图像处理作业二实验报告

姓名：果宏帅　　班级：软工一班　　学号：3017218058

一. 本次实验需要实现均值滤波器、顺序统计滤波器、自适应滤波器。针对每种滤波器分别展示其算法，相应实验结果及分析展示在算法后面。(对于高斯噪声与椒盐噪声,每种滤波器算法一致,故只展示一种)

二. 噪声生成算法

本次实验噪声生成算法采用 Matlab 的 imnose()函数,分别生成高斯噪声和椒盐噪声,对应 matlab 函数为 addGaussianNoise()和 addSaltNoise()
高斯噪声生成算法具体代码:

```
function [noise_picture] = addGaussianNoise()
img = imread('test.jpg');
%添加高斯噪声,均值为 0，方差为 0.01
noise_picture = imnoise(img,'gaussian',0,0.01);
%subplot(1,2,1);
%imshow(img),title('原始图像');
%subplot(1,2,2);
%imshow(noise_picture),title('均值为 0，方差为 0.01 的高斯噪声图像');
```

椒盐噪声生成算法具体代码:

```
%使用 imnoise 函数添加噪声
function [noise_picture] =  addSaltNoise()
img = imread('test.jpg');
%添加椒盐噪声 ，密度为 0.01
noise_picture = imnoise(img,'salt & pepper',0.01);
%subplot(1,2,1);
%imshow(img),title('原始图像');
%subplot(1,2,2);
%imshow(noise_picture),title('密度为 0.05 的椒盐噪声图像');
```

三. 均值滤波器

3.1 算数均值滤波器(ArithmeticMeanFilter)

代码如下：

```
function [] = ArithmeticMeanFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        new_salt(i,j,:)=sum(sum(sa_temp))/9;
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('算数均值处理后');
```

实验结果：

高斯噪声

椒盐噪声：

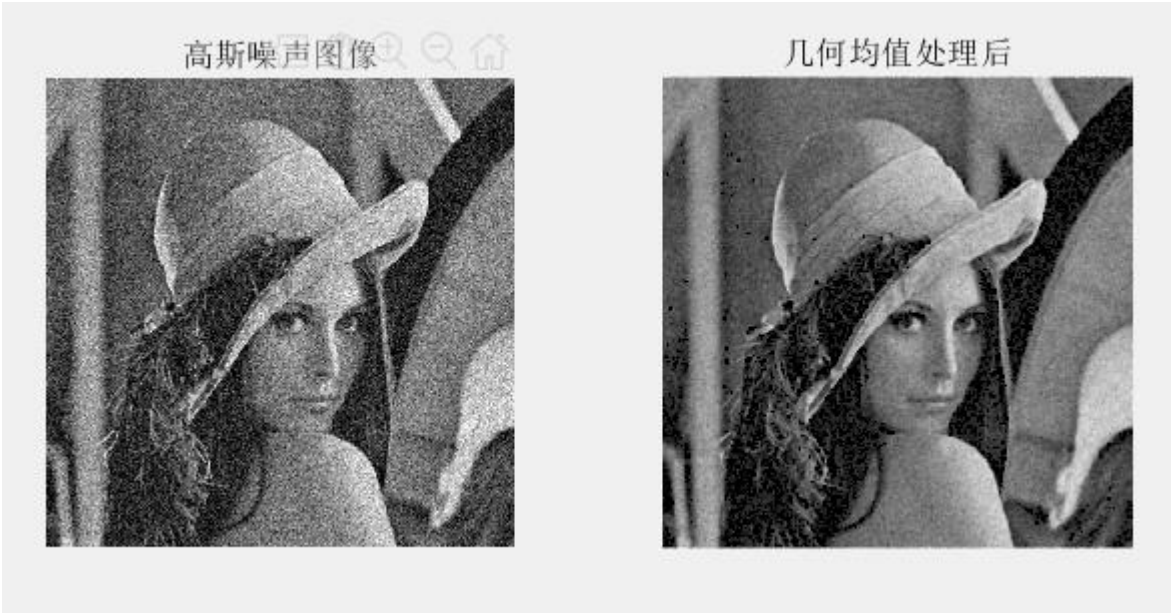
3.2几何均值滤波器(GeometricMeanFilter)

代码如下:

```
function [] = GeometricMeanFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        new_salt(i,j,:)=(prod(prod(sa_temp)))^(1/9);
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('几何均值处理后');
```

实验结果:

高斯噪声:



椒盐噪声:

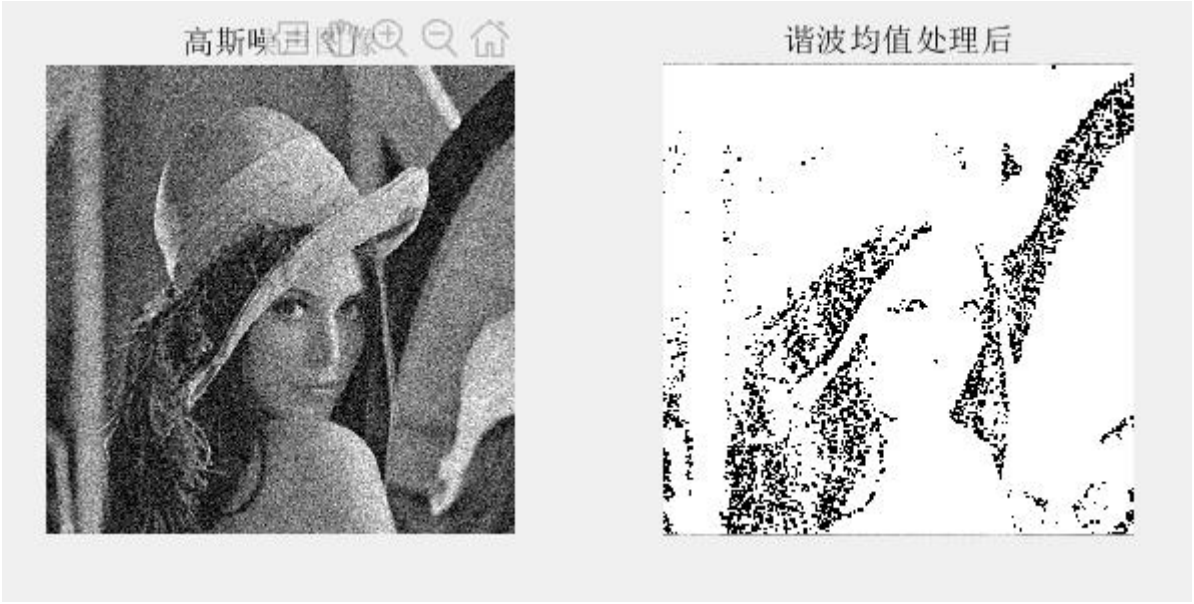
3.3谐波均值滤波器(HarmonicMeanFilter)

代码如下:

```
function [] = HarmonicMeanFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        new_salt(i,j,:)=9/(sum(sum(1./sa_temp)));
    end
```

```
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('谐波均值处理后');
```

实验结果:
高斯噪声:



椒盐噪声:

3.4逆谐波均值滤波器(InverseHarmonicMeanFilte)

代码如下:

```
function [] = InverseHarmonicMeanFilte()
```

```
%阶数为 1.5
q=1.5;
%处理椒盐噪声
salt_img=addSaltNoise();
salt_img=im2double(salt_img);
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        sa_temp1=sum(sum(sa_temp.^(q+1)));
        sa_temp2=sum(sum(sa_temp.^q));
        new_salt(i,j,:)=sa_temp1/sa_temp2;
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('逆谐波均值处理后');
```

实验结果:
高斯噪声:

椒盐噪声:

四. 统计排序滤波器

4.1中值滤波器(MedianFilter)

代码如下:

```
function [] = MedianFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        new_salt(i,j,:)=median(sa_temp(:));
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('中值滤波器处理后');
```

实验结果:

高斯噪声:

椒盐噪声:

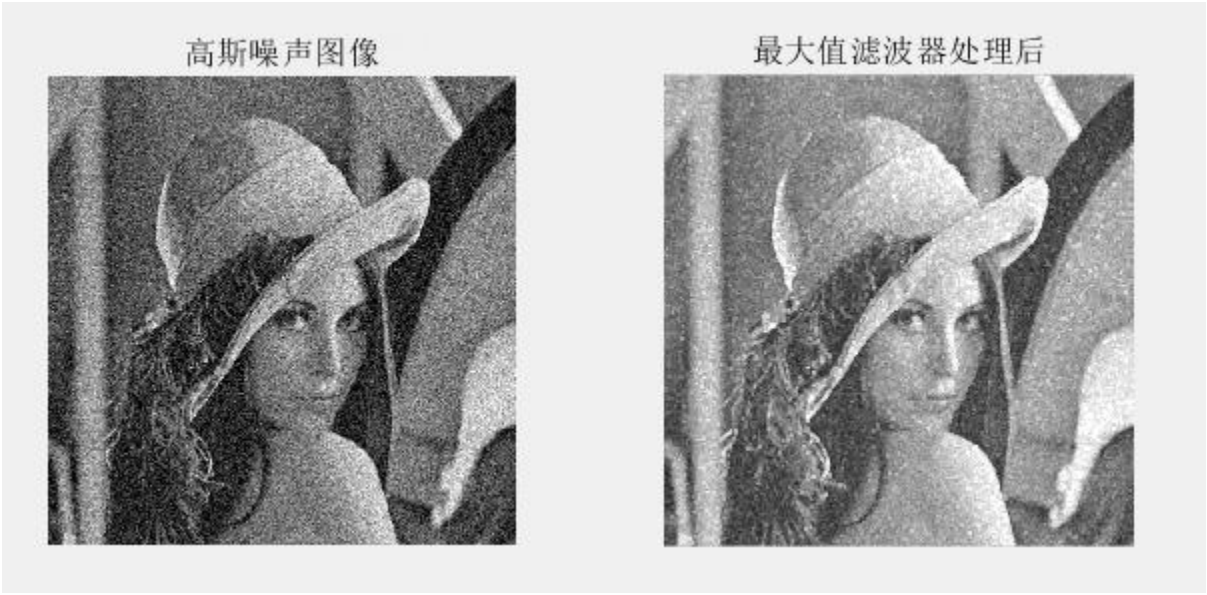
4.2最大值滤波器(MaximumFilter)

代码如下:

```
function [] = MaximumFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        new_salt(i,j,:)=max(sa_temp(:));
```

```
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('最大值滤波器处理后');
```

实验结果:
高斯噪声:



椒盐噪声:

4.3最小值滤波器(MinimumFilter)

代码如下:

```
function [] = MinimumFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
salt_img=im2double(salt_img);
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        new_salt(i,j,:)=min(sa_temp(:));
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('最小值滤波器处理后');
```

实验结果：
高斯噪声：

椒盐噪声：

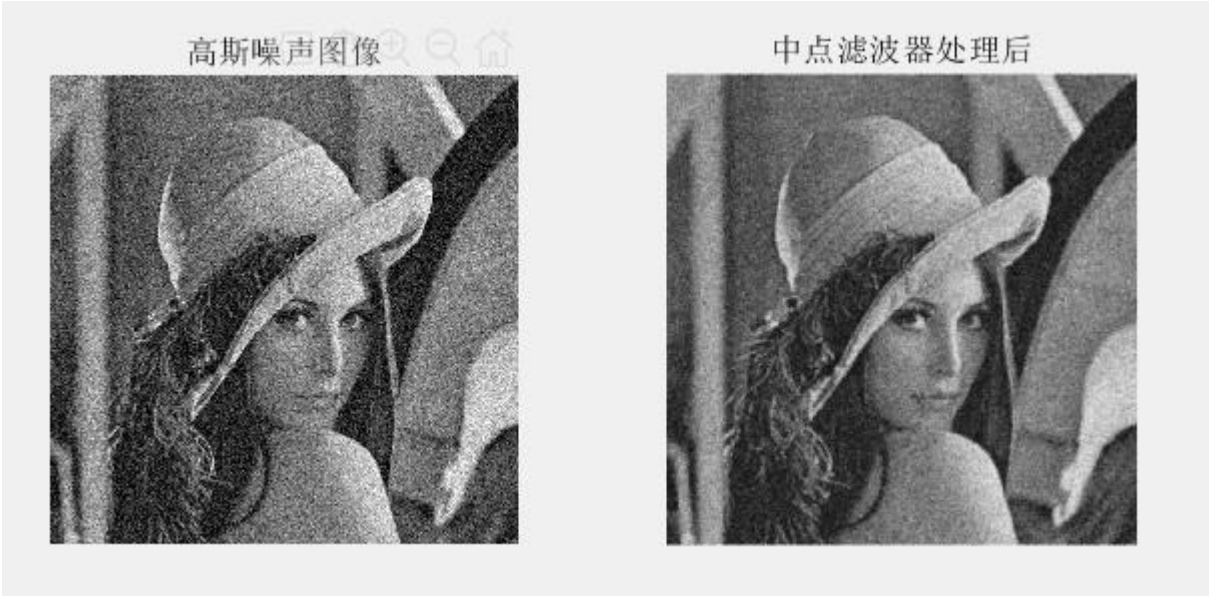
4.4中点滤波器(MidpointFilter)

代码如下:

```
function [] = MidpointFilter()
%处理椒盐噪声
salt_img=addSaltNoise();
salt_img=im2double(salt_img);
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        temp1=min(sa_temp(:));
        temp2=max(sa_temp(:));
        new_salt(i,j,:)=(temp1+temp2)/2;
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('中点滤波器处理后');
```

实验结果:

高斯噪声:



椒盐噪声：

4.5修正后的阿尔法均值滤波器(AlphaFilter)

代码如下：

```
function [] = AlphaFilter()
%设定 d=3
d=3;
d_max=floor(d/2);
d_min=ceil(d/2);
%处理椒盐噪声
salt_img=addSaltNoise();
salt_img=im2double(salt_img);
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
```

```
for i=2:sa_h-1
    for j=2:sa_w-1
        %获得 3*3 矩阵
        sa_temp=salt_img(i-1:i+1,j-1:j+1);
        temp1=sort(sa_temp(:));
        temp2=temp1(1+d_min:9-d_max);
        new_salt(i,j,:)=sum(temp2)/(9-d);
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('alpha 滤波器处理后');
```

实验结果：
高斯噪声：

椒盐噪声：

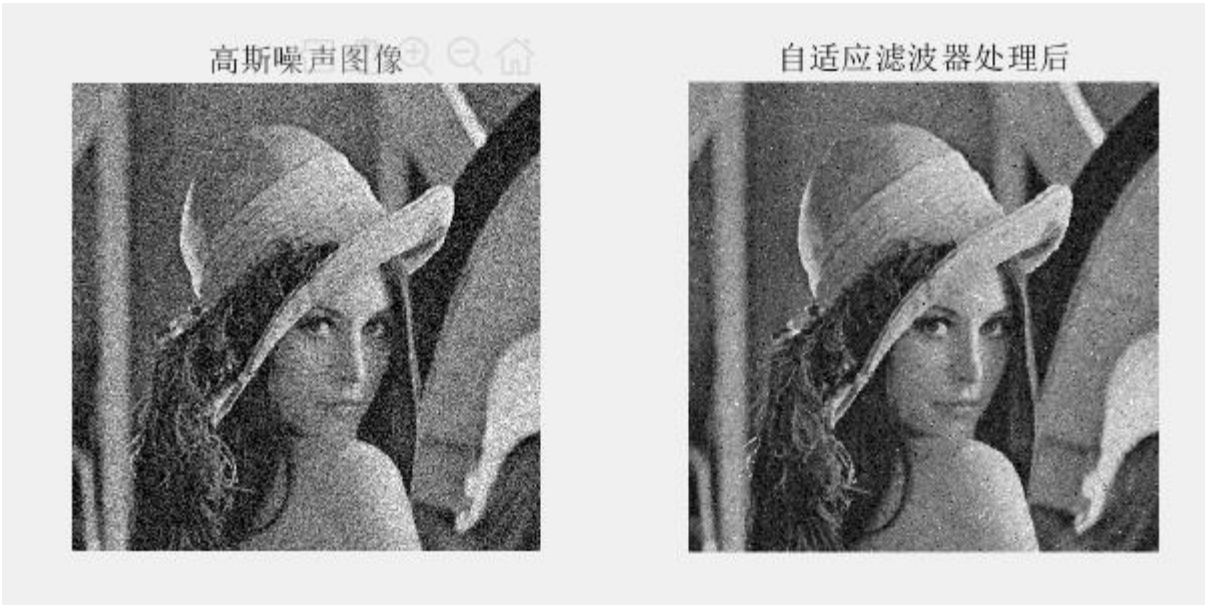
五．自适应滤波器

5.1自适应均值滤波器(SelfAdaptingFilter)

代码如下：

```
function [] = SelfAdaptingFilter()
%高斯噪声为0.01
gau_noise=0.01;
%处理高斯噪声
gau_img=addGaussianNoise();
gau_img=im2double(gau_img);
[gau_h,gau_w,~]=size(gau_img);
new_gau=gau_img;
for i=2:gau_h-1
    for j=2:gau_w-1
        gau_temp=gau_img(i-1:i+1,j-1:j+1);
        temp=gau_temp(:);
        ML=mean(temp);
        gau_l=var(temp);
        gau_org=gau_img(i,j,:);
        new_gau(i,j,:)=gau_org-(gau_noise/gau_l)*(gau_org-ML);
    end
end
figure('name','高斯噪声');
subplot(1,2,1);
imshow(gau_img),title('高斯噪声图像');
subplot(1,2,2);
imshow(new_gau),title('自适应滤波器处理后');
end
```

实验结果：
高斯噪声：



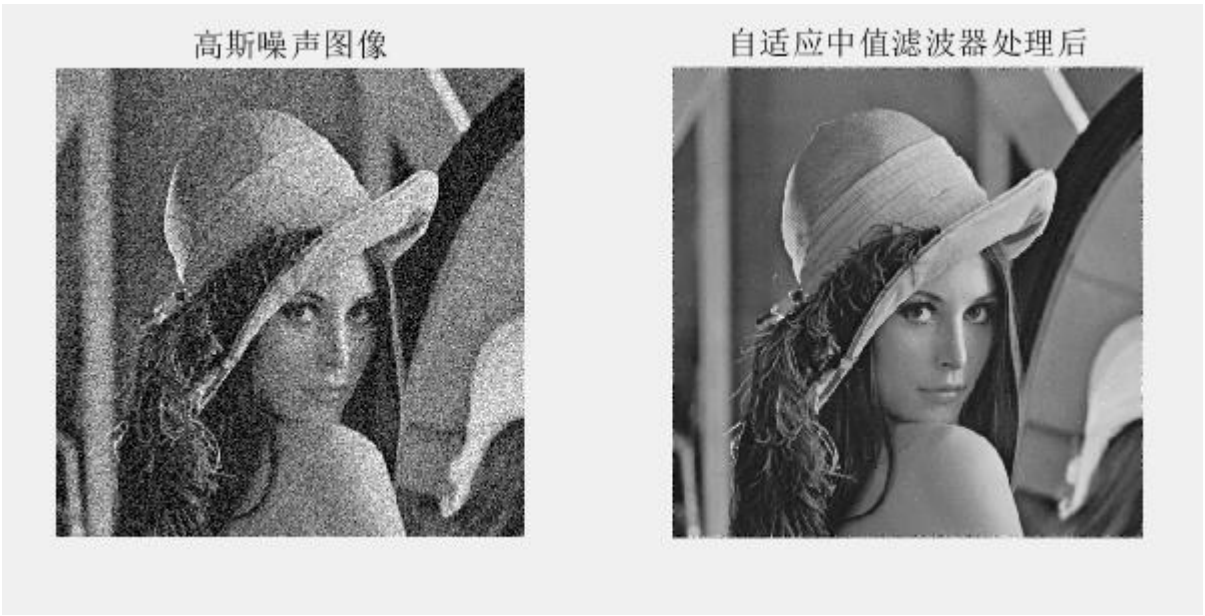
5.2自适应中值滤波器(AdaptiveMedianFilter)

代码如下

```
function [] = AdaptiveMedianFilter()
%设定 Smax=7
Smax=7;
%处理椒盐噪声
salt_img=addSaltNoise();
salt_img=im2double(salt_img);
[sa_h,sa_w,~]=size(salt_img);
new_salt=salt_img;
for i=4:sa_h-3
    for j=4:sa_w-3
        r=1;
        sa_Zxy=salt_img(i,j,:);
        while(r<=Smax)
            sa_temp=salt_img(i-r:i+r,j-r:j+r);
            sa_Zmin=min(sa_temp(:));
            sa_Zmax=max(sa_temp(:));
            sa_Zmed=median(sa_temp(:));
            if(sa_Zmed>sa_Zmin&&sa_Zmed<sa_Zmax)
                break;
            else
                r=r+1;
            end
        end
```

```
        if(sa_Zxy>sa_Zmin&&sa_Zxy<sa_Zmax)
            new_salt(i,j,:)=sa_Zxy;
        else
            new_salt(i,j,:)=sa_Zmed;
        end
    end
end
figure('name','椒盐噪声');
subplot(1,2,1);
imshow(salt_img),title('椒盐噪声图像');
subplot(1,2,2);
imshow(new_salt),title('自适应中值滤波器处理后');
```

实验结果:
高斯噪声:



椒盐噪声:

六.结论

对于图像修复,大部分滤波器针对某种噪声效果好,例如中点滤波器对高斯噪声效果好,对椒盐噪声效果一般.但自适应中值滤波器针对两种噪声的修复效果均较为明显,在自适应滤波器修复效果之下的则是中值滤波器.