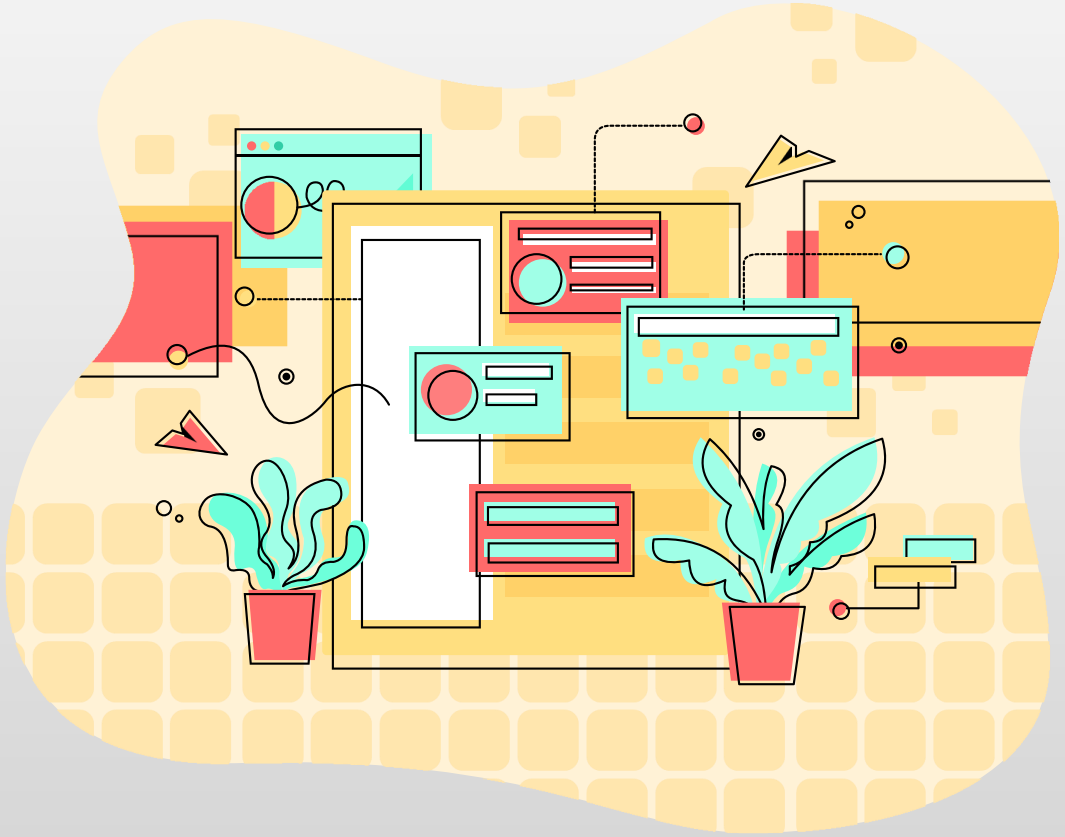


数字图像处理
实验二
——空间滤波器

实验内容

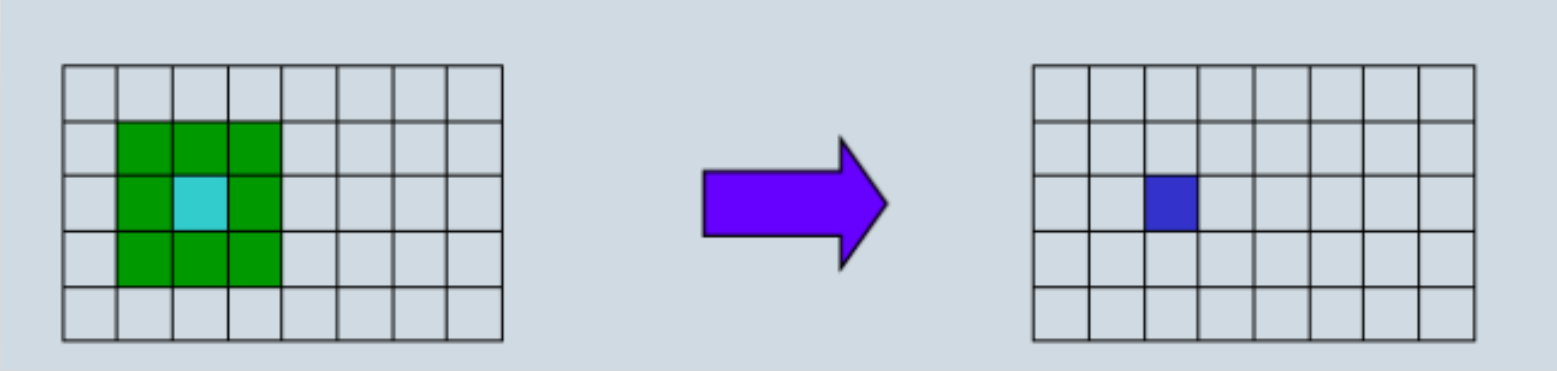
- 基础实验内容：
- 利用课件上关于均值滤波器的代码，分别对一幅图像实现3*3,5*5,7*7,9*9的均值滤波，并对实验结果进行分析；
- 利用课件上关于锐化滤波器的代码，分别对一幅图像实现3*3,5*5,7*7,9*9的sobel、prewitt滤波，Robert锐化滤波和拉普拉斯锐化滤波并对实验结果进行分析。
- 提高实验内容：
- 自编均值滤波器，对一幅图像实现填充后，并完成3*3,5*5,7*7,9*9,11*11的均值滤波。
- 自编锐化滤波器，对一幅图像实现填充后，并完成sobel、prewitt滤波，Robert锐化滤波和拉普拉斯3*3,5*5,7*7,9*9,11*11的锐化滤波

实验平台：Windows10 + Matlab

Part 1
均值滤波器

1.1 均值滤波器的原理：

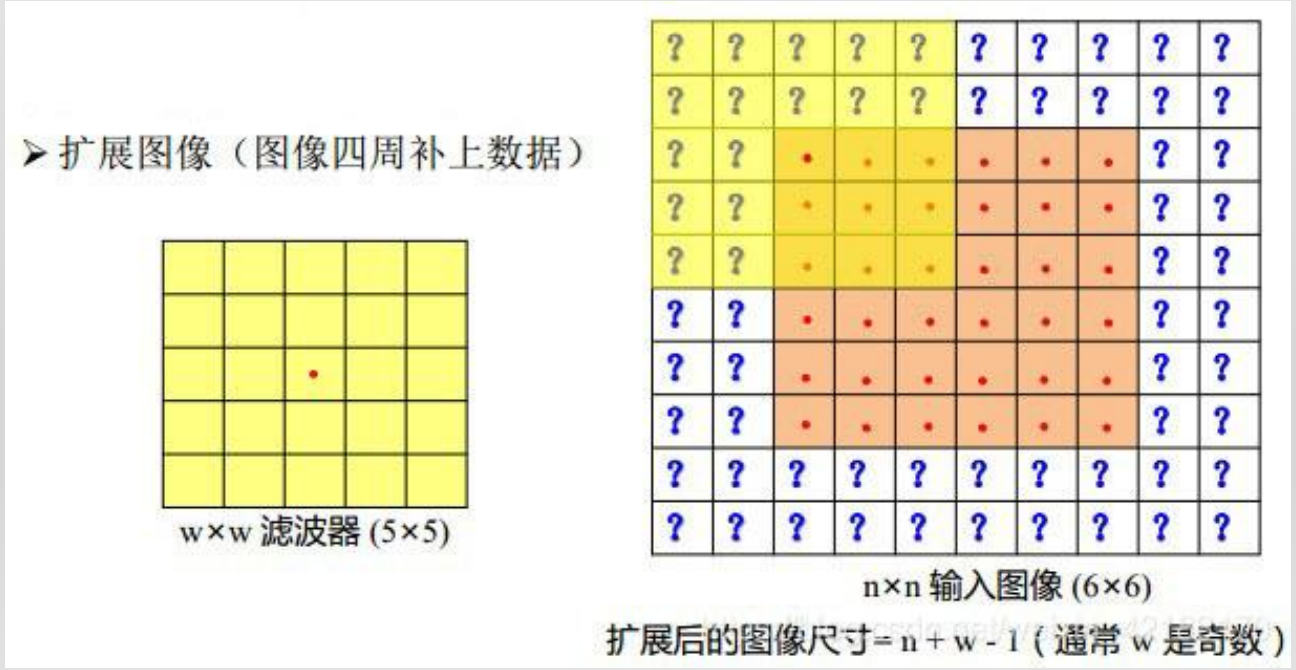
- 在图像上，对待处理的像素给定一个模板，模板包括这个像素周围的邻近像素（例如3*3的滤波器模板就包括中心像素和他的8领域像素）。将模板中的全体像素的均值来替代原来的中心像素值的方法就是均值滤波。
- 对图像的每一个像素，使用模板进行滤波，即可实现对图像的处理。



1.2 边界填充：

在实现均值滤波时，我们会遇到如何处理边界像素的问题。对于边界像素，模板无法覆盖，从而不能直接使用均值滤波器进行处理。我们可以选择忽略边界数据，但会丢失一部分图像内容。因此，为了合理处理图像边界，需要对图像进行边界填充，对图像进行拓展。对于灰度图像，有四种填充方式。

1.2.1 使用特定值n填充拓展（四周补上数据）

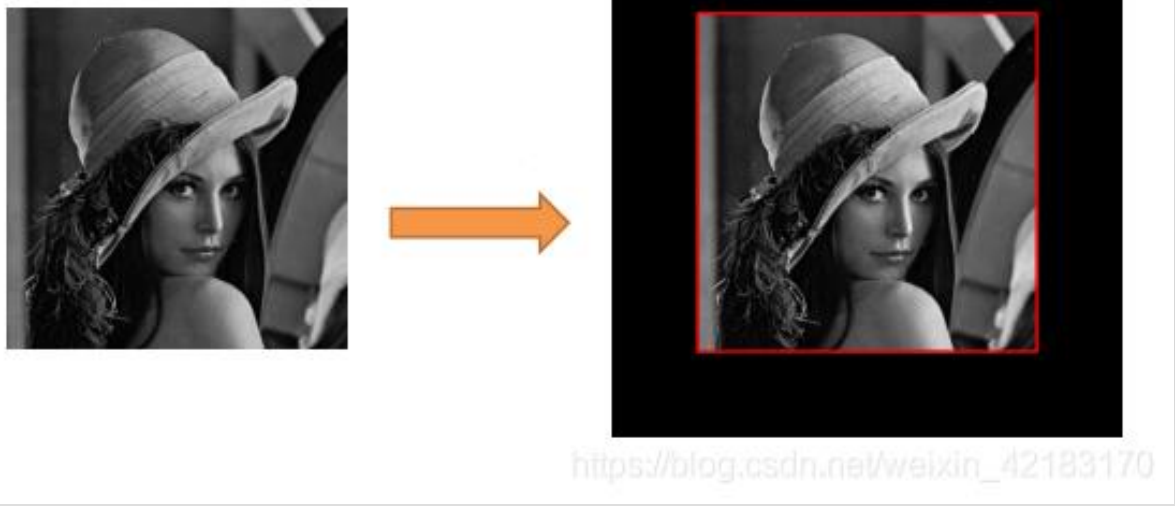


例如，使用n=0来填充。

101	100	99	99	98	98
100	99	98	98	97	98
100	99	98	98	97	99
100	99	99	98	98	100
100	100	99	98	98	101
100	100	101	102	103	97

使用n=0
填充

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	101	100	99	99	98	98	0	0
0	0	100	99	98	98	97	98	0	0
0	0	100	99	98	98	97	99	0	0
0	0	100	99	99	98	98	100	0	0
0	0	100	100	99	98	98	101	0	0
0	0	100	100	101	102	103	97	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



https://blog.csdn.net/weixin_42183170

1.2.2 复制图像边界

101	100	99	99	98	98
100	99	98	98	97	98
100	99	98	98	97	99
100	99	99	98	98	100
100	100	99	98	98	101
100	100	101	102	103	97

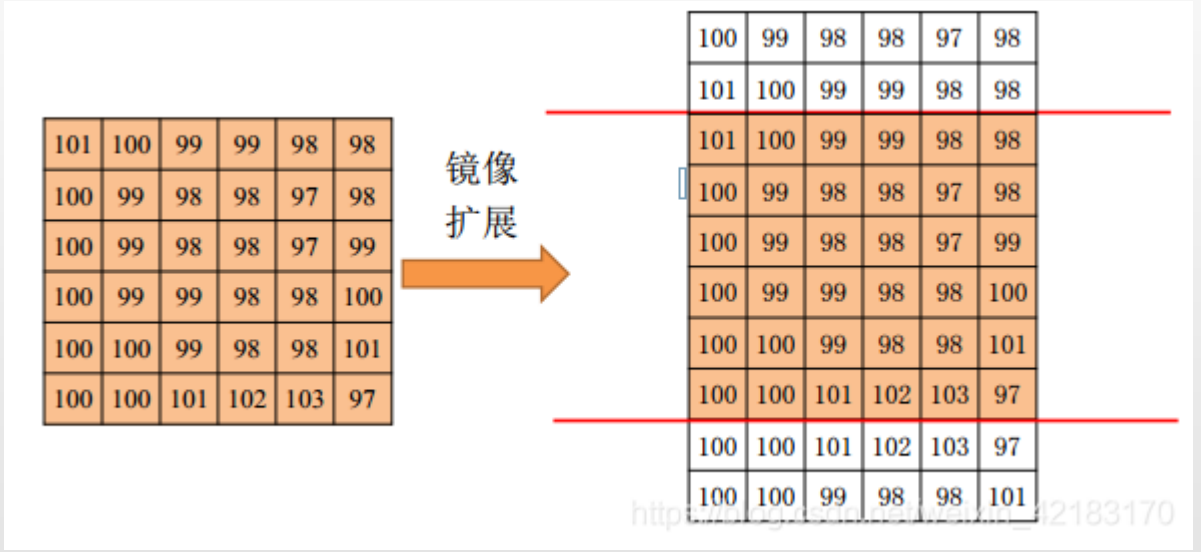
复制图像边界的值

101	101	101	100	99	99	98	98	98	98
101	101	101	100	99	99	98	98	98	98
101	101	101	100	99	99	98	98	98	98
100	100	100	99	98	98	97	98	98	98
100	100	100	99	98	98	97	99	99	99
100	100	100	99	99	98	98	100	100	100
100	100	100	100	99	98	98	101	101	101
100	100	100	100	101	102	103	97	97	97
100	100	100	100	101	102	103	97	97	97
100	100	100	100	101	102	103	97	97	97

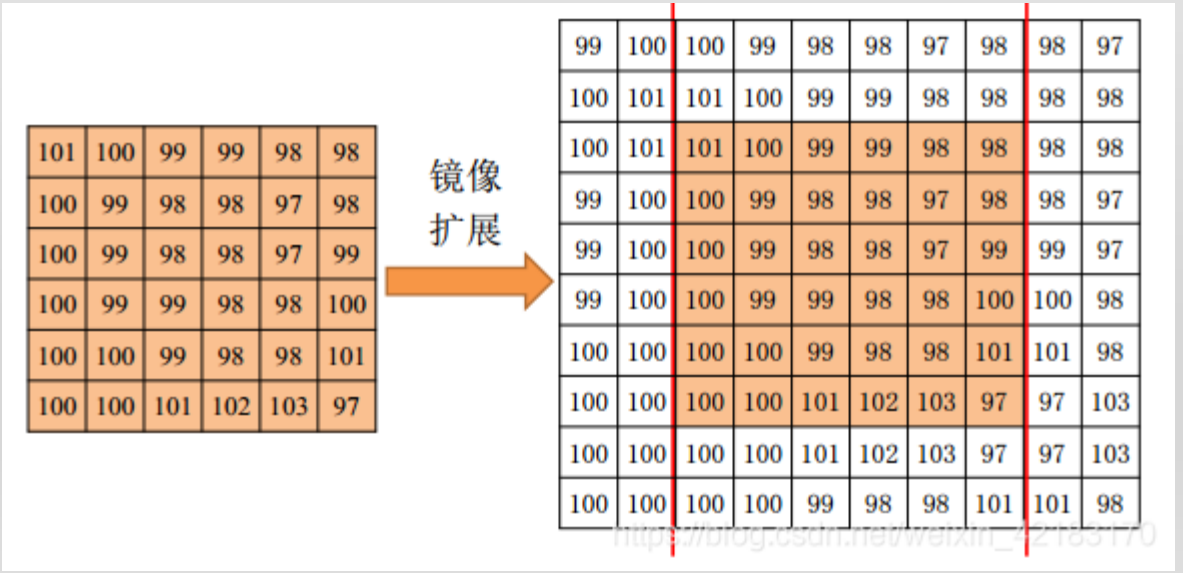
复制图像边界的值

1.2.3 镜像拓展

先在垂直方向镜像拓展



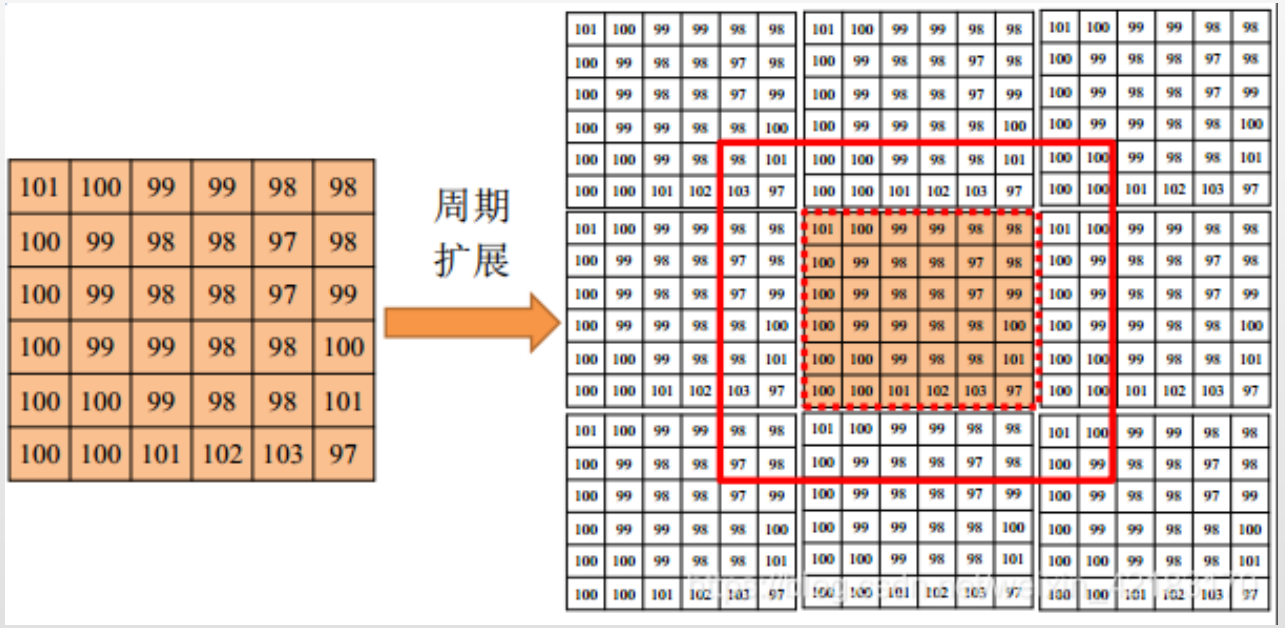
再在水平方向镜像拓展



1.2.4 周期拓展

101	100	99	99	98	98
100	99	98	98	97	98
100	99	98	98	97	99
100	99	99	98	98	100
100	100	99	98	98	101
100	100	101	102	103	97

周期
扩展



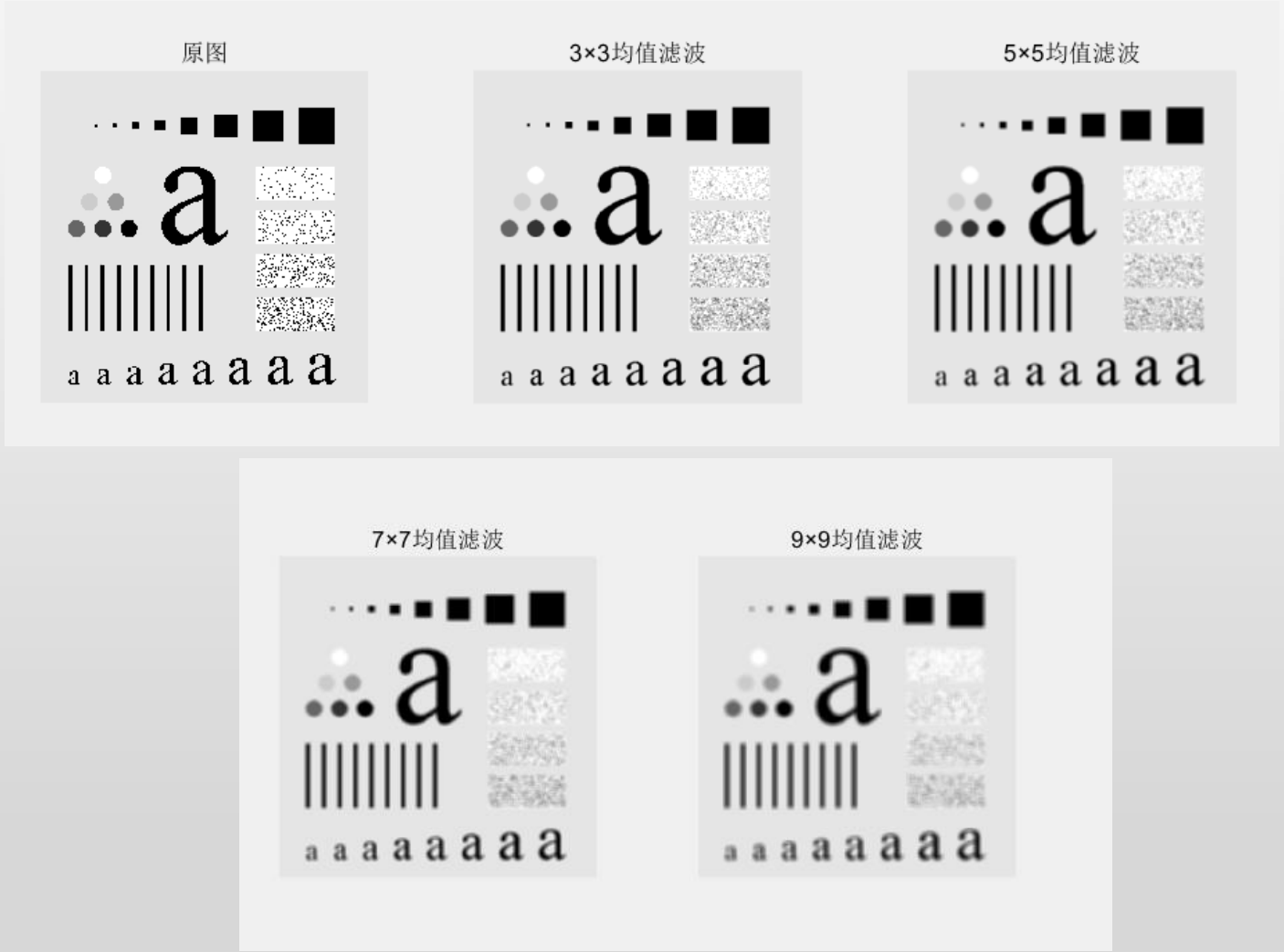
周期
扩展

对于边界填充，它的优缺点如下：

优点：与原图的尺寸相等。
缺点：若扩展方法不当，补在靠近图像边缘的部分会给处理后的图像带来不良影响，而且会随着滤波器尺寸的增加而增大。

调用Matlab相关函数完成均值滤波器

```
%% 使用matlab内置函数的均值滤波器
Image=imread('Letters-a.jpg');
w1=fspecial('average',3);      % 3×3均值滤波模板
w2=fspecial('average',5);      % 5×5均值滤波模板
w3=fspecial('average',7);
w4=fspecial('average',9);
result1=imfilter(Image,w1,'conv','replicate');
result2=imfilter(Image,w2,'conv','replicate');
result3=imfilter(Image,w3,'conv','replicate');
result4=imfilter(Image,w4,'conv','replicate');
figure
subplot(151),imshow(Image),title('原图')
subplot(152),imshow(uint8(result1)),title('3×3均值滤波')
subplot(153),imshow(uint8(result2)),title('5×5均值滤波')
subplot(154),imshow(uint8(result3)),title('7×7均值滤波')
subplot(155),imshow(uint8(result4)),title('9×9均值滤波')
```



可以发现，均值滤波后的图像内容边缘变模糊了，因为它对所有的点平等对对待，在对噪声进行分摊的同时，图像的边界也被分摊了。

均值滤波器的自编

自己写代码部分

Part 2
锐化滤波器

2.1 边缘分析：

对图像的锐化的目的是为了加强图像中景物的边缘和轮廓，因此需要分析图像边缘，检测出图像边缘，从而使边缘突出，实现图像锐化。

2.1.1 Sobel算子（3*3）

该算子包含两组3x3的矩阵，分别为横向及纵向，将之与图像作平面卷积，即可分别得出横向及纵向的亮度差分近似值。如果以A代表原始图像，Gx及Gy分别代表经横向及纵向边缘检测的图像灰度值，其公式如下：

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

2.1.2 Prewitt算子（3*3）

Prewitt算子与sobel算子思路类似，但模板系数不一样。

-1	-1	-1
0	0	0
1	1	1

水平

-1	0	1
-1	0	1
-1	0	1

垂直

2.1.3 Robert算子

-1	0
0	1

0	-1
1	0

Roberts 算子

2.1.4 拉普拉斯算子

拉普拉斯算子是二阶微分算子，其对应的滤波器模板为：

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

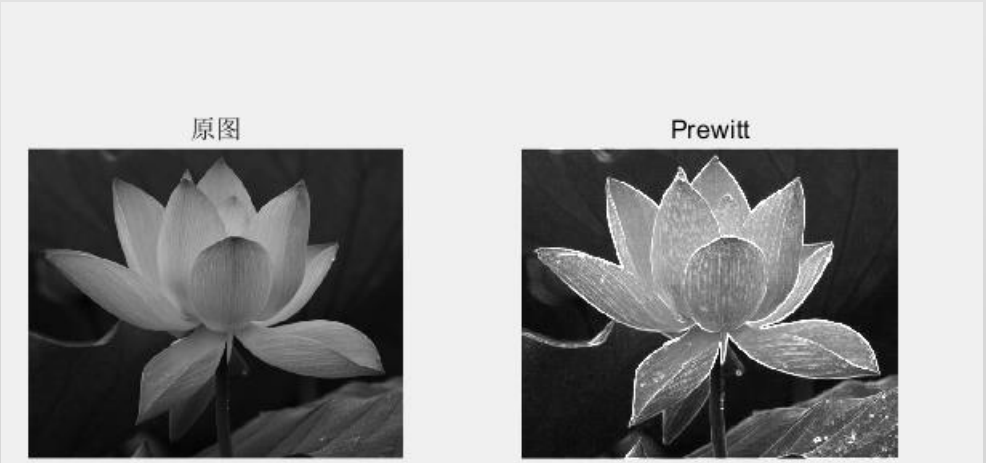
确定滤波器模板后，对图像内每一个像素点，使用对应的滤波器在模板范围内对中心的像素值进行计算和处理，即可实现图像的锐化。

2.2 调用Matlab相关函数完成锐化滤波器

调用imfilter函数，对图像进行锐化滤波

```
%% Sobel锐化
I = imread('lotus.bmp');
I=rgb2gray(I);
H1=[-1 -2 -1;0 0 0;1 2 1];H2=[-1 0 1;-2 0 2;-1 0 1];        %Sobel算子模板
R1= imfilter(I,H1);
R2= imfilter(I,H2);
edge=abs(R1)+abs(R2);          % 基于模板运算获取Sobel梯度图像
img=I+edge;     %锐化图像
figure,
subplot(121),imshow(I),title('原图')
subplot(122),imshow(img),title('Sobel')
```


```
%% Prewitt锐化
I = imread('lotus.bmp');
I=rgb2gray(I);
I=im2double(I);
H1=[-1 -1 -1;0 0 0;1 1 1];H2=[-1 0 1;-1 0 1;-1 0 1];        %Prewitt算子模板，两个模板
R1= imfilter(I,H1);
R2= imfilter(I,H2);
edge=abs(R1)+abs(R2);          % 基于模板运算获取Prewitt梯度图像
img=I+edge;
figure,
subplot(121),imshow(I),title('原图')
subplot(122),imshow(img),title('Prewitt')
```



```
%% Robert锐化
I = imread('lotus.bmp');
I=rgb2gray(I);
H1= [1 0;0 -1]; H2 = [0 1; -1 0];          %Robert算子模板
R1 = imfilter(I,H1);
R2 = imfilter(I,H2);
edge= abs(R1) + abs(R2);        % 基于模板运算获取Robert梯度图像
img = I + edge;
figure
subplot(121),imshow(I),title('原图')
subplot(122),imshow(img),title('Robert')
```


```
%% laplacian 锐化
I = imread('lotus.bmp');
I=rgb2gray(I);
H1=[0 -1 0;-1 5 -1;0 -1 0];        %拉普拉斯锐化模板
img=imfilter(I,H1);
figure
subplot(121),imshow(I),title('原图')
subplot(122),imshow(img),title('Laplacian')
```

2.3 自编相关函数完成锐化滤波器

自编代码
1
2
3
4
5
67
7
8
6
45
7676768