上海应用技术大学

计算机科学与信息工程学院

课题报告

课程学期： 2020~2021 第一学期

课程名称： 机器视觉应用

课题名称： 基于 MATLAB 语言对比梯度算子：Roberts 算子、Sobel 算子、Prewitt 算子区别

学组成员： 刘子健、王恺昊、张珅铖

指导教师： 张晴

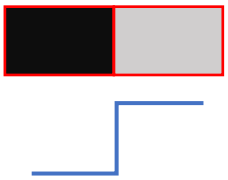
一、基础内容摘要：

（一）边缘检测的基本原理：

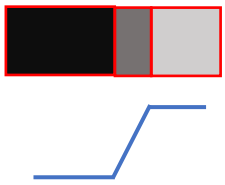
1）边缘：

图像边缘是图像最基本的特征，所谓边缘是指图像局部特性的不连续性。灰度或结构等信息的突变处称之为边缘。

例如，灰度级的突变、颜色的突变,、纹理结构的突变等。边缘是一个区域的结束，也是另一个区域的开始，利用该特征可以分割图像。



(图 1.1.1)理想边缘模型



(图 1.1.2)斜坡边缘模型

如图 1.1.1 及 1.1.2 为灰度级跃变的边缘模型，图 1.1.1 是一个理想的边缘所具备的特性。每个灰度级跃变到一个垂直的台阶上。而实际上，在图像采集系统的性能、采样率和获取图像的照明条件等因素的影响，得到的边缘往往是模糊的，边缘被模拟成具有“斜坡面”的剖面，如图 1.1.2 所示，在这个模型中，模糊边缘变得“宽”了，而清晰的边缘变得“窄”了。

2）边缘检测：

图像的边缘有方向和幅度两种属性。

边缘通常可以通过一阶导数或二阶导数检测得到。一阶导数是以最大值作为对应的边缘的位置，而二阶导数则以过零点作为对应边缘的位置。

（二）边缘检测算子分类：

1）一阶导数的边缘算子：

通过模板作为核与图像的每个像素点做卷积和运算，然后选取合适的阈值来提取图像的边缘。常见的有 Roberts、Sobel 和 Prewitt 算子。

2）二阶导数的边缘算子：

依据二阶导过零点，常见的有 Laplacian 算子，此类算子对噪声敏感。

3）其他边缘算子：

前面两类均是通过微分算子来检测图像边缘，还有一种就是 Canny 算子，其是在满足一定约束条件下推导出来的边缘检测最优化算子。

本课题报告将着重对 Roberts、Sobel、Prewitt 算子进行探究。

（三）梯度：

1）图像梯度：

为了达到寻找边缘的目的，检测灰度变化可用一阶导数或二阶导数来完成。**下面将讨论一阶导数**。

为了在一幅图像f的(x,y)位置处寻找边缘的强度和方向，所以选择的工具就是**梯度**，梯度用∇f来表示并用向量来定义，定义如下所示：

$$\nabla f \equiv gard(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} \equiv \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

其中，梯度∇f为一个向量，它表示f在位置(x,y)处最大变化率方向。

梯度∇f的大小用$M(x,y)$表示：

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

注：mag 函数在 matlab 中为波特图在频率为 w 时，频率响应的幅值。

其中$M(x,y)$表示梯度向量方向变化率的值。

2）数学梯度的简单推导：

对于以为函数$f(x)$在点x处的导数的近似：将函数$f(x+\Delta x)$展开为x的泰勒级数，令$\Delta x = 1$，且只保该级数的线性项，则函数$f(x)$的梯度∇f计算为：

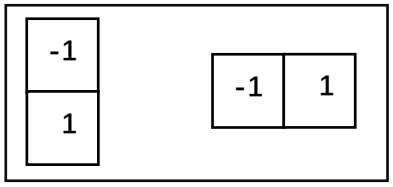
$$\nabla f = \frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$$

3）梯度算子：

由上面的数学推导可知，要得到一幅图像的梯度，则要求在图像的每个像素点位置处计算偏导数。我们处理的是数字量，因此要求关于一点的邻域上的偏导数的数字近似，因此一幅图像f，在 (x,y) 位置处的x和 y方向上的梯度大小g_x和g_y分别计算为：

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \quad g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y)$$

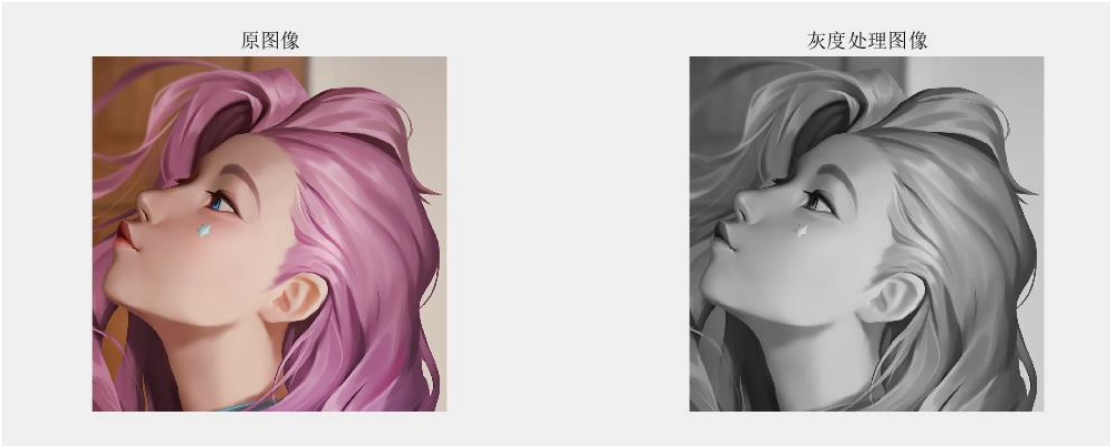
上述两个公式对所有 x 和 y 的有关值可用下图的一维模板对 $f(x,y)$的滤波得到:



用于计算梯度偏导数的滤波器模板，通常称之为梯度算子、边缘算子和边缘检测子等。

二、Roberts 算子、Sobel 算子、Prewitt 算子基本原理介绍及代码实践：

如图 2.0 所示，为本次实践部分处理的原图像及灰度处理后的图像



（图 2.0）原图像（左）及灰度处理后的图像（右）

（一）Roberts 算子：

1）基本原理：

Roberts 算子又称为交叉微分算法，它是基于交叉差分的梯度算法，通过局部差分计算检测边缘线条。常用来处理具有**陡峭的低噪声图像**，当图像边缘接近于正 45 度或负 45 度时，该算法处理效果更理想。其缺点是对边缘的定位不太准确，提取的边缘线条较粗。

Roberts 算子的模板分为水平方向和垂直方向，如下式所示，从其模板可以看出，Roberts 算子能较好的增强正负 45 度的图像边缘。

$$d_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad d_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

例如，下面给出 Roberts 算子的模板，在像素点 e 处x和y方向上的梯度大小g_x和g_y分别计算为：

a	b	c
d	e	f
g	h	i

-1	0
0	1

0	-1
1	0

Roberts 算子

$$g_x = \frac{\partial f}{\partial x} = i - e \quad g_y = \frac{\partial f}{\partial y} = h - f$$

2）代码实例：

首先创建与图像长宽相同的 0 矩阵：　imr=zeros(m,n);

设置边界阈值为 10/15（当前）/20：　roberts_edge=15;

注：不同的边界阈值会得到不同的结果如图 2.1 所示

利用双循环对原图像进行卷积计算，由于 Roberts 算子模板的大小为 2×2大小，故在进行像素处理时，可以选择在边界处内一圈进行处理：

```
for i=2:m-1
    for j=2:n-1
        …(函数实现)
    end
end
```

使用 Roberts 算子进行计算，并输出至结果矩阵中：

```
imr(i,j)=abs(imgray(i+1,j+1)-imgray(i,j))+abs(imgray(i,j+1)-imgray(i+1,j));
```

阈值处理以确定边界：

```
if imr(i,j)<roberts_edge
    imr(i,j)=0;
else
    imr(i,j)=255;
end
```

如图 2.1 所示，图像在 Roberts 算子中，不同阈值下的处理结果：

（图 2.1）使用 Roberts 算子分别在 10、15、20 阈值下的处理结果

3）Roberts 算子完整代码呈现：

Matlab 实现图示：

```
%上海应用技术大学 计算机科学与信息工程学院
%刘子健 1910400604
img=imread("原图1350x1350.png");  %读入图像
imgray=rgb2gray(img);             %灰度处理
[m,n]=size(imgray);               %读入图像大小：m行n列
subplot(2,3,1);
imshow(img);
title("原图像");                  %输出原图像
subplot(2,3,2);
imshow(imgray);
title("灰度处理图像");            %输出灰度处理图像
%###Roberts算子###处理部分：
roberts_edge=15;                  %边界处理阈值
imr=zeros(m,n);                   %创建m行n列的零矩阵，用于输出robers算子处理后的结果图像
for i=2:m-1                       %roberts算子模板为2*2，所以从内圈开始卷积
    for j=2:n-1
        imr(i,j)=abs(imgray(i+1,j+1)-imgray(i,j))+abs(imgray(i,j+1)-imgray(i+1,j));
                                  %绝对值（右下角-左上角）+绝对值（左下角-右上角）
        if imr(i,j)<roberts_edge  %若低于此值，将其变为黑色，反之白色（边界）
            imr(i,j)=0;
        else
            imr(i,j)=255;
        end
    end
end
subplot(2,3,5);
imshow(imr);
title("Roberts算子处理（阈值15）：");          %输出图像
```

文字版本：

```
roberts_edge=15;
imr=zeros(m,n);
for i=2:m-1
    for j=2:n-1
        imr(i,j)=abs(imgray(i+1,j+1)-
imgray(i,j))+abs(imgray(i,j+1)-imgray(i+1,j));
        if imr(i,j)<roberts_edge
            imr(i,j)=0;
        else
            imr(i,j)=255;
        end
    end
end
imshow(imr);
```

（二）Prewitt 算子：

1）基本原理：

Prewitt 算子是一种图像边缘检测的微分算子，其原理是利用特定区域内像素灰度值产生的差分实现边缘检测。由于 Prewitt 算子采用 3×3 模板对区域内的像素值进行计算，故 Prewitt 算子的边缘检测结果在水平方向和垂直方向更加明显。

Prewitt 算子适合用来识别噪声较多、灰度渐变的图像，其计算公式如下所示：

$$d_y=\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}\quad d_x=\begin{bmatrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{bmatrix}$$

例如，下面给出 Prewitt 算子的模板，在像素点 e 处x和y方向上的梯度大小g_x和g_y分别计算为：

a	b	c
d	e	f
g	h	i

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

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

Prewitt 算子

$$g_x=\frac{\partial f}{\partial x}=g+h+i-a-b-c$$

$$g_y=\frac{\partial f}{\partial y}=c+f+i-a-d-g$$

2）代码实例：

使用 Prewitt 算子对图像进行计算：

```
imp(i,j)=abs(g(i-1,j-1)+g(i,j-1)+g(i+1,j-1)-g(i-1,j+1)-g(i,j+1)-g(i+1,j+1))
+abs(g(i+1,j-1)+g(i+1,j)+g(i+1,j+1)-g(i-1,j-1)-g(i-1,j)-g(i-1,j+1));
```

注：上述公式中 imgray 已使用 g 代替；

运算方式以外部分均与上文提及的 Roberts 算子相同；

如图 2.2 所示，图像在 Prewitt 算子中，不同阈值下的处理结果：

（图 2.2）使用 Prewitt 算子分别在 5、10、15 阈值下的处理结果

3）Prewitt 算子完整代码呈现：

Matlab 实现图示：

```
34      %###Prewitt算子###处理部分：
35 —    prewitt_edge=15;                    %prewitt算子边界处理阈值
36 —    imp=zeros(m,n);                     %创建m行n列的零矩阵，用于输出Prewitt算子处理后的结果图像
37
38 —    for i=2:m-1
39 —        for j=2:n-1                     %prewitt算子模板为3*3，从内圈开始卷积
40 —            imp(i,j)=abs(imgray(i-1,j-1)+imgray(i,j-1)+imgray(i+1,j-1)-imgray(i-1,j+1)-imgray(i
41              %运算
42 —            if imp(i,j)<prewitt_edge  %若低于此值，将其变为黑色，反之白色（边界）
43 —                imp(i,j)=0;
44 —            else
45 —                imp(i,j)=255;
46 —            end
47 —        end
48 —    end
49
50 —    subplot(2,3,6);
51 —    imshow(imp);
52 —    title("Prewitt算子处理（阈值15）：");            %输出图像
```

文字版本：

```
prewitt_edge=15;
imp=zeros(m,n);
for i=2:m-1
    for j=2:n-1
        imp(i,j)=abs(imgray(i-1,j-1)+imgray(i,j-1)+imgray(i+1,j-1)-
imgray(i-1,j+1)-imgray(i,j+1)-imgray(i+1,j+1))+abs(imgray(i+1,j-
1)+imgray(i+1,j)+imgray(i+1,j+1)-imgray(i-1,j-1)-imgray(i-1,j)-
imgray(i-1,j+1));
        if imp(i,j)<prewitt_edge
            imp(i,j)=0;
        else
            imp(i,j)=255;
        end
    end
end

subplot(2,3,6);
imshow(imp);
title("Prewitt 算子处理（阈值 15）：");
```

（三）Sobel 算子：

1）基本原理：

Sobel 算子是一种用于边缘检测的离散微分算子，它结合了高斯平滑和微分求导。该算子用于计算图像明暗程度近似值，根据图像边缘旁边明暗程度把该区域内超过某个数的特定点记为边缘。Sobel 算子在 Prewitt 算子的基础上增加了**权重**的概念，认为相邻点的距离远近对当前像素点的影响是不同的，距离越近的像素点对应当前像素的影响越大，从而实现图像锐化并突出边缘轮廓。

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。对噪声具有平滑作用，提供较为精确的边缘方向信息。因为 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。

Sobel 算子的边缘定位更准确，常用于噪声较多、灰度渐变的图像。其算法模板如下面的公式所示，其中d_x表示水平方向，d_y表示垂直方向:

$$d_y=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\quad d_x=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

例如，下面给出 Sobel 算子的模板，在像素点 e 处x和y方向上的梯度大小g_x和g_y分别计算为：

a	b	c
d	e	f
g	h	i

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

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

Sobel 算子

$$g_x=\frac{\partial f}{\partial x}=g+2h+i-a-2b-c$$

$$g_y=\frac{\partial f}{\partial y}=c+2f+i-a-2d-g$$

2）代码实例：

使用 Prewitt 算子对图像进行计算：

```
ims(i,j)=abs(g(i-1,j-1)+2.*g(i,j-1)+g(i+1,j-1)-g(i-1,j+1)-2.*g(i,j+1)-
g(i+1,j+1))+abs(g(i+1,j-1)+2.*g(i+1,j)+g(i+1,j+1)-g(i-1,j-1)-2.*g(i-1,j)-g(i-1,j+1));
```

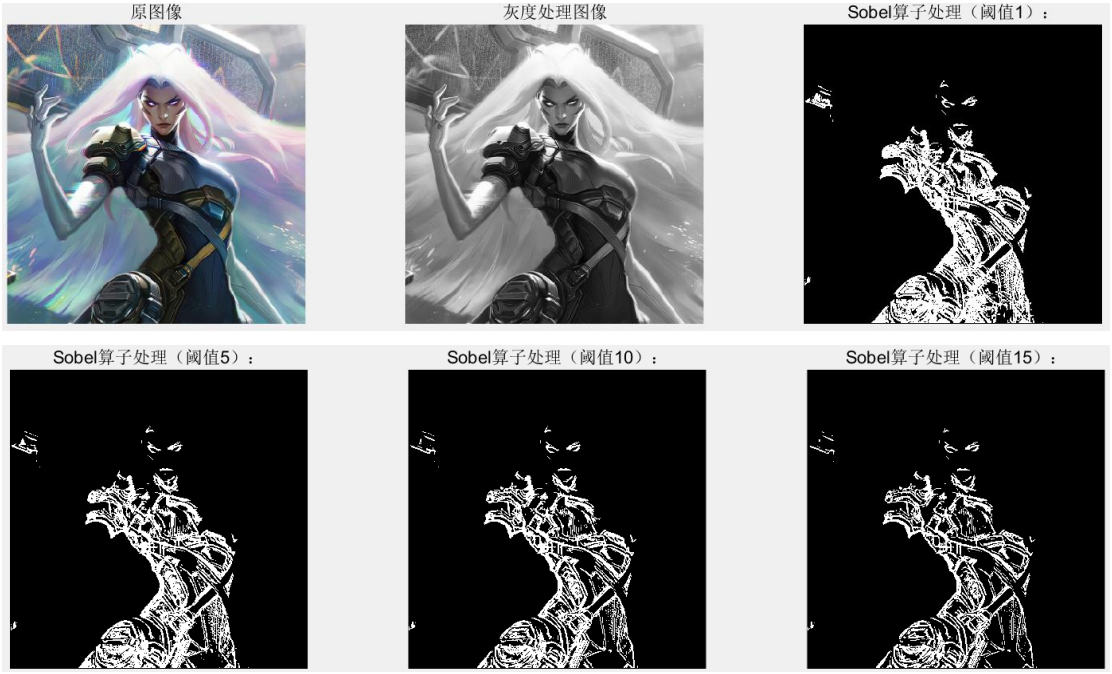
注：上述公式中 imgray 已使用 g 代替；

运算方式以外部分均与上文提及的 Roberts 算子相同；

如图 2.3 所示，图像在 Prewitt 算子中，不同阈值下的处理结果：

（图 2.3）使用 Sobel 算子分别在 5、10、15 阈值下的处理结果

尽管阈值很低，但部分边界仍然未被合理显示出来。我们更换原图以继续测试：



（图 2.4）使用 Sobel 算子分别在 2、5、10、15 阈值下的处理结果

如图 2.4 所示，此张原图相对之前的原图具有更多的噪点，但无论阈值多么低（本次测试中阈值为 1），所以我们可以看出 Sobel 算子的结果会具有更多的抗噪性，从新的原图中我们可以直接看出，背景较为微小的细节可能被处理为噪点导致无法视为边缘。

但是，Sobel 算子提供较为精确的边缘方向信息，在细节上更加丰富。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。

3）Sobel 算子完整代码呈现：

Matlab 实现图示：

```
54      %###Sobel 算子###处理部分：
55 —    sobel_edge=5;
56 —    ims=zeros(m,n);
57
58 —    for i=2:m-1                          %Sobel算子模板为3*3，从内圈开始卷积
59 —        for j=2:n-1
60 —              ims(i,j)=abs(imgray(i-1,j-1)+2.*imgray(i,j-1)+imgray(i+1,j-1)-img
61                 %运算，注意：权重部分需要点乘
62 —            if ims(i,j)<sobel_edge  %若低于此值，将其变为黑色，反之白色（边界）
63 —                ims(i,j)=0;
64 —            else
65 —                ims(i,j)=255;
66 —            end
67 —        end
68 —    end
69
70 —    subplot(2,3,4);
71 —    imshow(ims);
72 —    title("Sobel算子处理（阈值5）："");            %输出图像
```

文字版本：

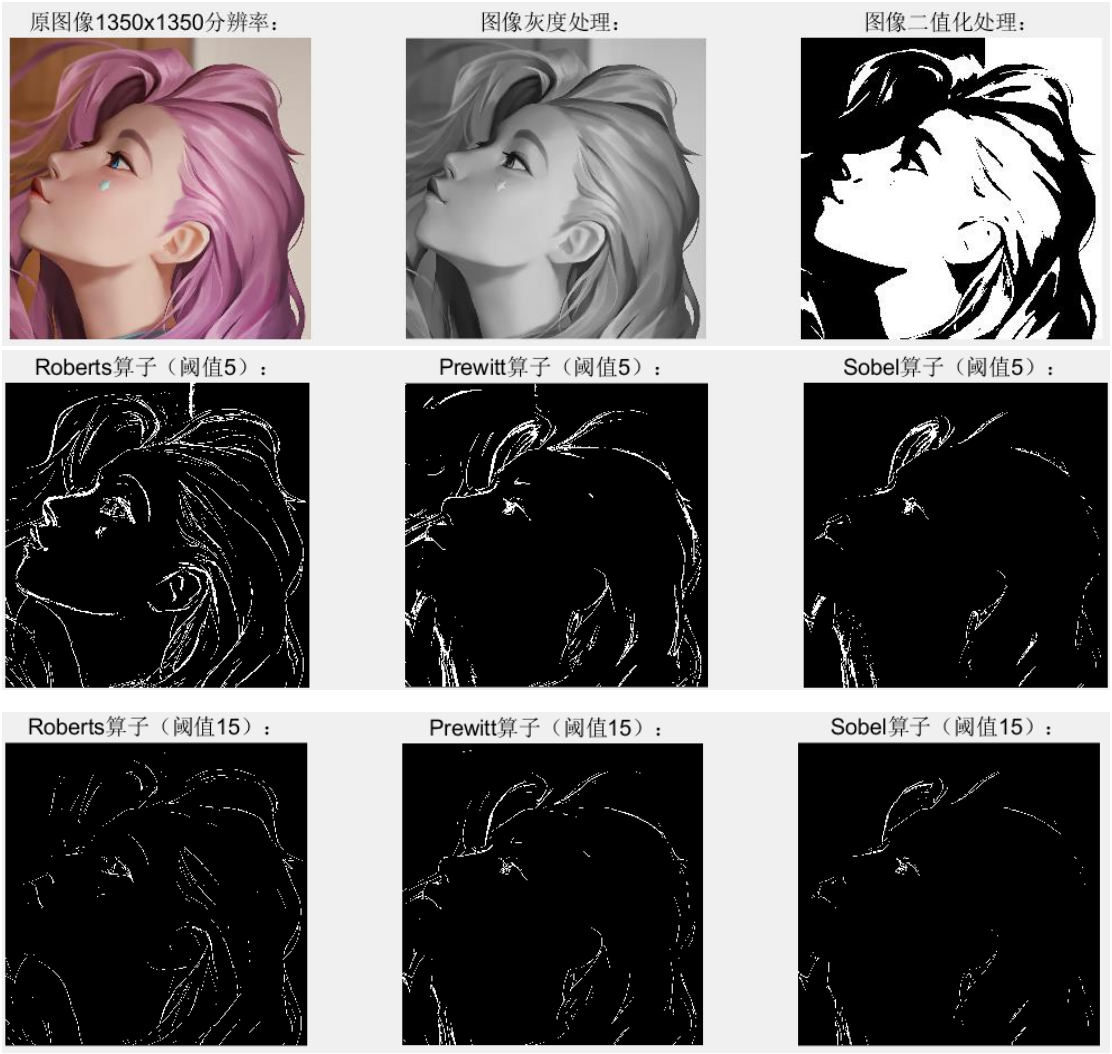
```
sobel_edge=5;
ims=zeros(m,n);
for i=2:m-1
    for j=2:n-1
          ims(i,j)=abs(imgray(i-1,j-1)+2.*imgray(i,j-
1)+imgray(i+1,j-1)-imgray(i-1,j+1)-2.*imgray(i,j+1)-
imgray(i+1,j+1))+abs(imgray(i+1,j-
1)+2.*imgray(i+1,j)+imgray(i+1,j+1)-imgray(i-1,j-1)-2.*imgray(i-
1,j)-imgray(i-1,j+1));
          if ims(i,j)<sobel_edge
              ims(i,j)=0;
          else
              ims(i,j)=255;
          end
    end
end
```

三、Roberts、Prewitt、Sobel 算子横向对比及小结：

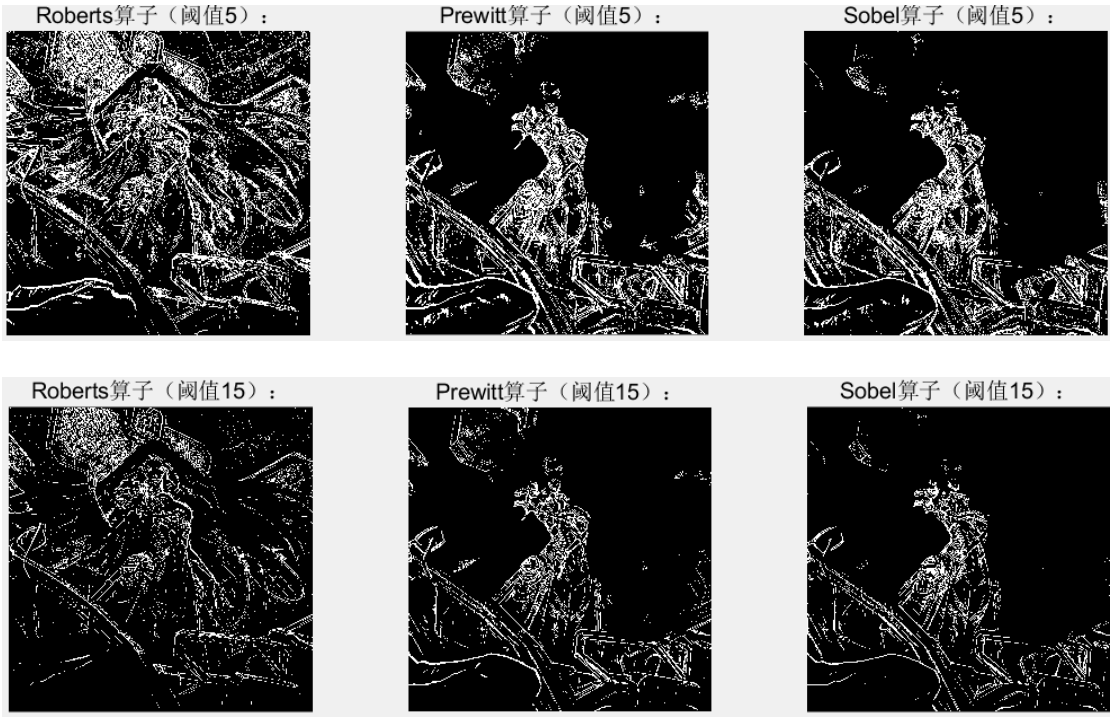
（一）三种算子的实验比较：

本报告探究的边缘检测算法主要是基于图像强度的一阶导数，但导数通常对噪声很敏感，因此需要采用滤波器来过滤噪声，并调用图像增强或阈值化算法进行处理，最后再进行边缘检测。下面对三种不同的算子进行实验对比：

示例图像 1：分辨率 1350*1350：



示例图像 2：分辨率 1080*1080：



结果分析：

本次报告中所选择的两幅图像分别背景单一及背景具有更多噪点与细节的图像，我们从对两幅图像的对比中我们可以得出以下结论：

1）**Robert 算子**对陡峭的低噪声图像效果较好，尤其是边缘正负 45 度较多的图像，但定位准确率较差；

显然，我们可以从第二幅原图中的 Roberts 算子中可以看出，由于 Roberts 算子模板为2×2大小，边缘定位精度较高，容易丢失一部分边缘，不具备抑制噪声的能力。该算子对具有陡峭边缘且含噪声少的图像效果较好，尤其是边缘正负 45 度较多的图像，但定位准确率较差；

2）**Prewitt 算子**对灰度渐变的图像边缘提取效果较好，而没有考虑相邻点的距离远近对当前像素点的影响；对比 Roberts 算子，Prewitt 对噪声较多的图像处理效果更好。

相对示例图 1，Prewitt 算子相对 Sobel 算则在示例图 2，即在细节较多的图像上表现更加。

3）**Sobel 算子**考虑了综合因素，对噪声较多的图像处理效果更好。

Sobel 算子，在 Prewitt 基础上增加了权重的概念，结果会具有更多的抗噪性，虽然 Prewitt 对噪声较多的图像处理效果更好，Sobel 算子边缘定位效果不错，但检测出的边缘容易出现多像素宽度。

参考文献：
[1] [Python 图像处理] 十七.图像锐化与边缘检测之 Roberts 算子、Prewitt 算子、Sobel 算子和 Laplacian 算子 blog.csdn.net/Eastmount/article/details/89001702
[2] 数字图像处理(19): 边缘检测算子(Roberts 算子、Prewitt 算子、Sobel 算子 和 Laplacian 算子) blog.csdn.net/zaishuiyifangxym/article/details/89840396
[3] 图像边缘检测——一阶微分算子 Roberts、Sobel、Prewitt、Kirsch、Robinson（Matlab 实现）blog.csdn.net/u014485485/article/details/78339420
[4] 图像边缘提取——梯度算子、Roberts 算子、prewitt 算子、Sobel 算子、Kirsch 算子、LOG 算子的 matlab 实现 blog.csdn.net/qq_25729757/article/details/75050178
[5] 冈萨雷斯. 数字图像处理（MATLAB 版）(第二版)

感谢国家