数字图像处理

实验四 边缘检测与图像分割

专　　业：计算机科学与技术

年　　级：2020 级

学　　号：20201060330

姓　　名：胡诚皓

2022 年 10 月 31 日

实验四： 边缘检测与图像分割

1．实验目的

1）掌握图像边缘检测的基本原理和方法。
2）比较不同边缘检测方法对检测结果的影响。
3）根据应用需求能设计简单边缘检测算子。
4）掌握基于区域相似性的分割方法，能用阈值处理方法进行图像分割。

2．实验原理

2.1 边缘检测原理

边缘检测是为了将其周围像素灰度有阶跃变化的像素检测出来，这些像素组成的集合就是该图像的边缘。比较常用的边缘检测方法就是考察每个像素在某个领域内灰度的变化，然后利用边缘临近一阶或二阶方向导数变化规律检测边缘，即边缘检测局部算法。

而常用的边缘检测算子有 Sobel，Roberts，Prewitt，Laplace 等等，这里先介绍 Sobel 算子的检测过程。主要的方法就是将图像的每一个点都用 Sobel 算子做卷积：一个用来检测垂直边缘，一个用来检测水平边缘，而最后两个卷积的最大值将作为该点的输出，即检测后的灰度。

示例：Sobel 算子可以看到 Sobel 算子包括两组 3*3 的矩阵，左边的表示垂直，右边的表示水平。将它与图像作平面卷积，即可分别得出垂直及水平的亮度差分近似值。

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

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

例如 A 表示的是原始图像，G_x和G_y分别表示经过水平和垂直边缘检测的图像灰度值，

则有：$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A$，$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A$

然后我们通过右边公式得到最后该点灰度的大小：$G = \sqrt{{G_x}^2 + {G_y}^2}$.

```
% Matlab 自带函数边缘检测
% K 为获取得到的关键帧的灰度图
E = edge(image,'sobel', 0.09);
subplot(1,3,3);imshow(E,[]):title('Matlab 自带函数边缘检测');
```

2.2 基于区域相似性的分割原理

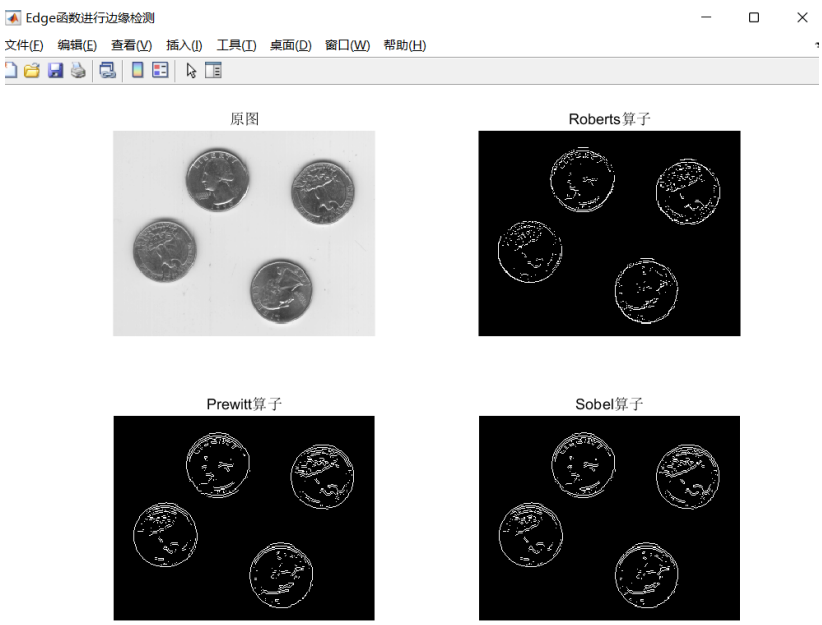
基于灰度的分割最简单的处理思想是，高于某一灰度的像素划分到一个区域中，低于某灰度的像素划分到另一区域中，这种基于灰度阈值的分割方法称为灰度门限法，它也是基于区域的分割方法。该方法中的关键是通过直方图找出合理的分割阈值（具体方法见课件 PPT）。

3、实验内容与要求

（1）使用 edge 函数进行 Roberts、Prewitt、Sobel 算子的边缘检测

```
E1 = edge(origin_std, 'Roberts');
E2 = edge(origin_std, 'Prewitt');
E3 = edge(origin_std, 'Sobel');
```

运行结果如下图：



（2）自己实现 Sobel 算子的边缘检测

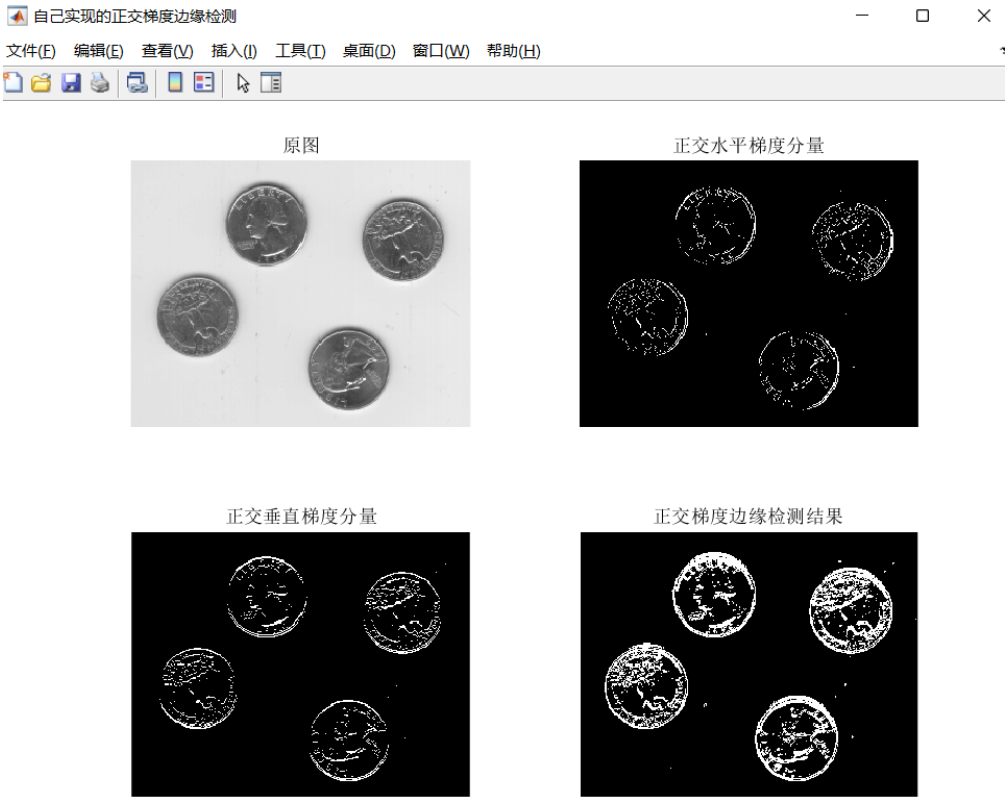
```
% Sobel 算子
sobel = [-1 0 1;-2 0 2;-1 0 1]/4;
% 水平方向和垂直方向的梯度分量
sobel_hori = imfilter(origin_std, sobel);
sobel_vert = imfilter(origin_std, sobel');
%  得到边缘信息
sobel_edge = (sobel_hori.^2+sobel_vert.^2).^0.5;
```

分别显示水平方向、垂直方向以及整体模值的二值图像，结果如下图：

（3）自己实现普通的正交梯度边缘检测

```
% 普通正交梯度的算子
orth = [0 0 0; -1 1 0; 0 0 0]; % 计算两个梯度分量
orth_hori = imfilter(origin_std, orth);
orth_vert = imfilter(origin_std, orth');
% 得到边缘信息
orth_edge = (orth_vert.^2+orth_hori.^2).^0.5;
```

事实上，都是根据像素计算的式子的形式做出算子，再和图像直接做卷积即可。最终结果如下图：

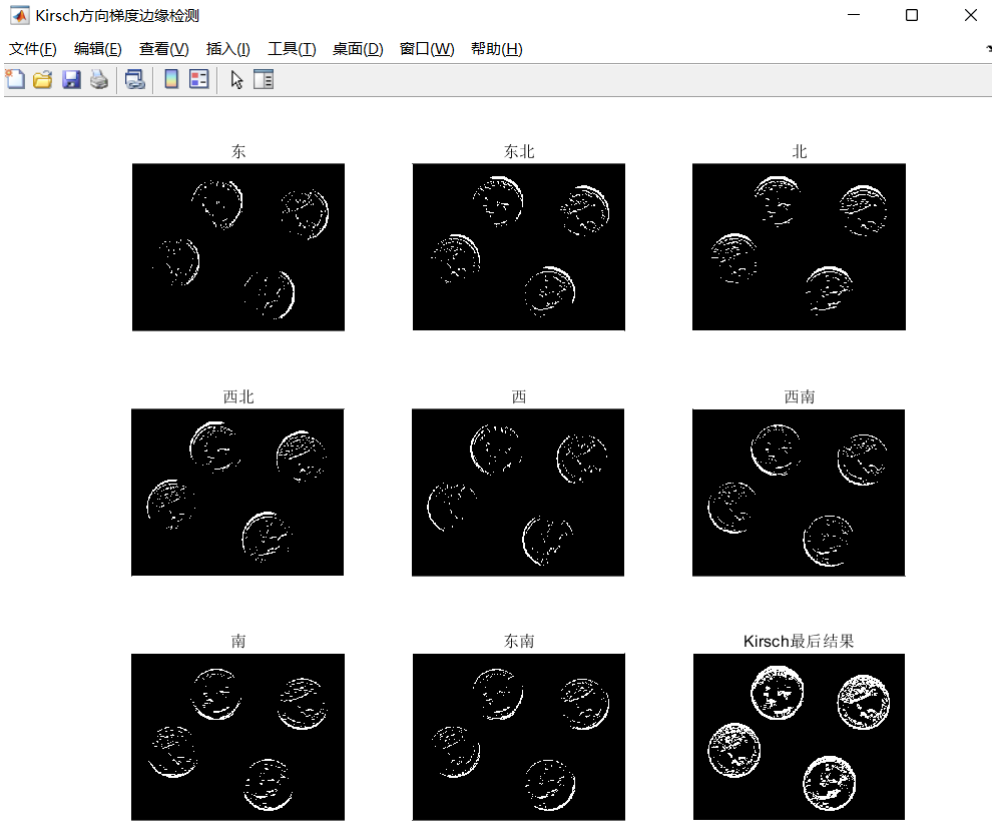


（4）Kirsch 算子进行方向梯度的边缘检测

```
kirsch(:,:,1) = [-3 -3 5;-3 0 5;-3 -3 5];
kirsch(:,:,2) = [-3 5 5;-3 0 5; -3 -3 -3];
kirsch(:,:,3) = [5 5 5;-3 0 -3;-3 -3 -3];
kirsch(:,:,4) = [5 5 -3;5 0 -3;-3 -3 -3];
kirsch(:,:,5) = [5 -3 -3;5 0 -3;5 -3 -3];
kirsch(:,:,6) = [-3 -3 -3;5 0 -3; 5 5 -3];
kirsch(:,:,7) = [-3 -3 -3;-3 0 -3;5 5 5];
kirsch(:,:,8) = [-3 -3 -3;-3 0 5;-3 5 5];
kirsch = kirsch./15;

% 计算各方向上的梯度分量
for i=1:8
    kirsch_res(:,:,i) = imfilter(origin_std, kirsch(:,:,i));
end
% 取最大值得到最终的边缘
kirsch_edge = max(kirsch_res, [], 3);
```

全部尽量使用矩阵整体进行计算，将各个方向的梯度分量存储再 kirsch_res 的各通道中，最后由 kirsch_edge 存储第三维（即“通道”维度）中各像素的最大值即可。最终结果如下图：



（5）基于图像直方图计算最优分割阈值

总体的处理流程为：

① 获取直方图

② 计算累计直方图

③ 从 0-255 遍历尝试各阈值，找出一个使得混合总错分概率最小的阈值

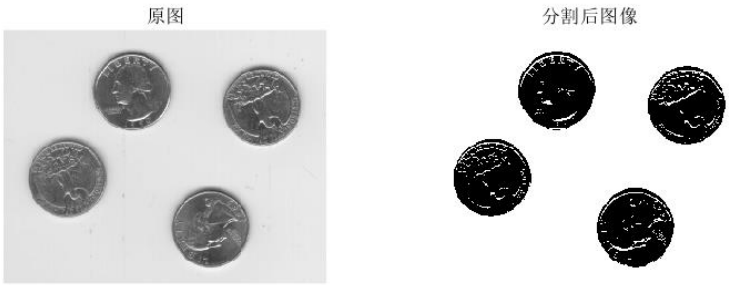
④ 计算最终阈值并进行图像分割

```
h = imhist(origin_img);
h_count = h;
```

```
% 计算累计直方图
for i=2:256
    h_count(i) = h_count(i) + h_count(i-1);
end
last_diff = -1;
idx = -1;
% 测试阈值
for T=1:256
    P1 = h_count(T)/(img_height*img_width);
    P2 = (h_count(256)-h_count(T))/(img_height*img_width);
    if (last_diff < 0 && P1-P2 > 0)
        idx = T;
        break
    end
end

% 分别计算两类像素的均值
left = [1:T-1]*h(1:T-1)/h_count(T-1);
right = [T:256]*h(T:256)/h_count(256);
% 取均值的平均值为最后的最佳阈值
threshold = mean([left right]);
```

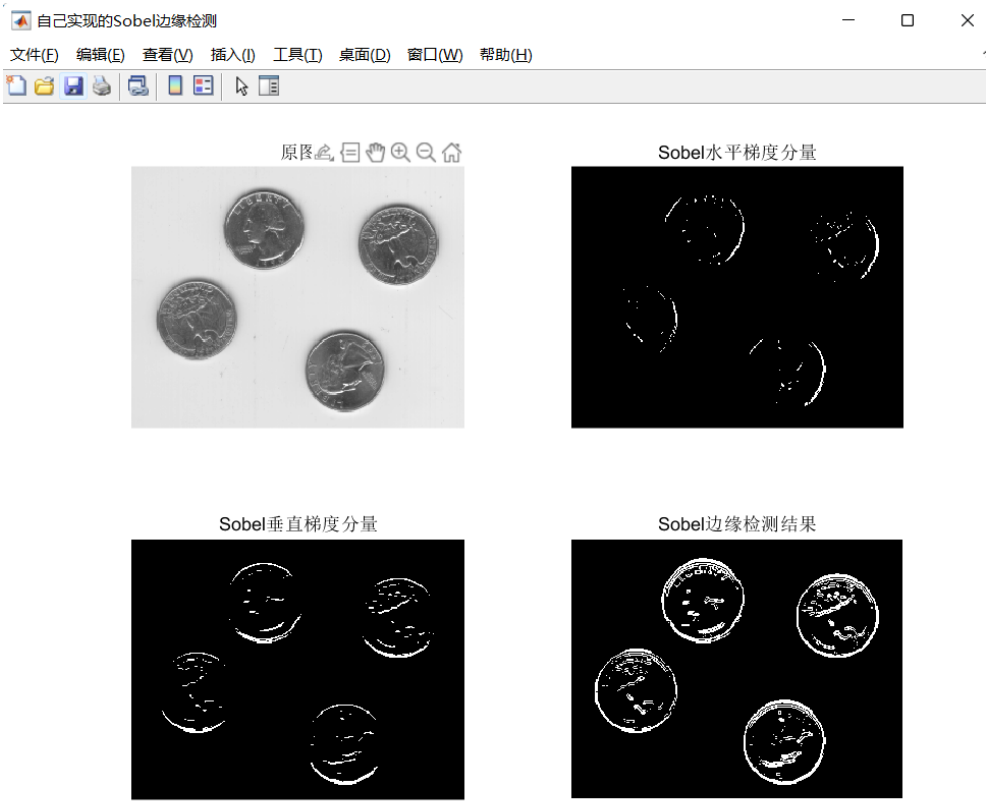
最终处理结果如下图：



4、实验分析

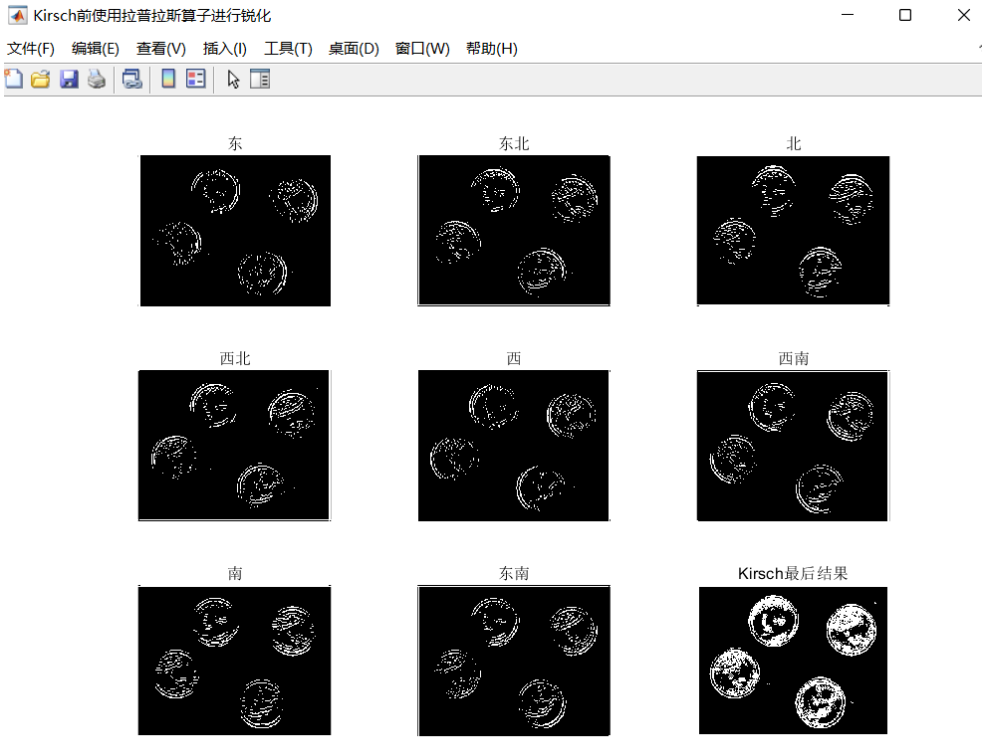
（1）尝试对边缘检测中出现的问题进行一定的改进

对比使用 matlab 自带函数 edge 和自己使用卷积计算得到的边缘，有明显的不同。对比 Sobel 算子的处理结果，可以发现在自己实现版本的处理结果中，边缘非常粗；而在 edge 的处理结果中，边缘比较细，甚至出现了一些断掉的地方。由于代码中使用 edge 处理时，没有给入阈值，阈值是 matlab 自己计算使用的，因此判断是自己实现时设置的阈值太小。调整阈值至 0.18 后，得到和 edge 处理结果差不多的结果，如下图所示：



事实上，在边缘检测位置偏移不大的情况下，将边缘计算的越细越好。对于固定阈值的二值化来说，由于计算出来的边缘粗细是不均匀的，这就导致边缘某些地方达到想要的细度时，另外一些地方的边缘可能已经断了。考虑到边缘检测利用的是像素值的“突变”，为了

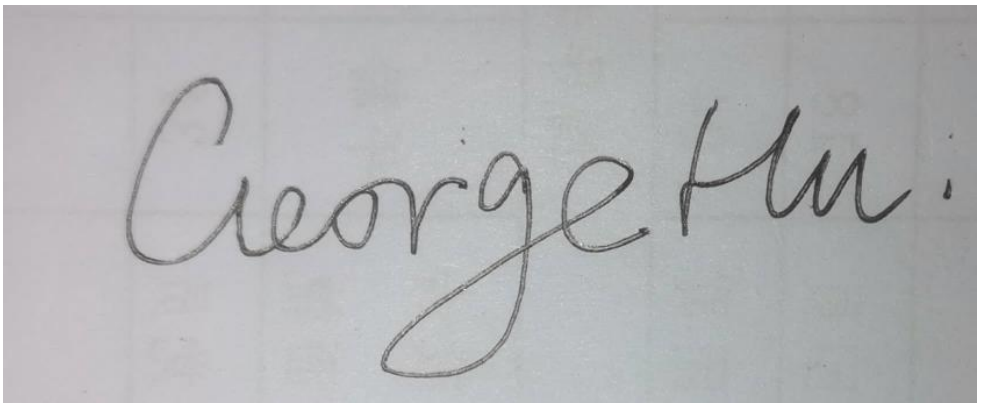
使边缘的粗细更加均匀，使得调节阈值时边缘的各部分能够一起变细，尝试先对图片进行均值滤波去噪后进行锐化以增强边缘利于检测。此处选用的时 3x3 均值滤波以及大小为 3 的拉普拉斯算子锐化，使用的阈值为 0.09 得到的结果如下图：



可以观察到，对“使得边缘更均匀”上确实有一定作用，硬币最外圈的边缘可以调节到比较细的程度。

（2）对图像分割应用的尝试

很显然，在图片中提取文字、物品等信息时，必然先要进行图像分割，提取出目标物。此处对两张实际的图像进行处理。原图如下，一个是车牌，另一个是手写的文字。



使用的是上面自己实现的直方图最优阈值法，得到的处理结果如下：

可以观察到，这种直方图最优阈值法对图片中光照明暗特别敏感。在车牌的处理中，上方有一定阴影，直接导致结果非常不理想。在手写英文名中，水笔写的字会有反光，导致分割出来的字非常不连续。想到对原图先进行对比度增强以克服这些问题，使用了直方图修正的增强，效果非常不好，如下图：

原图

分割后图像

原图

分割后图像

这就给我们了一个启示：在图像处理的过程中，尤其要注意在现实生活中图像多是来源于照片，非常容易收到现场光照情况的影响，需要特别考虑这一点进行克服。