作业 1：边缘检测

学号：19049100002　　姓名：张泽群　　任课老师：李宇楠　　课程号：CS205201

1. *Gaussion*滤波

高斯滤波是一种线性平滑滤波，适用于消除高斯噪声，广泛应用于图像处理的减噪过程。通俗的讲，高斯滤波就是对整幅图像进行加权平均的过程，每一个像素点的值，都由其本身和邻域内的其他像素值经过加权平均后得到。

我们将其权重成为高斯核。理论上，高斯分布在所有定义域上都有非负值，这就需要一个无限大的高斯核。而实际上，仅需要取均值周围奇数倍标准差内的值，以外部份可以直接去掉。

我们可以通过二维高斯分布求得高斯核：

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

我们假定模糊半径为2，sigma = 0.5 ，则获得[5,5]的高斯核。

$$S=\begin{bmatrix} 6.96E-08 & 2.81E-05 & 0.000207549 & 2.81E-05 & 6.96E-08 \\ 2.81E-05 & 0.011331767 & 0.083731061 & 0.011331767 & 2.81E-05 \\ 0.000207549 & 0.083731061 & 0.618693507 & 0.083731061 & 0.000207549 \\ 2.81E-05 & 0.011331767 & 0.083731061 & 0.011331767 & 2.81E-05 \\ 6.96E-08 & 2.81E-05 & 0.000207549 & 2.81E-05 & 6.96E-08 \end{bmatrix}$$

再对高斯核和灰度图像进行卷积运算即可得到高斯滤波后的处理噪声的图像。

$$S=G*Image$$

2. *Sobel*滤波

2.1 简介

Sobel是基于梯度图像模值大小的检测算子，通常有水平和垂直两种算子，主要用于边缘检测，在技术上它是以离散型的差分算子，用来运算图像亮度函数的梯度的近似值， Sobel算子是典型的基于一阶导数的边缘检测算子，由于该算子中引入了类似局部平均的运算，因此对噪声具有平滑作用，能很好的消除噪声的影响

2.2 计算过程

*Sobel*算子包含两组[3,3]的矩阵，分别为横向及纵向模板(如下)，将之与图像作平面卷积，即可分别得出横向及纵向的灰度差分近似值，求得x 方向的梯度、y 方向的梯度。

$$S_x=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\quad S_y=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

计算公式所下所示：

$$g_x=\frac{\partial f}{\partial x}=(z_7+2z_8+z_9)-(z_1+2z_2+z_3)$$
$$g_y=\frac{\partial f}{\partial x}=(z_3+2z_6+z_9)-(z_1+2z_4+z_7)$$

$$G_x=S_x*Image$$
$$G_y=S_y*Image$$

Sobel 算子根据像素点上下，左右邻点灰度加权差，在边缘处达到极值这一现象检测边缘，结合高斯平滑，对噪声具有平滑作用，提供较为精确的边缘方向信息图像的每一个像素的横向及纵向梯度近似值可用如下的公式结合，来计算梯度幅值（正确的梯度幅值应乘1/8）。

$$G=\sqrt{G_x^2+G_y^2}$$
$$G_{true}=\frac{1}{8}\sqrt{G_x^2+G_y^2}$$

然后用如下公式计算梯度方向。(在Canny算子中梯度方向一般取近似 0 度，45 度，90 度和 135度四个方向)

$$\theta=arctan(\frac{G_y}{G_x})$$

2.3 算子优缺点

*Sobel*算子结合了高斯平滑和微分求导（分化），因此结果会具有更多的抗噪性，当对精度要求不是很高时，*Sobel* 算子是一种较为常用的边缘检测方法。

缺点是*Sobel*算子并没有将图像的主题与背景严格地区分开来，换言之就是*Sobel*算子并没有基于图像灰度进行处理，由于*Sobel*算子并没有严格地模拟人的视觉生理特征，所以提取的图像轮廓有时并不能令人满意。

3. *Canny*滤波

3.1 简介

John F.Canny 于 1986年发明了一个多级边缘检测算法——*Canny*边缘检测算子，并创立了边缘检测计算理论（*Computational theory of edge detection*），该理论有效的解释了这项技术的工作理论。

*Canny*边缘检测是一种比较新的边缘检测算子，具有很好地边缘检测性能，该算子功能比前面几种都要好，但是它实现起来较为麻烦，*Canny*算子是一个具有滤波，增强，检测的多阶段的优化算子。

3.2 计算过程

- **step1:** 用高斯滤波器平滑图象
- **step2:** 计算图像中每个像素点的梯度强度和方向（用一阶偏导的有限差分来计算梯度的幅值和方向）
- **step3:** 对梯度幅值进行非极大值抑制（**Non-Maximum Suppression**），以消除边缘检测带来的杂散响应
- **step4:** 用双阈值算法**(Double-Threshold)**检测来确定真实和潜在的边缘，通过抑制孤立的弱边缘最终完成边缘检测

  其中step 1,step2在前文已经提到，不多赘述。

3.3 非极大值抑制

通过非极大值抑制（*Non - maximum Suppression*）过滤掉非边缘像素，将模糊的边界变得清晰。该过程保留了每隔像素点上梯度强度的极大值，过滤掉其他的值。

对于每个像素点，它进行如下计算：

1. 将其梯度方向近似为0，45，90，135四个值中的一个，表示上下左右和45度方向。

2. 在每一点上，领域的中心像素I[x.y]与沿着梯度线的两个像素比较。如果I[x,y]的梯度值不比沿梯度线的两个相邻像素梯度值大，则令I[x,y]=0（抑制）。

3.4 双阈值算法检测和连接边缘

经过非极大抑制后图像中仍然有很多噪声点，此时需要通过双阈值技术处理(即设定一个阈值上界和阈值下界)来确定潜在边界。图像中的像素点如果大于阈值上界则认为必然是边界（称为强边界，*strong edge*），小于阈值下界则认为必然不是边界，两者之间的则认为是候选项（称为弱边界，*weak edge*）。

对非极大值抑制图像作用两个阈值*threshold1*和*threshold2*,我们把梯度值小于*threshold1*的像素的灰度值设为0，得到图像1。然后把梯度值小于*threshold2*,的像素的灰度值设为0，得到图像2。由于图像2的阈值较高，去除大部分噪音，但同时也损失了有用的边缘信息。而图像1的阈值较低，保留了较多的信息，我们可以以图像2为基础，以图像1为补充来连结图像的边缘。

链接边缘的具体步骤如下：

1. 对图像2进行扫描，当遇到一个非零灰度的像素p(x,y)时，跟踪以p(x,y)为开始点的轮廓线，直到轮廓线的终点q(x,y)。

2. 考察图像1中与图像2中q(x,y)点位置对应的点s(x,y)的8邻近区域。如果在s(x,y)点的八邻近区域中有非零像素s(x,y)存在，则将其包括到图像2中，作为r(x,y)点。从r(x,y)开始，重复第一步，直到我们在图像1和图像2中都无法继续为止。

当完成对包含p(x,y)的轮廓线的连结之后，将这条轮廓线标记为已经访问。回到第一步，寻找下一条轮廓线。重复第一步、第二步、第三步，直到图像2中找不到新轮廓线为止。

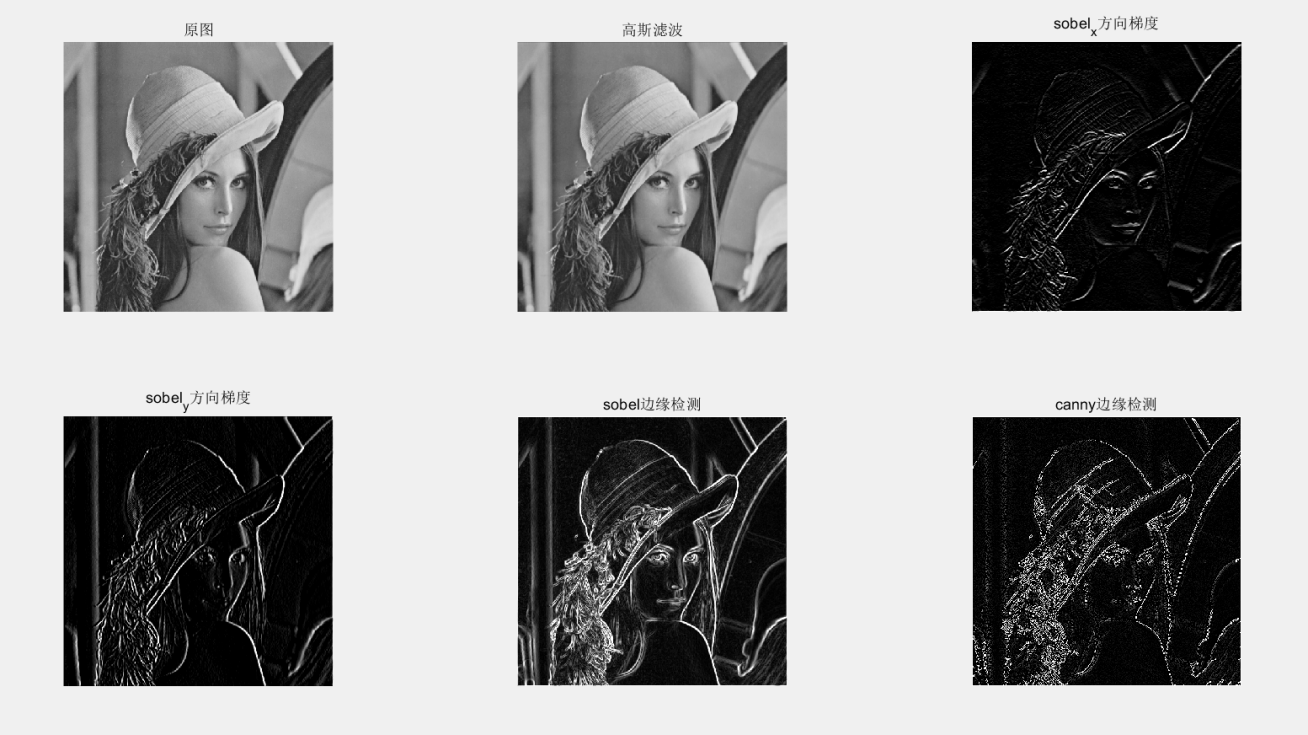
3.5 算子优缺点

*Canny*可以认为是*Sobel*的改进算子。它具有低错误率，边缘点被很好定位，单一的边缘响应优点。

4. 实验结果及最终分析

4.1 实验结果

实验结果如下图所示： //x 方向的梯度、y 方向的梯度、梯度幅度、梯度角度的结果见 result.xlsx

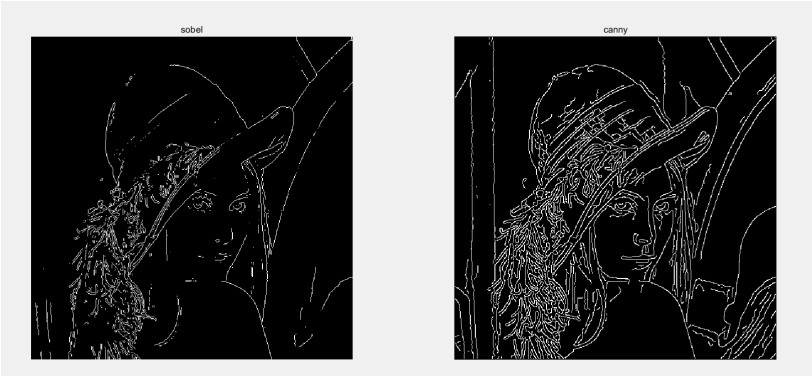


4.2 最终分析

分析：sobel、canny算子边缘检测功能基本实现。

对比：和matlab自带函数edge()生成的sobel、canny算子边缘检测生成图相比，自己写的sobel边缘检测在噪音处理方面有所欠缺，canny边缘检测在轮廓处理上也有较大欠缺。

（对比测试代码见.edgeFunc.m）



5. 源程序附录

1. test.m

```
clear all;
clc;

I=imread('testPhoto.png');
I=rgb2gray(I);
figure;
subplot(2,3,1);
imshow(I),title('原图');
I=double(I);

%高斯滤波
%h= fspecial('gaussian',[5 5]);
%I=imfilter(I,h);
I_g = Gaussian(I,[5 5],0.5);
subplot(2,3,2);
imshow(uint8(I_g)),title('高斯滤波');

%sobel边缘检测
[I_s,I_sx,I_sy,GradValue,GradDirection] =Sobel(I_g);
subplot(2,3,3);
imshow(uint8(I_sx)),title('sobel_x方向梯度');

subplot(2,3,4);
imshow(uint8(I_sy)),title('sobel_y方向梯度');

subplot(2,3,5);
imshow(uint8(I_s)),title('sobel边缘检测');

xlswrite('result.xlsx',I_sx,'X方向梯度');
xlswrite('result.xlsx',I_sy,'Y方向梯度');
xlswrite('result.xlsx',GradValue,'梯度幅度');
xlswrite('result.xlsx',GradDirection,'梯度角度');

%canny边缘检测
I_c =Canny(I_s,I_sx,I_sy);
subplot(2,3,6);
imshow(uint8(I_c)),title('canny边缘检测');
```

2.Gaussian.m

```
function I_gaussian=Gaussian(I,hsize, sigma)        %高斯滤波
    size = (hsize-1)/2;
    [m,n] = meshgrid(-size(2):size(2),-size(1):size(1));
    arg = -(m.*m + n.*n)/(2*sigma*sigma);
    h = exp(arg);
    h(h<eps*max(h(:))) = 0;
    sumh = sum(h(:));
    if sumh ~= 0
       h = h/sumh;
    end

    I_gaussian = double(I);
    I_gaussian=conv2( I_gaussian,h,'same');
end
```

3.Sobel.m

```
function [I_sobel,I_sobel_x,I_sobel_y,GradValue,GradDirection]=Sobel(I)
      %sobel算子
    hx = [-1 0 1;-2 0 2;-1 0 1;];
    hy = hx';
    sobel = double(I);
    I_sobel_x = conv2( sobel,hy,'same');
    I_sobel_y = conv2( sobel,hx,'same');
    I_sobel = sqrt(I_sobel_x.^2+I_sobel_y.^2);
    GradValue = I_sobel/8;
    GradDirection  = atan2(I_sobel_y,I_sobel_x);

    %for x=2:m-1
        %for y=2:n-1
            %Gx = I(x+1,y-1) + 2*I(x+1,y) + I(x+1,y+1)-I(x-1, y-1) -
2*I(x-1,y) - I(x-1,y+1);
            %Gy = I(x-1,y-1) + 2*I(x,y-1) + I(x+1,y-1)-I(x-1, y+1) -
2*I(x,y+1) - I(x+1,y+1);
            %I_sobel_x(x,y) = Gx;
            %I_sobel_y(x,y) = Gy;
            %I_sobel(x,y) = sqrt(Gx^2+Gy^2);
        %end
    %end

end
```

4.connect.m

```
function I_connect=connect(I_connect,y,x,low)          %种子定位后的连通分析
    neighbour=[-1 -1;-1 0;-1 1;0 -1;0 1;1 -1;1 0;1 1];   %八连通搜寻
    [m, n]=size(I_connect);
    for k=1:8
        y2=y+neighbour(k,1);
        x2=x+neighbour(k,2);
        if y2>=1 &&y2<=m &&x2>=1 && x2<=n
            if I_connect(y2,x2)>=low && I_connect(y2,x2)~=255
                I_connect(y2,x2)=255;
                I_connect=connect(I_connect,y2,x2,low);
            end
        end
    end
end
```

5.Canny.m

```
function I_canny=Canny(I,I_x,I_y)          %canny算子
    [m,n] = size(I);
    newGradDricetion=zeros(m,n);
    for i=1:m
        for j=1:n
            Mx=I_x(i,j);
            My=I_y(i,j);

            if My~=0
                o = atan(Mx/My);          %求边缘的法线弧度
            elseif My == 0 && Mx > 0
                o = pi/2;
            else
                o = -pi/2;
            end
       % 将梯度方向取0°，45°，90°，135°四个方向
            if ( ( o>=( -22.5/180*pi ) && o<( 22.5/180*pi ) ) || ( o>=(
157.5/180*pi ) && o<=pi ) || ( o<=( -157.5/180*pi ) && o>=-pi ) )
                dir=0;
            elseif ( ( o>= ( 22.5/180*pi ) && o<( 67.5/180*pi ) ) || (
o>= ( -157.5/180*pi ) && o<( -112.5/180*pi)))
                dir=45;
            elseif ( ( o>= ( 67.5/180*pi ) && o<( 112.5/180*pi) ) || (
o>= ( -112.5/180*pi ) && o<( -67.5/180*pi )))
                dir=90;
            else
                dir=135;
            end
            newGradDricetion(i,j)=dir;
        end
```

```
    end

    I_canny = I;

    %非极大值抑制
    for i=3:m-2
        for j=3:n-2
            if( newGradDricetion(i,j)==0 )
                A=[I_canny(i-1,j),I_canny(i+1,j)];
                if( abs(I_canny(i,j))< max( abs(A) ) )
                    I_canny(i,j)=0;
                end
            elseif( newGradDricetion(i,j)==45 )
                A=[I_canny(i-1,j-1),I_canny(i+1,j+1)];
                if( abs(I_canny(i,j))< max( abs(A) ))
                    I_canny(i,j)=0;
                end
            elseif( newGradDricetion(i,j)==90 )
                A=[I_canny(i,j-1),I_canny(i,j+1)];
                if( abs(I_canny(i,j))< max( abs(A) ) )
                    I_canny(i,j)=0;
                end
            elseif( newGradDricetion(i,j)==135 )
                A=[I_canny(i-1,j+1),I_canny(i+1,j-1)];
                if( abs(I_canny(i,j))< max( abs(A) ) )
                   I_canny(i,j)=0;
                end
            end
        end
    end

    %用双阈值算法检测和连接边缘
    up=200;      %上阈值
    low=35;      %下阈值
    set(0,'RecursionLimit',10000);   %设置最大递归深度
    for i=1:m
        for j=1:n
          if I_canny(i,j)>up &&I_canny(i,j)~=255
                I_canny(i,j)=255;
                I_canny=connect(I_canny,i,j,low);
          end
        end
    end
end
```