数字图像处理实验报告

学院： 计算机学院 专业：计算机科学与技术

姓名	//	学号	19//	班级	计科//
实验时间	2021//	指导教师	//	成绩	

实验目的

掌握空间滤波的基本思想，了解均值滤波和锐化滤波，使用 Matlab 实现均值滤波和锐化滤波。

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

Matlab R2019a

实验程序及结果

```
%% 使用matlab内置函数的均值滤波器,untitled.m
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')
```


```
%% 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')
```

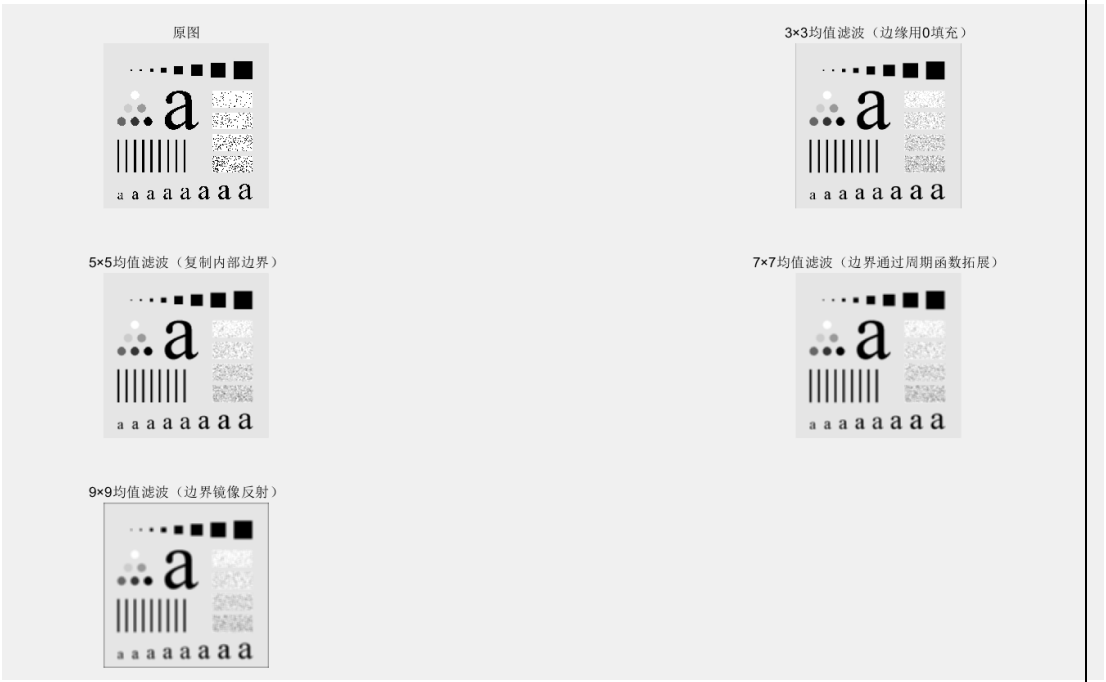

```
%% laplacian 锐化
I = imread('lotus.bmp');
```

```
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')
```


```
clear;
clc;
%自编函数实现均值滤波器  part1.m
I=imread('Letters-a.jpg');
figure,subplot(321),imshow(I)
title('原图');
img1=avg(I,3,0);
subplot(322),imshow(img1)
title('3×3均值滤波（边缘用0填充）')
img2=avg(I,5,'replicate');
subplot(323),imshow(img2);
title('5×5均值滤波（复制内部边界）')
img4=avg(I,7,'circular');
subplot(324),imshow(img4);
title('7×7均值滤波（边界通过周期函数拓展）')
img5=avg(I,9,'symmertic');
subplot(325),imshow(img5);
title('9×9均值滤波（边界镜像反射）')
function G=fill_border(img,n,mode)
function new_img=avg(img,n,mode)
%均值滤波  avg.m
%img是图像矩阵，n为模板大小(n×n),mode是边缘填充方式
if(nargin<3)
    mode=0;      %图像外部边界默认用0填充
end
[x y]=size(img);
```

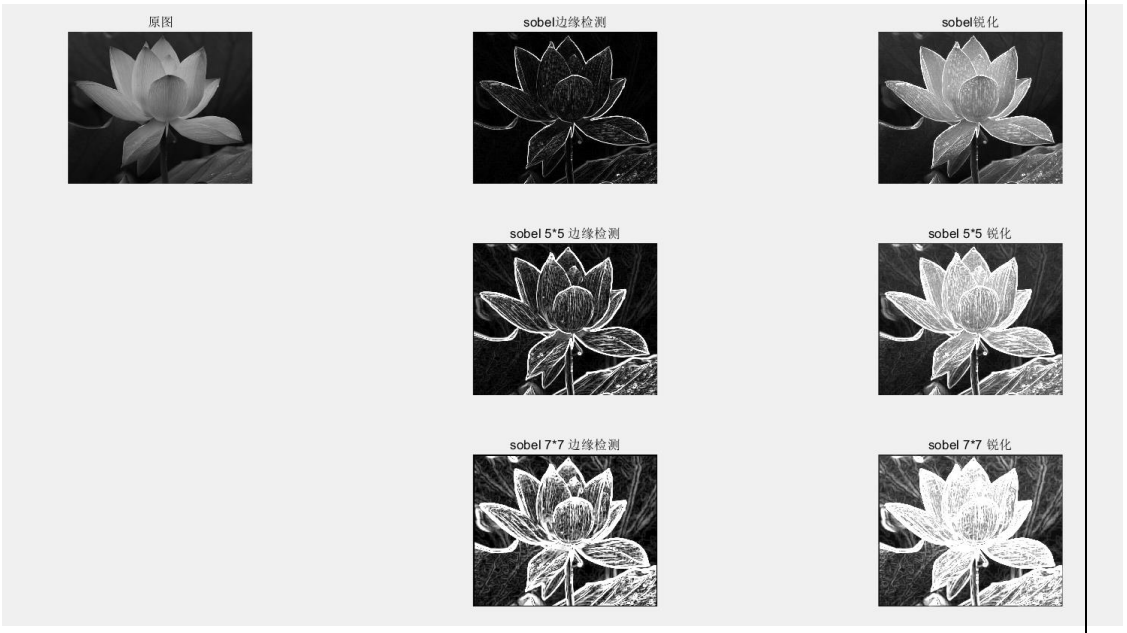
```
new_img=zeros(x,y);
filled_img=fill_border(img,n,mode);        %对图像进行边界填充
%求均值，填充到新的图像矩阵new_img
for i=1:x
    for j=1:y
        %每一个n×n的模板内求出均值，填入中心的灰度
      new_img(i,j)=sum(sum(filled_img(i:i+n-1,j:j+n-1)))/(n*n);
    end
end
new_img=uint8(new_img);
end

%对图像边缘进行填充  fill_border.m
[x y]=size(img);
G=zeros(x+n-1,y+n-1);    %对边界填充的图像矩阵大小为(x+n-1)*(y+n-1)
G((n-1)/2+1:x+((n-1)/2),(n-1)/2+1:y+((n-1)/2))=img;        %将原图的灰度值填入新的图像矩阵
%根据mode的类型对新的图像矩阵进行填充
if isa(mode,'double')==1
    %外部边界通过x来填充
    G(1:(n-1)/2,1:y+n-1)=mode;
    G(x+(n-1)/2+1:x+n-1,1:y+n-1)=mode;
    G(1:x+n-1,1:(n-1)/2)=mode;
    G(1:x+n-1,y+(n-1)/2+1:y+n-1)=mode;
elseif strcmp(mode,'replicate')==1
    %复制边界
    %先复制四个角的灰度值
    G(1:(n-1)/2,1:(n-1)/2)=img(1,1);
    G(x+(n-1)/2:x+n-1,y+(n-1)/2:y+n-1)=img(x,y);
    G(1:(n-1)/2,y+(n-1)/2:y+n-1)=img(1,y);
    G(x+(n-1)/2:x+n-1,1:(n-1)/2)=img(x,1);
    %其他边界复制
    for i=1:(n-1)/2
        G(i,(n-1)/2+1:y+(n-1)/2)=img(1,1:y);
        G(x+(n-1)/2+i,(n-1)/2+1:y+(n-1)/2)=img(x,1:y);
        G((n-1)/2+1:x+(n-1)/2,i)=img(1:x,1);
        G((n-1)/2+1:x+(n-1)/2,y+(n-1)/2+i)=img(1:x,y);
    end
elseif strcmp(mode,'symmetric')==1
    %沿边界进行镜像反射
    %先上下镜像反射
    for i=1:(n-1)/2
        G(i,(n-1)/2+1:y+(n-1)/2)=img((n-1)/2-i+1,1:y);
        G(x+(n-1)/2+i,(n-1)/2+1:y+(n-1)/2)=img(x-i+1,1:y);
    end
```

```
    %再左右反射
    for i=1:(n-1)/2
        G(1:x+n-1,i)=G(1:x+n-1,(n-1)/2-i+1);
        G(1:x+n-1,x+(n-1)/2+i)=G(1:x+n-1,y+(n-1)/2-i+1);
    end
elseif strcmp(mode,'circular')==1
    %作为二维周期函数进行扩展
    G1=zeros(3*x,3*y);
    G1(1:x,1:y)=img;
    G1(1:x,y+1:2*y)=img;
    G1(1:x,2*y+1:3*y)=img;
    G1(x+1:2*x,1:y)=img;
    G1(x+1:2*x,y+1:2*y)=img;
    G1(x+1:2*x,2*y+1:3*y)=img;
    G1(2*x+1:3*x,1:y)=img;
    G1(2*x+1:3*x,y+1:2*y)=img;
    G1(2*x+1:3*x,2*y+1:3*y)=img;
    G=G1(x-(n-1)/2+1:2*x+(n-1)/2,y-(n-1)/2+1:2*y+(n-1)/2);
end
end
```



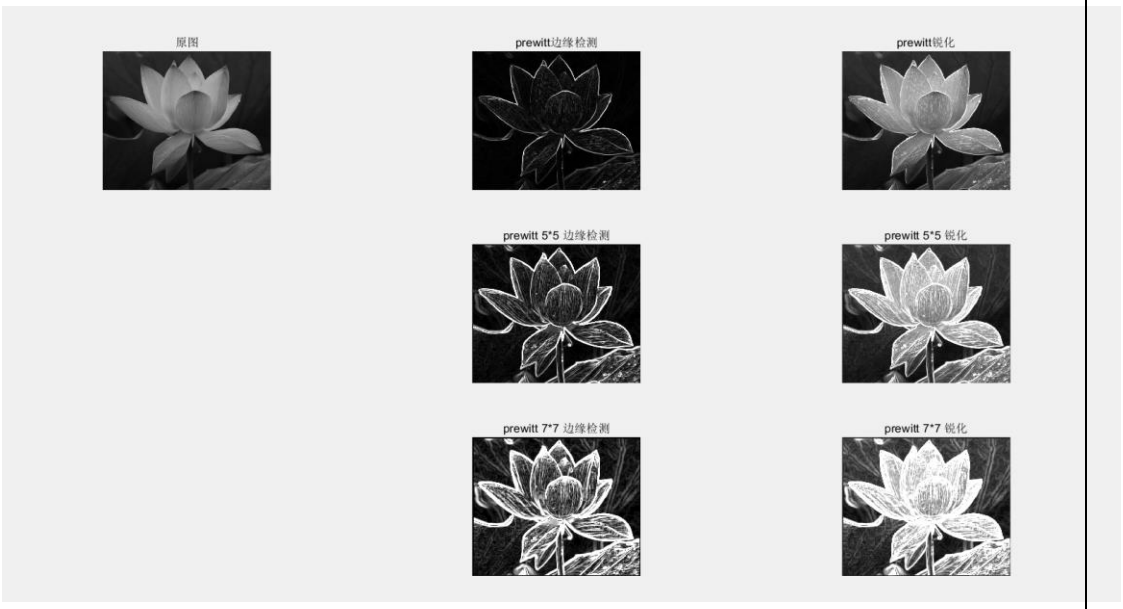

```
%自编函数实现锐化滤波  part2.m
img=imread('lotus.bmp');
img=rgb2gray(img);
%sobel
img1=sobel(img);
```

```
figure,subplot(331),imshow(img);
title('原图')
subplot(332),imshow(img1);
title('sobel边缘检测')
img2=img+img1;
subplot(333),imshow(img2);
title('sobel锐化')
img_sobel5=sobel(img,5);
subplot(335),imshow(img_sobel5);
title('sobel 5*5 边缘检测')
img5s=img+img_sobel5;
subplot(336),imshow(img5s)
title('sobel 5*5 锐化')
img_sobel7=sobel(img,7);
subplot(338),imshow(img_sobel7);
title('sobel 7*7 边缘检测')
img7s=img+img_sobel7;
subplot(339),imshow(img7s)
title('sobel 7*7 锐化')
```




```
%prewitt
img3=prewitt(img);
figure,subplot(331),imshow(img);
title('原图')
subplot(332),imshow(img3);
title('prewitt边缘检测')
img4=img+img3;
subplot(333),imshow(img4);
title('prewitt锐化')
```

```
img_pre5=prewitt(img,5);
subplot(335),imshow(img_pre5);
title('prewitt 5*5 边缘检测')
img5p=img+img_pre5;
subplot(336),imshow(img5p)
title('prewitt 5*5 锐化')
img_pre7=prewitt(img,7);
subplot(338),imshow(img_pre7);
title('prewitt 7*7 边缘检测')
img7p=img+img_pre7;
subplot(339),imshow(img7p)
title('prewitt 7*7 锐化')
```




```
%robert
figure,subplot(131),imshow(img);
title('原图')
img5=robert(img);
subplot(132),imshow(img5);
title('robert边缘检测')
img6=img+img5;
subplot(133),imshow(img6);
title('robert锐化')
```


```
%拉普拉斯
figure,subplot(131),imshow(img);
title('原图')
img7=laplacian(img);
subplot(132),imshow(img7);
title('拉普拉斯缘检测')
img8=img+img7;
subplot(133),imshow(img8);
title('拉普拉斯锐化')
```


```
%sobel边缘检测  sobel.m
function new_img=sobel(img,n)
if nargin<2
    n=3;            %默认模板为3*3
end
d=(n-1)/2;
[x y]=size(img);
img=double(img);
new_img=zeros(x,y);
%Sobel算子模板
if n==3
    sobel_x = [-1,-2,-1;0,0,0;1,2,1];
    sobel_y = [-1,0,1;-2,0,2;-1,0,1];
elseif n==5
    sobel_x=[-1,-1,-2,-1,-1;-1,-1,-2,-1,-1;0,0,0,0,0;1,1,2,1,1;1,1,2,1,1];
```

```
    sobel_y=[-1,-1,0,1,1;-1,-1,0,1,1;-2,-2,0,2,2;-1,-1,0,1,1;-1,-1,0,1,1];
elseif n==7

sobel_x=[-1,-1,-1,-3,-1,-1,-1;-1,-1,-1,-3,-1,-1,-1;-1,-1,-1,-3,-1,-1,-1;0,0,0,0,0,0,0;1,1,1,3,1,1,1;1,1,
1,3,1,1,1;1,1,1,3,1,1,1];

sobel_y=[-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-3,-3,-3,0,3,3,3;-1,-1,-1,0,1,1,1;-1,-1,-1,0
,1,1,1;-1,-1,-1,0,1,1,1];
end
for i=1+d:x-d
    for j=1+d:y-d
        t=img(i-d:i+d,j-d:j+d);
        G_x=sobel_x.* t;
        G_x=abs(sum(G_x(:)));
        G_y=sobel_y.*t;
        G_y=abs(sum(G_y(:)));
        sobel_xy=sqrt(G_x^2+G_y^2);
        new_img(i+1,j+1)=sobel_xy;
    end
end
new_img=uint8(new_img);
end
%prewitt边缘检测  prewitt.m
function new_img=prewitt(img,n)
if nargin<2
    n=3;            %默认模板为3*3
end
d=(n-1)/2;
[x y]=size(img);
img=double(img);
new_img=zeros(x,y);
%Prewitt算子模板
if n==3
    prewitt_x = [-1,-1,-1;0,0,0;1,1,1];
    prewitt_y = [-1,0,1;-1,0,1;-1,0,1];
elseif n==5
    prewitt_x=[-1,-1,-1,-1,-1;-1,-1,-1,-1,-1;0,0,0,0,0;1,1,1,1,1;1,1,1,1,1];
    prewitt_y=[-1,-1,0,1,1;-1,-1,0,1,1;-1,-1,0,1,1;-1,-1,0,1,1;-1,-1,0,1,1];
elseif n==7

prewitt_x=[-1,-1,-1,-1,-1,-1,-1;-1,-1,-1,-1,-1,-1,-1;-1,-1,-1,-1,-1,-1,-1;0,0,0,0,0,0,0;1,1,1,1,1,1,1;1,
1,1,1,1,1,1;1,1,1,1,1,1,1];

prewitt_y=[-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-1,-1,-1,0,1,1,1;-1,-1,-1
```

```
,0,1,1,1;-1,-1,-1,0,1,1,1];
end
for i=1+d:x-d
    for j=1+d:y-d
        t=img(i-d:i+d,j-d:j+d);
        G_x=prewitt_x.* t;
        G_x=abs(sum(G_x(:)));
        G_y=prewitt_y.*t;
        G_y=abs(sum(G_y(:)));
        prewitt_xy=sqrt(G_x^2+G_y^2);
        new_img(i+1,j+1)=prewitt_xy;
    end
end
new_img=uint8(new_img);
end
%robert边缘检测  robert.m
function new_img=robert(img)
[x y]=size(img);
img=double(img);
new_img=zeros(x,y);
H1= [1 0;0 -1]; H2 = [0 1; -1 0];          %Robert算子模板
for i=1:x-1
    for j=1:y-1
        t=img(i:i+1,j:j+1);
        G_x=H1.*t;
        G_x=abs(sum(G_x(:)));
        G_y=H2.*t;
        G_y=abs(sum(G_y(:)));
        G_xy=sqrt(G_x^2+G_y^2);
        new_img(i+1,j+1)=G_xy;
    end
end
new_img=uint8(new_img);
end
%拉普拉斯边缘检测  laplacian.m
function new_img=laplacian(img)
[x y]=size(img);
img=double(img);
new_img=zeros(x,y);
%拉普拉斯模板系数
H=[0,-1,0;-1,4,-1;0,-1,0];
for i=1:x-2
    for j=1:y-2
        t=img(i:i+2,j:j+2);
```

	G=H.*t;
	G=abs(sum(G(:)));
	new_img(i+1,j+1)=G;
	end
	end
	new_img=uint8(new_img);
	end

实验总结	结果分析： 1. 均值滤波后的图像，内容边缘出现了模糊，且模糊的程度随滤波器的大小增大而增大。是因为因为它对所有的点平等对待，在对噪声进行分摊的同时，图像的边界也被分摊了。 2. Sobel 算子引入平均因素，对图像中的随机噪声有一定的平滑作用；相隔两行和两列求差分，故边缘两侧的元素得到了增强，边缘显得粗且亮。 3. Prewitt 算子和 Sobel 算子思路类似，但模板系数不同。相比较 Prewitt 算子，Sobel 模板能够较好的抑制噪声。 4. Robert 边缘算子采用的是对角方向相邻的两个像素之差，通过交叉求微分检测局部变化，这种边缘检测的方式对噪声敏感，但检测出的边缘不够有所缺失。 5. 拉普拉斯算子是二阶微分算子，在图像边缘处理中，二阶微分的边缘定位能力更强，锐化效果更好，同时又能保留背景信息。 通过对图像进行空间滤波，能够实现图像平滑或锐化的效果，对图像锐化或平滑的程度取决于空间滤波器的类型和大小。对于图像边缘，在进行空间滤波时，需要对图像的边缘进行填充，再进行滤波计算。
指导教师意见	签名：　　年　　月　　日