图像处理第二次作业

姓名：万焱 学号：3017218106 班级： 2班

算术均值滤波器:

算术均值滤波器很简单，就是求某个区域内的所有像素的算术均值，可用下式子表示。

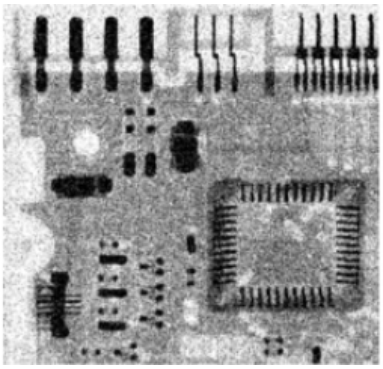
$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t)$$

这就是一个低通滤波器，会使得画面模糊，有些许去噪能力

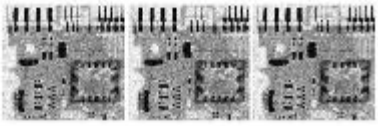
```
function [] = Arithemticmeanfilter()
close all;
f= imread ('test.png');
[w,~]=size(f);
image= f(:,:);
fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        resultImage(x,y)=sum(is(:))/numel(is);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('3*3算术均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
```

```
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        resultImage(x,y)=sum(is(:))/numel(is);

    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5算术均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        resultImage(x,y)=sum(is(:))/numel(is);

    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9算术均值');
```

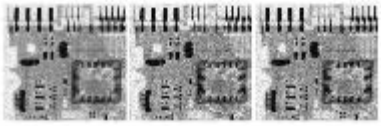
样例原图:



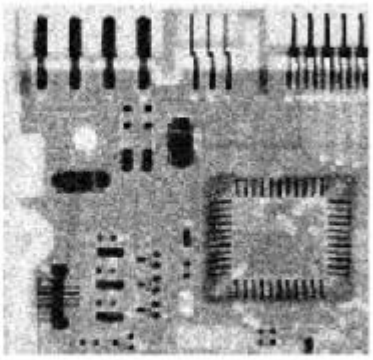
结果生成图:



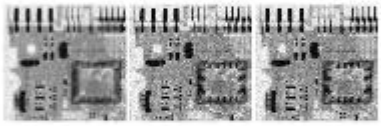
原图像



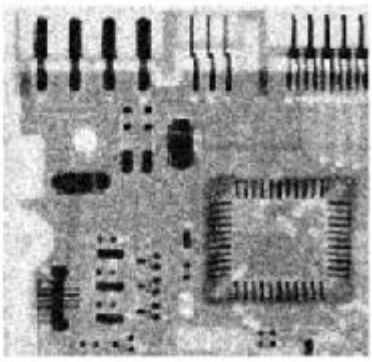
3*3算术均值



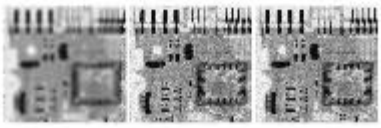
原图像



5*5算术均值



原图像



9*9算术均值

几何均值滤波器

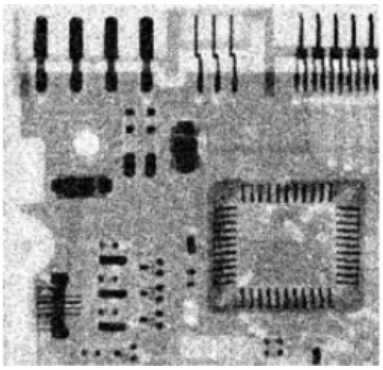
几何均值滤波器，求某个区域内的几何平均值。这个滤波器产生的去噪效果可以与算术平均值滤波器基本一样，但是可以更少的丢失细节。

实验代码:

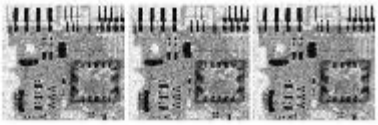
```
function [] = Arithemticmeanfilter()
close all;
f= imread ('test.png');
[w,~]=size(f);
image= f(:,:);
fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        resultImage(x,y)=sum(is(:))/numel(is);
    end
end
```

```
subplot(1,2,2);
imshow(resultImage);
xlabel('3*3算术均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        resultImage(x,y)=sum(is(:))/numel(is);

    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5算术均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        resultImage(x,y)=sum(is(:))/numel(is);

    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9算术均值');
```

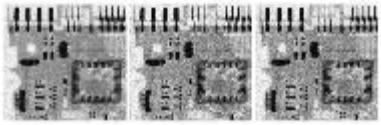
实验原图：



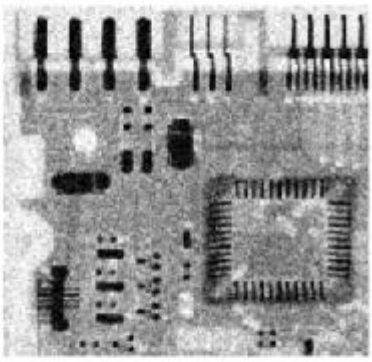
实验结果图：



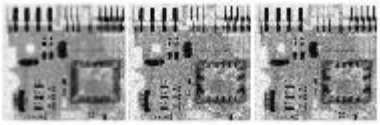
原图像



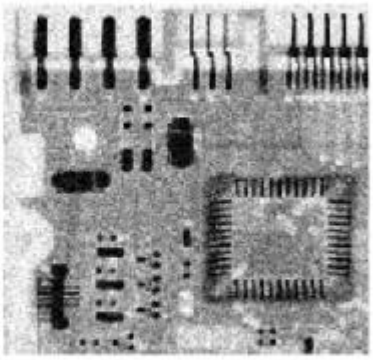
3*3几何均值



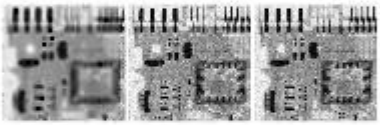
原图像



5*5几何均值



原图像



9*9几何均值

谐波均值滤波器

其表达式如下所示，。 $$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

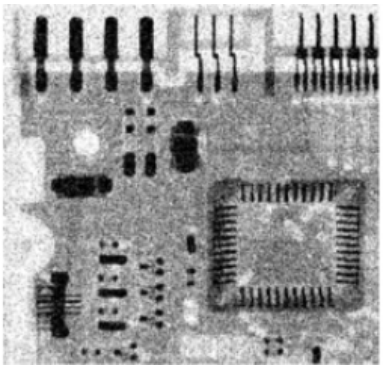
注意式子的分母，这个滤波器不但不能去除椒盐噪声，对于灰度过黑的图像而言，也是要坏事的。这个滤波器，擅长于高斯噪声的去噪。

实验代码：

```
function [] = Harmonicmeanfilter()
close all;

f=imread('test.png');

[w,~]=size(f);
image= f(:,:);
fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        is=1./is;
        resultImage(x,y)=numel(is)/sum(is(:));

    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('3*3谐波均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        %遍历每个点的四周
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        is=1./is;
        resultImage(x,y)=numel(is)/sum(is(:));
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5谐波均值');

resultImage= f(:,:);
```

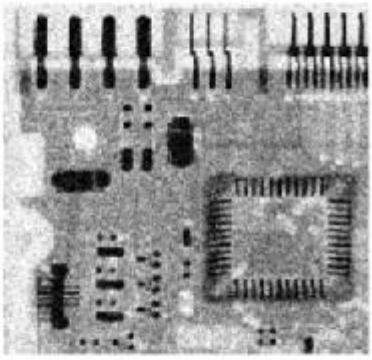
```
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        %遍历每个点的四周
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        is=1./is;
        resultImage(x,y)=numel(is)/sum(is(:));
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9谐波均值');
```

实验原图：

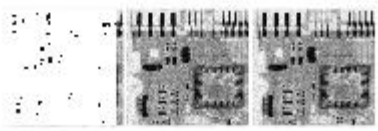


实验结果图：

原图像　　3*3谐波均值



原图像



5*5谐波均值

逆谐波均值滤波器

逆谐波滤波器可以对应多种噪声，式子如下。

$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

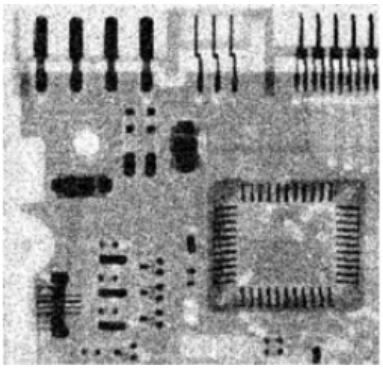
这个滤波器，可以通过Q值的变化，来获得一定的效果。当Q为正，这个滤波器可以去除胡椒噪声，当Q为负，这个滤波器可以去除盐粒噪声

实验代码：

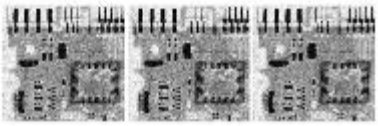
```
function [] = Inverseharmonicmeanfilter()
close all
clear all

f=imread('test.png');

[w,~]=size(f);
image= f(:,:);
Q1 = 2;
fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
```

```
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        resultImage(x,y) = sum(is(:).^(Q1+1))/sum(is(:).^(Q1));
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('3*3逆谐波均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        %遍历每个点的四周
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        resultImage(x,y) = sum(is(:).^(Q1+1))/sum(is(:).^(Q1));

    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5逆谐波均值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        %遍历每个点的四周
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        resultImage(x,y) = sum(is(:).^(Q1+1))/sum(is(:).^(Q1));
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9逆谐波均值');
```

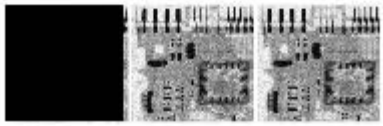
实验原图：



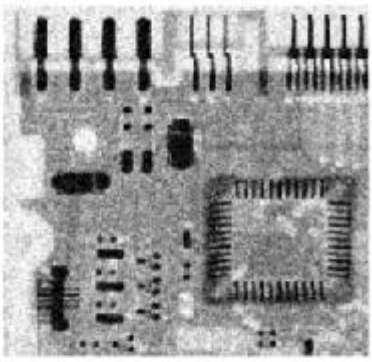
实验结果图：



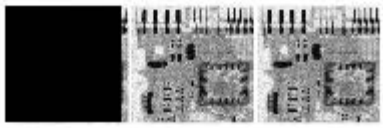
原图像



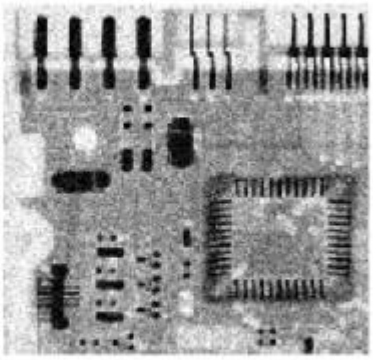
3*3逆谐波均值



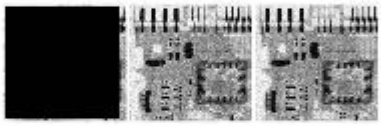
原图像



5*5逆谐波均值



原图像



9*9逆谐波均值

中值滤波器

中值滤波采用非线性的方法，它在平滑脉冲噪声方面非常有效,同时它可以保护图像尖锐的边缘，选择适当的点来替代污染点的值，所以处理效果好，对椒盐噪声表现较好，对高斯噪声表现较差。

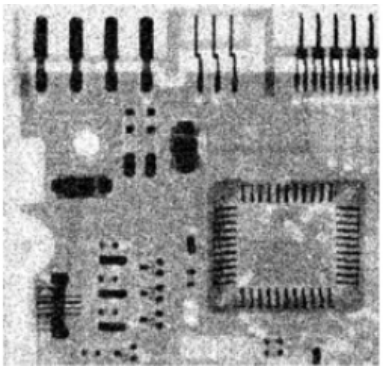
实现代码：

```
function [] = medianfilter()
close all
```

```
clear all

f=imread('test.png');

[w,~]=size(f);
image= f(:,:);

fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        temp = sort(is(:));
        resultImage(x,y)= temp((numel(temp)-1)/2);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('3*3中值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        %遍历每个点的四周
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        temp = sort(is(:));
        resultImage(x,y)= temp((numel(temp)-1)/2);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5中值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
```

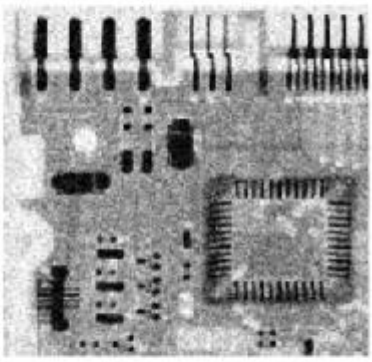
```
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        %遍历每个点的四周
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        temp = sort(is(:));
        resultImage(x,y)= temp((numel(temp)-1)/2);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9中值');
```

实验原图：

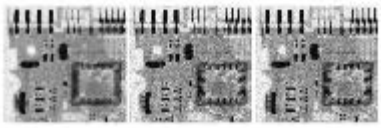


实验结果图：

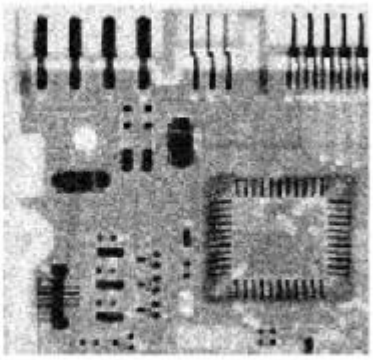
原图像　　3*3中值



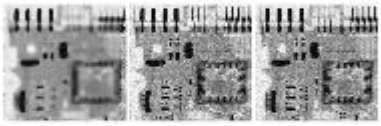
原图像



5*5中值



原图像



9*9中值

最大值和最小值滤波器

最大值滤波，即以模板内进行有序排列后最大像素值代替中心像素值，可以去除图像中的暗斑，使亮斑增大。

最小值滤波，即以模板内进行有序排列后最小像素值代替中心像素值，可以去除图像中的亮斑，使暗斑增大。

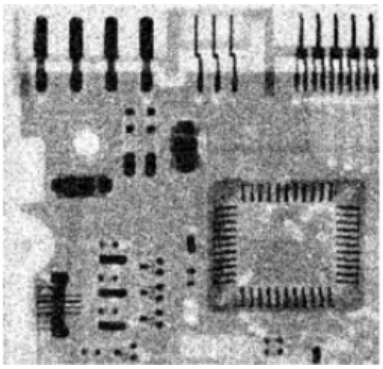
实验代码：

MAX:

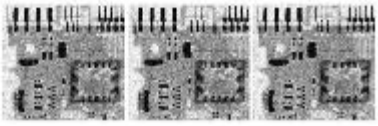
```
function [] = Maxfilter()
close all
clear all

f=imread('test.png');

[w,~]=size(f);
image= f(:,:);

fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        temp = is(:);
        resultImage(x,y)= max(temp);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('3*3最大值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        %遍历每个点的四周
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        temp = is(:);
        resultImage(x,y)= max(temp);
    end
end


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

```
xlabel('5*5最大值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        %遍历每个点的四周
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        temp = is(:);
        resultImage(x,y)= max(temp);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9最大值');
```

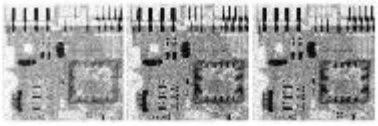
实验原图:



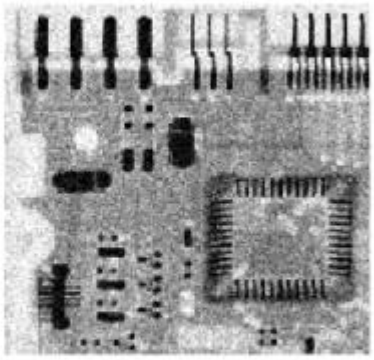
实验结果图:



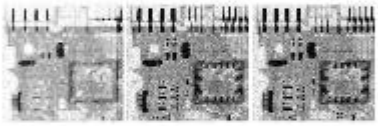
原图像



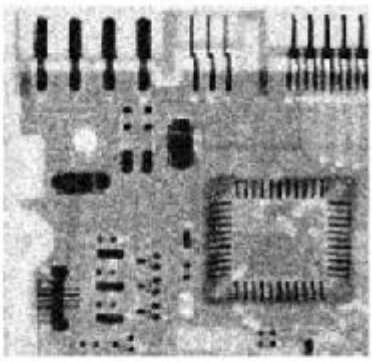
3*3最大值



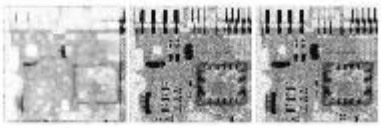
原图像



5*5最大值



原图像



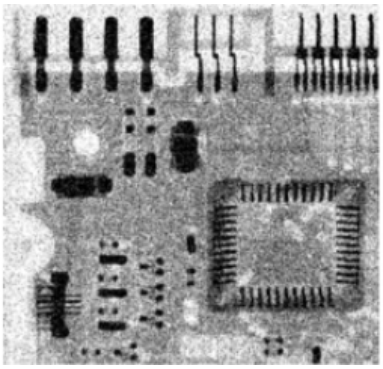
9*9最大值

MIN：

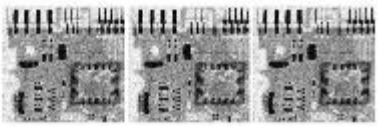
```
function [] = minfilter()
close all
clear all

f=imread('test.png');

[w,~]=size(f);
image= f(:,:);

fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        temp = is(:);
        resultImage(x,y)= min(temp);
    end
end
```

```
subplot(1,2,2);
imshow(resultImage);
xlabel('3*3最小值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        %遍历每个点的四周
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        temp = is(:);
        resultImage(x,y)= min(temp);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5最小值');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        %遍历每个点的四周
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        temp = is(:);
        resultImage(x,y)= min(temp);
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9最小值');
```

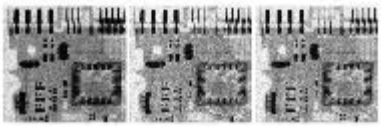
实验原图：



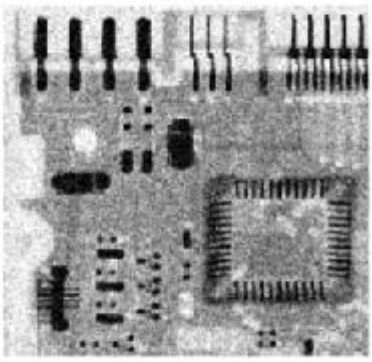
实验结果图：



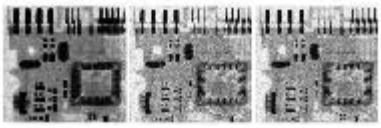
原图像



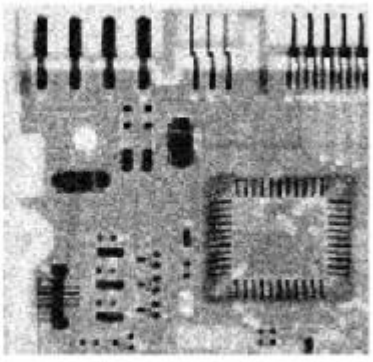
3*3最小值



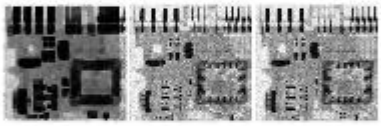
原图像



5*5最小值



原图像



9*9最小值

中点滤波器

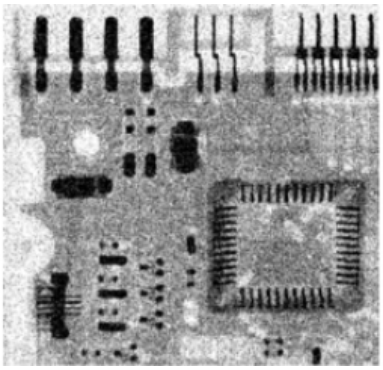
实验代码：

```
function [] = middlefilter()
close all
clear all

f=imread('test.png');
```

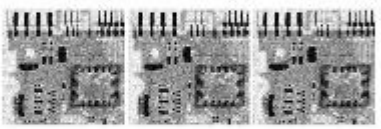
```
[w,~]=size(f);
image= f(:,:);

fsize1=3;
fsize2=5;
fsize3=9;

fssize1=(fsize1-1)/2;
fssize2=(fsize2-1)/2;
fssize3=(fsize3-1)/2;

figure();
subplot(1,2,1);
imshow(image);
xlabel('原图像');

resultImage = image;

for x=1+fssize1:1:w-fssize1
    for y=1+fssize1:1:w-fssize1
        is=f(x-fssize1:1:x+fssize1,y-fssize1:1:y+fssize1);
        temp = is(:);
        resultImage(x,y)= (max(temp) + min(temp))/2;
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('3*3中点');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');

for x=1+fssize2:1:w-fssize2
    for y=1+fssize2:1:w-fssize2
        %遍历每个点的四周
        is=f(x-fssize2:1:x+fssize2,y-fssize2:1:y+fssize2);
        temp = is(:);
        resultImage(x,y)= (max(temp) + min(temp))/2;
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('5*5中点');

resultImage= f(:,:);
figure();
subplot(1,2,1);
imshow(f);
xlabel('原图像');
```

```
for x=1+fssize3:1:w-fssize3
    for y=1+fssize3:1:w-fssize3
        %遍历每个点的四周
        is=f(x-fssize3:1:x+fssize3,y-fssize3:1:y+fssize3);
        temp = is(:);
        resultImage(x,y)= (max(temp) + min(temp))/2;
    end
end


subplot(1,2,2);
imshow(resultImage);
xlabel('9*9中点');
```

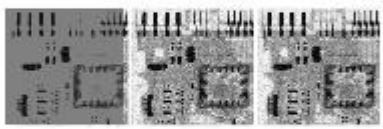
实验原图：



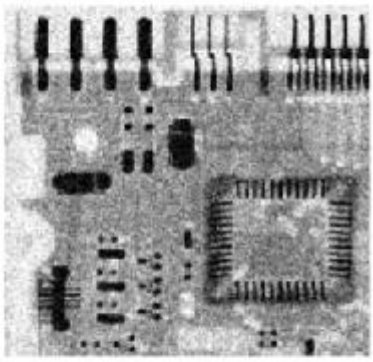
实验结果图：



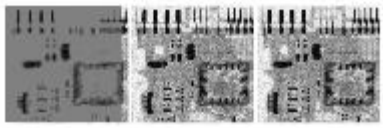
原图像



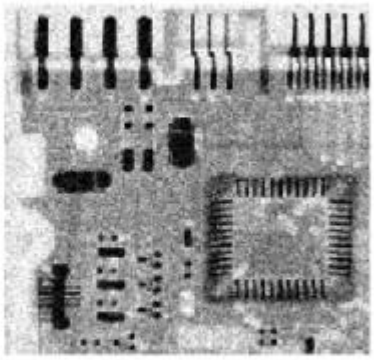
3*3中点



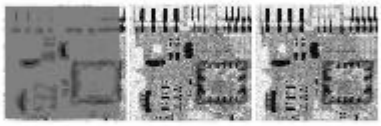
原图像



5*5中点



原图像



9*9中点

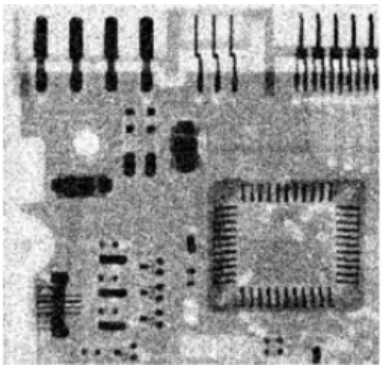
修正后的阿尔法均值滤波器

实验代码：

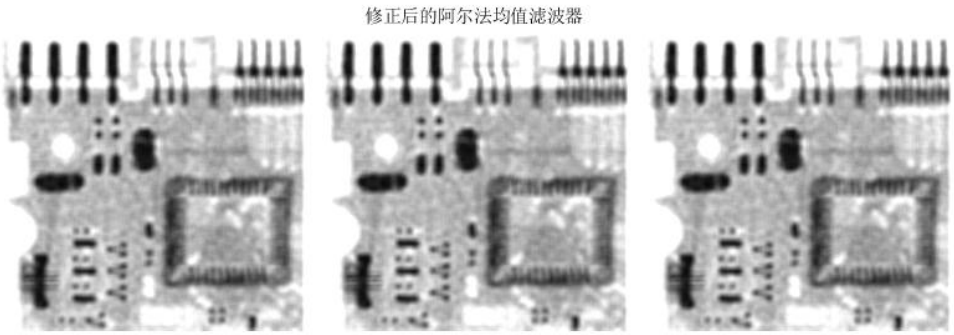
```
function [] = Spatialfilter()
clc;
clear;
%读入图像，并转换为double型
I=imread('test.png');
```

```
I_D=im2double(I);
[MM,NN]=size(I_D);

%%%%%----------------------2、顺序统计滤波器-------------------------------
%%======================= 2.3 修正后的阿尔法均值滤波器====================
%定义子窗口的尺寸
m=5;
n=5;
%确定要扩展的行列数
len_m=floor(m/2);
len_n=floor(n/2);
%将原始图像进行扩展，这里采用了镜像扩展，以进行图像边缘计算
I_D_pad=padarray(I_D,[len_m,len_n],'symmetric');
%获得扩展后的图像尺寸
[M,N]=size(I_D_pad);
d=5;
J_Alpha=zeros(MM,NN);
%逐点计算子窗口的谐波平均
for i=1+len_m:M-len_m
    for j=1+len_n:N-len_n
        %从扩展图像中取出子图像
        Block=I_D_pad(i-len_m:i+len_m,j-len_n:j+len_n);
        %计算矩阵的阿尔法均值
        J_Alpha(i-len_m,j-len_n)=sum(sum(Block))/(m*n-d);
    end
end
figure;
imshow(J_Alpha);
title('修正后的阿尔法均值滤波器');
```

实验原图：



实验结果图：



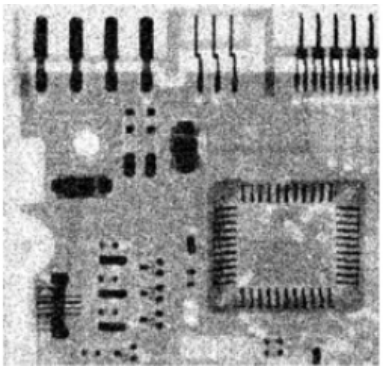
自适应滤波器

使用LMS算法对进行信号分离

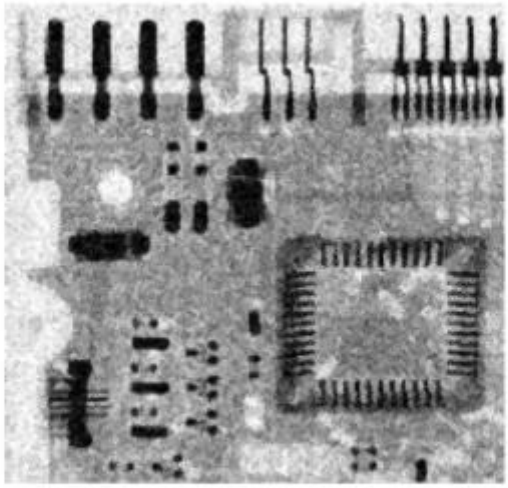
实验代码：

```
function [] = Adaptivefilter()
clc;
clear;
f=imread('test.png');
image_gray=rgb2gray(f);%得到灰度图像
ff =image_gray;
ff(:) = 0;
alreadyProcessed = false(size(image_gray));%生成逻辑非的矩阵
% 迭代.
Smax=7;
for k = 3:2:Smax
   zmin = ordfilt2(image_gray, 1, ones(k, k), 'symmetric');
   zmax = ordfilt2(image_gray, k * k, ones(k, k), 'symmetric');
   zmed = medfilt2(image_gray, [k k], 'symmetric');

   processUsingLevelB = (zmed > zmin) & (zmax > zmed) & ...
       ~alreadyProcessed;
   zB = (image_gray > zmin) & (zmax > image_gray);
   outputZxy  = processUsingLevelB & zB;
   outputZmed = processUsingLevelB & ~zB;
   ff(outputZxy) = image_gray(outputZxy);
   ff(outputZmed) = zmed(outputZmed);

   alreadyProcessed = alreadyProcessed | processUsingLevelB;
   if all(alreadyProcessed(:))
      break;
   end
end
ff(~alreadyProcessed) = zmed(~alreadyProcessed);
f1=imnoise(image_gray,'salt & pepper',0.3);%添加椒盐噪声后的图像

figure;
imshow(ff);
figure;
imshow(f1);
```

实验原图：



实验结果图：



自适应中值滤波器

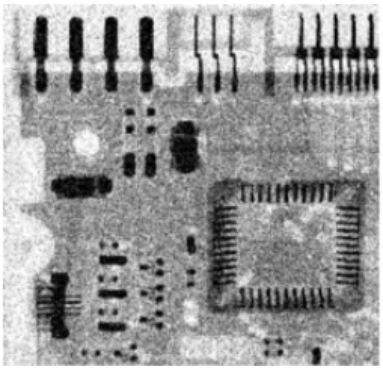
自适应中值滤波器是以m*n的矩形窗口Sxy定义的滤波器区域内图像的统计特性为基础的，可以处理具有更大概率的脉冲噪声如椒盐噪声。

实验代码：

```
function [] =adaptivemedianfilter()
clc;
clear;
f=imread('test.png');
image_gray=rgb2gray(f);%得到灰度图像
ff =image_gray;
ff(:) = 0;
alreadyProcessed = false(size(image_gray));%生成逻辑非的矩阵
% 迭代.
Smax=7;
for k = 3:2:Smax
   zmin = ordfilt2(image_gray, 1, ones(k, k), 'symmetric');
   zmax = ordfilt2(image_gray, k * k, ones(k, k), 'symmetric');
   zmed = medfilt2(image_gray, [k k], 'symmetric');

   processUsingLevelB = (zmed > zmin) & (zmax > zmed) & ...
       ~alreadyProcessed;
   zB = (image_gray > zmin) & (zmax > image_gray);
   outputZxy  = processUsingLevelB & zB;
   outputZmed = processUsingLevelB & ~zB;
   ff(outputZxy) = image_gray(outputZxy);
   ff(outputZmed) = zmed(outputZmed);

   alreadyProcessed = alreadyProcessed | processUsingLevelB;
   if all(alreadyProcessed(:))
      break;
   end
end
ff(~alreadyProcessed) = zmed(~alreadyProcessed);
f1=imnoise(image_gray,'salt & pepper',0.3);%添加椒盐噪声后的图像
f2=medfilt2(f1,[3,3]);%中值滤波后的图像
```

```
subplot(2,2,1);
figure;
imshow(image_gray);
title('原图');
subplot(2,2,2);
imshow(f1);
title('椒盐噪声污染后的图像');
subplot(2,2,3);
imshow(f2);
title('中值滤波');
subplot(2,2,4);
imshow(ff);
title('自适应中值滤波');
```

实验原图：



在Sxy定义的滤波器区域内定义如下变量：

Zmin=Sxy中的最小灰度值

Zmax=Sxy中的最大灰度值

Zmed=Sxy中的灰度值的中值

Zxy=坐标（x，y）处的灰度值

实验结果图：