计算机科学与技术　　学院

计算机视觉 课程实验报告

学号：	姓名：	班级：
实验题目：实验 E7：图像结构 2		

实验内容：

实验 7.1 Canny 边缘检测

- 了解 OpenCV 中 canny 边缘检测函数的用法，并选取图像进行测试，观察阈值对结果的影响。

实验 7.2 霍夫变换

- 实现基于霍夫变换的图像圆检测（边缘检测可以用 opencv 的 canny 函数）。

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

一、 Canny 边缘检测

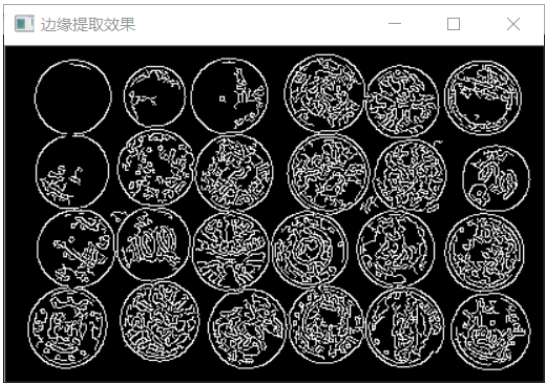
主要测试代码如下，调用 opencv 库的 Canny 函数，注意需要先将图像转为灰度图，然后进行滤波去除噪声可以使边缘提取的效果更好。

```
Mat img = imread("9999.jpg");
imshow("原始图", img);
Mat DstPic, edge, grayImage;
//创建与 src 同类型和同大小的矩阵
DstPic.create(img.size(), img.type());
DstPic = Scalar::all(0);
//将原始图转化为灰度图
cvtColor(img, grayImage, COLOR_BGR2GRAY);
//先使用 3*3 内核来降噪
blur(grayImage, edge, Size(5, 5));
//运行 canny 算子
Canny(edge, edge, T1, T2, 3);
imshow("边缘提取效果", edge);
waitKey(0);
```

调整 Canny 的两个上下阈值得到的边缘图像如下：

（原图）

（1）固定 T1 = 10, 增大 T2:



(2)固定 T2 = 90, 递增 T1:

测试后发现：

1、高的那个阈值是将要提取轮廓的物体与背景区分开来，就像阈值分割的参数一样，是决定目标与背景对比度的；

2、低的阈值是用来平滑边缘的轮廓，有时高的阈值设置太大了，可能边缘轮廓不连续或者不够平滑，通过低阈值来平滑轮廓线，或者使不连续的部分连接起来。

3、T1，T2。大于 T1 的称为强边界。T1 和 T2 之间的为弱边界。如果只有强边界，那么边界可能断断续续。而且会少分割。所以弱边界的作用就是解决这个问题。如果强边界点的连通区域内有弱边界点，那么认为该弱边界点为强边界。

二、 基于霍夫变换的图像圆检测

1. 背景介绍

查找圆的标准算法是 Hough 变换和 RANSAC（随机抽样一致）。

（1） 霍夫圆检测基本方法分类 [2]

1) **经典 HT，类似直线检测的变换空间方法，将圆参数化，在三维空间进行投票得到圆的方程；该方法对图像质量要求不高，噪声不敏感，效果较好，但计算量巨大，耗费时间和存储空间，难用于实际应用；**

2) **随机 HT,主要思想是在图像上随机选取几个点来确定一个圆，利用累加器来统计可能的圆心和半径；点的选取多种多样，如对称取点（利用圆上三个点便能确定一个圆），也可利用圆上的弦的垂直平分线过圆心的性质，随机取点后算平分线交点对圆心的位置对应累加等；随机 HT 相较于经典 HT，降低了内存需求和计算量，并且参数范围可无限大精度可无限高，但噪声敏感，对于复杂的图像引入了大量无效累积，需要针对性的复杂优化；**

3) **广义的 HT,(opencv 中实现的圆检测算法)，一般算法为取参考点，对于边缘像素点计算梯度角，对每一个梯度角，存储对应于参考点的距离和角度；算法具有较好的抗干扰性，但也需要较大的存储空间和计算量，如下是 opencv 圆检测算法的思路：**

1. 首先也是用 Canny 算子找边缘；

2.不同的是 HoughTransform 还利用 cvSobel 算子求梯度，得到 x 轴方向和 y 轴方向的梯度后就能够求边缘点梯度的方向，即是半径的方向了；

3.然后在半径方向上进行累加器计数，范围是所设定的最小半径到最大半径；

4.最后只有计数值大于一定阈值的点，并且是四邻域范围内最大的点才能作为圆心；

5.半径值是通过边缘点到已找到的圆心的距离来求得的，只要有足够多

的边缘点投某圆心一票，那该圆心和半径就能准确找到。

（2） RANSAC 圆检测基本思想 [4]

随机抽样一致是一种对观测数据进行最大化模型检验的方法，可概括为一个求解已知模型的参数的框架，传统 RANSAC 圆检测算法过程如下：

1)对待检测的图像进行预处理及边缘提取,建立由所有边缘点坐标构成的点集 D.令当前循环次数 k=0；

2)初始化局内点点集 Inpts=NULL.从边界点点集 D 中随机抽取 3 个点,计算这 3 个点所确定的圆的参数[a, b, r](圆心(a,b).半径 r),若圆的半径 r 的范围在预设的范围之内,则转 3);否则转 6).

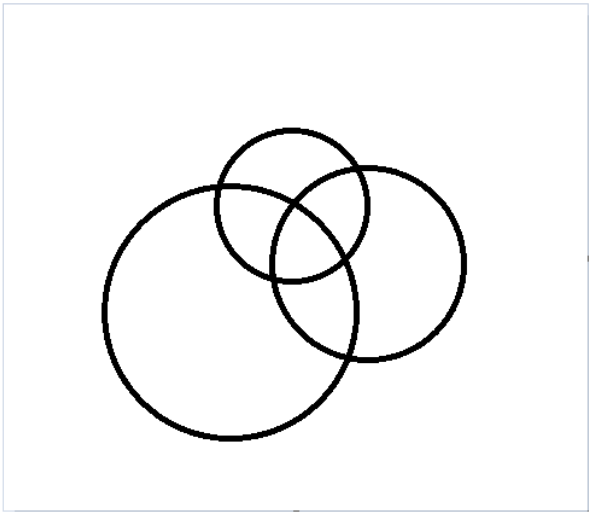
3)计算各边界点到 2)中所得到的圆的圆心的距离 d,若|d-r|≤ε（ε 为可接受的局内点偏离裕度),则认为该点为局内点，将其坐标存人局内点点集 Inpts，否则视为局外点.

4)计算该圆上的局内点点数 M,若 M 大于阈值 Mmin ,则认为此次估计出的圆模型足够合理,这些局内点也可视为有效点，转 5);否则转 6).

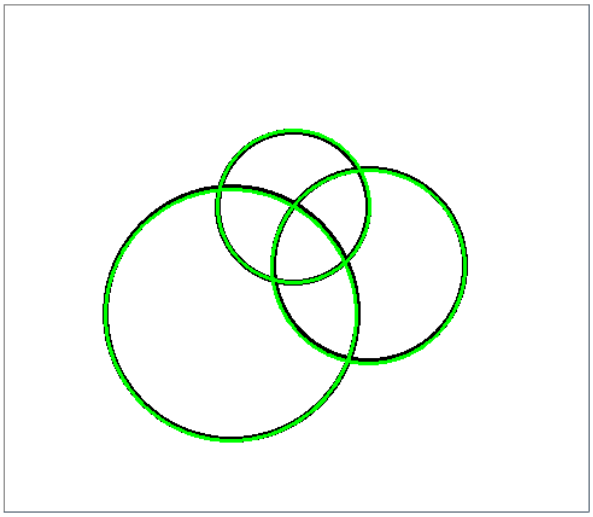
5)对点集 Inpts 中的所有点用最小二乘法重新计算圆的参数模型，得到最终结果.

6) k=k+1,若 h> K，则结束;否则转 2).

根据 git 上的实现代码测试效果如下：



* Original Image



* Circle detection

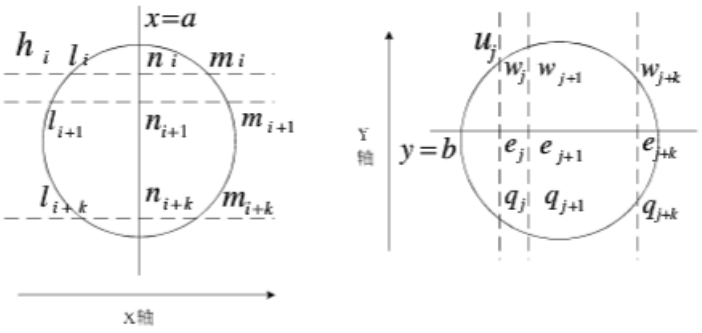
经过实验后发现，RANSAC 圆检测对于较简单的图像速度较快，效率较高检测结果较好，但对于复杂多噪声的图像，无目的抽样耗时较长（一张 500*500 的复杂图得二十分钟以上），检测结果准确性和稳定性较差。

2. 本次实验使用的圆检测算法介绍[3]

基本思想是利用圆的中心对称性，且圆的对称轴必过圆心，算法分两部分，先算出所有可能的圆心点，再对每个可能的圆心算半径。

利用霍夫的一维转换，对于每一行，扫描每一个非边缘像素点 pxj，对 pxj 的左右相邻像素点进行搜索，找到左右相邻的第一个边缘像素点，记为 mxp（向左搜索），lxq（向右搜索）。看是否有：pxj-mxp=lxq-pxj（pxj，mxp，lxq 取横坐标值） 如果有证明这一列有可能是圆的对称轴，在一维空间中采用 Hough 变换对它们对应的点进行累加计数，计数最大值对应的参数坐标即为圆心的 x 轴的坐标。如下图：

对每一列也可进行同样的扫描，得到圆心可能在的行号，即圆心的 y 轴坐标，最后根据得到行号和列号，构建一个二维空间（rows*cols），空间的坐标点上的值即为对应行号和列号在扫描中得到的累加值之和，取若干（提前设定最大检测圆的数量）累加值最大的坐标位置作为候选圆心。



（a）获取圆心横坐标　　（b）获取圆心纵坐标

图 1　圆心的获取

Fig.1 Detect circle center

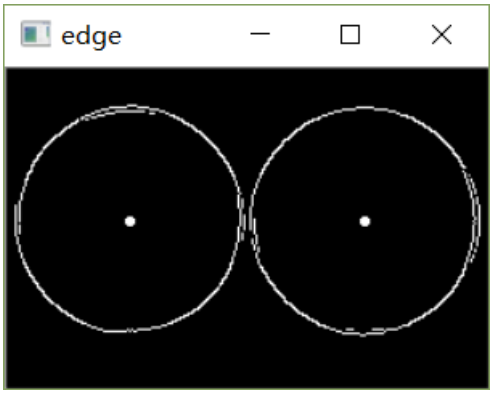
计算圆心半径：利用圆心坐标（a，b），将边缘像素点 pij 代入圆方程（x-a）2+（y-b）2=r2，计算出一个候选半径 r，在一维空间中采用 Hough 变换对候选半径 r 进行累加计数。同时进行检测排除（1）：看 r 的计数值 A（r）是否大于构成圆允许的最小点数 Tm=λ×πr（λ为比例系数，本文中λ=0.5）来确定真圆，r 即为该圆的半径。（2）对于某一圆心若其算出来的最可能半径大与图像宽度或高度的一半，则可判定该圆为无效圆进行舍弃。

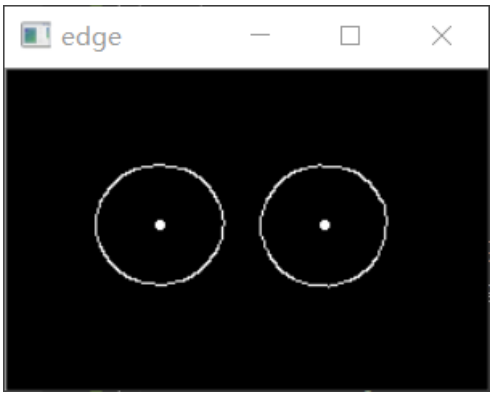
该算法首先大大降低了复杂度，按照该方法，对复杂度进行分析。计算圆心时，在一维空间中进行累加计数求出圆心坐标，共进行了 2×N 次计算，复杂度函数为：f1（N）=O（2N）（假设图中有 N 个像素点），同理，计算半径时，共进行了 N 次计算，故算法在执行时共进行了 3×N 次计算，复杂度函数：

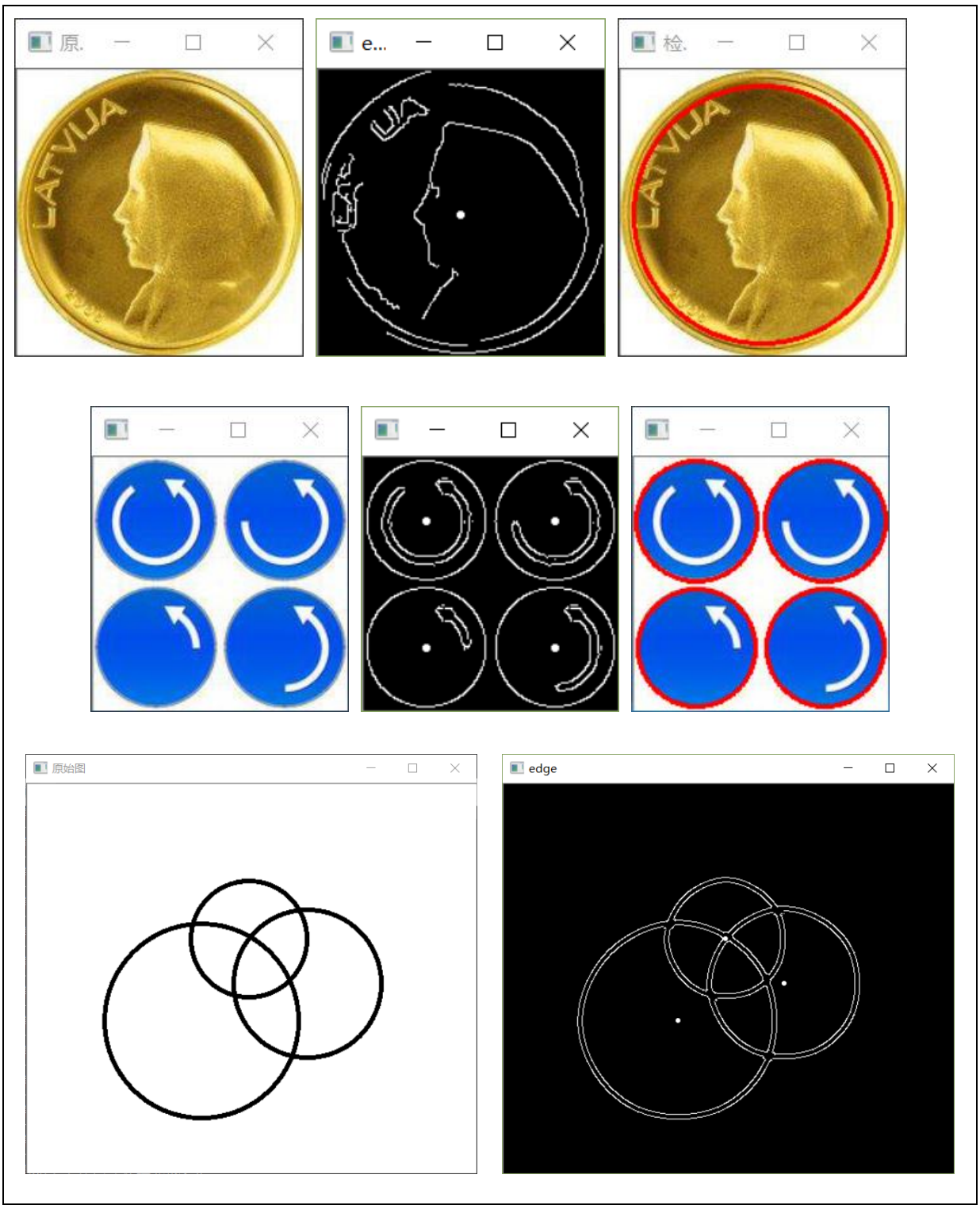
f（N）=f1（N）+f2（N）=O（3N）

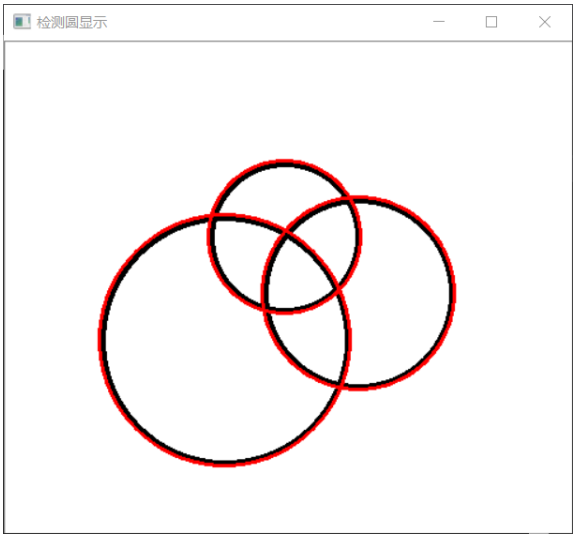
比传统 HTO(N^3)大大降低了运算成本。

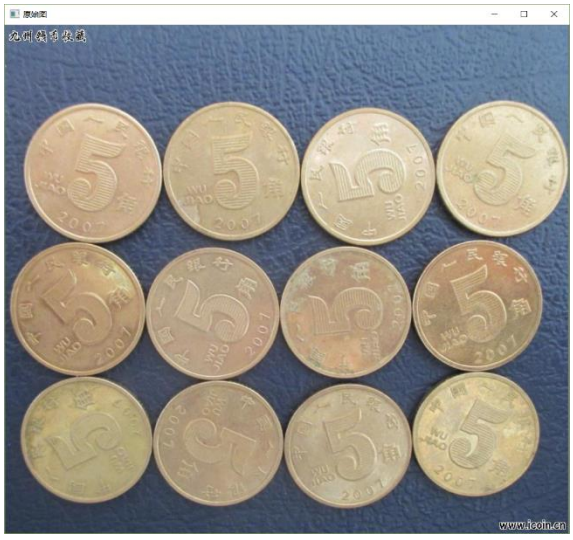
测试结果展示：

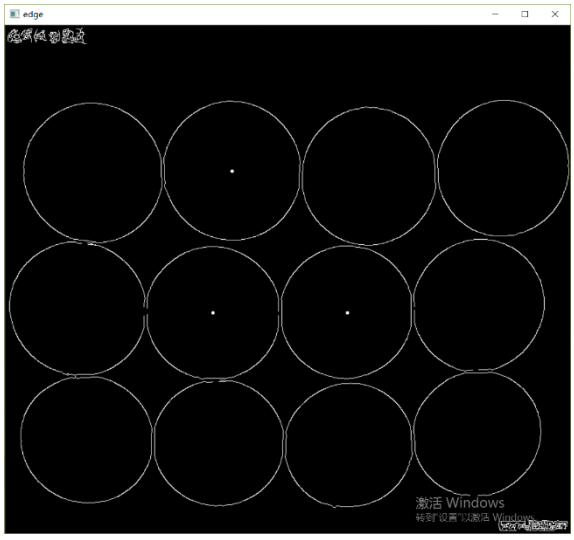


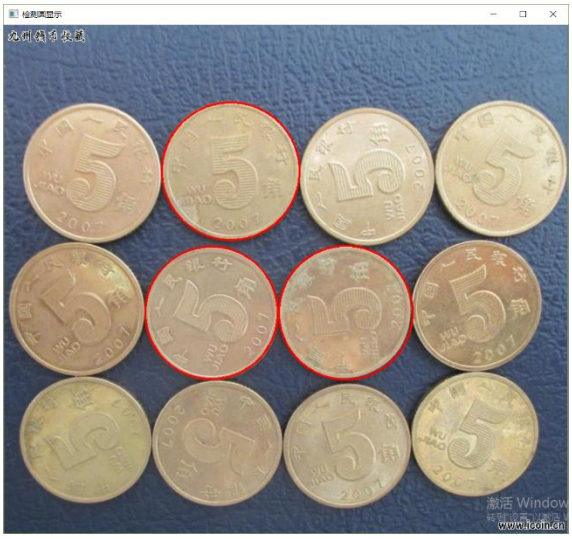



原.
e...
检.
LATVIJA
LATVIJA
LATVIJA
原始图
edge









算法总结，利用圆对称性该进的 Hough 圆检测算法对比传统的霍夫检测（包括）和 RANSAC 算法在检测较少的圆、噪声少的图像时有明显优势，速度非常快（基本秒算），检测的圆效果较好，比随机的 HT 有更高的可靠和稳定性，但是在抗其它干扰（如多个圆对称排布、正方形同样的中心对称的图形时）效果较差，能保证检测速度（不超过 30

秒），但不一定所有圆都能检测出来。另外本次实验的代码还需要进一步优化，特别对于候选圆的筛选判断可以进一步完善，可能回达到更好效果。

参考文献与链接：

[1]改进的 RANSAC 圆检测算法——邓仕超，高 阳，韩海媚
[2]基于 Hough 变换的圆检测方法——朱桂英，张瑞林
[3]改进的 Hough 变换圆检测算法（Improved Hough transform circle detection algorithm）——尚璐，李锐，宋信玉，（重庆大学 光电技术及系统教育部重点实验室，重庆 400030）
[4]RANSAC 圆检测实现代码 GitHub 链接：https://github.com/Yiphy/Ransac-2d-Shape-Detection/