《数字图像处理》实验报告

实验名称　　：实验 5 数字图像空域线性滤波处理

实验日期　　：2022.9.30

姓　　名　　：傅康

学　　号　　：084520126

班　　级　　：医信 20

成　　绩　　：

信息技术学院

南京中医药大学

实验目的：

1. 进一步了解 MatLab 软件/语言，学会使用 MatLab 对图像作滤波处理，掌握滤波算法，体会滤波效果。
2. 掌握图像滤波的基本定义及目的。
3. 理解空间域滤波的基本原理及方法。
4. 掌握进行图像的空域滤波的方法。
5. 了解几种不同滤波方式的使用和使用的场合，培养处理实际图像的能力，并为课堂教学提供配套的实践机会。

实验内容和要求

建立一个名为“xxxxx 实验 5”的解决方案（xxxxx 为自己的学号）
数据准备：一个灰度图像文件，一个 8*8 的 uint8 的二维矩阵

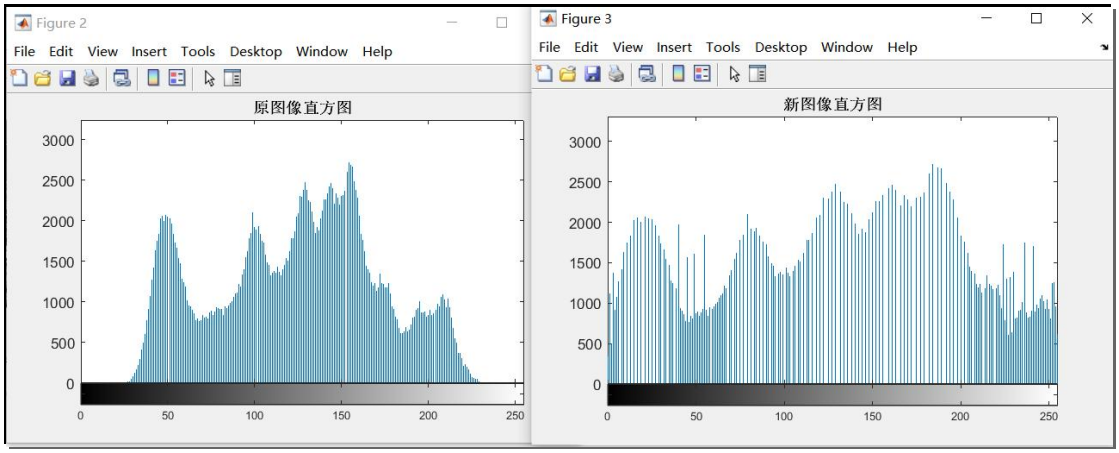
1. 自主编程实现灰度直方图均衡化处理算法，并与 histeq()函数的运算效率进行比较。
2. 自主编程实现灰度直方图规定化处理算法，并与 adapthisteq ()函数的运算效率进行比较。
3. 读取图像及矩阵数据，使用 imnoise()函数给图像分别加入椒盐噪声和高斯噪声后并显示在图像窗口中。
4. 对加入噪声的图像及矩阵分别选用均值滤波、加权滤波等不同的平滑模板做运算，对比不同模板所形成的数据特点，并在图像窗口中显示。
5. 使用函数 fspecial 和 imfilter，分别采用不同的填充方法（或边界选项，如零填充、’replicate’、’symmetric’、’circular’）进行滤波，并显示处理后的图像及矩阵。
6. 运用 for 循环，将加有椒盐噪声的图像进行 10 次，20 次均值滤波，查看其特点，显示均值处理后的图像及矩阵。

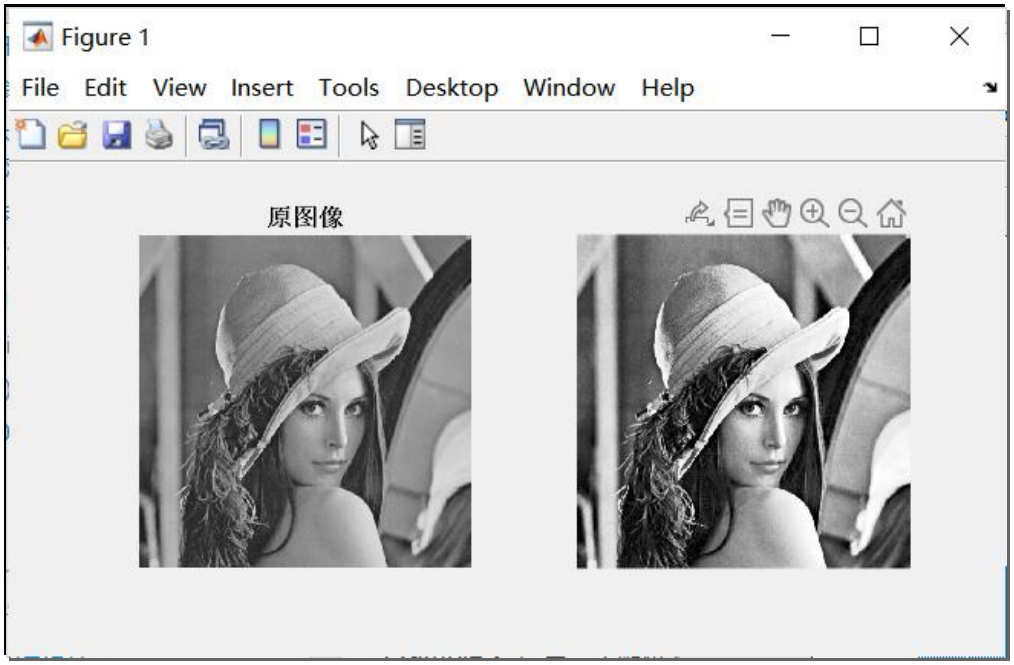
运行结果（写清题号）

描述实验的基本步骤，用数据和图片给出各个步骤中取得的实验结果和源代码，并进行必要的讨论，必须包括原始图像及其计算/处理后的图像。

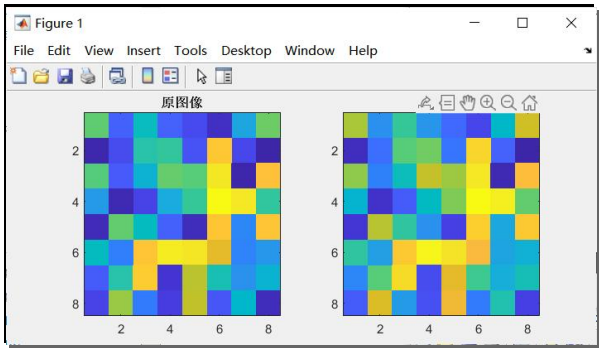
1. 自主编程实现灰度直方图均衡化处理算法，并与 histeq()函数的运算效率进行比较。

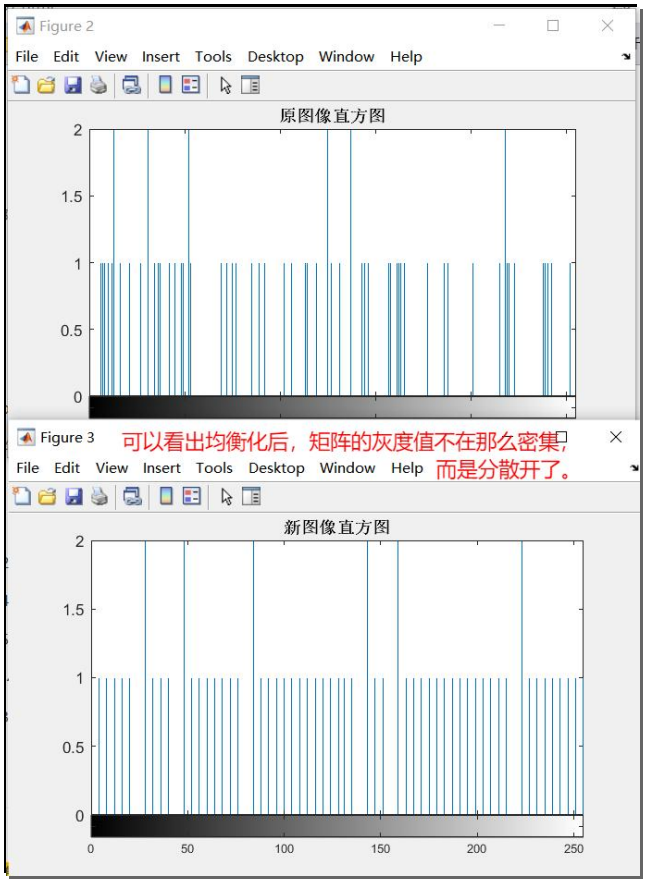
图像：





自定义矩阵：





	1	2	3	4	5	6	7	8
1	75	181	86	204	33	149	94	52
2	191	200	155	187	185	189	175	99
3	2	73	189	13	28	60	153	141
4	12	177	26	18	30	188	202	58
5	170	142	32	22	164	248	94	164
6	154	101	140	204	84	221	52	124
7	134	15	124	241	167	22	22	38
8	186	199	227	175	191	93	197	200

以[3,1]为例。由直方图可得
原灰度级r=2,nr(2)=1,
pr(2)=1/64
新灰度级ps=T(r)=T(2)=
pr(1)+pr(2)=(0+1)/64
≈4/256，即s新值为4。

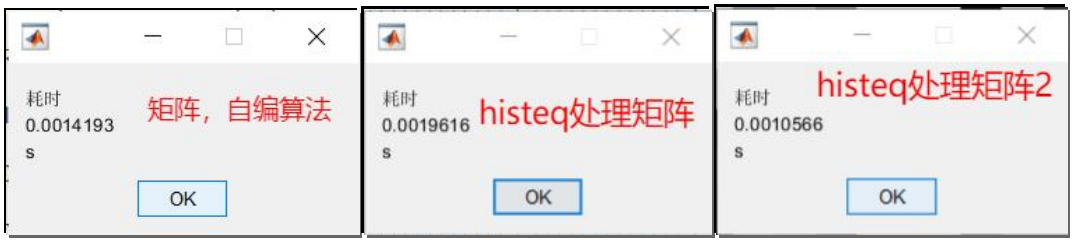
	1	2	3	4	5	6	7	8
1	80	179	88	239	52	135	100	64
2	211	227	147	191	183	203	171	104
3	4	76	203	12	40	72	139	128
4	8	175	36	20	44	195	231	68
5	163	131	48	32	155	255	100	155
6	143	108	124	239	84	243	64	116
7	120	16	116	251	159	32	32	56
8	187	219	247	171	211	92	215	227

代码：

ex5_1.m

```
%直方图均衡化采用灰度级r的累积分布函数作为变换函数，s(新灰度级)=T(r)
% imA=imread("C:\Users\Knight6\Pictures\matlabimage\Fig4.bmp");%图像
imA=randi([0,255],8,8,'uint8');%自定义矩阵
[h,w]=size(imA);
imB=zeros(h,w);%初始化新图像
%1 统计原始图像直方图
imAhistogram=imhist(imA);
tic;
%2 计算新的灰度级
imBhistogram=zeros(256,1);%建一个256的新直方图灰度级数组
imBhistogram(1)=imAhistogram(1);%设置累加起始条件
for i=2:256
    %通过累加的得到新的灰度级
    imBhistogram(i)=imBhistogram(i-1)+imAhistogram(i);
end
%3 计算新的直方图
for i=1:w
    for j=1:h
        %总像素数不变,一除就能得到对应的比例，也直接完成了归一化
        %w*h是图像的所有像素，因为像素0-255，数组1-256
        imB(j,i)=imBhistogram(imA(j,i)+1)/(w*h);
    end
end
t=0+toc;
msgbox(["耗时";t;'s'],"modal");
%4 处理后的新灰度代替处理前的灰度，生成新图像显示
    %图像
% figure,subplot(1,2,1),imshow(im2uint8(imA));title('原图像');
% subplot(1,2,2),imshow(im2uint8(imB));title('新图像');
    %自定义矩阵
figure,subplot(1,2,1),imagesc(im2uint8(imA));title('原图像');
subplot(1,2,2),imagesc(im2uint8(imB));title('新图像');
uitable("Data",im2uint8(imA),'Parent',uifigure(1)),title('原矩阵');
uitable("Data",im2uint8(imB),'Parent',uifigure(2)),title('新矩阵');
figure(2),imhist(imA);title('原图像直方图');
figure(3),imhist(im2uint8(imB));title('新图像直方图');

```

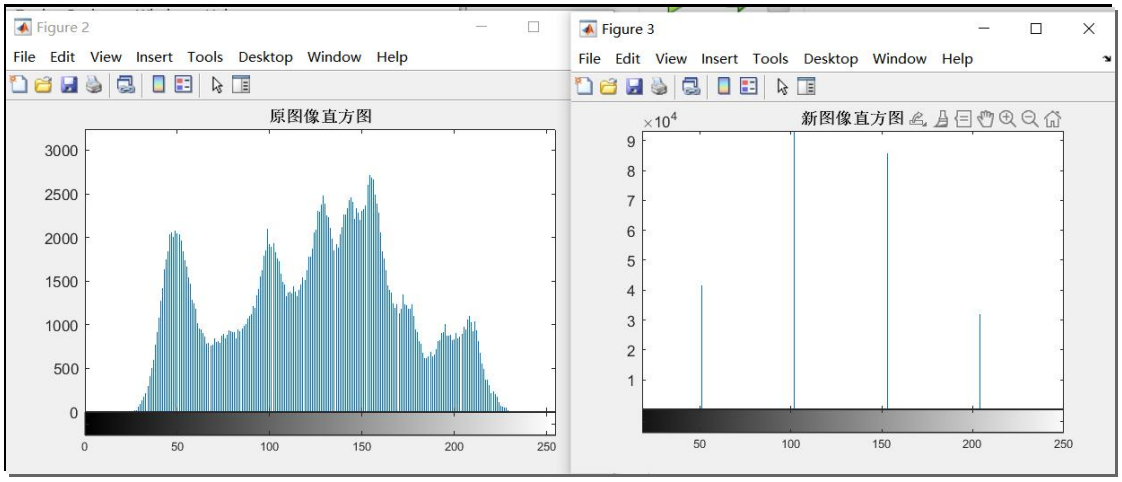
运算效率：对于图像，可以很明显看出 **histeq** 效率比自编算法快了一倍以上；但是对于矩阵，发现有时候自编算法居然运算效率更快，猜想可能是因为自编算法考虑的情况比较简单，鲁棒性不强，而 **histeq** 函数对数据处理更严谨。

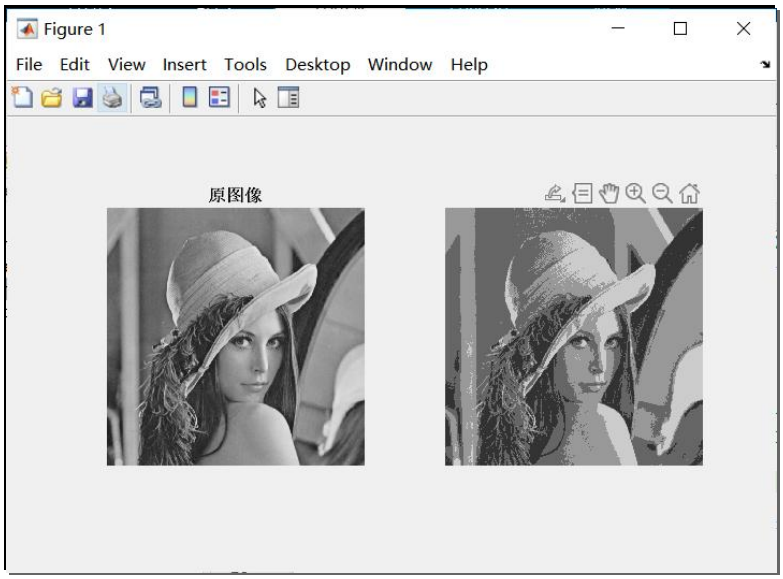


2. 自主编程实现灰度直方图规定化处理算法，并与 adapthisteq ()函数的运算效率进行比较。

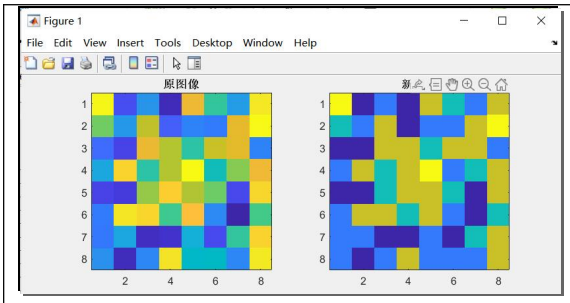
和均衡化相似，规定化以规定化

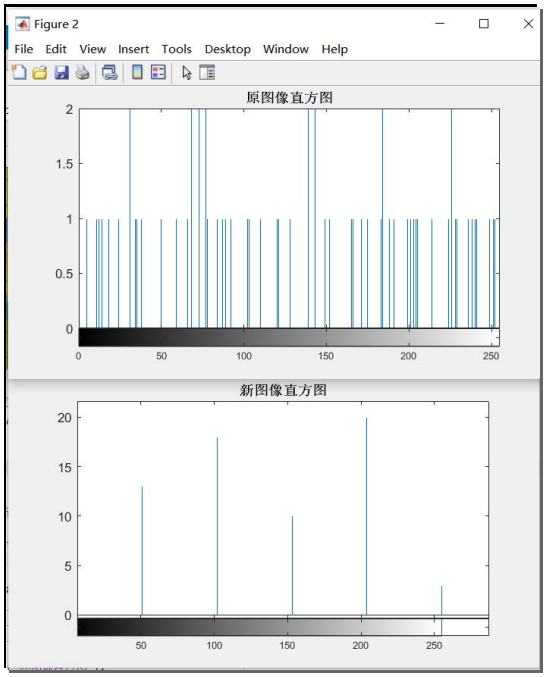
图像：





自定义矩阵:





原矩阵

	1	2	3	4	5	6	7	8
1	249	38	87	11	204	143	92	240
2	166	89	188	50	73	68	201	251
3	59	31	203	184	139	191	199	73
4	103	226	139	184	252	128	171	205
5	31	24	175	224	183	165	34	229
6	68	238	228	149	214	78	5	152
7	66	102	14	18	110	35	143	226
8	84	12	77	236	120	121	77	241

新矩阵

	1	2	3	4	5	6	7	8
1	255	51	102	51	204	153	102	204
2	153	102	204	51	102	102	204	255
3	51	51	204	204	153	204	204	102
4	102	204	153	204	255	102	153	204
5	51	51	153	204	204	153	51	204
6	102	204	204	153	204	102	51	153
7	102	102	51	51	102	51	153	204
8	102	51	102	204	102	102	102	204

规定化直方图
[2 64 136 182 245]
五个灰度级概率均为0.2

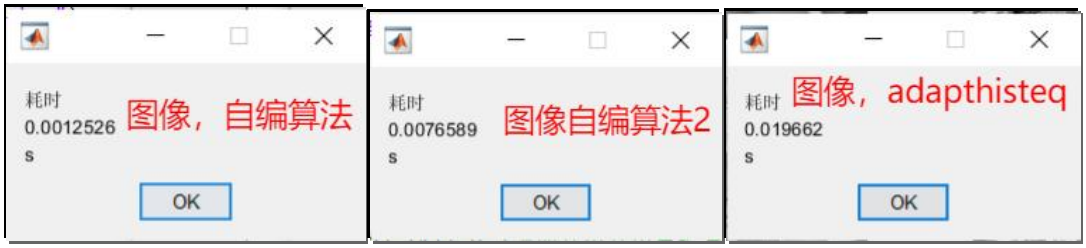
和均衡化类似，
变换函数为规定化直方图的累计分布函数，
最后得出的新矩阵灰度级分布于事先规定化矩阵映射：
51，102，153，204，255
五个灰度级中

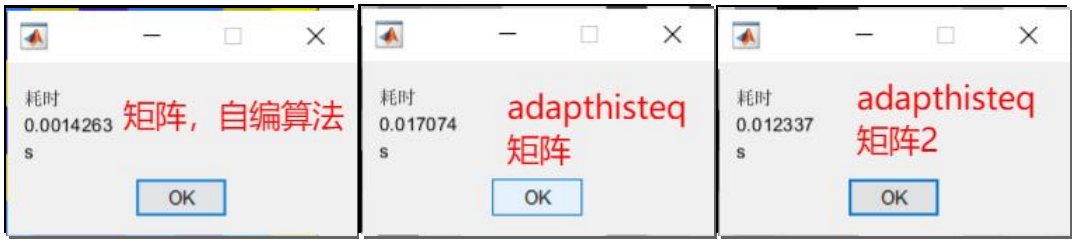
由[1,1]可直观验证，249对应的累计分布函数由于245已经达到1了，所以249映射之后应该是1，即255/255。

代码：

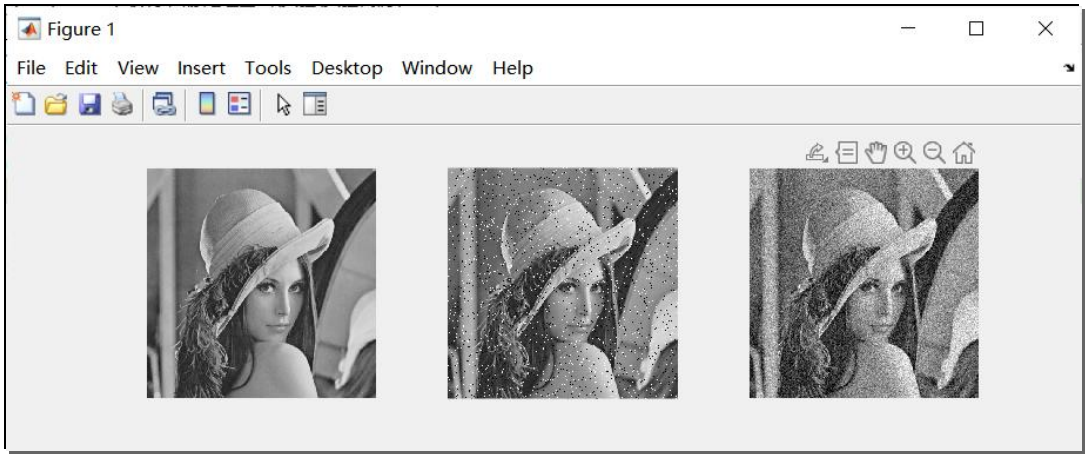
```
%直方图规定化采用 规定直方图累积分布函数 作为变换函数，s(新灰度级)=T(r)
% imA=imread("C:\Users\Knight6\Pictures\matlabimage\Fig4.bmp");
imA=randi([0,255],8,8,'uint8');%自定义矩阵
[h,w]=size(imA);
imB=zeros(h,w);%初始化处理后图像
tic;
%2 设定规定直方图并求其累计直方图，计算新的灰度级
specification=zeros(256,1);
specification([2 64 136 182 245],(1))=0.2;%取其中几个点设置规定化灰度级,理论上取几个点，新图像就有几个灰度级
cul_specification=zeros(256,1);%规定直方图累积分布函数
cul_specification(1)=specification(1);%设置累加起始条件
for i=2:256
    %通过累加的得到新的灰度级
    cul_specification(i)=cul_specification(i-1)+specification(i);%已经归一化累加
end

%3 计算新的直方图
for i=1:w
    for j=1:h
        %w*h是图像的所有像素，因为像素0-255，数组1-256
        imB(j,i)=cul_specification(imA(j,i)+1);
    end
end
t=0+toc;
msgbox(["耗时";t;'s'],"modal");

%4 处理后的新灰度代替处理前的灰度，生成新图像显示
    %图像
% figure,subplot(1,2,1),imshow(im2uint8(imA));title('原图像');
% subplot(1,2,2),imshow(im2uint8(imB));title('新图像');
    %自定义矩阵
figure,subplot(1,2,1),imagesc(im2uint8(imA));title('原图像');
subplot(1,2,2),imagesc(im2uint8(imB));title('新图像');
uitable("Data",im2uint8(imA),'Parent',uifigure(1));
uitable("Data",im2uint8(imB),'Parent',uifigure(2));

figure(2);imhist(im2uint8(imA));title('原图像直方图');
figure(3);imhist(im2uint8(imB));title('新图像直方图');
```

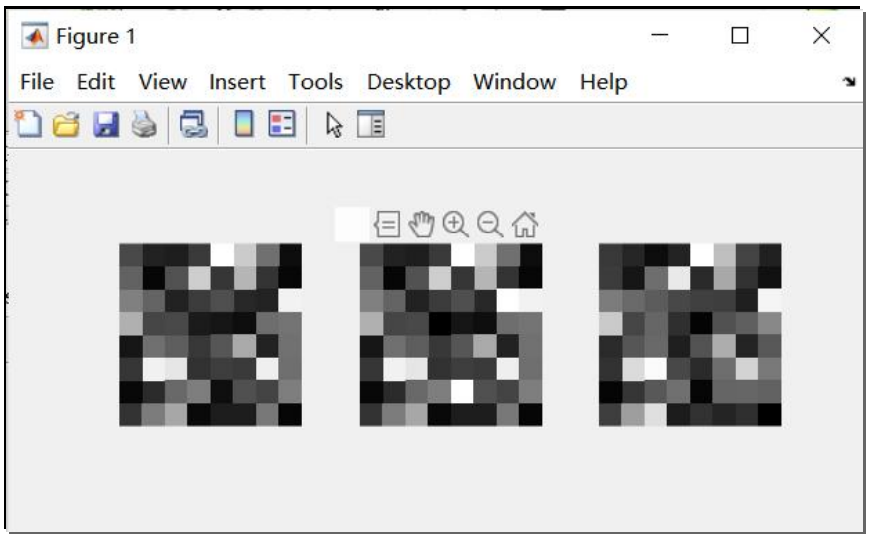
运算效率：对于矩阵，还是同上，**adapthisteq** 有时快有时慢，但都相差不大；对于图像，自编算法快了很多，验证发现是由于给定的规定化直方图的问题，开始我是规定化直方图只给了五个灰度级，如左图，后来，我给了十个灰度级，时间迅速增加。

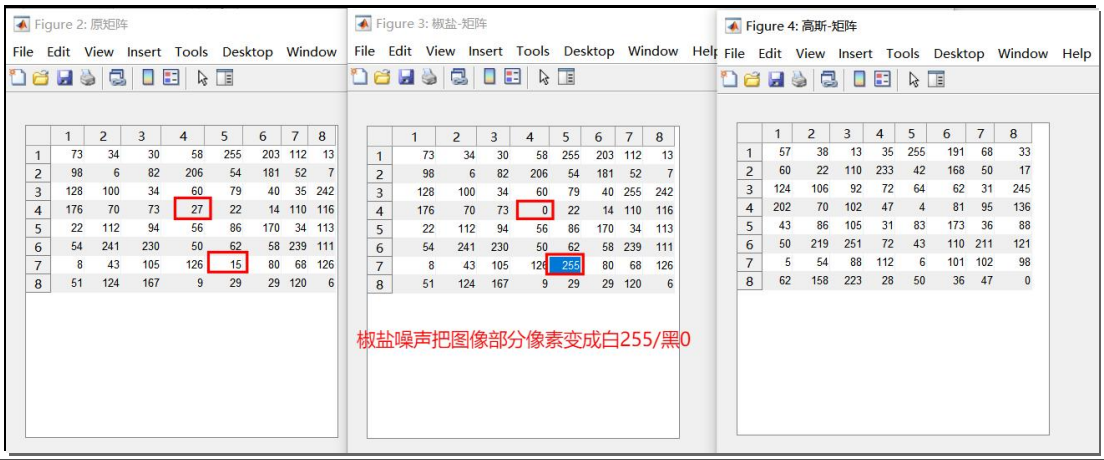




3. 读取图像及矩阵数据，使用 imnoise()函数给图像分别加入椒盐噪声和高斯噪声后并显示在图像窗口中。

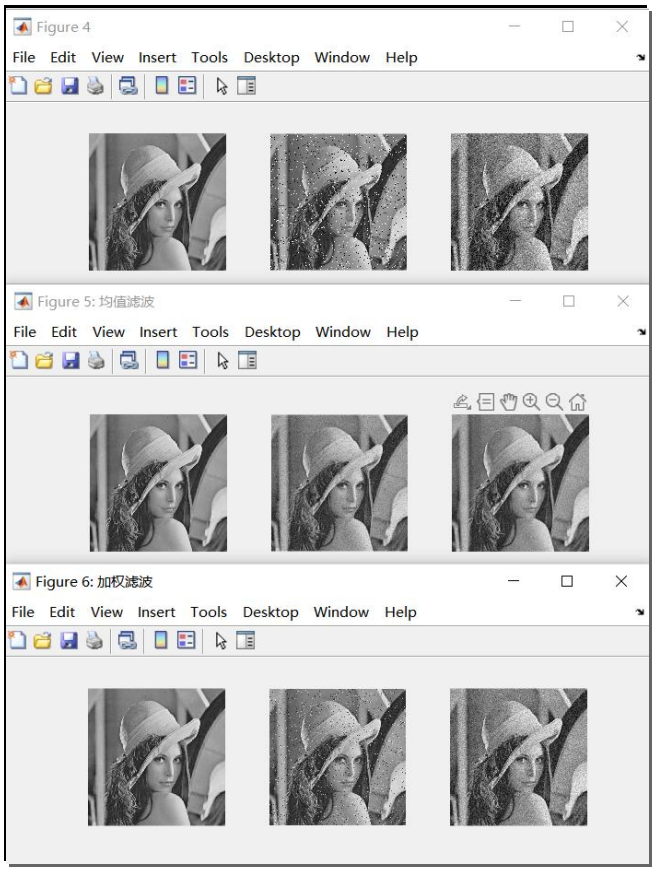




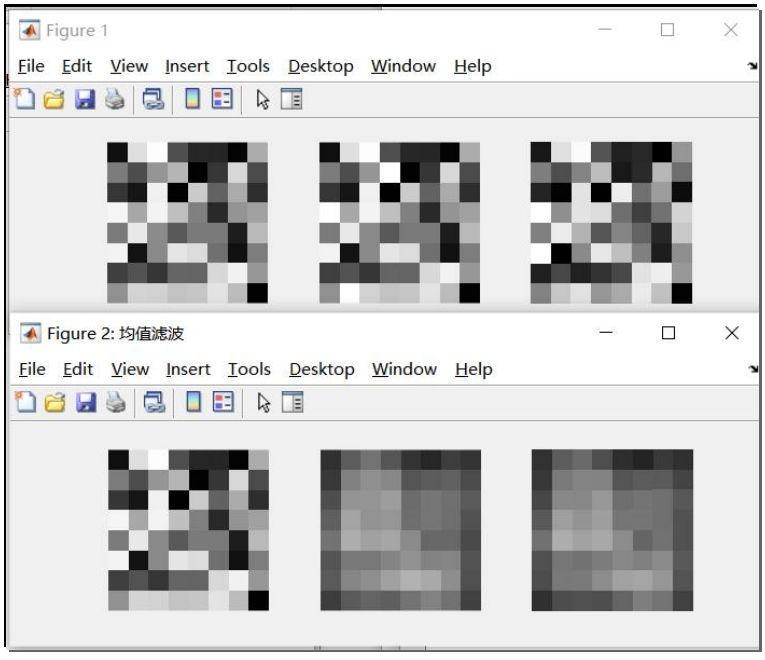


4. 对加入噪声的图像及矩阵分别选用均值滤波、加权滤波等不同的平滑模板做运算，对比不同模板所形成的数据特点，并在图像窗口中显示。

均值和加权均值滤波模板[1 2 1;2 4 2;1 2 1]./16，通过结果我们可以看见均值滤波对噪声能够有较好的消除效果，而加权均值滤波则效果不太明显。



矩阵-均值滤波



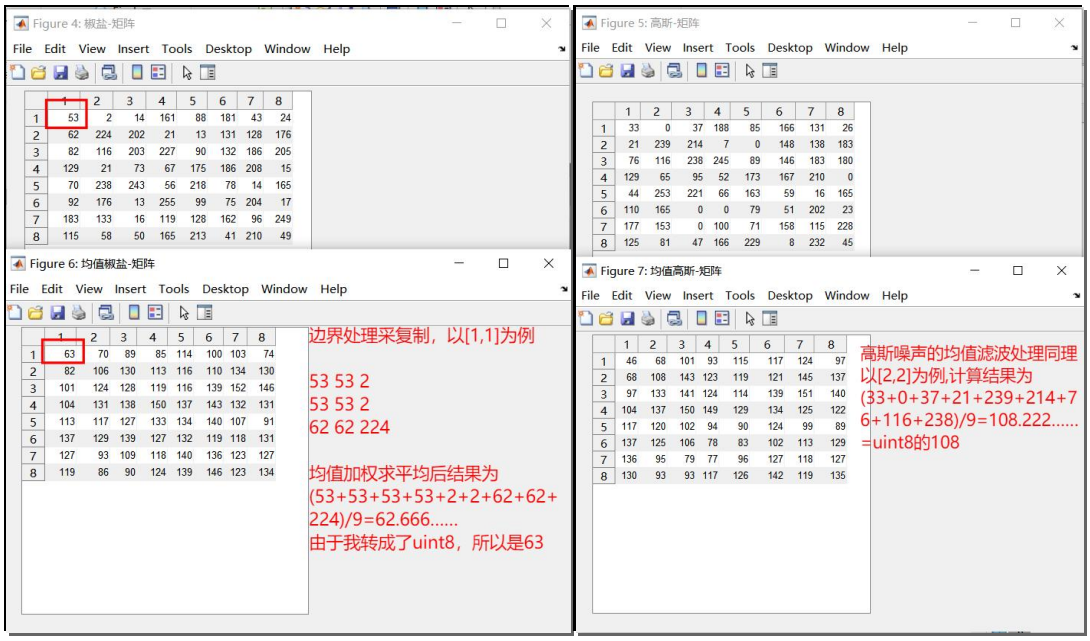


Figure 4: 椒盐-矩阵
File Edit View Insert Tools Desktop Window Help
Figure 5: 高斯-矩阵
File Edit View Insert Tools Desktop Window Help
Figure 6: 均值椒盐-矩阵
File Edit View Insert Tools Desktop Window Help
边界处理采复制，以[1,1]为例
53 53 2
53 53 2
62 62 224
均值加权求平均后结果为
(53+53+53+53+2+2+62+62+224)/9=62.666.....
由于我转成了uint8，所以是63
Figure 7: 均值高斯-矩阵
File Edit View Insert Tools Desktop Window Help
高斯噪声的均值滤波处理同理
以[2,2]为例,计算结果为
(33+0+37+21+239+214+76+116+238)/9=108.222.....
=uint8的108

矩阵-加权滤波

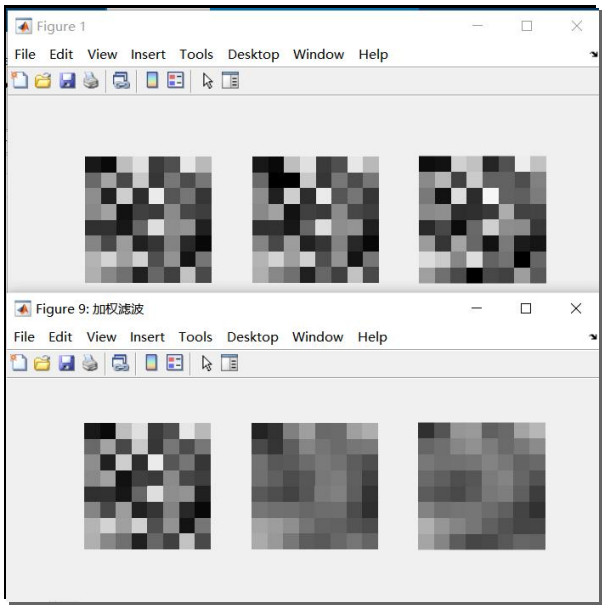
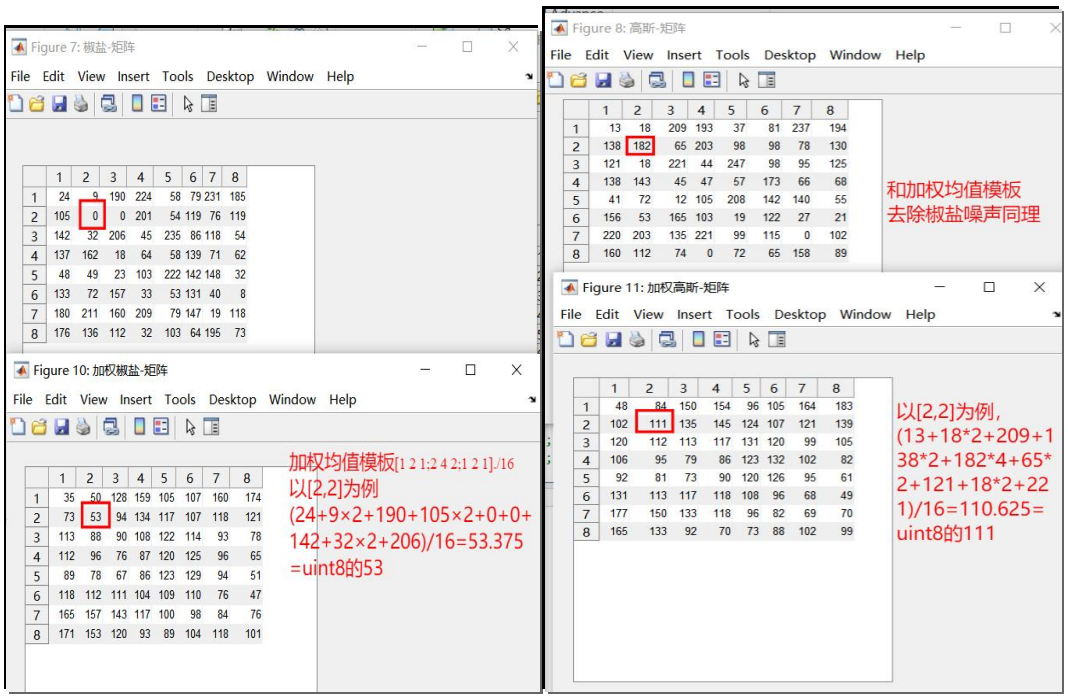


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 9: 加权滤波
File Edit View Insert Tools Desktop Window Help



代码：

```
figure,subplot(1,3,1),imshow(imA);
subplot(1,3,2),imshow(imB);
subplot(1,3,3),imshow(imC);

% gB=imfilter(imB,fspecial("average",3),'replicate');
% gC=imfilter(imC,fspecial("average",3),'replicate');
% figure("Name","均值滤波"),subplot(1,3,1),imshow(imA);
% subplot(1,3,2),imshow(uint8(gB));
% subplot(1,3,3),imshow(uint8(gC));

uitable("Data",imA,'Parent',figure("Name","原矩阵"));
uitable("Data",imB,'Parent',figure("Name","椒盐-矩阵"));
uitable("Data",imC,'Parent',figure("Name","高斯-矩阵"));
% uitable("Data",uint8(gB),'Parent',figure("Name","均值椒盐-矩阵"));
% uitable("Data",uint8(gC),'Parent',figure("Name","均值高斯-矩阵"));

model=[1 2 1;2 4 2;1 2 1]./16;
gB=imfilter(imB,model,'replicate');
gC=imfilter(imC,model,'replicate');
figure("Name","加权滤波"),subplot(1,3,1),imshow(imA);
subplot(1,3,2),imshow(uint8(gB));
subplot(1,3,3),imshow(uint8(gC));
uitable("Data",uint8(gB),'Parent',figure("Name","加权椒盐-矩阵"));
uitable("Data",uint8(gC),'Parent',figure("Name","加权高斯-矩阵"));
```

5. 使用函数 fspecial 和 imfilter，分别采用不同的填充方法（或边界选项，如零填充、’replicate’、’symmetric’、’circular’）进行滤波，并显示处理后的图像及矩阵。

图像

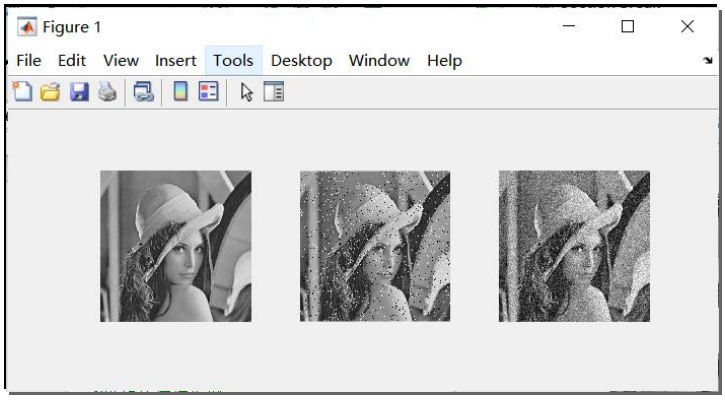


Figure 1
File Edit View Insert Tools Desktop Window Help

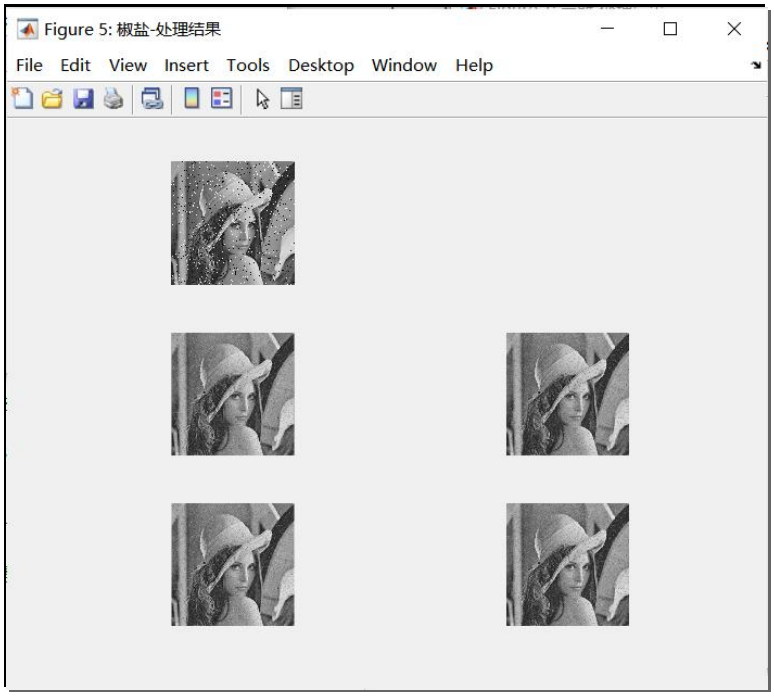


Figure 5: 椒盐-处理结果
File Edit View Insert Tools Desktop Window Help

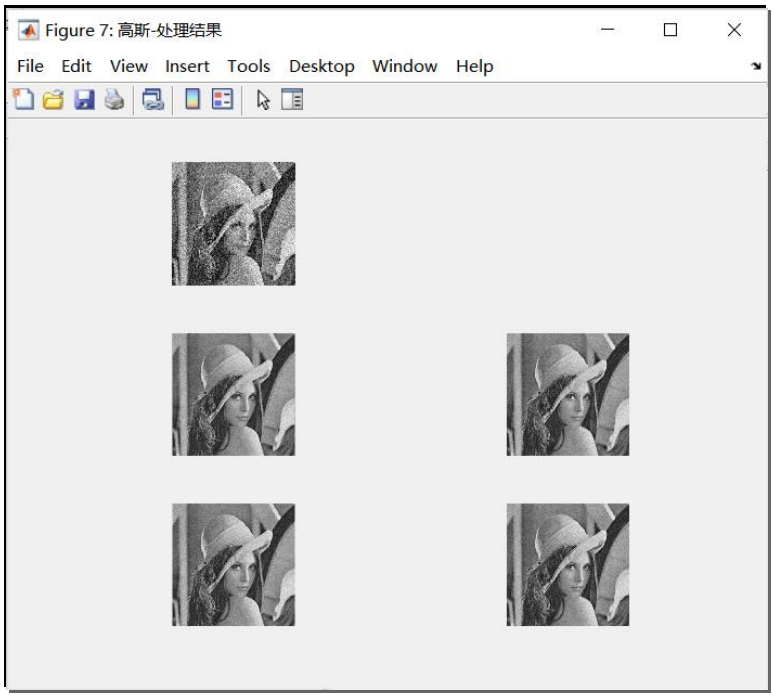
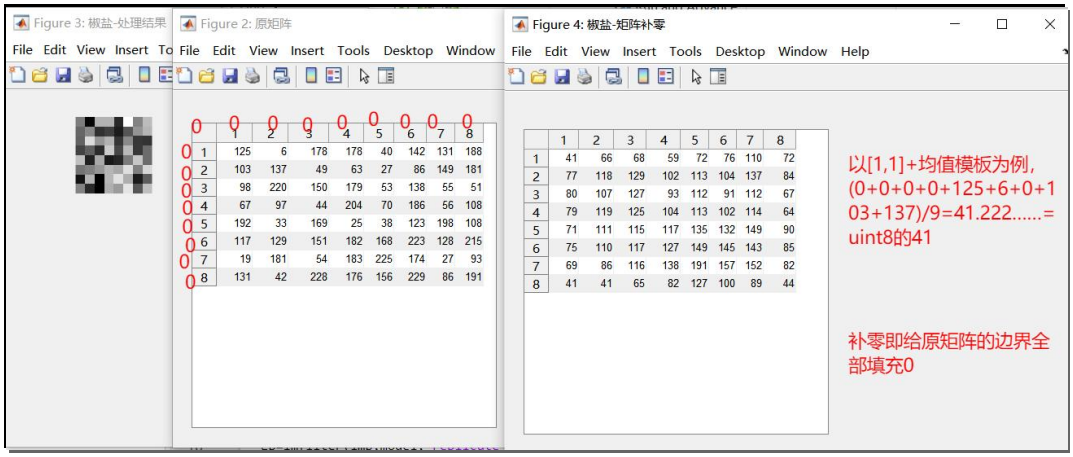
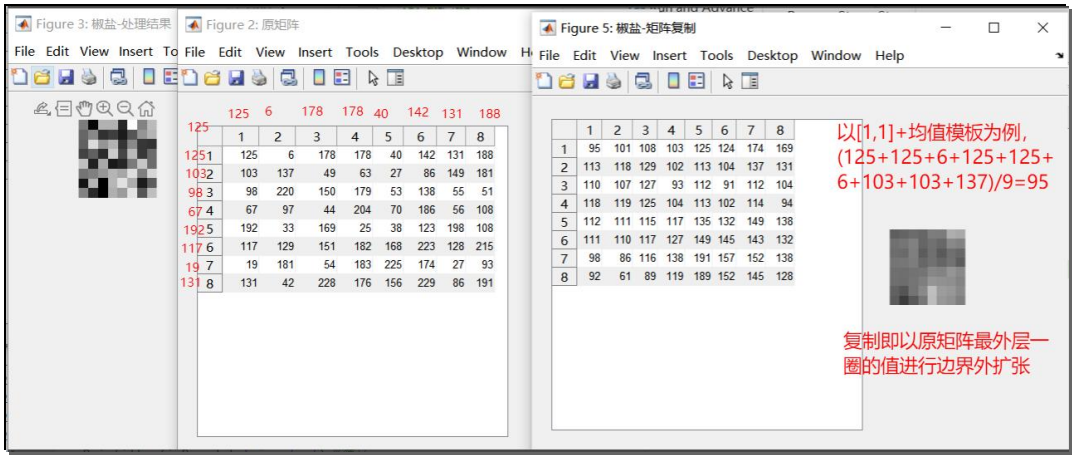
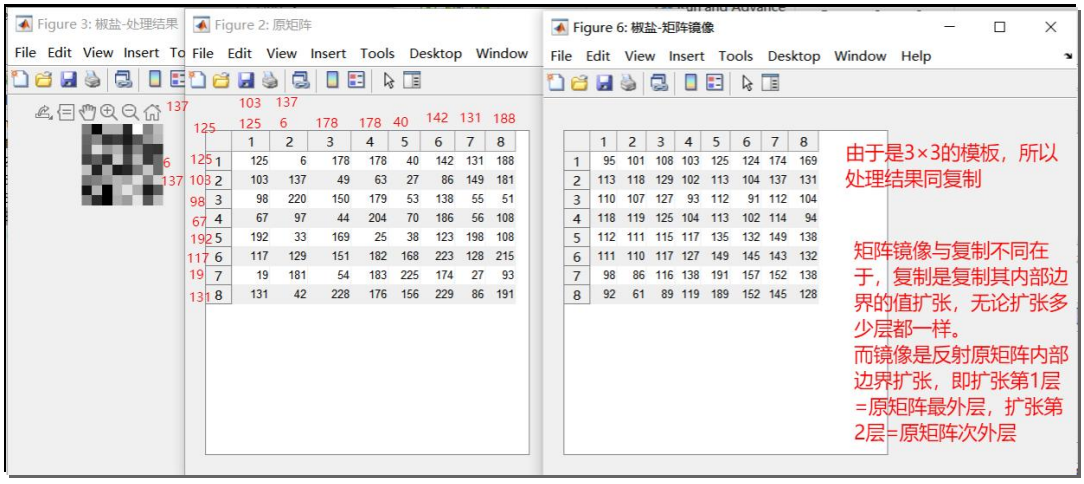


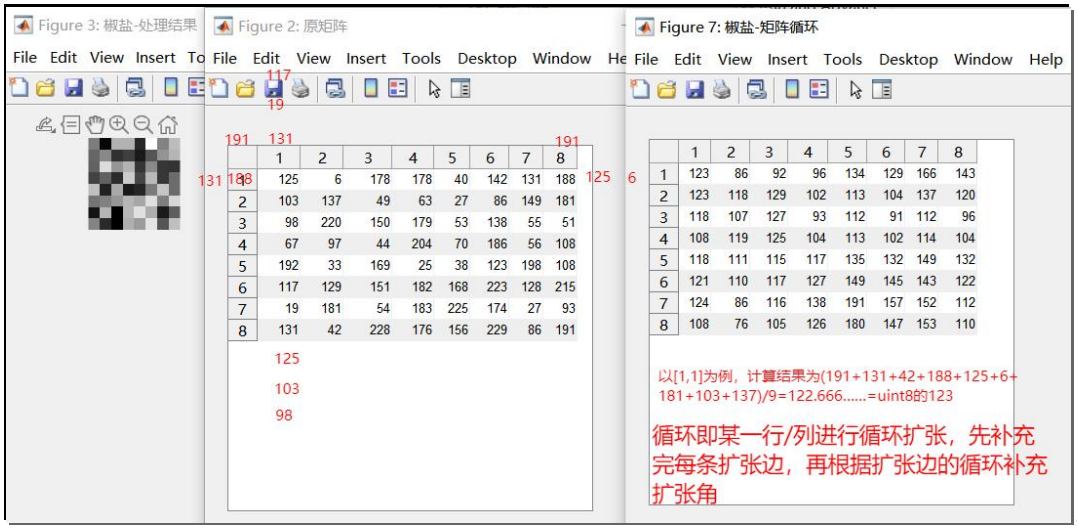
Figure 7: 高斯-处理结果
File Edit View Insert Tools Desktop Window Help

矩阵—以椒盐噪声的处理为例







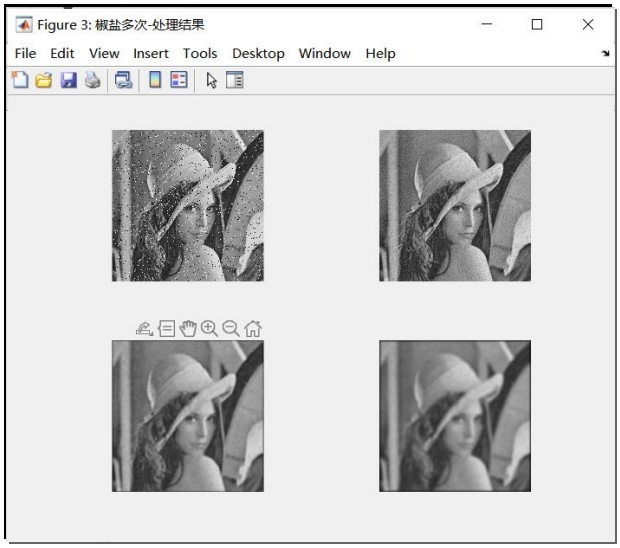


代码：

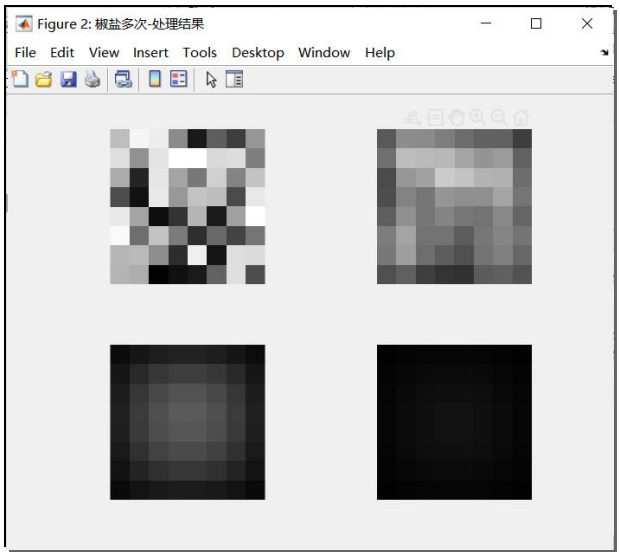
```
% imA=imread("C:\Users\Knight6\Pictures\matlabimage\Fig4.bmp");
imA=randi([0,255],8,8,'uint8');%自定义矩阵
%加噪声
imB=imnoise(imA,"salt & pepper");%椒盐噪声
imC=imnoise(imA,"gaussian");%高斯噪声
%显示噪声图像
figure,subplot(1,3,1),imshow(imA);
subplot(1,3,2),imshow(imB);
subplot(1,3,3),imshow(imC);
%以均值滤波为例
model=fspecial("average",3);%均值滤波模板
%椒盐噪声处理
uitable("Data",imA,'Parent',figure("Name","原矩阵"));
figure('Name','椒盐-处理结果'),subplot(3,2,1),imshow(imB);
gB=imfilter(imB,model,0);%补零
uitable("Data",gB,'Parent',figure("Name","椒盐-矩阵补零"));
subplot(3,2,3),imshow(uint8(gB));
gB=imfilter(imB,model,'replicate');%复制
uitable("Data",gB,'Parent',figure("Name","椒盐-矩阵复制"));
subplot(3,2,4),imshow(uint8(gB));
gB=imfilter(imB,model,'symmetric');%镜像
uitable("Data",gB,'Parent',figure("Name","椒盐-矩阵镜像"));
subplot(3,2,5),imshow(uint8(gB));
gB=imfilter(imB,model,'circular');%循环
uitable("Data",gB,'Parent',figure("Name","椒盐-矩阵循环"));
subplot(3,2,6),imshow(uint8(gB));
%高斯噪声处理
figure('Name','高斯-处理结果'),subplot(3,2,1),imshow(imC);
gC=imfilter(imC,model,0);%补零
subplot(3,2,3),imshow(uint8(gC));
gC=imfilter(imC,model,'replicate');%复制
subplot(3,2,4),imshow(uint8(gC));
gC=imfilter(imC,model,'symmetric');%镜像
subplot(3,2,5),imshow(uint8(gC));
gC=imfilter(imC,model,'circular');%循环
subplot(3,2,6),imshow(uint8(gC));
```

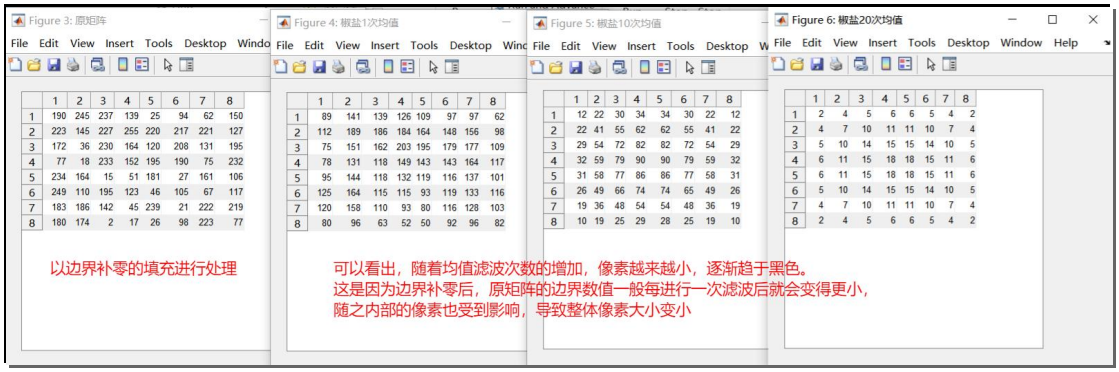
6. 运用 for 循环，将加有椒盐噪声的图像进行 10 次，20 次均值滤波，查看其特点，显示均值处理后的图像及矩阵。

图像：

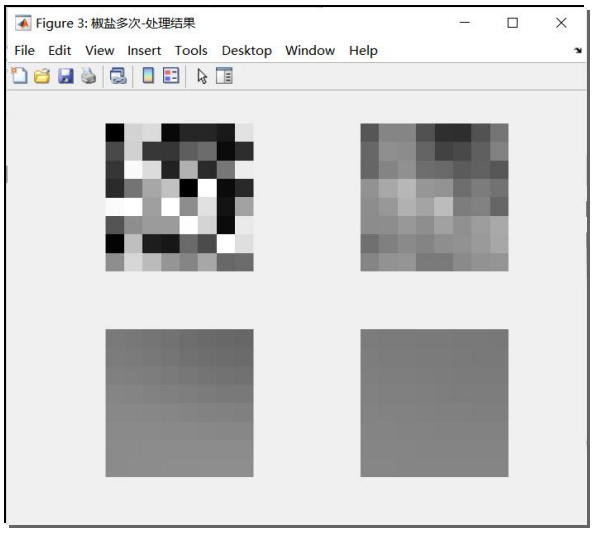


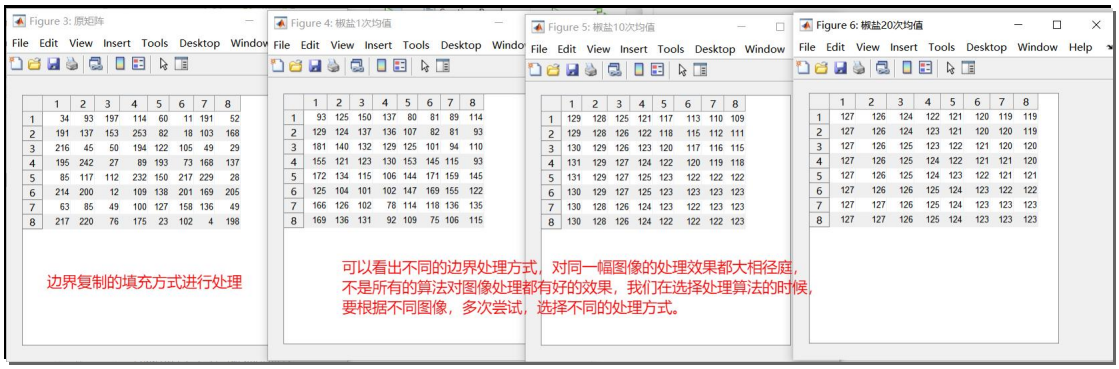
矩阵-补零：





矩阵-复制：





多次的均值滤波，可以理解成一个均匀化的橡皮，不断重复地打磨一张图片，将卷积核所覆盖的区域进行均匀化，从而消除突兀的噪声点。

代码：

```m
% imA=imread("C:\Users\Knight6\Pictures\matlabimage\Fig4.bmp");
imA=randi([0,255],8,8,'uint8');%自定义矩阵
%加噪声
imB=imnoise(imA,"salt & pepper");%椒盐噪声
imC=imnoise(imA,"gaussian");%高斯噪声
%显示噪声图像
figure,subplot(1,3,1),imshow(imA);
subplot(1,3,2),imshow(imB);
subplot(1,3,3),imshow(imC);
%均值滤波
model=fspecial("average",3);%均值滤波模板
%椒盐噪声处理
figure('Name','椒盐多次-处理结果'),subplot(2,2,1),imshow(imB);
gB=imfilter(imB,model,'replicate');
gB1=gB;
subplot(2,2,2),imshow(gB);
for i=1:10
    gB=imfilter(gB,model,'replicate');
end
gB10=gB;
subplot(2,2,3),imshow(gB);
for j=1:20
    gB=imfilter(gB,model,'replicate');
end
gB20=gB;
subplot(2,2,4),imshow(gB);
%矩阵数据显示
uitable("Data",imA,'Parent',figure("Name","原矩阵"));
uitable("Data",gB1,'Parent',figure("Name","椒盐1次均值"));
uitable("Data",gB10,'Parent',figure("Name","椒盐10次均值"));
uitable("Data",gB20,'Parent',figure("Name","椒盐20次均值"));
```

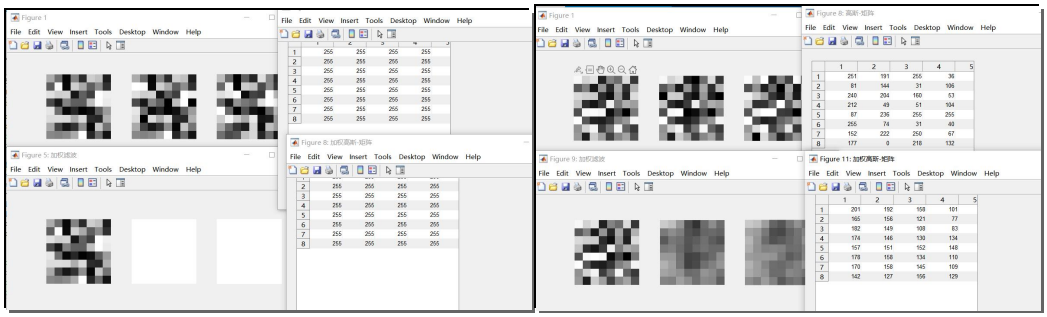
讨论/注意：

①imBhistogram(imA(j,i)+1)之所以+1 是因为数组从 1 开始（1-256）而像素从 0 开始（0-255）。所以最开始输入 imBhistogram(0)会报错。

```
disp(imBhistogram(0));
```

Array indices must be positive integers or logical values.

②第 4 步，最开始，用加权均值滤波模板[1 2 1;2 4 2;1 2 1]./16，只是简单设置[1 2 1;2 4 2;1 2 1]，并未实际上的去求平均，结果导致处理后的图像全白 255，后来通过 command window 输入代码验证发现模板未求平均所致，遂修正。



③对直方图均衡化原理不太理解

虽然了解了直方图均衡化和规定化的方法步骤，直观感受了其均衡化的效果。但是对其中的

数学原理，还是不能理解得很充分，还需要继续花点时间看看。

【数字图像处理】 直方图的均衡与规定化 - 司念 - 博客园 (cnblogs.com)

实验的体会与思考题

1. 总结不同平滑处理算法适用的图像特点

①均值滤波

它利用卷积运算对图像邻域的像素灰度进行平均，从而达到减小图像中噪声影响、降低图像对比度的目的。但邻域平均值主要缺点是在降低噪声的同时使图像变得模糊，特别在边缘和细节处，而且邻域越大，在去噪能力增强的同时模糊程度越严重。所以适用于零散突兀的椒盐噪声处理。

②高斯滤波

图像高斯平滑也是邻域平均的思想对图像进行平滑的一种方法，在图像高斯平滑中，对图像进行平均时，不同位置的像素被赋予了不同的权重。有利于保持边缘、细节，对于这一类需求的图像适用。

③中值滤波

在中值滤波算法中，对于孤立像素的属性并不非常关注，而是认为图像中的每个像素都跟邻域内其他像素有着密切的关系，对于每一个邻域，算法都会在采样得到的若干像素中，选择一个最有可能代表当前邻域特征的像素的灰度作为中心像素灰度，对于需要避免离散型杂点对图像影响的图像适用。