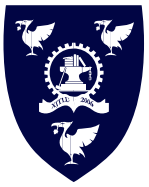

Xi'an Jiaotong-Liverpool University

西交利物浦大學

Image Processing EEE412

Lab 2：Image enhancement

XUN XU 1926930

2019/10/24

Task1

i

Using imread() function to load into the image

Code:

```
im= imread('lenna512.bmp');
```

ii

Result:

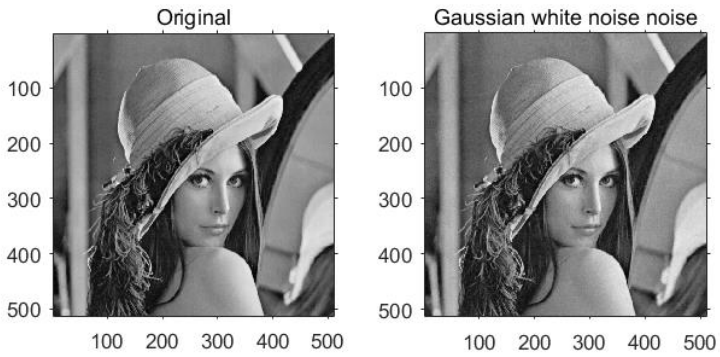

Figure 1

The calculated noise variance and mean are: 9.9762 and -0.0035. Basically consistent with the statistical characteristics of Gaussian white noise

Code:

```
function [img_noise] =im_wn(img)


img_noise=double(img)+sqrt(10)*randn(size(img));%using randn function to produce random number


end
```

```
%Check if the properties of the noise are reasonable

noise = img_noise1 - double(im);

std=std2(noise)    ;       % Standard deviation

mean=mean2(noise)  ;       % Mean of two-dimensional array

disp(std*std);

disp(mean);
```

iii

Result:

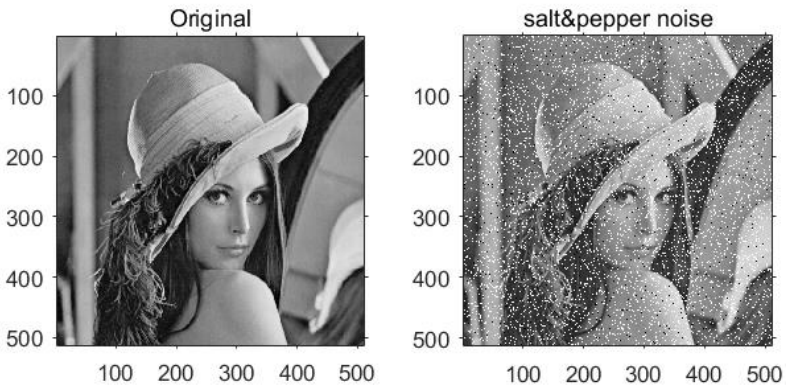

Figure 2

Code:

```
function [ image ] =    im_SP( image )


[width,height,z]=size(image);

if(z>1)

      image=rgb2gray(image);%if it is not a grey image,then transform it

end

k1=0.1;%Set the ratio of the random number

k2=0.1;
```

```
a1=rand(height,width)<k1;%Generate a random number for the ordinate

a2=rand(height,width)<k2;%Generate random numbers for the abscissa

image(a1&a2)=0;

image(a1& ~a2)=255;

subplot(1,2,2);

end
```

iv

Image	im_wn	im_SP	im_low_dynamic_range
PSNR (dB)	38.1068	15.2281	23.9912

By comparing the psnr values, we can know that adding Gaussian white noise, low dynamic range, and adding three pictures of salt and pepper noise decreases in comparison with the original picture, which is consistent with our visual observation, in which Gaussian white noise is the most common and basic noise, and the salt and pepper noise is also called impulse noise. The appearance of black and white points has a greater impact on the image.

v

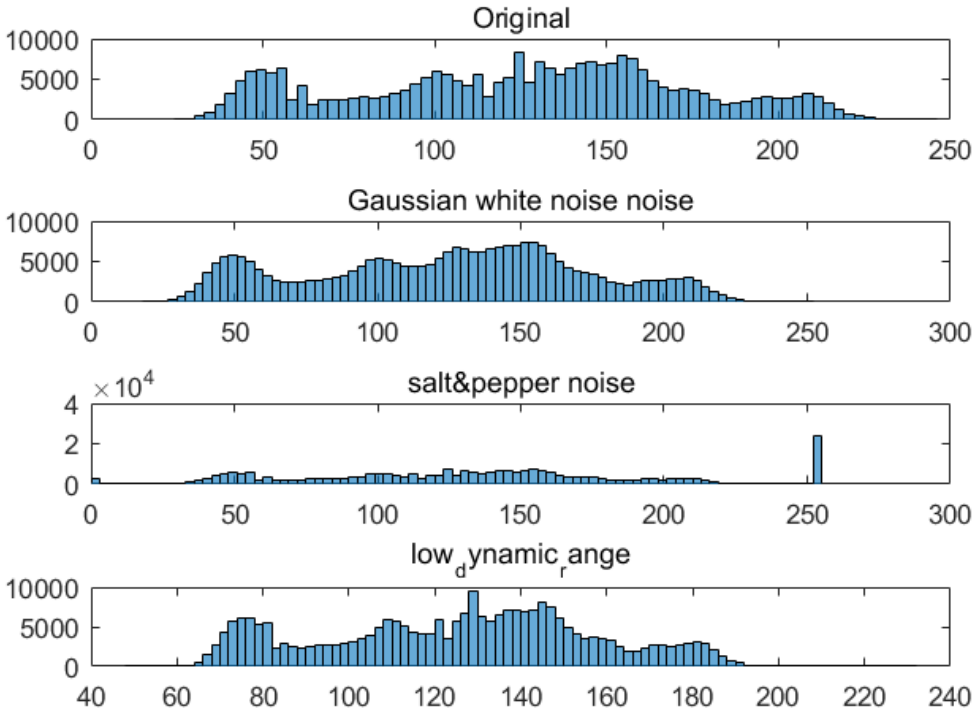

Figure 3

It can be seen that the histogram of the image with higher contrast is more extended.

Vi

Image	im_wn10	im_wn100	im_wn1000
PSNR (dB)	47.7645	57.6143	Inf

Because the noise of the pictures we add is subject to the normal distribution of ~(0,σ^2), the noise of the image processed by this method will have a mean of 0, and as k increases , it will tend to 0. That is to say, the variance of the image noise tends to zero, and the image is closer and closer to the original image. Therefore, as k increases, the value of psnr becomes larger and larger, and the image is getting closer to the original image.

Code:

```
im= imread('lenna512.bmp');


im_sum=im_wn(im);
k=10;
for i=2:k
    im_k=im_wn(im);
    im_sum=double(im_sum)+im_k;
end
im_ave=im_sum/k;
im_wn10=im_ave;


im_sum=im_wn(im);
k=100;
for i=2:k
    im_k=im_wn(im);
    im_sum=double(im_sum)+im_k;
end
im_ave=im_sum/k;
im_wn100=im_ave;


im_sum=im_wn(im);
```

```
k=1000;
for i=2:k
        im_k=im_wn(im);
        im_sum=double(im_sum)+im_k;
end
im_ave=im_sum/k;
im_wn1000=im_ave;


disp(psnr(uint8(im_wn10),im));
disp(psnr(uint8(im_wn100),im));
disp(psnr(uint8(im_wn1000),im));


figure;
subplot(141);imshow(im,[]);title('Original');axis on;
subplot(142);imshow(im_wn10,[]);title('imwn10');axis on;
subplot(143);imshow(im_wn100,[]);title('imwn100');axis on;
subplot(144);imshow(im_wn1000,[]);title('imwn1000');axis on;
```

Task2

(1)

Result:

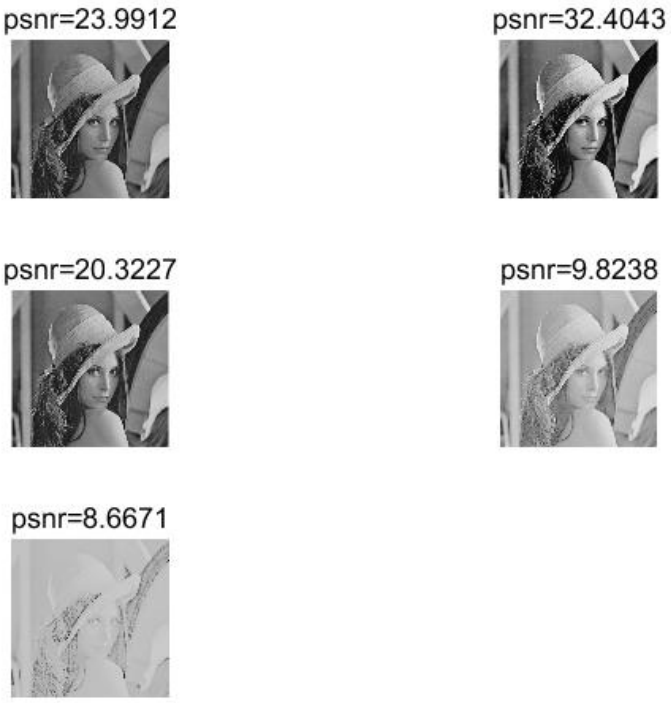

Figure 4

For different groups of (X1,Y2) and (X2,Y2),its psnr values are as follows:

0.05,0.05,0.95,0.95	0.3,0.15,0.7,0.83	0.1,0.15,0.85,0.95	0.3,0.7,0.4,0.75	0.3,0.8,0.8,0.85
23.9912	32.4043	20.3227	9.8238	8.6671

Code:

```
im=imread('lenna512_low_dynamic_range.bmp');

im1=imread('lenna512.bmp');



figure;

out1=linear_mapping_transform(im,0.05,0.05,0.95,0.95);

subplot(3,2,1);

imshow(out1,[]);
```

```
disp(psnr(out1,im2double(im1)));
title(['psnr=',num2str(psnr(out1,im2double(im1)))]);

out2=linear_mapping_transform(im,0.3,0.15,0.7,0.83);
subplot(3,2,2);
imshow(out2,[]);
disp(psnr(out2,im2double(im1)));
title(['psnr=',num2str(psnr(out2,im2double(im1)))]);

out3=linear_mapping_transform(im,0.1,0.15,0.85,0.95);
subplot(3,2,3);
imshow(out3,[]);
disp(psnr(out3,im2double(im1)));
title(['psnr=',num2str(psnr(out3,im2double(im1)))]);

out4=linear_mapping_transform(im,0.3,0.7,0.4,0.75);
subplot(3,2,4);
imshow(out4,[]);
disp(psnr(out4,im2double(im1)));
title(['psnr=',num2str(psnr(out4,im2double(im1)))]);

out5=linear_mapping_transform(im,0.3,0.8,0.8,0.85);
subplot(3,2,5);
imshow(out5,[]);
disp(psnr(out5,im2double(im1)));
title(['psnr=',num2str(psnr(out5,im2double(im1)))]);

figure;
histeq(im);
```

```
subplot(1,2,1);

imshow(out2);

title(['psnr=',num2str(psnr(out2,im2double(im1)))]);

subplot(1,2,2);

imshow(b);

title(['psnr=',num2str(psnr(histeq(im),im1))]);
```

(2)

Results:

Figure 5

Using linear mapping transform to enhance the contrast of im_low_dynamic_range, although the psnr value is larger than the histogram equalization, the contrast effect is not as good as the histeq(), indicating that the contrast is not necessarily closer to the original picture.

Task3

The sobel operator is calculated as follows:

$$\Delta_x f(x,y) = [f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)] - [f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)]$$
$$\Delta_y f(x,y) = [f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)] - [f(x+1,y) + 2f(x+1,y) + f(x+1,y+1)]$$

Results:

Figure 6

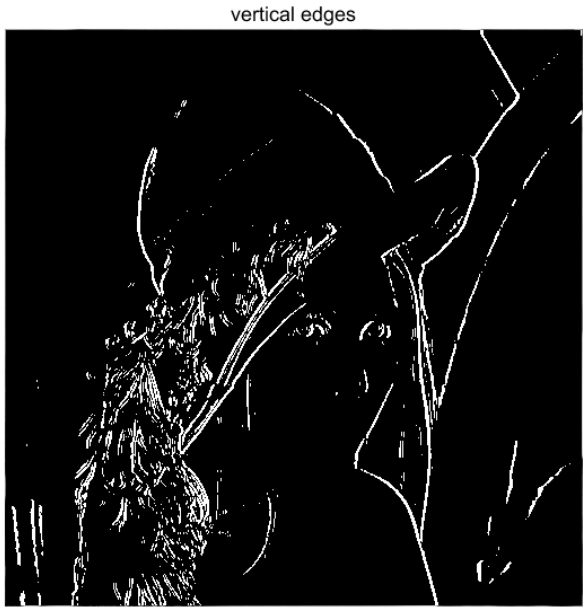

Figure 7

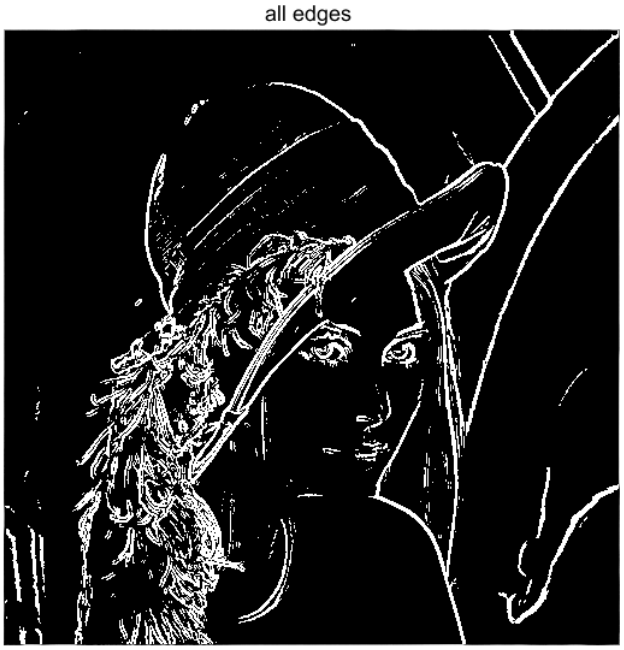

Figure 8

Comparing the three figures, it can be seen that the sobel operator is better for vertical edge recognition.

Code:

```
clear;
im=imread('lenna512.bmp');
grayimage=mat2gray(im);
[m,n]=size(grayimage);

newgrayimage=grayimage;
sobelNum=0;
sobelThreshold=0.7;
for j=2:m-1
    for k=2:n-1
        sobelNum=abs(grayimage(j-1,k-1)+2*grayimage(j-1,k)+grayimage(j-1,k+1)-grayimage(j+1,k-1)-2*grayimage(j+1,k)-
grayimage(j+1,k+1));
        if(sobelNum > sobelThreshold)
            newgrayimage(j,k)=255;
        else
            newgrayimage(j,k)=0;
        end
    end
end
figure;
imshow(newgrayimage);
title('horizontal edges');

newgrayimage=grayimage;
sobelNum=0;
sobelThreshold=0.7;
for j=2:m-1
    for k=2:n-1
        sobelNum=abs(grayimage(j-1,k+1)+2*grayimage(j,k+1)+grayimage(j+1,k+1)-grayimage(j-1,k-1)-2*grayimage(j,k-1)-
```

```
grayimage(j+1,k-1));
            if(sobelNum > sobelThreshold)
                newgrayimage(j,k)=255;
            else
                newgrayimage(j,k)=0;
            end
        end
end
figure;
imshow(newgrayimage);
title('vertical edges');


newgrayimage=grayimage;
sobelNum=0;
sobelThreshold=0.7;
for j=2:m-1
        for k=2:n-1
            sobelNum=abs(grayimage(j-1,k+1)+2*grayimage(j,k+1)+grayimage(j+1,k+1)-grayimage(j-1,k-1)-2*grayimage(j,k-1)-
grayimage(j+1,k-1))+abs(grayimage(j-1,k-1)+2*grayimage(j-1,k)+grayimage(j-1,k+1)-grayimage(j+1,k-1)-2*grayimage(j+1,k)-
grayimage(j+1,k+1));
            if(sobelNum > sobelThreshold)
                newgrayimage(j,k)=255;
            else
                newgrayimage(j,k)=0;
            end
        end
end
figure;
imshow(newgrayimage);
```

```
title('all edges ');
```

Task4

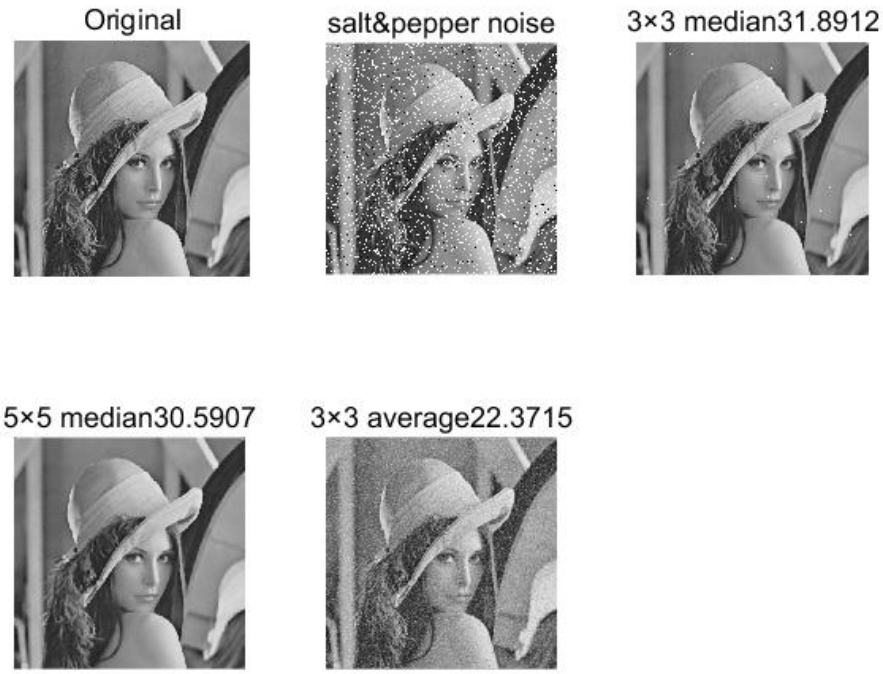

Figure9

(1)

3×3

Sharp edges and features with 3*3 filter in the image are preserved well, but there is a little noise that has not been eliminated

5×5

The noise is basically eliminated, but sharp edges and features in the image are blurred, so the contrast is reduced

(2)

For salt&pepper noise, median filtering performs better than mean filtering (average filtering) .

Both mean filtering and median filtering can smooth out the image and take care of the noise.

The mean value filtering uses a linear method to average the pixel values in the entire window range. The mean value filtering itself has inherent defects, that is, it does not protect the image details well, and the image denoising also destroys the details of the image. The image is blurred and the noise points are not well removed. Mean filtering performs better for Gaussian noise and performs poorly for salt&pepper noise.

The median filtering uses a nonlinear method, which is very effective in smoothing the impulse

noise. At the same time, it can protect the sharp edges of the image and select the appropriate point to replace the value of the pollution point. Therefore, the processing effect is good and the salt and pepper noise performance is better. Poor performance for Gaussian noise

Code:

```
im=imread('lenna512.bmp');
J=im_SP(im);
subplot(231),imshow(im);title('Original');
subplot(232),imshow(J);title('salt&pepper noise')
k1=medfilt2(J); %3×3 template median filtering
k2=medfilt2(J,[5 5]); %5×5 template median filtering
h1=fspecial('average',3);
k3=imfilter(J,h1);%     3×3 template average filtering

subplot(233),imshow(k1);title(['3×3 median',num2str(psnr(k1,im))])
subplot(234),imshow(k2);title(['5×5 median',num2str(psnr(k2,im))])
subplot(235),imshow(k3);title(['3×3 average',num2str(psnr(k3,im))])

a=psnr(k1,im);
disp(a);
b=psnr(k2,im);
disp(b);
c=psnr(k3,im);
disp(c);
```

```
   psnr1=31.8912


   psnr2=30.5907
```

```
psnr3=22.3715
```