作业七：

计算作业

5.1
5.22

实验作业：

1 、用 matlab（或 C 语言）实现自适应均值滤波器，并和算数均值滤波器的结果做对比。
2 、用 matlab（或 C 语言）实现自适应中值滤波器，并和中值滤波器的结果做对比。
3 、用式（5.6-3）对图像进行模糊（空气湍流模型）

$$\mathrm{H(u,v)} = e^{-k(u^2+v^2)^{5/6}}$$

处理，然后加白高斯噪声，得到降质图像。用逆滤波和维纳滤波恢复图像，对结果进行分析

代码：

1.习题 5.1&5.2

```
%%%%%%%%%% 习题 5.1

clear;
clc;

I=zeros(244,233);
for i=18:226
    for j=1:233
        for k=1:9
            if 24*k-7<=j&&j<=24*k
                I(i,j)=255;
            end
        end
    end
end
figure
subplot(2,2,1),imshow(I),title('原始图像');
K1=filter2(fspecial('average',3),I)/255;
K2=filter2(fspecial('average',7),I)/255;
```

```
K3=filter2(fspecial('average' ,9),I)/255;
subplot(2,2,2),imshow(K1 ),title('3*3 算术均值滤波图像');
subplot(2,2,3) ,imshow(K2),title('7*7 算术均值滤波图像');
subplot(2,2,4),imshow(K3),title('9*9 算术均值滤波图像');

%%%%%%%%% 习题 5.2

I=zeros(244,233);
for i=18:227  % 白道道行 7px
    for j=1:233
        for k=1:9
            if 24*k-7<=j&&j<=24*k
                I(i,j)=255;
            end
        end
    end
end
figure;
subplot(221),imshow(I),title('原始图像');
F=I;G=I;H=I;M=I;
for i=2:243
    for j=2:232
        P=F(i-1:i+1,j-1:j+1);
        X=prod(prod(P),2);
        G(i,j)=X^(1/9);
    end
end
subplot(222),imshow(G),title('3*3 几何平均滤波图')
for i=4:241
    for j=4:230
        P=F(i-3:i+3,j-3:j+3);
        X=prod(prod(P),2);
        H(i,j)=X^(1/49);
    end
end
subplot(223),imshow(H),title('7*7 几何平均滤波图')
for i=5:240
    for j=5:229
        P=F(i-4:i+4j-4j+4);
        X=prod(prod(P),2);
        M(i,j)=X^(1/81);
    end
end
subplot(224),imshow(M),title('9*9 几何平均滤波图')
```

2.3 个均值滤波器

```
%%%%%%%%%% 1.算术均值滤波  &  几何均值滤波器  &  自适应均值滤波

img=imread('ckt-board-orig.tif');
figure;
subplot(231);imshow(img);xlabel('a.原图');
% 加噪密度 0.1 高斯白噪声
imgNoise=imnoise(img,'gaussian',0,0.01);
subplot(232);imshow(imgNoise);xlabel('b.加噪');
[m,n]=size(imgNoise);
% 选择 mask 大小，mask 为可调奇数 3、5、7、9 ……
mask=5;
imgNoise=double(imgNoise);
% varNoise=var(imgNoise(:)); % 噪声方差方法 1
% varNoise=(std2(imgNoise))^2; % 噪声方差方法 2
% 以上两种方法试验后发现不对，应该如下。
varNoise=0.01
% 进行拓展
imgEx=[255*ones((mask-1)/2,n+(mask-1));255*ones(m,(mask-
1)/2),imgNoise,255*ones(m,(mask-1)/2);255*ones((mask-1)/2,n+(mask-
1))];
temp=zeros(mask,mask);
arithMean=zeros(m,n);
geomeMean=zeros(m,n);
adaptMean=zeros(m,n);

for x=1+(mask-1)/2:m+(mask-1)/2
    for y=1+(mask-1)/2:n+(mask-1)/2
        temp=imgEx(x-(mask-1)/2:x+(mask-1)/2,y-(mask-1)/2:y+(mask-
1)/2); % mask 方块
        varLocal=var(temp(:)); % 局部方差方法 1
        % varLocal=(std2(temp))^2; % 局部方差方法 2
        arithMean(x-(mask-1)/2,y-(mask-1)/2)=mean(mean(temp)); % 算数均
值
        geomeMean(x-(mask-1)/2,y-(mask-
1)/2)=(prod(prod(temp),2))^(1/(mask*mask)); % 几何均值
        adaptMean(x-(mask-1)/2,y-(mask-1)/2)=imgEx(x,y)-
varLocal/varNoise*(imgEx(x,y)-arithMean(x-(mask-1)/2,y-(mask-
1)/2)); % 自适应均值
    end
end
```

```
subplot(234);imshow(uint8(arithMean));xlabel('c.算术平均');
subplot(235);imshow(uint8(geomeMean));xlabel('d.几何平均');
subplot(236);imshow(uint8(adaptMean));xlabel('e.自适应平均');
```

3.2 个中值滤波器

```
%%%%%%%%%% 2.自适应中值滤波  &  中值滤波

im=imread('ckt-board-orig.tif');
figure;
subplot(221);imshow(im);xlabel('a.原图');
% 加噪密度 0.4 椒盐噪声
iNoise=imnoise(im,'salt & pepper',0.4);
subplot(222);imshow(iNoise);xlabel('b.加噪');
% 两次普通中值滤波
imMed=medfilt2(iNoise,[3 3]);
imMed=medfilt2(imMed,[3 3]);
subplot(223);imshow(imMed);xlabel('c.2 次中值滤波');
% 最大尺寸 9*9 的自适应中值滤波器
[w,h]=size(iNoise);
nmin=3;nmax=9; % 起始窗 nmin*nmin,最大窗 nmax*nmax
imAdaptMed=iNoise; % 定义复原后图像
% 将 iNoise 扩展
imEx=[zeros((nmax-1)/2,h+(nmax-1));zeros(w,(nmax-
1)/2),iNoise,zeros(w,(nmax-1)/2);zeros((nmax-1)/2,h+(nmax-1))];
for x=1:w
    for y=1:h
        for n=nmin:2:nmax
            % iNoise 中某点(x,y)的邻域
            Sxy=imEx(x+(nmax-1)/2-(n-1)/2:x+(nmax-1)/2+(n-1)/2,y+(nmax-
1)/2-(n-1)/2:y+(nmax-1)/2+(n-1)/2);
            Smax=max(max(Sxy));
            Smin=min(min(Sxy));
            Smed=median(median(Sxy));
            if Smed>Smin && Smed<Smax
                if iNoise(x,y)<=Smin || iNoise(x,y)>=Smax
                    imAdaptMed(x,y)=Smed;
                end
                break  % 有输出则不再循环判断
            end
        end
        imAdaptMed(x,y)=Smed; % 达到最大窗口都没满足上述条件姑且取最大窗中
值
        % imAdaptMed(x,y)=iNoise(x,y); % 保持原值误差很大舍弃
```

```
    end
end
subplot(224);imshow(imAdaptMed);xlabel('d.自适应中值滤波');
```

4.模糊加高斯噪声逆滤波与维纳滤波（手打函数版）

```
%%%%%%%%%% 3.对图像进行模糊 H(u,v)=exp(-k*(u^2+v^2)^(5/6))处理，然后加白高斯噪声，得到降质图像。用逆滤波和维纳滤波恢复图像
img=imread('original_DIP.tif');
img=imread('aerial_view_no_turb.tif');
img=imread('lena256.bmp');
figure;
subplot(231);imshow(img);xlabel('a.原图');
f=double(img);
F=fftshift(fft2(img)); % 原图傅里叶
subplot(234);imshow(abs(F),[]);xlabel('b.傅里叶变换所得频谱');
[m,n]=size(F);
H=zeros(m,n);  % 传递函数
B=zeros(m,n);  % 模糊傅里叶
F1=zeros(m,n); % 逆滤波
F2=zeros(m,n); % 维纳滤波
a=0.1;b=0.1;T=1;K=0.00259;
for u=1:m
    for v=1:n
        H(u,v)=(T/(pi*(a*u+b*v)))*sin(pi*(a*u+b*v))*exp(-1i*pi*(a*u+b*v));
        B(u,v)=H(u,v)*F(u,v);
    end
end
blur=ifft2(ifftshift(B)); % 模糊
blur=256*blur/max(max(blur));
blur=uint8(real(blur));
subplot(232);imshow(blur);xlabel('c.傅里叶反变换所得模糊图像');


% 模糊加高斯白噪声
g=imnoise(blur,'gaussian',0,0.01);
subplot(235);imshow(abs(g));xlabel('d.模糊加噪图像');
g=double(g);
G=fftshift(fft2(g)); % 模糊加高斯白噪声傅里叶

for u=1:m
    for v=1:n
        F1(u,v)=1/H(u,v)*G(u,v);
        F2(u,v)=1/H(u,v)*(abs(H(u,v)))^2/((abs(H(u,v)))^2+K)*G(u,v);
```

```
    end
end

f1=ifft2(ifftshift(F1));
f1=256.*f1./max(max(f1));
f1=uint8(real(f1));
subplot(233);imshow(f1);xlabel('e.逆滤波图像');

f2=ifft2(ifftshift(F2));
f2=256.*f2./max(max(f2));
f2=uint8(real(f2));
subplot(236);imshow(f2);xlabel('f.维纳滤波图像');
```

5.模糊加高斯噪声逆滤波与维纳滤波（运用 matlab 自带函数）

```
% 读取原始图像
img = imread('original_DIP.tif');
img = im2double(img);
figure;
subplot(231);
imshow(img);
title('Original image');

% % 函数说明
% ①FSPECIAL('motion',LEN,THETA)为运动模糊算子，表示摄像物体逆时针方向以
theta 角度运动了 len 个像素，len 的默认值为 9，theta 的默认值为 0；
% ②imfilter(f, w, filtering_mode, boundary_options, size_options)，f 为输入
图像，w 为滤波掩模，
% filtering_mode 用于指定在滤波过程中是使用"相关"'corr'还是"卷积"'conv'，
% boundary_options 用于处理边界充零问题，边界的大小由滤波器的大小确定
('circular'图像大小通过将图像看成是一个二维周期函数的一个周期来扩展)。

% 模糊图像
PSF = fspecial('motion', 50, -45);
img1 = imfilter(img, PSF, 'conv', 'circular');
subplot(232);
imshow(img1);
title('Blurred image');
```

```
% 添加高斯噪声 noise_var 可变，按照书上分别设为 650,65,0.0065
noise_var = 650;
img2 = imnoise(img1, 'gaussian', 0, noise_var);
subplot(233);
imshow(img2);
title(['add Gaussian noise with variance is ',num2str(noise_var)]);

% % 函数说明
% deconvwnr(I, PSF, NSR)其中，I 值退化的图像，是原图像卷积一个点扩散函数 PSF
然后加上加性噪声而得到的;
% NSR 噪信比，可以为一标量，也可以为与 I 同样大小的矩阵，默认值为 0。

% 逆滤波，NSR 设为 0(噪信比为 0，参数维纳滤波退化为理想逆滤波)
% Specifying 0 for the NSR is equivalent to creating an ideal inverse
filter.
img3 = deconvwnr(img2, PSF, 0.0);
subplot(223);
imshow(img3);
title('Result of invense filtering');

% 参数维纳滤波，计算噪信比带入
estimated_NSR = noise_var / var(img(:));
img4 = deconvwnr(img2, PSF, estimated_NSR);
subplot(224);
imshow(img4);
title('Result of Wiener filtering');
```

6.手打函数与 MATLAB 自带函数逆滤波与维纳滤波的对比

```
% 读取原始图像
img=imread('lena256.bmp');
% img = imread('original_DIP.tif');
img = im2double(img);
figure;
subplot(331);
imshow(img);
title('Original image');

% % 函数说明

% ①FSPECIAL('motion',LEN,THETA)为运动模糊算子，表示摄像物体逆时针方向以

theta 角度运动了 len 个像素，len 的默认值为 9，theta 的默认值为 0；
```

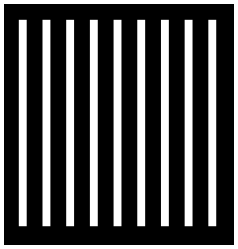
```
% ②imfilter(f, w, filtering_mode, boundary_options, size_options)，f 为输入图像，w 为滤波掩模，
% filtering_mode 用于指定在滤波过程中是使用“相关”‘corr’还是“卷积”‘conv’，
% boundary_options 用于处理边界充零问题，边界的大小由滤波器的大小确定（‘circular’图像大小通过将图像看成是一个二维周期函数的一个周期来扩展）。

% 模糊图像
PSF = fspecial('motion', 30, -45);
img1 = imfilter(img, PSF, 'conv', 'circular');
subplot(332);
imshow(img1);
title('Blurred image');

% 添加高斯噪声
noise_var = 0.000001;
img2 = imnoise(img1, 'gaussian', 0, noise_var);
subplot(333);imshow(img2);title(['add Gaussian noise with variance is ',num2str(noise_var)]);

% % 函数说明
% deconvwnr(I, PSF, NSR)其中，I 值退化的图像，是原图像卷积一个点扩散函数 PSF 然后加上加性噪声而得到的;
% NSR 噪信比，可以为一标量，也可以为与 I 同样大小的矩阵，默认值为 0。

% 逆滤波，NSR 设为 0(噪信比为 0，参数维纳滤波退化为理想逆滤波)
% Specifying 0 for the NSR is equivalent to creating an ideal inverse filter.
img3 = deconvwnr(img2, PSF, 0.0);
subplot(323);imshow(img3);title('Result of invense filtering using function of matlab');

% 参数维纳滤波，计算噪信比带入
estimated_NSR = noise_var / var(img(:));
img4 = deconvwnr(img2, PSF, estimated_NSR);
subplot(324);imshow(img4);title('Result of Wiener filtering using function of matlab');


g=double(img1);
G=fftshift(fft2(g)); % 模糊加高斯白噪声傅里叶
[m,n]=size(G);
H=zeros(m,n);  % 传递函数
F1=zeros(m,n); % 逆滤波
```

```
F2=zeros(m,n); % 维纳滤波
a=0.1;b=0.1;T=1;K=0.06;
for u=1:m
    for v=1:n
        H(u,v)=(T/(pi*(a*u+b*v)))*sin(pi*(a*u+b*v))*exp(-1i*pi*(a*u+b*v));
        F1(u,v)=1/H(u,v)*G(u,v);
        F2(u,v)=1/H(u,v)*(abs(H(u,v)))^2/((abs(H(u,v)))^2+K)*G(u,v);
    end
end

f1=ifft2(ifftshift(F1));
f1=256*f1/max(max(f1));
f1=uint8(real(f1));
subplot(325);imshow(f1);title('Result of invense filtering using function
typed');

f2=ifft2(ifftshift(F2));
f2=256*f2/max(max(f2));
f2=uint8(real(f2));
subplot(326);imshow(f2);title(['Result of Wiener filtering using function
typed K= ',num2str(K)]);
```

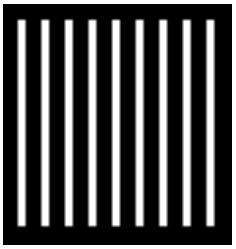
运行结果:

1.5.1&5.2

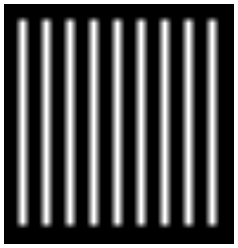
原始图像



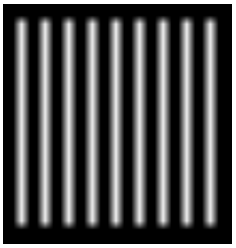
3*3算术均值滤波图像



7*7算术均值滤波图像

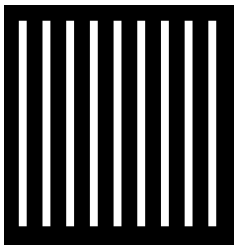


9*9算术均值滤波图像

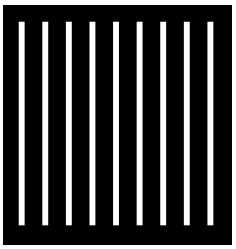


5.1

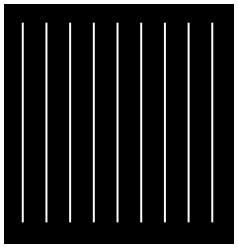
原始图像



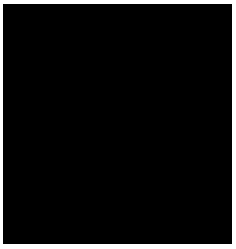
3*3几何平均滤波图



7*7几何平均滤波图

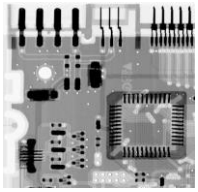


9*9几何平均滤波图

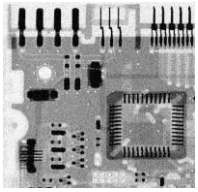


5.2

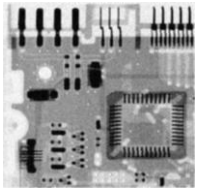
2.3 个均值滤波器



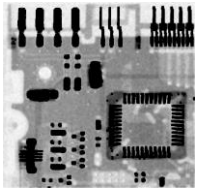
a.原图



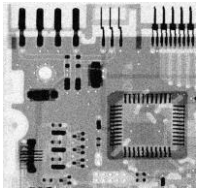
b.加噪



c.算术平均

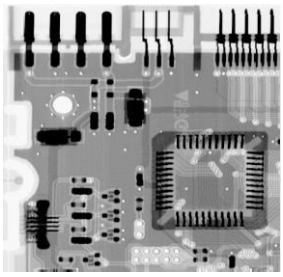


d.几何平均

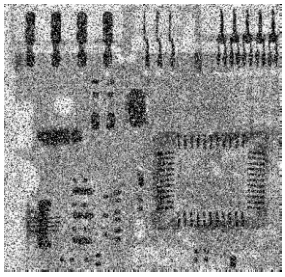


e.自适应平均

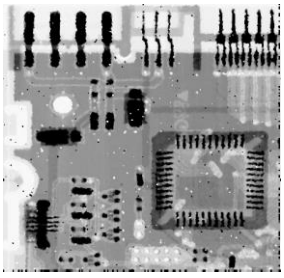
3.2 个中值滤波器



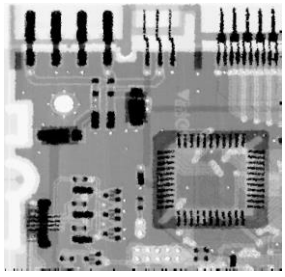
a.原图



b.加噪



c.2次中值滤波



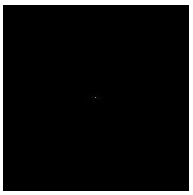
d.自适应中值滤波

4.模糊加高斯噪声逆滤波与维纳滤波（手打函数版）

a.原图

c.傅里叶反变换所得模糊图像

e.逆滤波图像

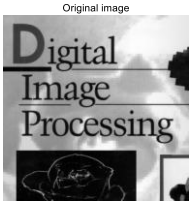


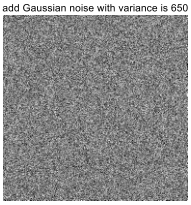
b.傅里叶变换所得频谱

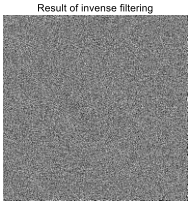
d.模糊加噪图像

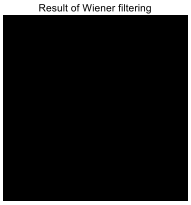
f.维纳滤波图像

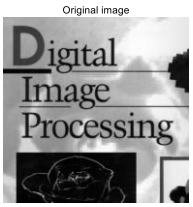
5.模糊加高斯噪声逆滤波与维纳滤波（运用 matlab 自带函数）（variance=650,65,0.0065,6.5e-7）



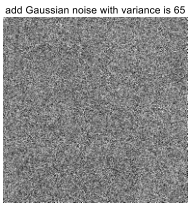


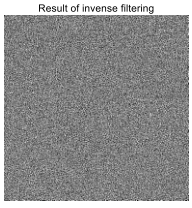


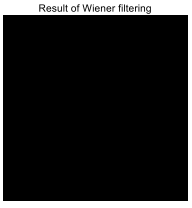


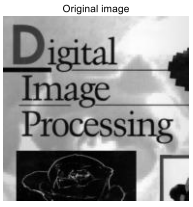

Original image


Blurred image

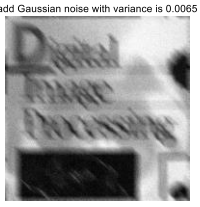

add Gaussian noise with variance is 65

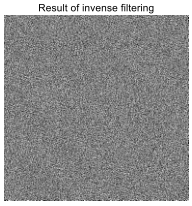

Result of invense filtering

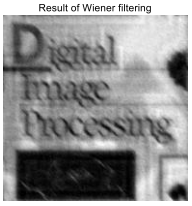

Result of Wiener filtering

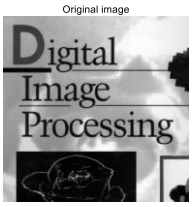

Original image


Blurred image

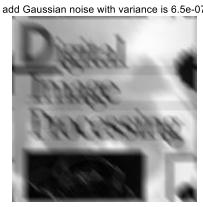

add Gaussian noise with variance is 0.0065

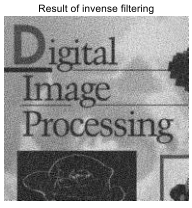

Result of invense filtering

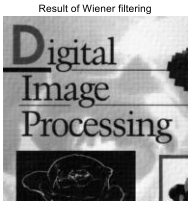

Result of Wiener filtering


Original image

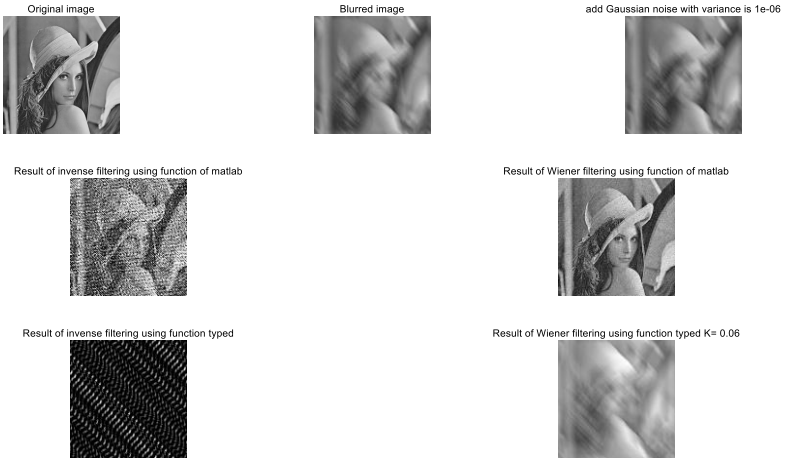

Blurred image


add Gaussian noise with variance is 6.5e-07


Result of invense filtering


Result of Wiener filtering

6.手打函数与 MATLAB 自带函数逆滤波与维纳滤波的对比



结果分析:

5.1 中随着 mask 的增大，边缘逐渐模糊。5.2 中随着 mask 的增大，白道道逐渐变窄直至消失。

3 个均值滤波器：算数均值滤波器 mask 越大越模糊；几何均值滤波器 mask 越大黑块越突出；自适应滤波器可以既解决边沿问题又基本不用滤除平滑块，边缘较好。

2 个中值滤波器中，两次中值滤波效果明显没有自适应中值滤波器效果好。

电脑自带的函数就是比手打的函数感觉更加形象，比如拟合运动模糊时

```
PSF = fspecial('motion', 50, -45);
img1 = imfilter(img, PSF, 'conv', 'circular');
```

比

```
H(u,v)=(T/(pi*(a*u+b*v)))*sin(pi*(a*u+b*v))*exp(-1i*pi*(a*u+b*v));
B(u,v)=H(u,v)*F(u,v);
```

拟合图像好，如下对比:

ab
cd　a、c为MATLAB函数拟合的模糊图像
b、d为手打函数拟合的模糊图像

遇到的问题以及解决办法：

1.

在做几何均值时，遇到了如何解决矩阵的所有元素积的问题，查资料了解到可以用`prod(prod(X),2)`函数。

2.

在做自适应均值滤波器时遇到了局部方差，噪声方差如何对应计算的问题，经过整理思路，看 PPT 和课本，以及上网查函数，找到如下 3 种方式（转自以下网址 https://www.cnblogs.com/denny402/p/4008210.html）

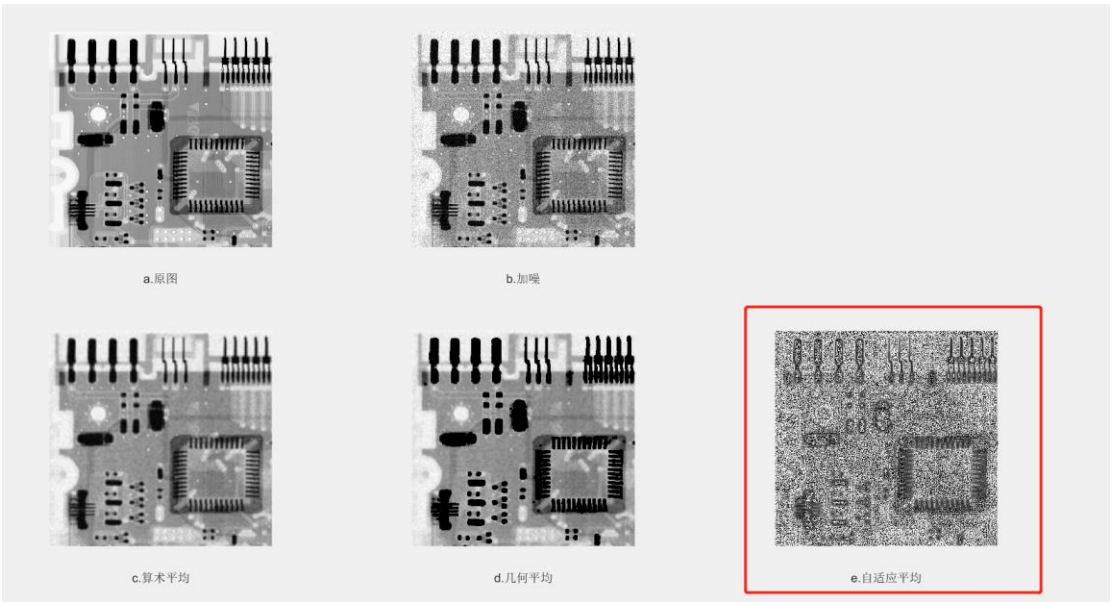
```
% 求一副灰度图像的方差
close all
clear
clc;
i=imread('d:/lena.jpg'); %载入真彩色图像
i=rgb2gray(i); %转换为灰度图
i=double(i);  %将 uint8 型转换为 double 型，否则不能计算统计量
% sq1=var(i,0,1); %列向量方差，第二个参数为 0，表示方差公式分子下面是 n-1，如果为 1 则是 n
% sq2=var(i,0,2); %行向量方差
avg=mean2(i);  %求图像均值
[m,n]=size(i);
s=0;
for x=1:m
    for y=1:n
    s=s+(i(x,y)-avg)^2; %求得所有像素与均值的平方和。
    end
end
%求图像的方差
a1=var(i(:)); %第一种方法：利用函数 var 求得。
a2=s/(m*n-1); %第二种方法：利用方差公式求得
a3=(std2(i))^2; %第三种方法：利用 std2 求得标准差，再平方即为方差。
```

但是使用

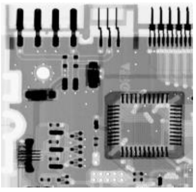
`adaptMean(x-(mask-1)/2,y-(mask-1)/2)=imgEx(x,y)- varNoise / varLocal *(imgEx(x,y)-arithMean(x-(mask-1)/2,y-(mask-1)/2));`

得到图像很奇怪，没有去噪反而加噪了，将 varNoise 和 varLocal 调换次序好像有所改变，但原理上又说不通，估计是赋值时赋得不对，还有待改正。（问题代码、奇怪图像如下，希望老师能够批评指正）

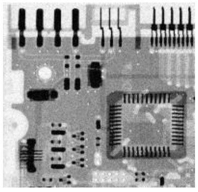
```
clear;
clc;

%%%%%%%%%% 1.算术均值滤波  &  几何均值滤波器  &  自适应均值滤波

img=imread('ckt-board-orig.tif');
figure;
subplot(231);imshow(img);xlabel('a.原图');
% 加噪密度0.1高斯白噪声
imgNoise=imnoise(img,'gaussian',0,0.01);
subplot(232);imshow(imgNoise);xlabel('b.加噪');
[m,n]=size(imgNoise);
% 选择mask大小，mask为可调奇数3、5、7、9 ……
mask=5;
imgNoise=double(imgNoise);
varNoise=var(imgNoise(:)); % 噪声方差方法1
% varNoise=(std2(imgNoise))^2; % 噪声方差方法2
% 进行拓展
imgEx=[255*ones((mask-1)/2,n+(mask-1));255*ones(m,(mask-1)/2),imgNoise,255*ones(m,(mask-1)/2);255*ones((mask-1)/2,n+(mask-1))];
temp=zeros(mask,mask);
arithMean=zeros(m,n);
geomeMean=zeros(m,n);
adaptMean=zeros(m,n);

for x=1+(mask-1)/2:m+(mask-1)/2
    for y=1+(mask-1)/2:n+(mask-1)/2
        temp=imgEx(x-(mask-1)/2:x+(mask-1)/2,y-(mask-1)/2:y+(mask-1)/2); % mask方块
        varLocal=var(temp(:)); % 局部方差方法1
        % varLocal=(std2(temp))^2; % 局部方差方法2
        arithMean(x-(mask-1)/2,y-(mask-1)/2)=mean(mean(temp)); % 算数均值
        geomeMean(x-(mask-1)/2,y-(mask-1)/2)=(prod(prod(temp,2)))^(1/(mask*mask)); % 几何均值
        adaptMean(x-(mask-1)/2,y-(mask-1)/2)=imgEx(x,y)-varNoise/varLocal*(imgEx(x,y)-arithMean(x-(mask-1)/2,y-(mask-1)/2)); % 自适应均值
    end
end
subplot(234);imshow(uint8(arithMean));xlabel('c.算术平均');
subplot(235);imshow(uint8(geomeMean));xlabel('d.几何平均');
subplot(236);imshow(uint8(adaptMean));xlabel('e.自适应平均');
```



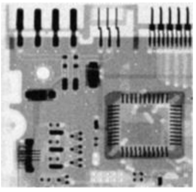
后来过了几天和同学讨论后改正了 varNoise 的算法，觉得不应该是所有点的方差，而应该是最初加噪时自己设置的 0.01，修正后 mask=5、7、9 如下三个图，算是成功解决问题：



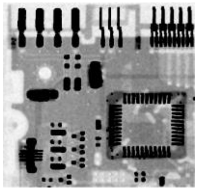
a.原图



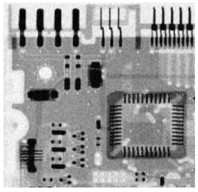
b.加噪



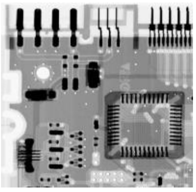
c.算术平均



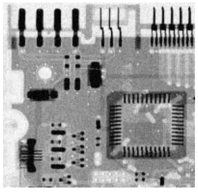
d.几何平均



e.自适应平均

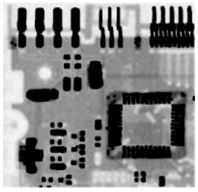


a.原图

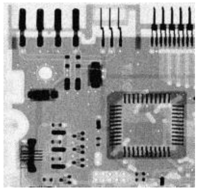


b.加噪

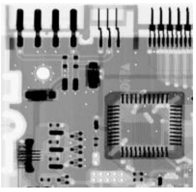
c.算术平均



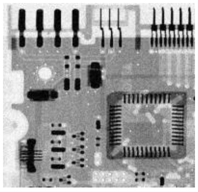
d.几何平均



e.自适应平均

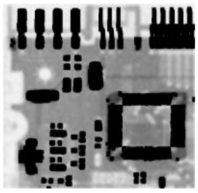


a.原图

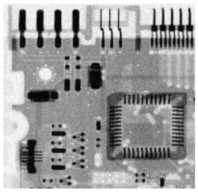


b.加噪

c.算术平均



d.几何平均



e.自适应平均

3

手打函数拟合度不高，查找资料之后采用如下函数进行模糊拟合和逆滤波、维纳滤波图像复原：

①FSPECIAL('motion',LEN,THETA)
为运动模糊算子，表示摄像物体逆时针方向以 theta 角度运动了 len 个像素，len 的默认值为 9，theta 的默认值为 0。

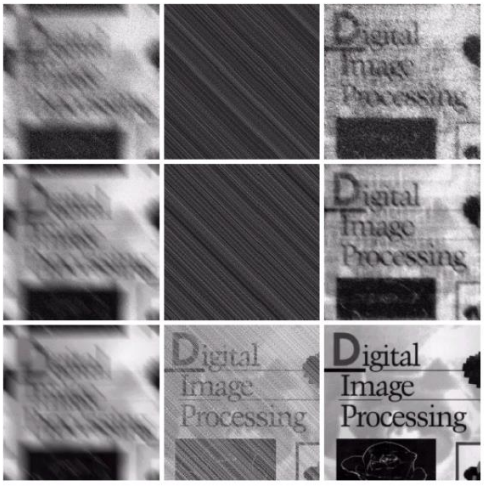
②imfilter(f, w, filtering_mode, boundary_options, size_options)
f 为输入图像；
w 为滤波掩模；
filtering_mode 用于指定在滤波过程中是使用“相关”‘corr’还是“卷积”‘conv’；
boundary_options 用于处理边界充零问题，边界的大小由滤波器的大小确定(‘circular’图像大小通过将图像看成是一个二维周期函数的一个周期来扩展)。

③deconvwnr(I，PSF，NSR)
I 值退化的图像，是原图像卷积一个点扩散函数 PSF 然后加上加性噪声而得到的;
NSR 噪信比，可以为一标量，也可以为与 I 同样大小的矩阵，默认值为 0。

4

实验结果 var=650、 65 、 0.0065 时和书上图片不一致不知为何，希望能与老师探讨解决问题。

$\sigma^2 = 650$



a b c
d e f
g h i

FIGURE 5.29 (a) Image corrupted by motion blur and additive noise. (b) Result of inverse filtering. (c) Result of Wiener filtering. (d)–(f) Same sequence, but with noise variance one order of magnitude less. (g)–(i) Same sequence, but noise variance reduced by five orders of magnitude from (a). Note in (h) how the deblurred image is quite visible through a "curtain" of noise.