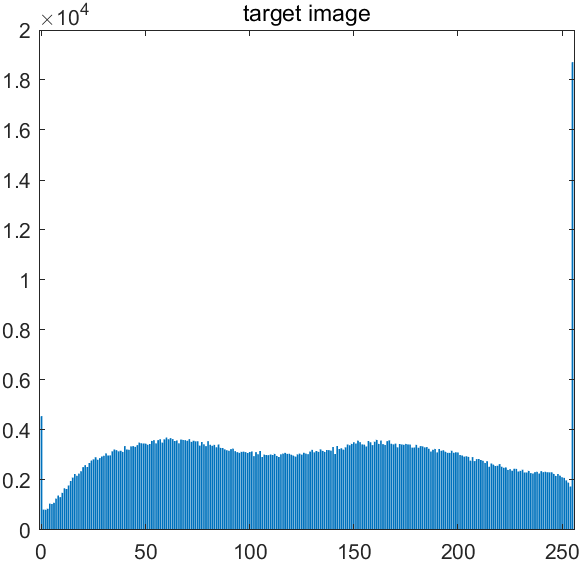


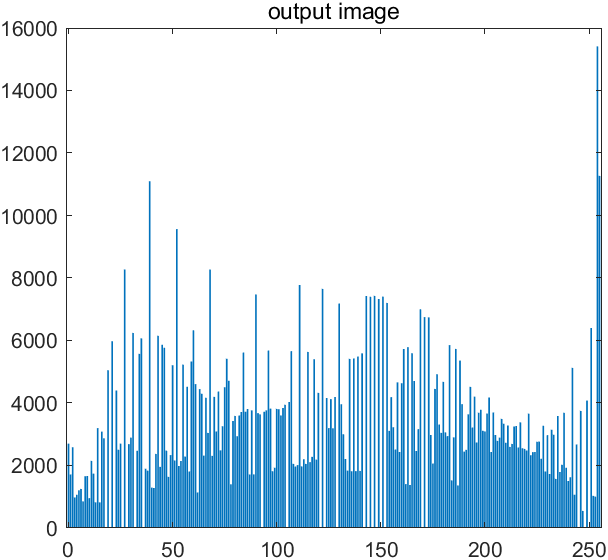


(3)*Histogram matching*









load Q3\original\_img.mat;

source=imread("Q3\source.jpg");

target=imread("Q3\target.jpg");

[n,m]=size(original\_img);

cnt=zeros(256,1);

bin=zeros(256,1);

for i=[1:256]

bin(i,1)=i-1;

end

for i=[1:n]

for j=[1:m]

cnt(original\_img(i,j)+1,1)=cnt(original\_img(i,j)+1,1)+1;

end

end

bar(bin,cnt);title("Histogram");

p=cnt/(n\*m);

p=cumsum(p);

p=p\*255;

p=round(p);

output=uint8(zeros(n,m));

for i=[1:n]

for j=[1:m]

output(i,j)=p(original\_img(i,j)+1,1);

end

end

imshow(output);title("output image");

[counts,binLocations]=imhist(output);

bar(binLocations,counts);title("output image");

[source\_n,source\_m,dim]=size(source);

[target\_n,target\_m,dim]=size(target);

x=uint8(zeros(source\_n,source\_m,3));

for k=[1:3]

source\_x=source(:,:,k);%获取原图像k通道

source\_hist=imhist(source\_x);%获取原图像k通道直方图

target\_x=target(:,:,k);%获取匹配图像k通道

target\_hist=imhist(target\_x);%获取匹配图像k通道直方图

%x(:,:,k)=histeq(source\_x,target\_hist);

source\_p=cumsum(source\_hist)/(source\_n\*source\_m);

target\_p=cumsum(target\_hist)/(target\_n\*target\_m);

map=zeros(1,256);%从0到255分别的映射

%help min

for i=[1:256]

[val,pos]=min(abs(source\_p(i)-target\_p));

%找到target\_p中与source(i)最接近的位置

map(i)=pos-1;

end

x(:,:,k)=map(double(source\_x)+1);

end

result=cat(3,x(:,:,1),x(:,:,2),x(:,:,3));

imshow(result);

[counts,binLocations]=imhist(source);

bar(binLocations,counts);title("source image");

[counts,binLocations]=imhist(target);

bar(binLocations,counts);title("target image");

[counts,binLocations]=imhist(result);

bar(binLocations,counts);title("output image");