《数字图像处理》实验报告

实验名称　　：实验 6 数字图像空域非线性滤波处理

实验日期　　：2022.10.7

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 对图像作非线性滤波处理，使学生有机会掌握非线性滤波算法，体会滤波效果。
2. 掌握图像非线性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 6”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. 使用函数 padarray 及 colfilt 进行图像及矩阵的非线性滤波
2. 使用函数 ordfilt2 对图像及矩阵做顺序统计滤波
3. 自主设计实现 2_D 中值滤波算法，并对比函数 medfilt2 对图像及矩阵做中值滤波的算法效率。
4. 用 conv2 函数进行线性锐化滤波，实现图像锐化滤波处理，采用 3×3 的拉普拉斯算子 w = [1 1 1;1 -8 1;1 1 1]处理。再分别采用 5×5，9×9，15×15 和 25×25 大小的拉普拉斯算子对图像及矩阵进行锐化滤波，观察其有何不同，要求在同一窗口中显示。
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’）对图像及矩阵进行拉普拉斯算子的锐化滤波处理，并显示处理后的结果（包括拉普拉斯图像及最终锐化结果图像）。
6. 查找资料，应用梯度算子，sobel 算子，prewitt 算子，roberts 算子分别对图像及矩阵进行锐化处理，并对比各种算子处理后的结果进行分析；（edge 函数）；

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。

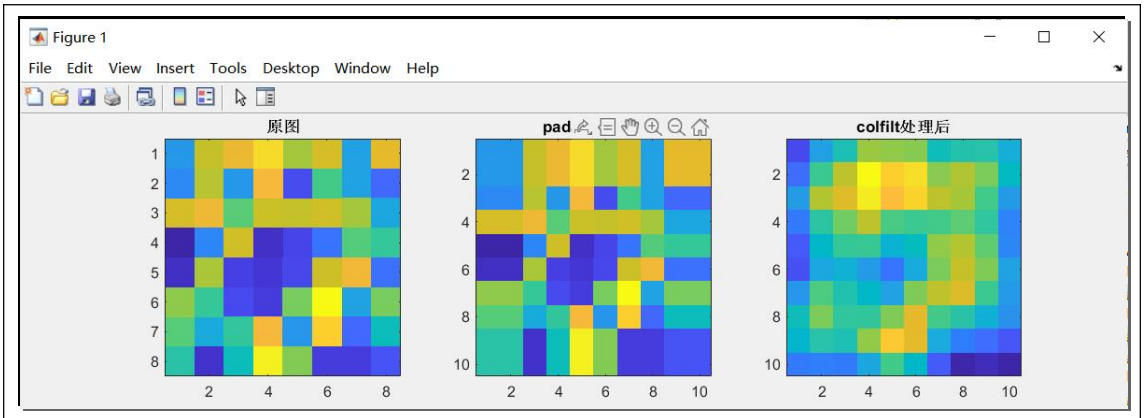
1. 使用函数 padarray 及 colfilt 进行图像及矩阵的非线性滤波

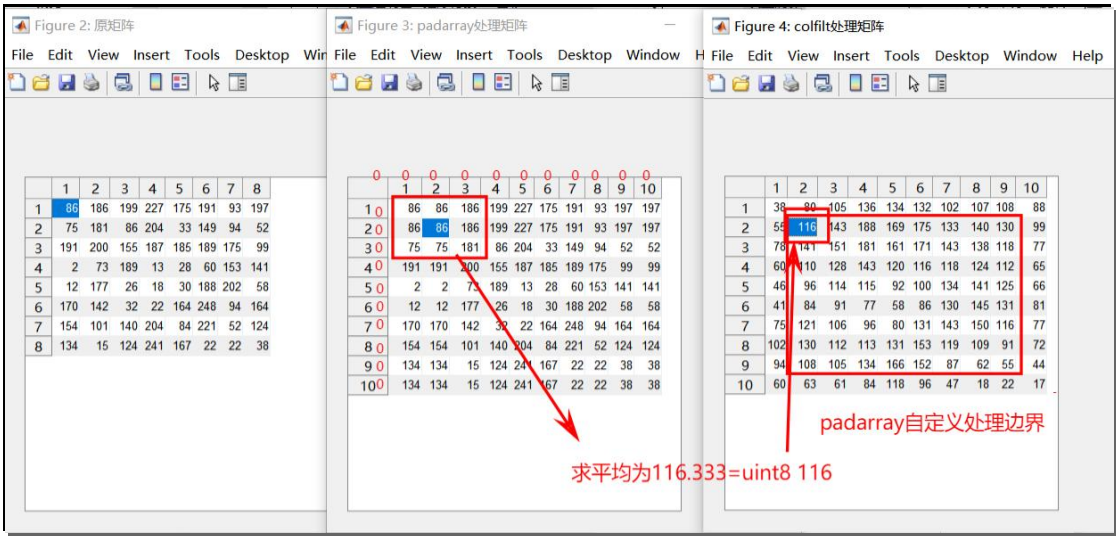
图像：

```
imA=imread('C:\Users\Knight6\Pictures\matlabimage\imC.bmp');
figure,subplot(131),imshow(imA),title('原图');
img=padarray(imA,[15 15],'replicate');
subplot(132),imshow(img),title('padarray处理');
img2=colfilt(img,[15 15],'sliding',@mean);
subplot(133),imshow(uint8(img2)),title('colfilt处理后');
```

矩阵：

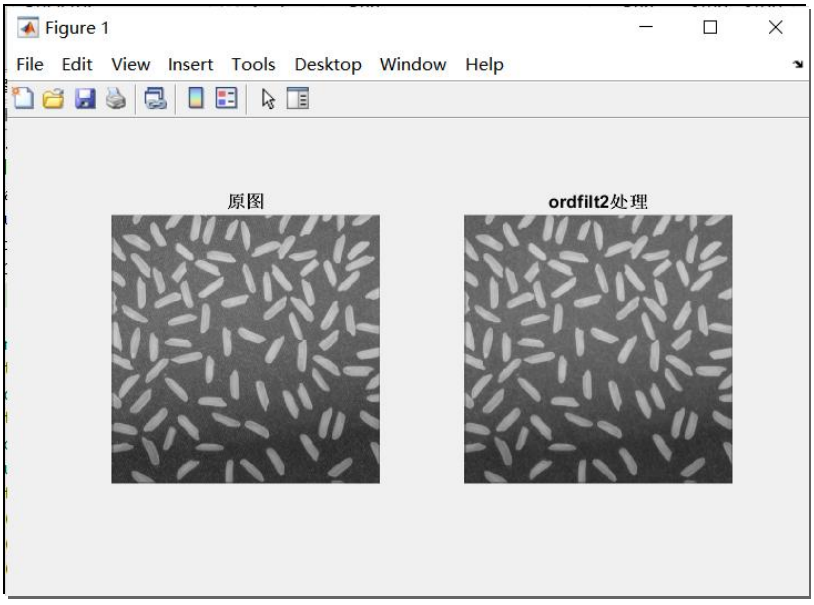
```
imA=randi([0 255],8);
subplot(131),imagesc(imA),title('原图');
img=padarray(imA,[1 1],'replicate');
subplot(132),imagesc(img),title('padarray处理');
img2=colfilt(img,[3 3],'sliding',@mean);
img2_=uint8(img2);
subplot(133),imagesc(img2_),title('colfilt处理后');
figure('Name','原矩阵'),uitable("Data",imA);
figure('Name','padarray处理矩阵'),uitable('Data',img);
figure('Name','colfilt处理矩阵'),uitable('Data',img2_);
```



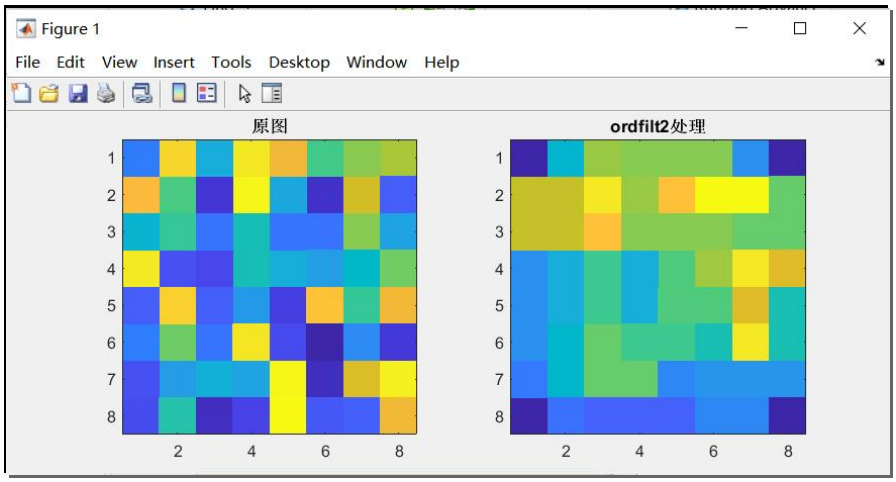


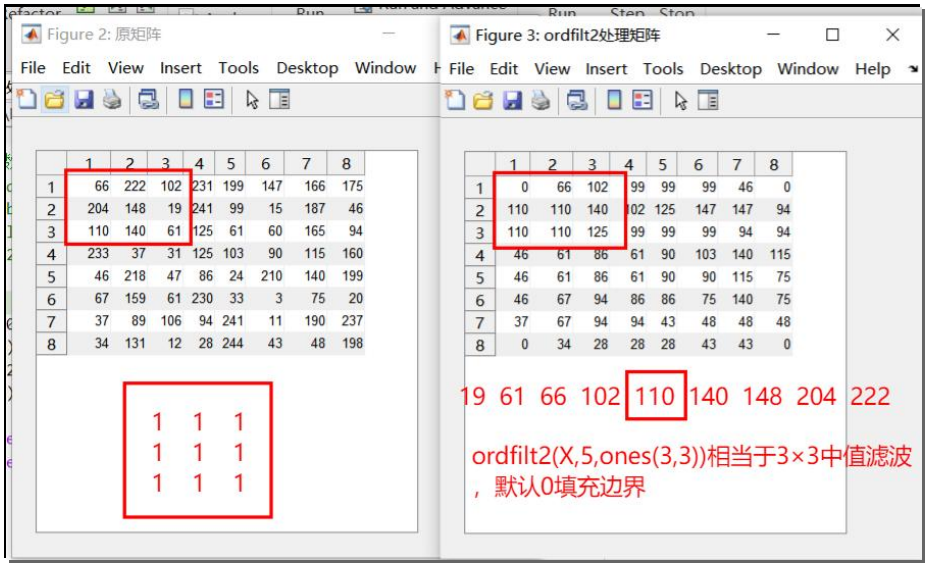
2. 使用函数 ordfilt2 对图像及矩阵做顺序统计滤波

```
imA=imread('C:\Users\Knight6\Pictures\matlabimage\imC.bmp');
figure,subplot(121),imshow(imA),title('原图');
img=ordfilt2(imA,5,ones(3,3));
subplot(122),imshow(img),title('ordfilt2处理');
```



```
imA=randi([0 255],8);
subplot(121),imagesc(imA),title('原图');
img=ordfilt2(imA,5,ones(3,3));
subplot(122),imagesc(img),title('ordfilt2处理');

figure('Name','原矩阵'),uitable("Data",imA);
figure('Name','ordfilt2处理矩阵'),uitable('Data',img);
```





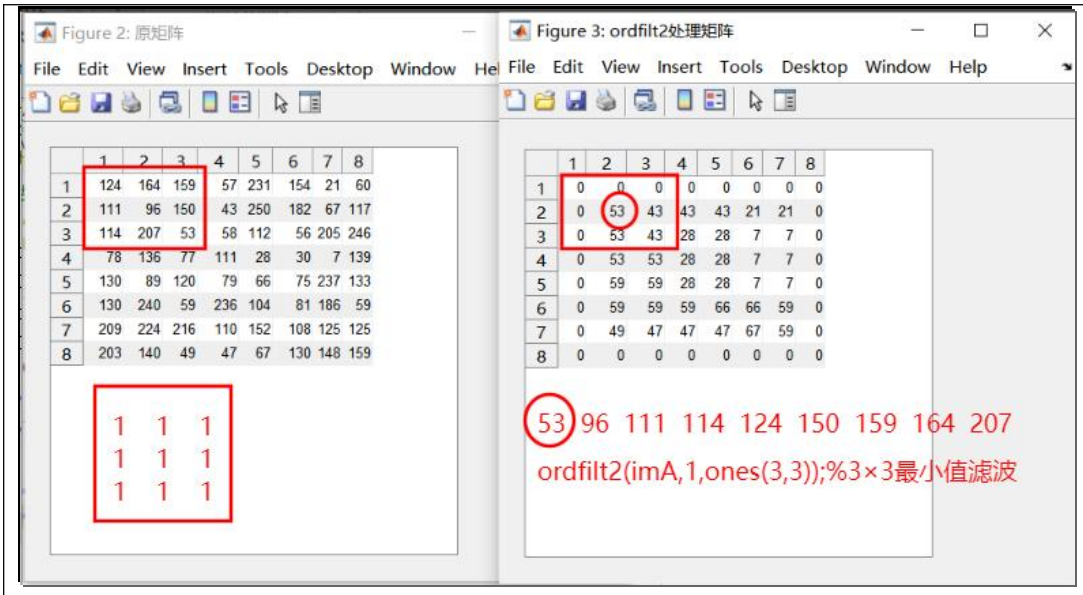


Figure 2: 原矩阵
File Edit View Insert Tools Desktop Window Hel
1 2 3 4 5 6 7 8
1 124 164 159 57 231 154 21 60
2 111 96 150 43 250 182 67 117
3 114 207 53 58 112 56 205 246
4 78 136 77 111 28 30 7 139
5 130 89 120 79 66 75 237 133
6 130 240 59 236 104 81 186 59
7 209 224 216 110 152 108 125 125
8 203 140 49 47 67 130 148 159
1 1 1
1 1 1
1 1 1
Figure 3: ordfilt2处理矩阵
File Edit View Insert Tools Desktop Window Help
1 2 3 4 5 6 7 8
1 0 0 0 0 0 0 0 0
2 0 53 43 43 43 21 21 0
3 0 53 43 28 28 7 7 0
4 0 53 53 28 28 7 7 0
5 0 59 59 28 28 7 7 0
6 0 59 59 59 66 66 59 0
7 0 49 47 47 47 67 59 0
8 0 0 0 0 0 0 0 0
53 96 111 114 124 150 159 164 207
ordfilt2(imA,1,ones(3,3));%3×3最小值滤波

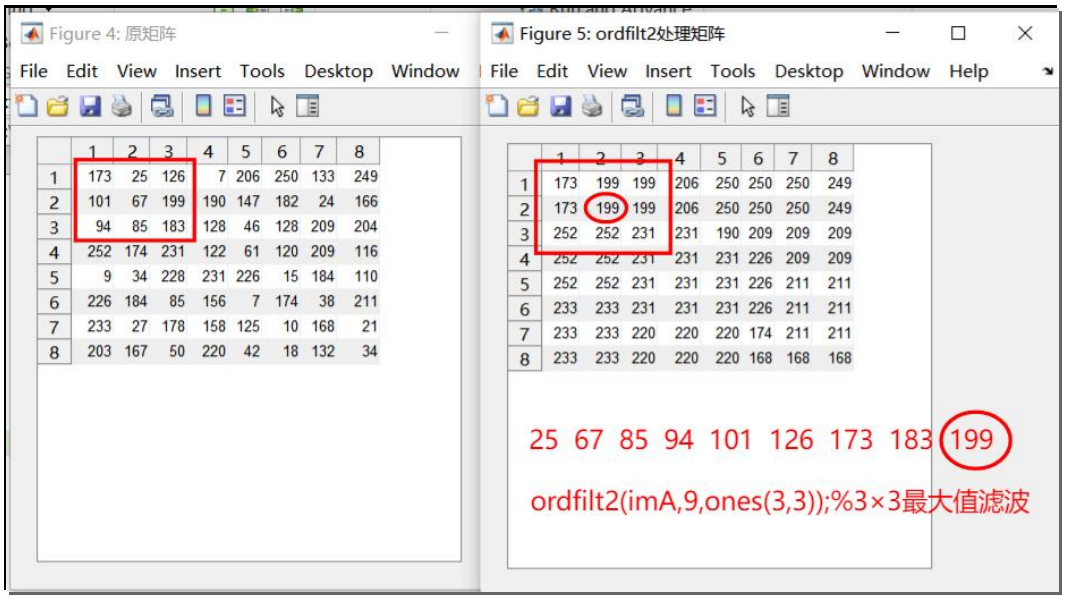


Figure 4: 原矩阵
File Edit View Insert Tools Desktop Window
1 2 3 4 5 6 7 8
1 173 25 126 7 206 250 133 249
2 101 67 199 190 147 182 24 166
3 94 85 183 128 46 128 209 204
4 252 174 231 122 61 120 209 116
5 9 34 228 231 226 15 184 110
6 226 184 85 156 7 174 38 211
7 233 27 178 158 125 10 168 21
8 203 167 50 220 42 18 132 34
Figure 5: ordfilt2处理矩阵
File Edit View Insert Tools Desktop Window Help
1 2 3 4 5 6 7 8
1 173 199 199 206 250 250 250 249
2 173 199 199 206 250 250 250 249
3 252 252 231 231 190 209 209 209
4 252 252 231 231 231 226 209 209
5 252 252 231 231 231 226 211 211
6 233 233 231 231 231 226 211 211
7 233 233 220 220 220 174 211 211
8 233 233 220 220 220 168 168 168
25 67 85 94 101 126 173 183 199
ordfilt2(imA,9,ones(3,3));%3×3最大值滤波

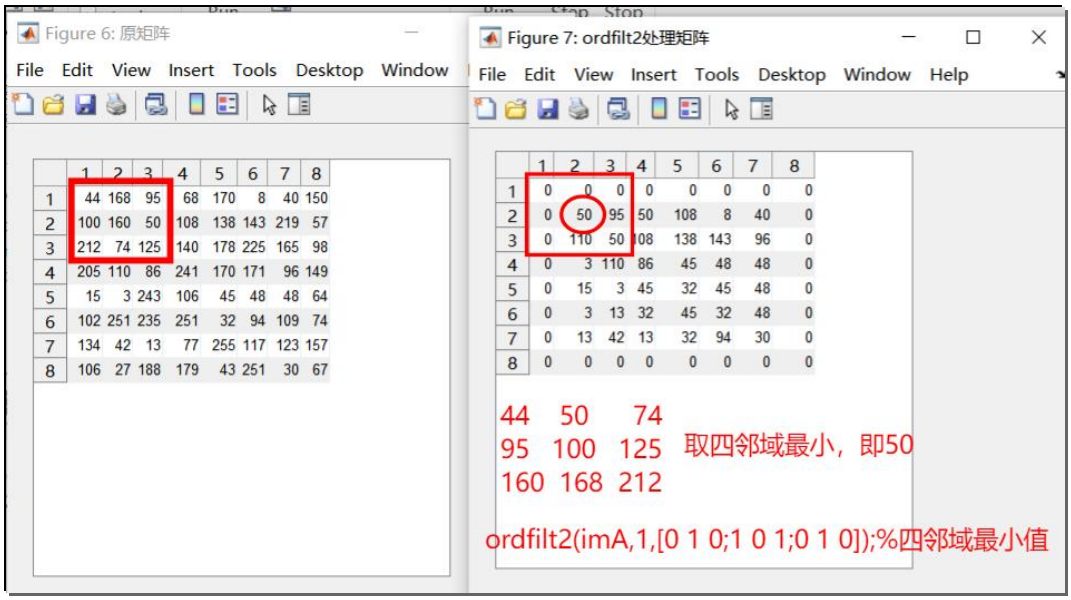
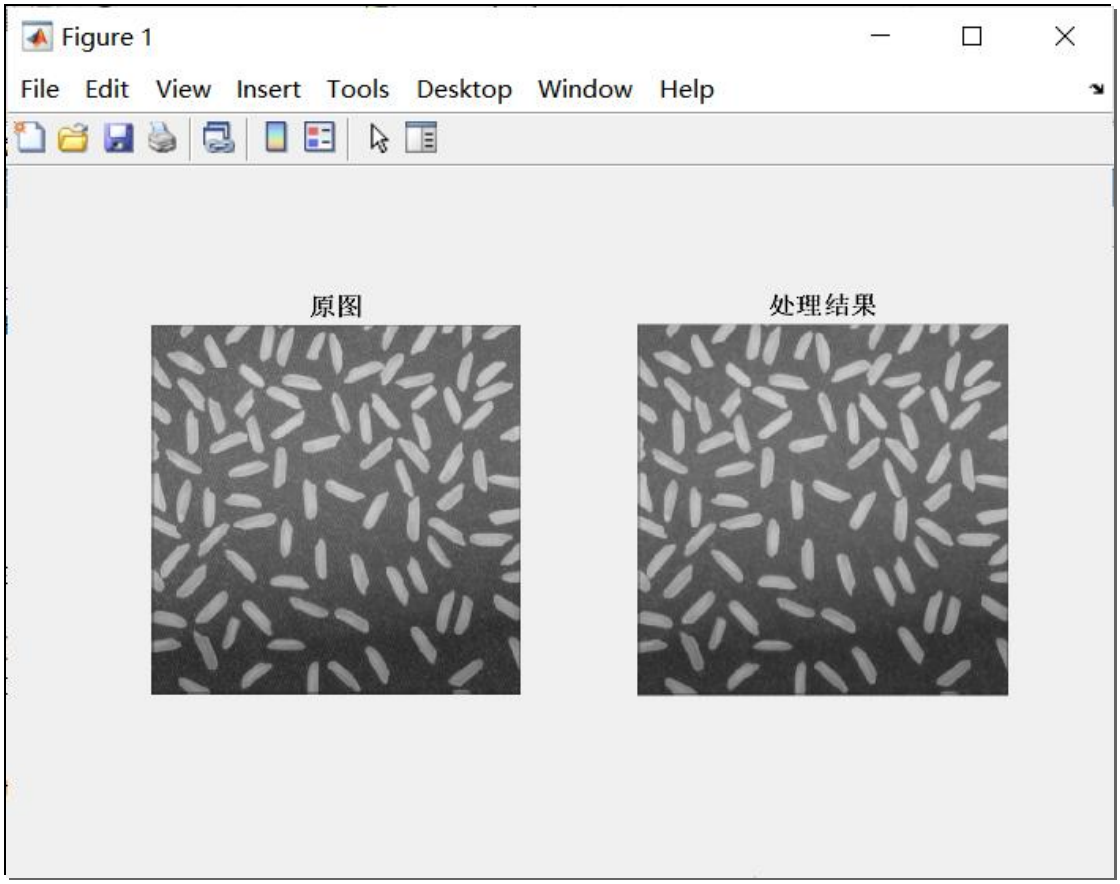
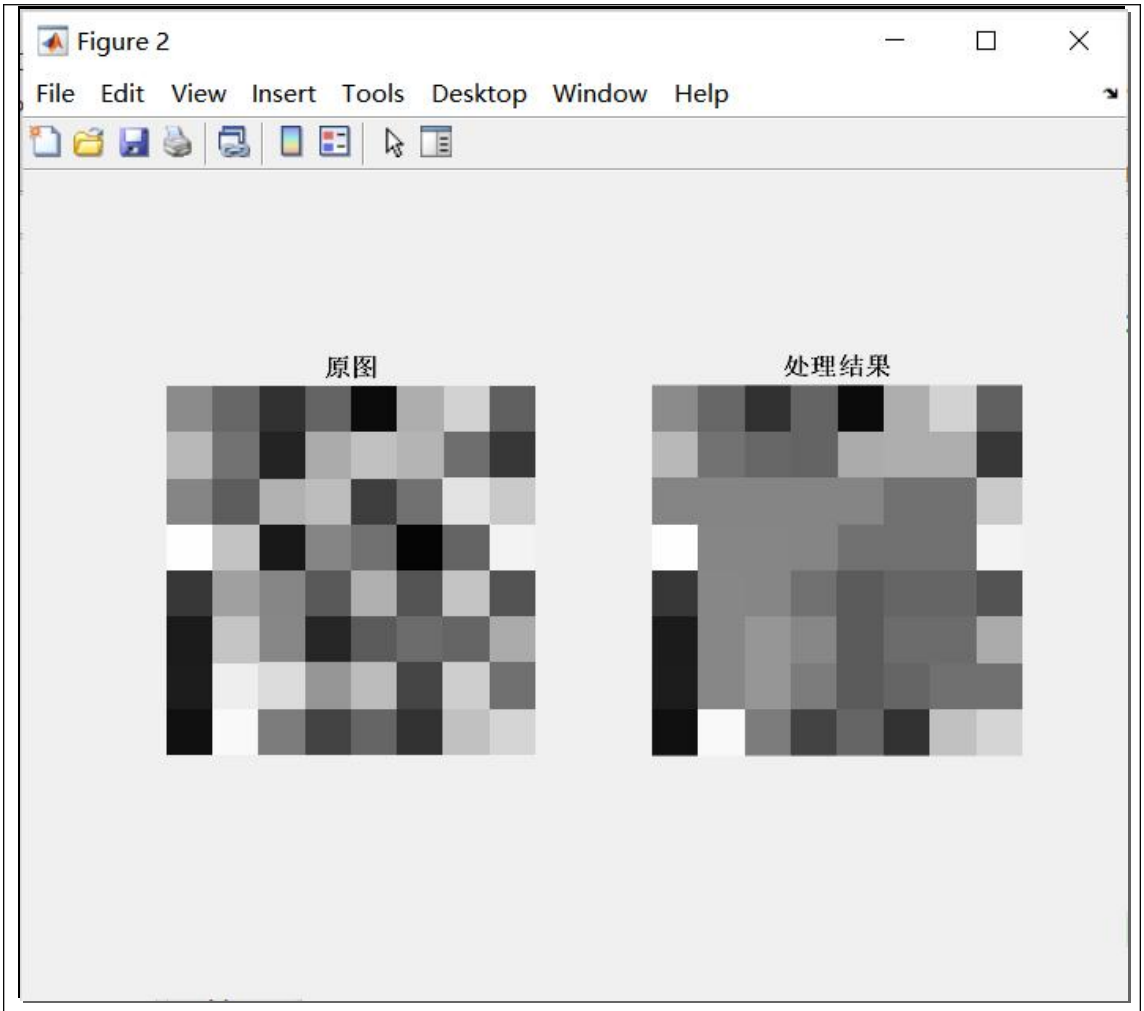


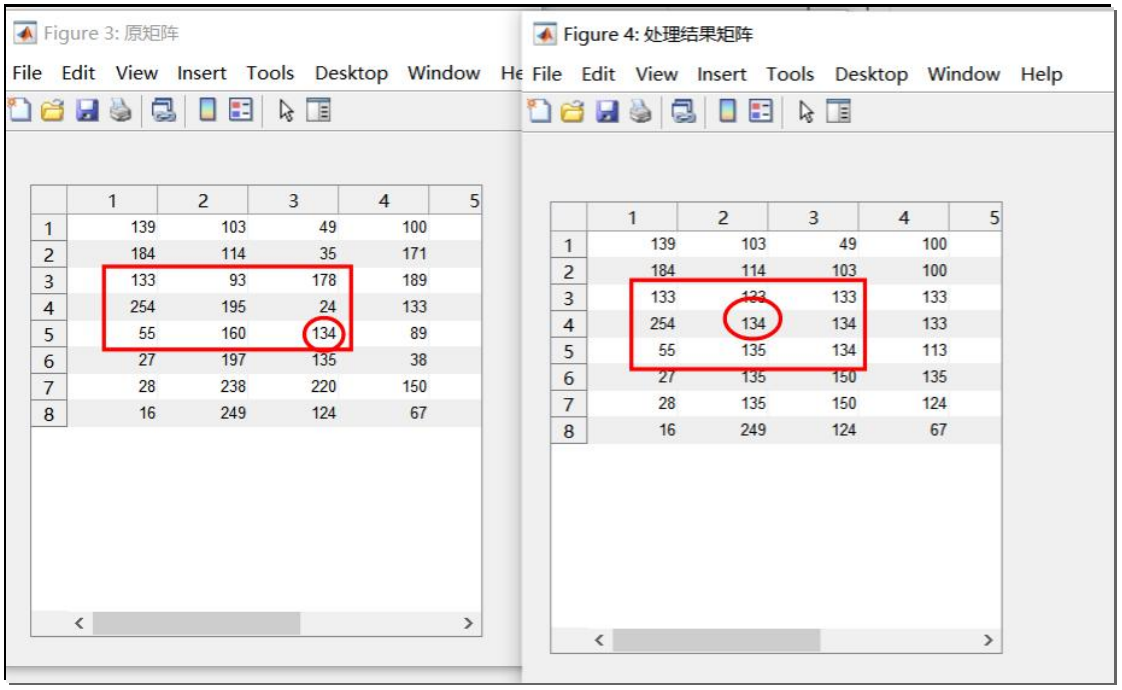
Figure 6: 原矩阵
File Edit View Insert Tools Desktop Window
1 2 3 4 5 6 7 8
1 44 168 95 68 170 8 40 150
2 100 160 50 108 138 143 219 57
3 212 74 125 140 178 225 165 98
4 205 110 86 241 170 171 96 149
5 15 3 243 106 45 48 48 64
6 102 251 235 251 32 94 109 74
7 134 42 13 77 255 117 123 157
8 106 27 188 179 43 251 30 67
Figure 7: ordfilt2处理矩阵
File Edit View Insert Tools Desktop Window Help
1 2 3 4 5 6 7 8
1 0 0 0 0 0 0 0 0
2 0 50 95 50 108 8 40 0
3 0 110 50 108 138 143 96 0
4 0 3 110 86 45 48 48 0
5 0 15 3 45 32 45 48 0
6 0 3 13 32 45 32 48 0
7 0 13 42 13 32 94 30 0
8 0 0 0 0 0 0 0 0
44 50 74
95 100 125 取四邻域最小，即50
160 168 212
ordfilt2(imA,1,[0 1 0;1 0 1;0 1 0]);%四邻域最小值

3. 自主设计实现 2_D 中值滤波算法，并对比函数 medfilt2 对图像及矩阵做中值滤波的算法效率。




```
imA=imread('C:\Users\Knight6\Pictures\matlabimage\imC.bmp');
figure,subplot(121),imshow(imA),title('原图');
[hang, lie]=size(imA);
imA1=double(imA);
imA2=imA1;

for i=1:hang-3+1
    for j=1:lie-3+1
        wdow=imA1(i:(i+3-1),j:(j+3-1));%取出元素
        wdow=wdow(:);
        med=median(wdow);%取中值
        imA2(i+(3-1)/2,j+(3-1)/2)=med;%为中心元素赋值
    end
end
subplot(122),imshow(uint8(imA2)),title('处理结果');
```





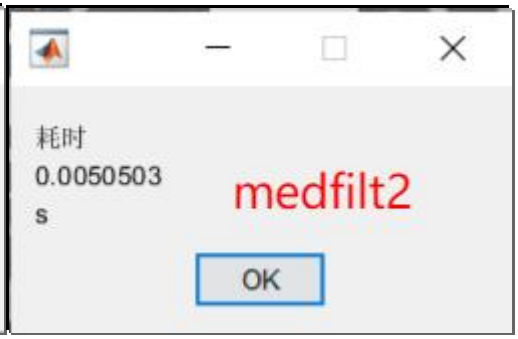
	1	2	3	4	5
1	139	103	49	100	
2	184	114	35	171	
3	133	93	178	189	
4	254	195	24	133	
5	55	160	134	89	
6	27	197	135	38	
7	28	238	220	150	
8	16	249	124	67	

	1	2	3	4	5
1	139	103	49	100	
2	184	114	103	100	
3	133	133	133	133	
4	254	134	134	133	
5	55	135	134	113	
6	27	135	150	135	
7	28	135	150	124	
8	16	249	124	67	

算法效率比较

矩阵：

图像：



如图可看出 **medfilt2** 的矩阵效率还是很高的，特别是像素点多的图像效率更为明显。

我也查找了更高效的中值滤波,原理就是去除最左侧一列，和增加最右侧一列。并尝试过，矩阵和图像的处理有所不同，**imhist** 统计图像的直方图，但是对于矩阵 **Mat2gray** 之后统计出来的和实际有偏差。之后，我又先将矩阵 **imA** 转为向量 **im=imA(:);**并尝试用 **tablulate** 统计矩阵的元素、个数及占比，以图获得跟直方图统计类似的效果，但是在移动矩阵的处理上，由于对移动处理不当，所以暂时搁置了这个算法。

高效中值滤波的方法及实现_51CTO 博客_中值滤波

三、高效中值滤波

从 2.5 可以看出，正是直方图本身的有序特性，使得求图像的中值非常简单，大大减少比较运算的次数，与图像的大小无关，最多比较 256 次。因此可以利用直方图，实现高效的图像中值滤波。

另外，在滑动窗口滤波中，对于相邻像素的两个滑动窗口而言，变化的内容只是丢掉了最左边的列，而增加了一个新的右边的列，窗口中间大面积的重叠区中的像素并没有变化，并不需要重新访问和处理。某参考文献中的高效中值滤波算法描述如下(假设窗口尺寸为 $N \times M$)：

1. 设置门限 $th = \frac{N \times M}{2} + 1$。 th即窗口所框住的范围矩阵的中间个数
2. 将窗口移动到一个新行的开始，建立窗口像素的直方图，通过直方图确定中值 med；记下亮度小于或 med 的像素数目到 $mNum$ 中。
3. 对于最左列的每个像素，设其灰度为 g_l，执行：（即去掉左侧）

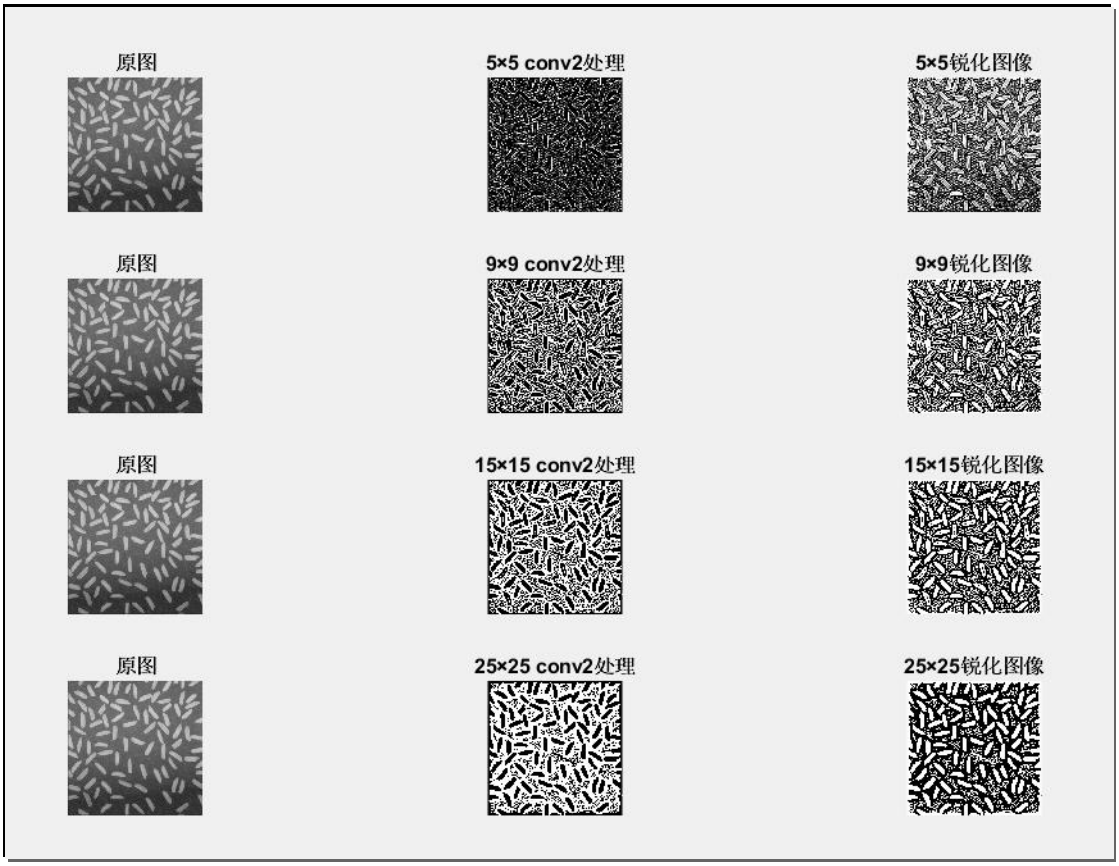
$H[g_l] = H[g_l] - 1$ H是直方图函数

若 $g_l \le med$，则 $mNum = mNum - 1$。

51CTO.com 技术博客 Blog

4. 用 conv2 函数进行线性锐化滤波，实现图像锐化滤波处理，采用 3×3 的拉普拉斯算子 w = [1 1 1;1 -8 1;1 1 1]处理。再分别采用 5×5，9×9，15×15 和 25×25 大小的拉普拉斯算子对图像及矩阵进行锐化滤波，观察其有何不同，要求在同一窗口中显示。

可以由下面图像及矩阵数值结果看出，随着模板的增大，模板中心和四周的对比度更明显，锐化更明显，但是图像的一些边线细节小点也被锐化丢失掉了。



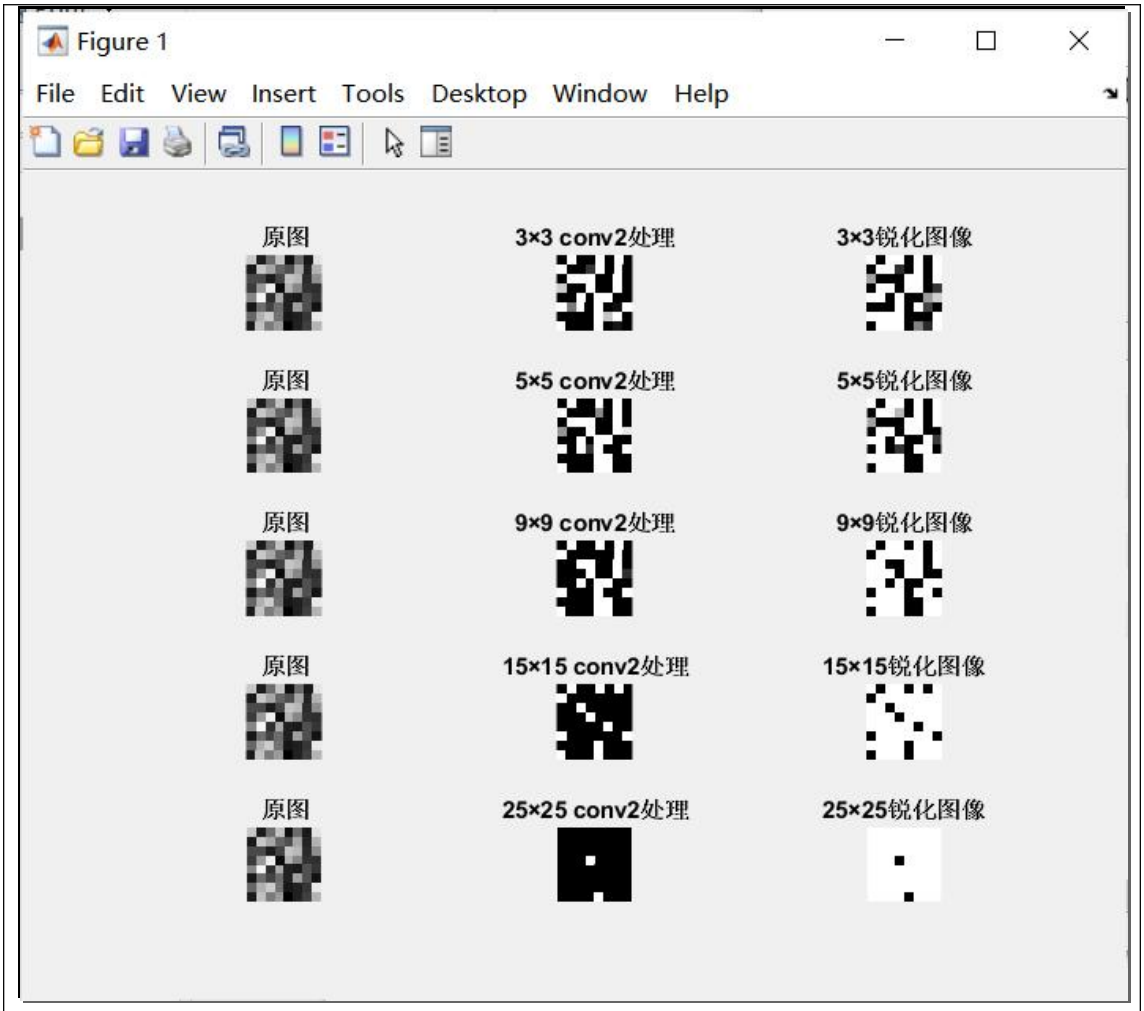


Figure 1
File Edit View Insert Tools Desktop Window Help
原图
3×3 conv2处理
3×3锐化图像
原图
5×5 conv2处理
5×5锐化图像
原图
9×9 conv2处理
9×9锐化图像
原图
15×15 conv2处理
15×15锐化图像
原图
25×25 conv2处理
25×25锐化图像

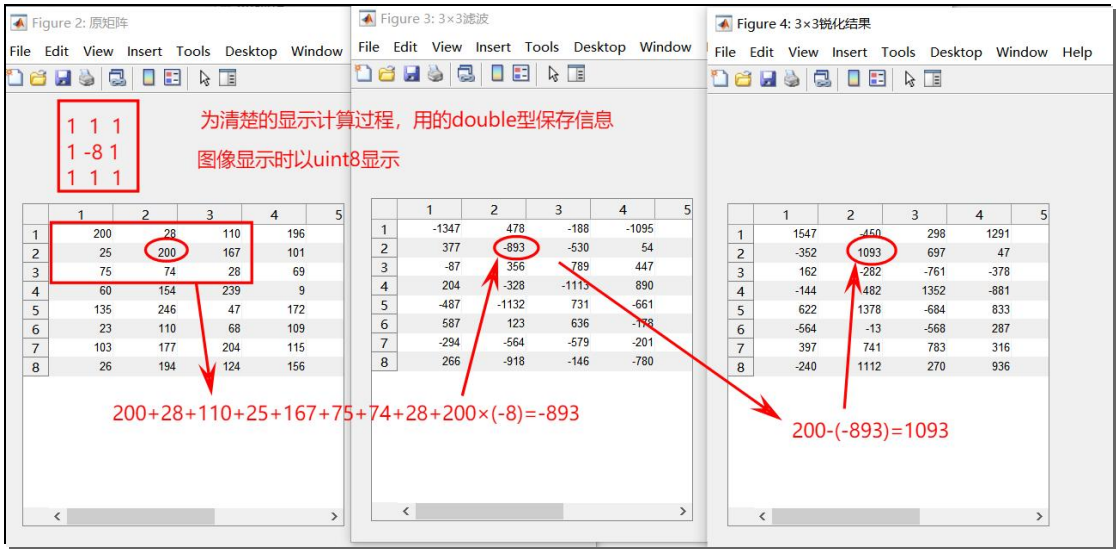
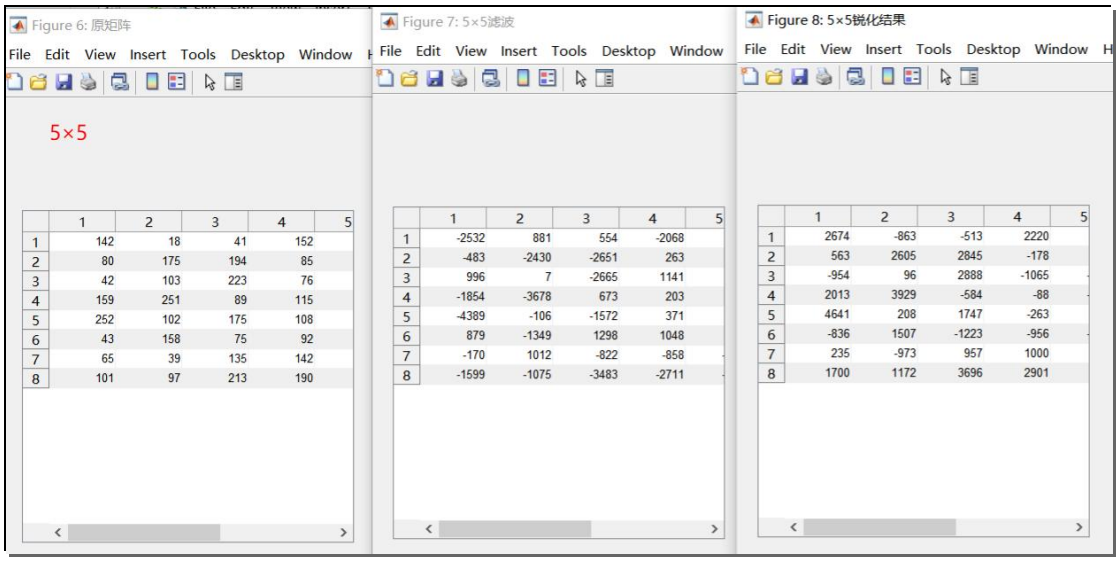


Figure 2: 原矩阵
Figure 3: 3×3滤波
Figure 4: 3×3锐化结果
1 1 1
1 -8 1
1 1 1
为清楚的显示计算过程，用的double型保存信息
图像显示时以uint8显示
200+28+110+25+167+75+74+28+200×(-8)=-893
200-(-893)=1093

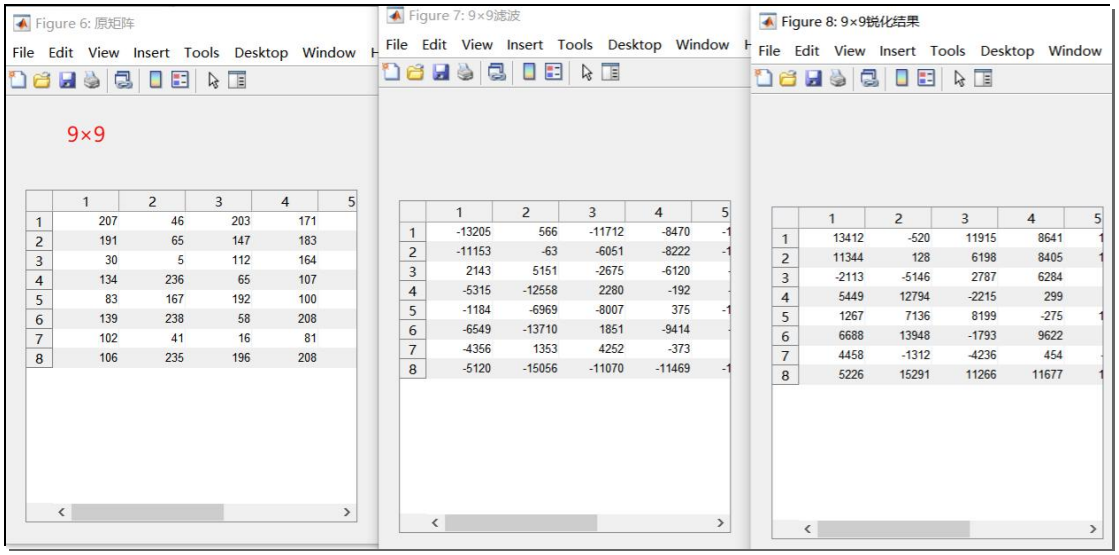
其他同理



	1	2	3	4	5
1	142	18	41	152	
2	80	175	194	85	
3	42	103	223	76	
4	159	251	89	115	
5	252	102	175	108	
6	43	158	75	92	
7	65	39	135	142	
8	101	97	213	190	

	1	2	3	4	5
1	-2532	881	554	-2068	
2	-483	-2430	-2651	263	
3	996	7	-2665	1141	
4	-1854	-3678	673	203	
5	-4389	-106	-1572	371	
6	879	-1349	1298	1048	
7	-170	1012	-822	-858	
8	-1599	-1075	-3483	-2711	

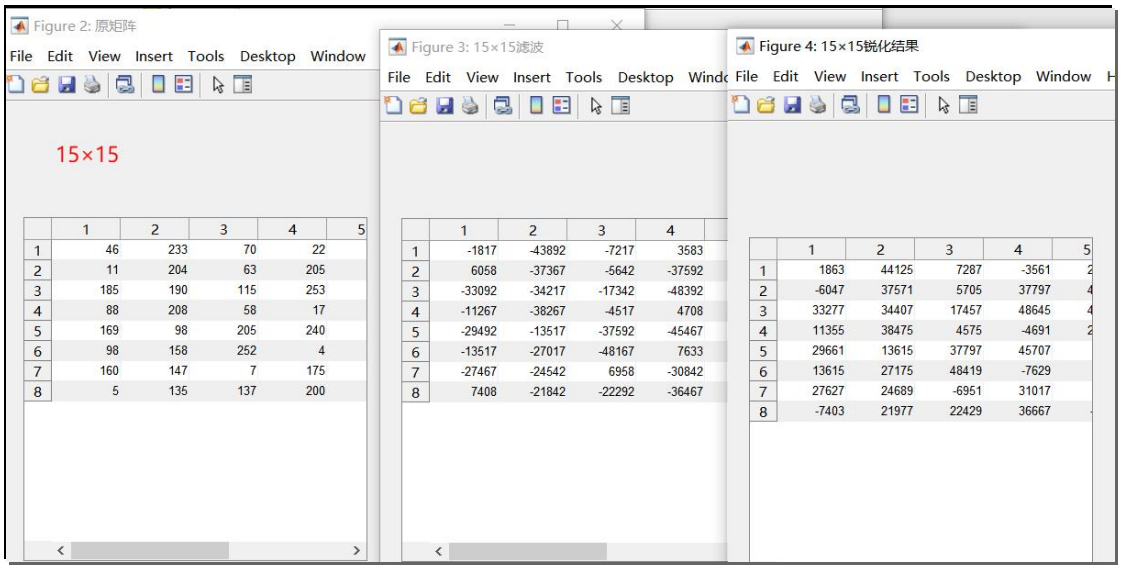
	1	2	3	4	5
1	2674	-863	-513	2220	
2	563	2605	2845	-178	
3	-954	96	2888	-1065	
4	2013	3929	-584	-88	
5	4641	208	1747	-263	
6	-836	1507	-1223	-956	
7	235	-973	957	1000	
8	1700	1172	3696	2901	



	1	2	3	4	5
1	207	46	203	171	
2	191	65	147	183	
3	30	5	112	164	
4	134	236	65	107	
5	83	167	192	100	
6	139	238	58	208	
7	102	41	16	81	
8	106	235	196	208	

	1	2	3	4	5
1	-13205	566	-11712	-8470	-1
2	-11153	-63	-6051	-8222	-1
3	2143	5151	-2675	-6120	
4	-5315	-12558	2280	-192	
5	-1184	-6969	-8007	375	-1
6	-6549	-13710	1851	-9414	
7	-4356	1353	4252	-373	
8	-5120	-15056	-11070	-11469	-1

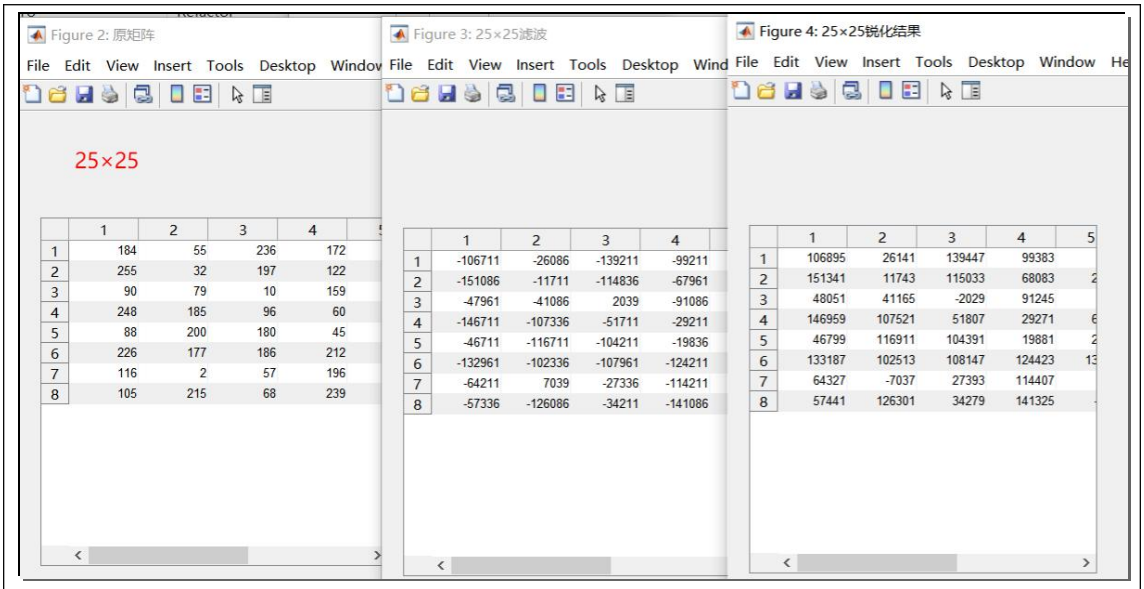
	1	2	3	4	5
1	13412	-520	11915	8641	1
2	11344	128	6198	8405	1
3	-2113	-5146	2787	6284	
4	5449	12794	-2215	299	
5	1267	7136	8199	-275	1
6	6688	13948	-1793	9622	
7	4458	-1312	-4236	454	
8	5226	15291	11266	11677	1



	1	2	3	4	5
1	46	233	70	22	
2	11	204	63	205	
3	185	190	115	253	
4	88	208	58	17	
5	169	98	205	240	
6	98	158	252	4	
7	160	147	7	175	
8	5	135	137	200	

	1	2	3	4
1	-1817	-43892	-7217	3583
2	6058	-37367	-5642	-37592
3	-33092	-34217	-17342	-48392
4	-11267	-38267	-4517	4708
5	-29492	-13517	-37592	-45467
6	-13517	-27017	-48167	7633
7	-27467	-24542	6958	-30842
8	7408	-21842	-22292	-36467

	1	2	3	4	5
1	1863	44125	7287	-3561	2
2	-6047	37571	5705	37797	4
3	33277	34407	17457	48645	4
4	11355	38475	4575	-4691	2
5	29661	13615	37797	45707	
6	13615	27175	48419	-7629	
7	27627	24689	-6951	31017	
8	-7403	21977	22429	36667	




```
%4. 用conv2函数进行线性锐化滤波，实现图像锐化滤波处理，采用 3×3 的拉普拉斯算子 w = [ 1 1 1;1 -8 1;1 1 1]处理。
% 再分别采用 5×5，9×9，15×15 和 25×25 大小的拉普拉斯算子对图像及矩阵进行锐化滤波，观察其有何不同，要求在同一窗口中显示。

%图像
imA=imread('C:\Users\Knight6\Pictures\matlabimage\imC.bmp');
%拉普拉斯算子
w3=ones(3);w3(2,2)=-8;%3×3拉普拉斯算子
w5=ones(5);w5(3,3)=-24;%5×5拉普拉斯算子
w9=ones(9);w9(5,5)=-80;%9×9拉普拉斯算子
w15=ones(15);w15(8,8)=-224;%15×15拉普拉斯算子
w25=ones(25);w25(13,13)=-624;%25×25拉普拉斯算子
%滤波图像
img_3=conv2(imA,w3,'same');
img_5=conv2(imA,w5,'same');
img_9=conv2(imA,w9,'same');
img_15=conv2(imA,w15,'same');
img_25=conv2(imA,w25,'same');
%锐化图像
img3=double(imA)-img_3;
img5=double(imA)-img_5;
img9=double(imA)-img_9;
img15=double(imA)-img_15;
img25=double(imA)-img_25;
%显示
figure,subplot(131),imshow(imA),title('原图');
subplot(132),imshow(uint8(img_3)),title('3×3 conv2处理');
subplot(133),imshow(uint8(img3)),title('3×3锐化图像');
subplot(534),imshow(imA),title('原图');
subplot(535),imshow(uint8(img_5)),title('5×5 conv2处理');
subplot(536),imshow(uint8(img5)),title('5×5锐化图像');
subplot(537),imshow(imA),title('原图');
subplot(538),imshow(uint8(img_9)),title('9×9 conv2处理');
subplot(539),imshow(uint8(img9)),title('9×9锐化图像');
subplot(5,3,10),imshow(imA),title('原图');
subplot(5,3,11),imshow(uint8(img_15)),title('15×15 conv2处理');
subplot(5,3,12),imshow(uint8(img15)),title('15×15锐化图像');
subplot(5,3,13),imshow(imA),title('原图');
subplot(5,3,14),imshow(uint8(img_25)),title('25×25 conv2处理');
subplot(5,3,15),imshow(uint8(img25)),title('25×25锐化图像');
```

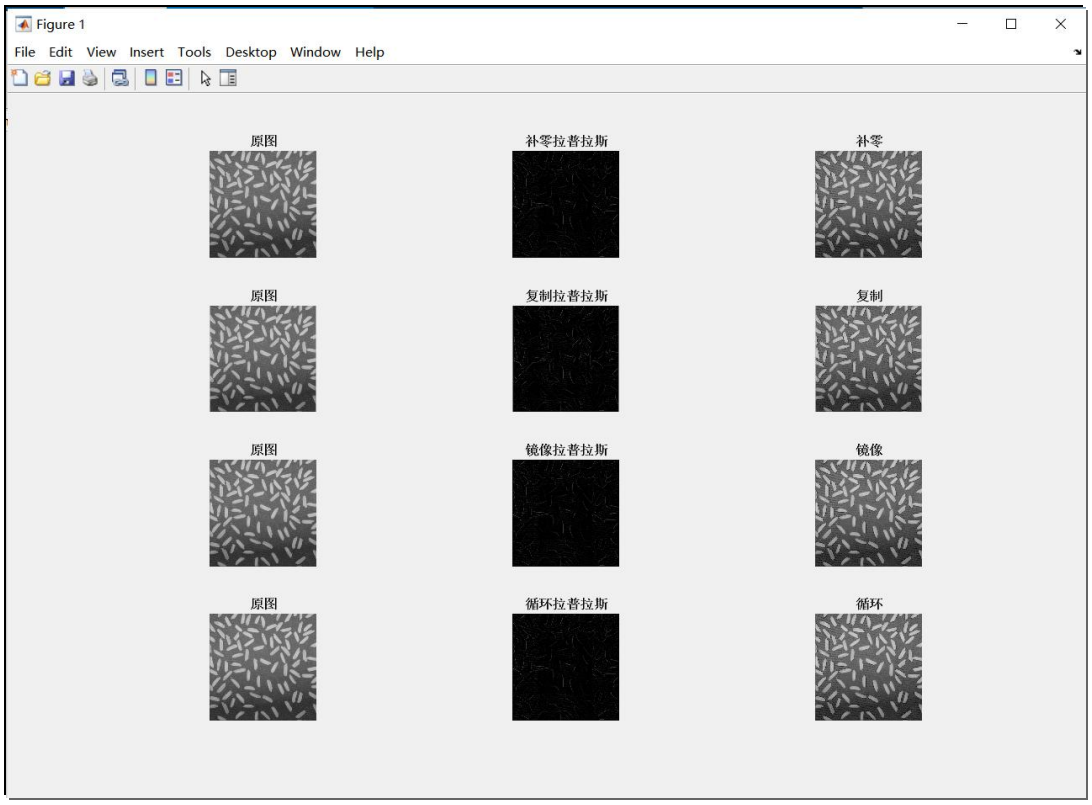
图像处理代码，矩阵类似

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'）对图像及矩阵进行拉普拉斯算子的锐化滤波处理，并显示处理后的结果（包括拉普拉斯图像及最终锐化结果图像）。

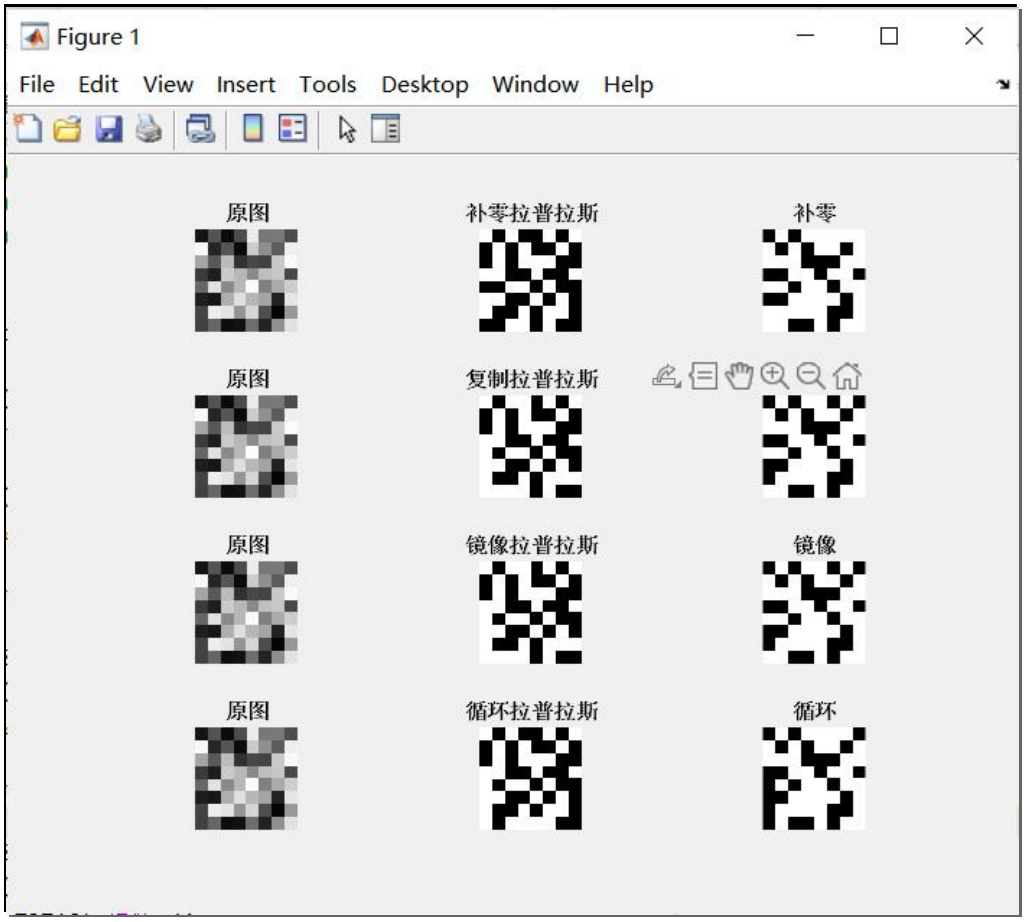
代码：

```
imA=imread('C:\Users\Knight6\Pictures\matlabimage\imC.bmp');
figure,subplot(431),imshow(imA),title('原图');

w=fspecial('laplacian',0);
img_=imfilter(imA,w,'corr',0);
img=imA-img_;
subplot(432);imshow(img_),title('补零拉普拉斯');
subplot(433);imshow(img),title('补零');

img_=imfilter(imA,w,'corr','replicate');
img=imA-img_;
subplot(434),imshow(imA),title('原图');
subplot(435);imshow(img_),title('复制拉普拉斯');
subplot(436);imshow(img),title('复制');

img_=imfilter(imA,w,'corr','symmetric');
img=imA-img_;
subplot(437),imshow(imA),title('原图');
subplot(438);imshow(img_),title('镜像拉普拉斯');
subplot(439);imshow(img),title('镜像');

img_=imfilter(imA,w,'corr','circular');
img=imA-img_;
subplot(4,3,10),imshow(imA),title('原图');
subplot(4,3,11);imshow(img_),title('循环拉普拉斯');
subplot(4,3,12);imshow(img),title('循环');
```

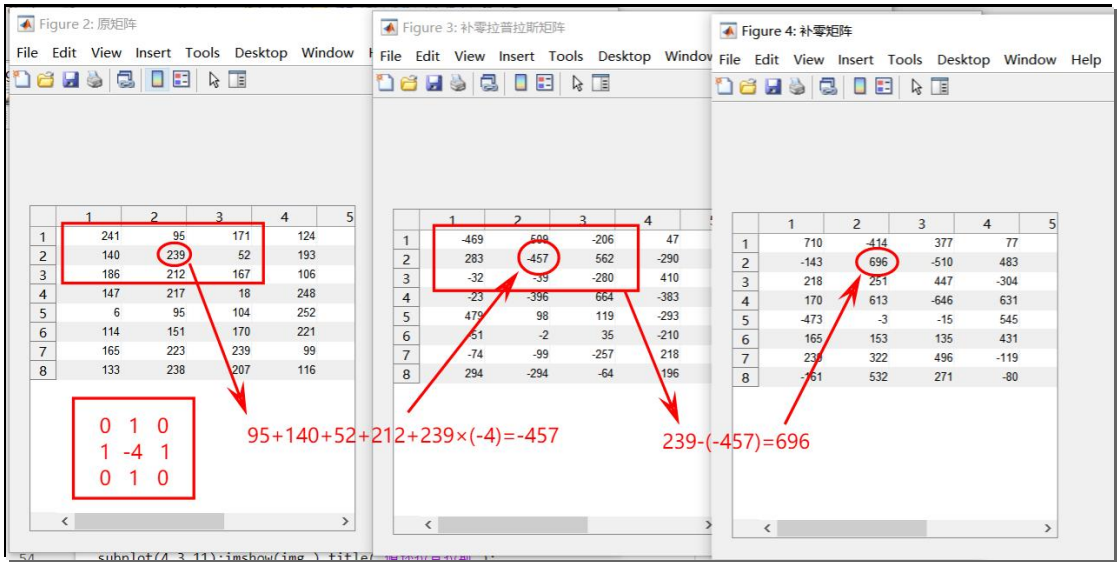
图像：



矩阵：



以边界补零为例



其他图像也类似，边界处理在实验 5 已经详细说明了，此处就不再赘述。值得注意的是 fspecial 产生的拉普拉斯算子是 4 邻域的拉普拉斯算子，而且 alpha 参数可以控制拉普拉斯算子的形状。而上面我使用的是 8 邻域的拉普拉斯算子。

```
>> fspecial('laplacian',0)

ans =

     0     1     0
     1    -4     1
     0     1     0
```

6. 查找资料，应用梯度算子，sobel 算子，prewitt 算子，roberts 算子分别对图像及矩阵进行锐化处理，并对比各种算子处理后的结果进行分析；（edge 函数）；

代码：

```
imA=imread('C:\Users\Knight6\Pictures\matlabimage\imC.bmp');

%sobel
figure(1),subplot(131),imshow(imA),title('原图');
w=edge(imA,'sobel');
subplot(132),imshow(w),title('sobel算子');
img=double(imA)-w;
subplot(133),imshow(uint8(img)),title('sobel算子结果');
%prewitt
figure(2),subplot(131),imshow(imA),title('原图');
w=edge(imA,'prewitt');
subplot(132),imshow(w),title('prewitt算子');
img=double(imA)-w;
subplot(133),imshow(uint8(img)),title('prewitt算子结果');
%roberts
figure(3),subplot(131),imshow(imA),title('原图');
w=edge(imA,'roberts');
subplot(132),imshow(w),title('roberts算子');
img=double(imA)-w;
subplot(133),imshow(uint8(img)),title('roberts算子结果');
```

图像：

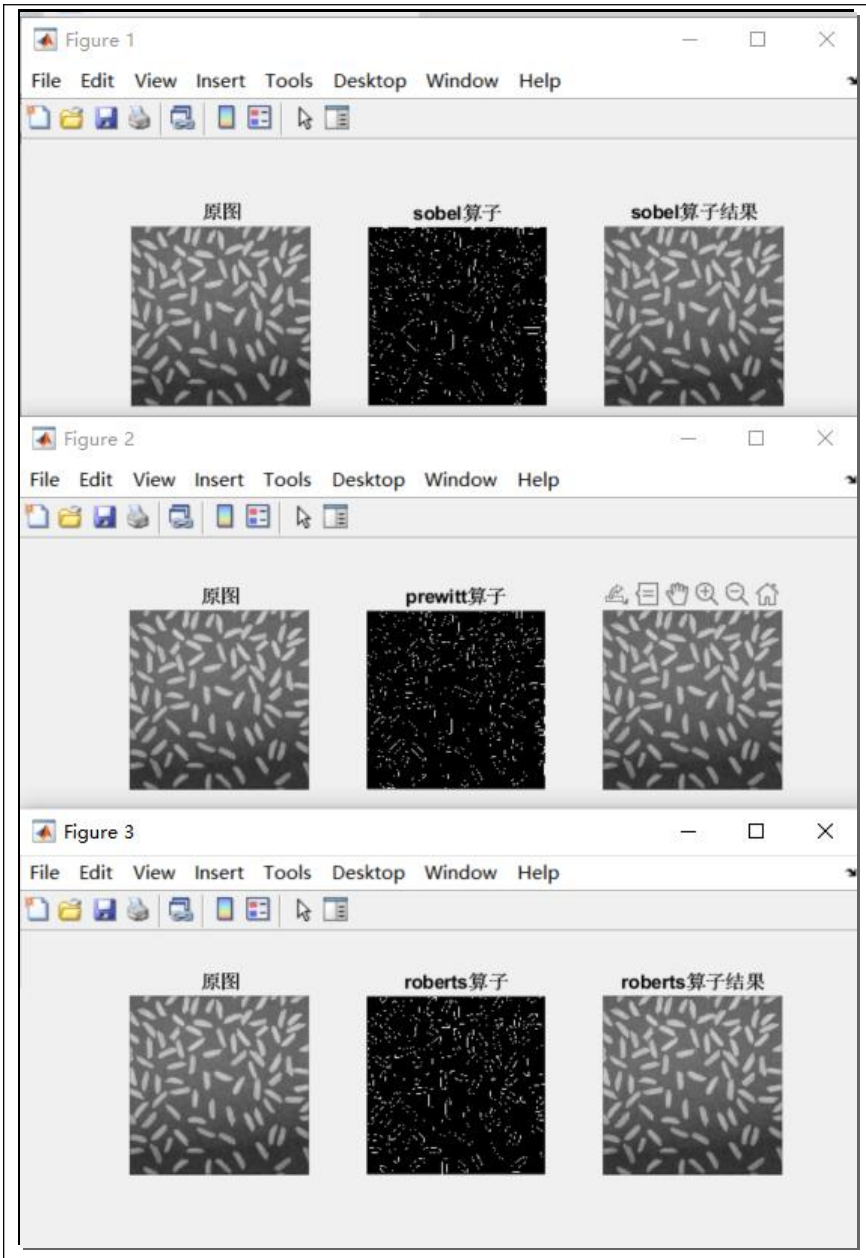
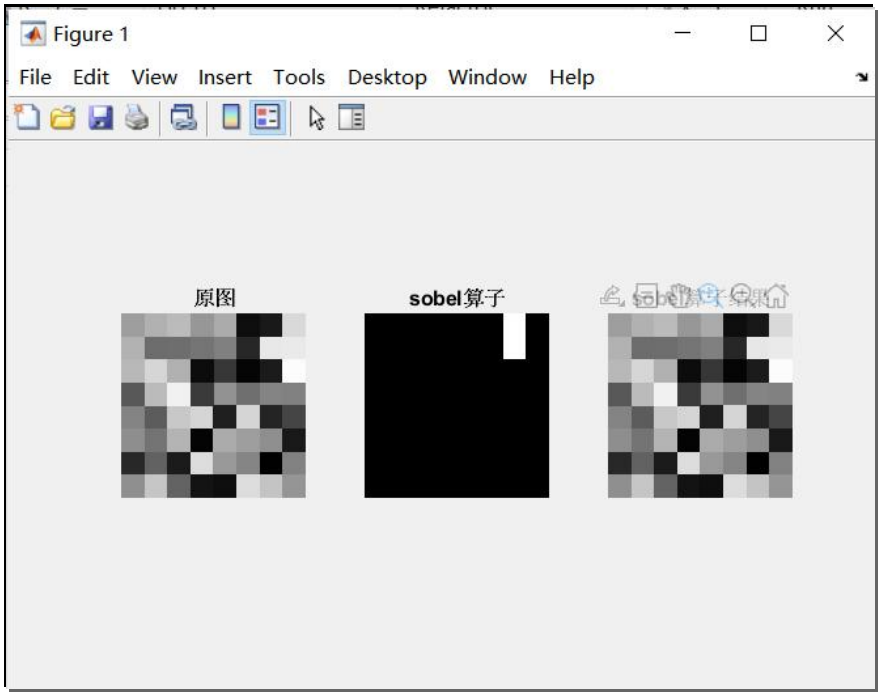
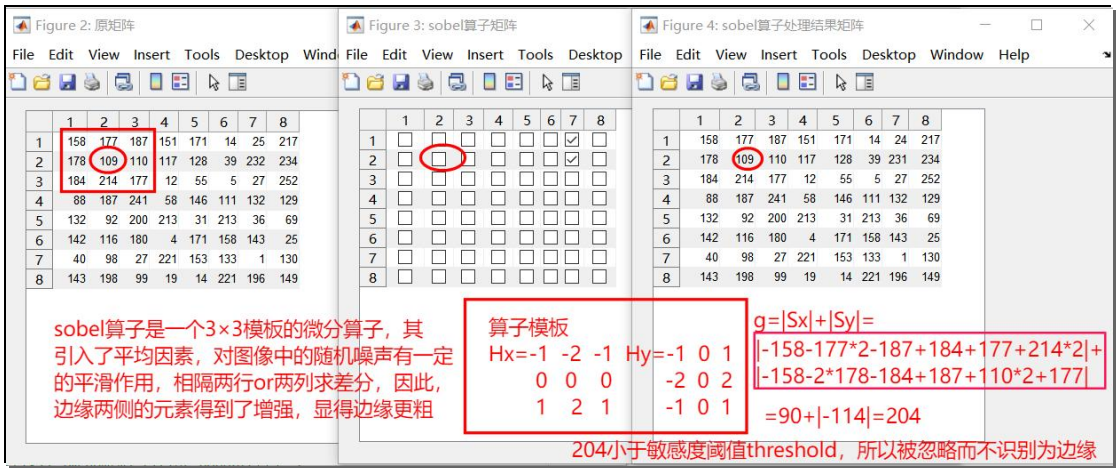


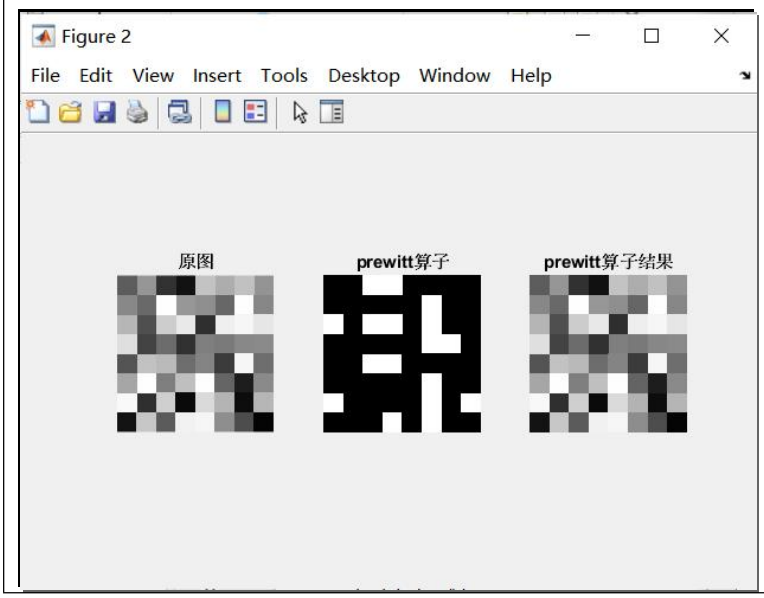
Figure 1
File Edit View Insert Tools Desktop Window Help
原图
sobel算子
sobel算子结果
Figure 2
File Edit View Insert Tools Desktop Window Help
原图
prewitt算子
Figure 3
File Edit View Insert Tools Desktop Window Help
原图
roberts算子
roberts算子结果

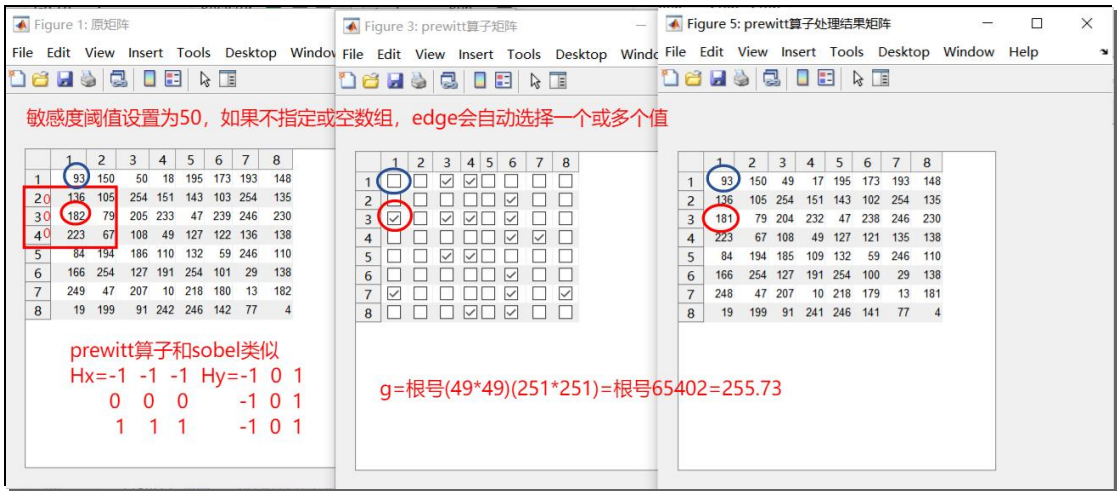
矩阵-sobel 算子：



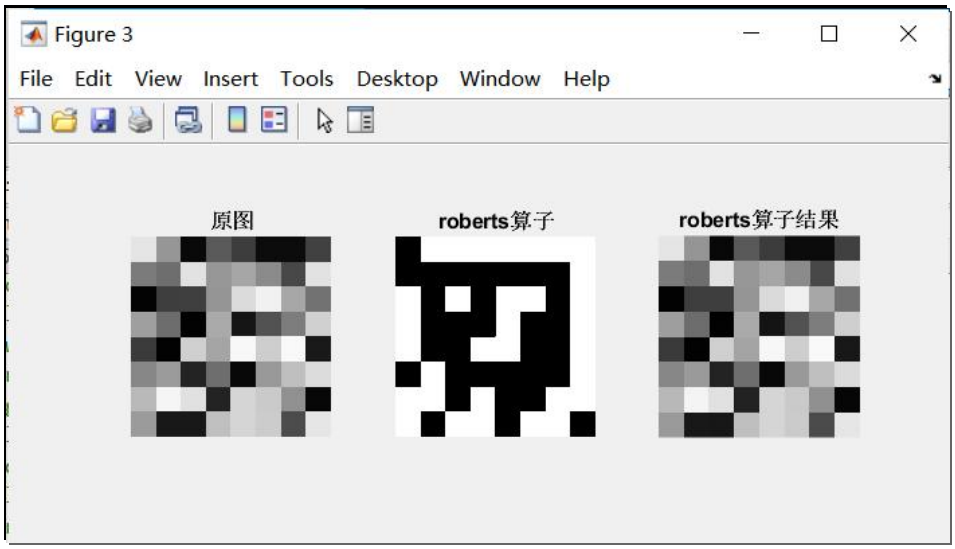


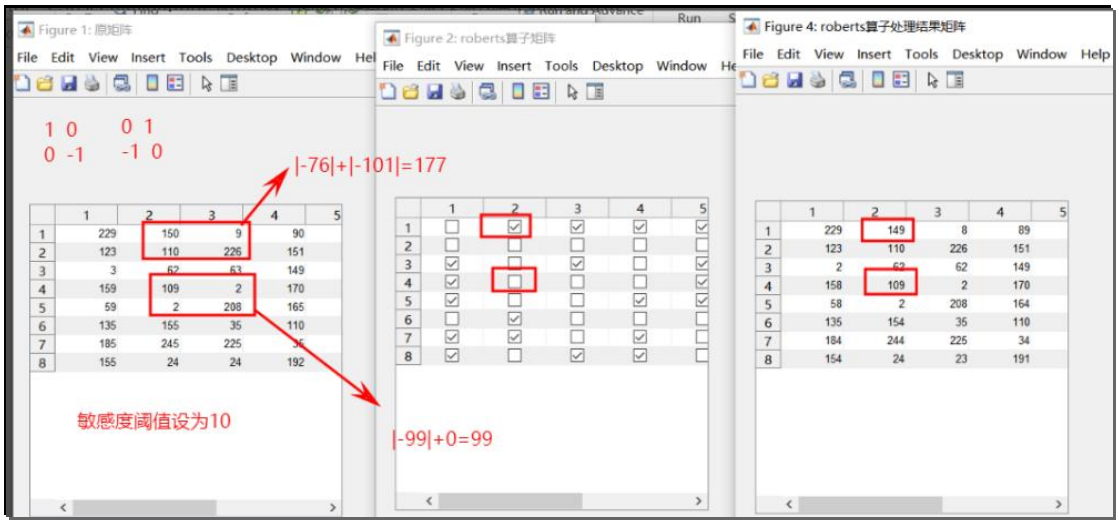
矩阵-prewitt 算子：





计算结果和预期相比还是有所偏差，查找了很多资料还是没有搞懂，为什么计算结果 g 明明大于敏感度阈值却没有被检测到边缘，不知道计算结果 g 是用的哪一种计算方法，也搞不懂 g 和敏感度阈值 threshold 的比较方式。另外，我看有的书上的锐化处理是原图和掩模图相加，应该也是一种增加对比度而锐化的方式。





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。