计算机视觉 Ex3 实验文档

梁俊华 数据科学与计算机学院 16340129

一、实验要求

1、实验 3-1 直线的霍夫变换

输入图像：普通 A4 打印纸，上面可能有手写笔记或者打印内容，但是拍照时可能角度不正.

输出图像：

1）图像的边缘点(输出边缘图像 I_{edge})

2）计算 A4 纸边缘的各直线方程，输出如下结果:

（a）各个直线的参数方程；

（b）在上面的边缘图 I_{edge} 绘制直线，用蓝色显示,得到图像 I_2 ；

（c）在 I_2 图上显示 A4 纸的相关边缘点，用红色点显示，得到图像 I_3 .

3）输出 A4 纸的四个角点（在 I_3 上用半径为 5 的圆绘制角点，得到图像 I_4 ）

2、实验 3-2 圆的霍夫变换

输入图像：图像上面有若干硬币.

输出图像：

1）图像的边缘（输出边缘图像 I_{edge} ）

2）把图像中边缘拟合成圆（在图像 I_{edge} 绘制出对应圆形，用蓝色显示，得到图像 I_2 ），圆周相关的像素（在 I_2 上显示与圆形相关的像素，用红色显示，得到图像 I_3 ）.

3）输出图像中硬币的数量

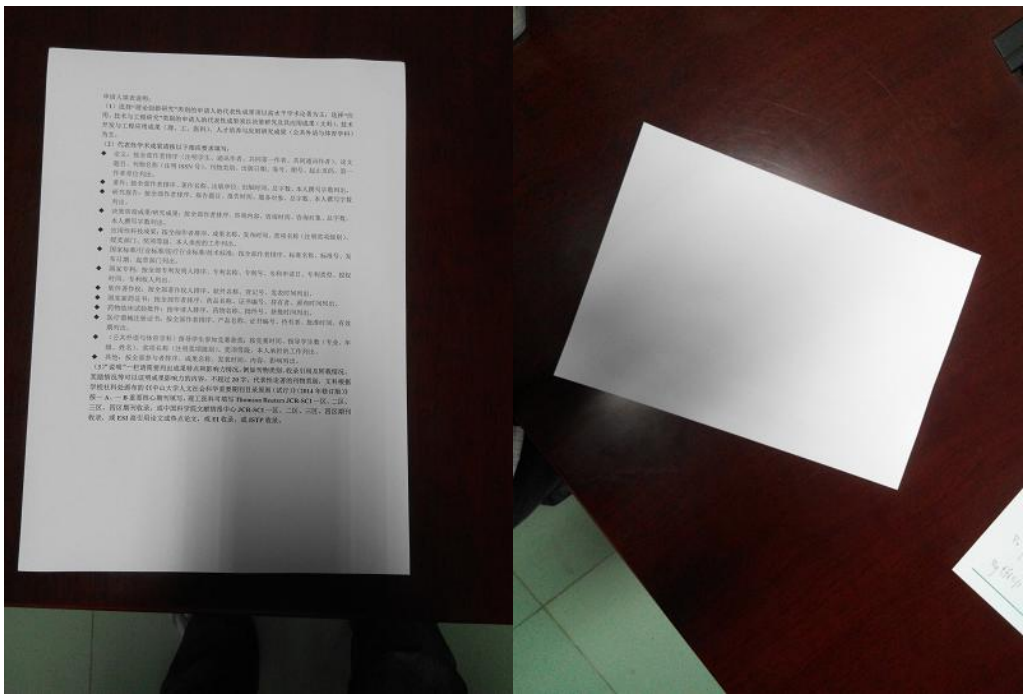
二、测试环境

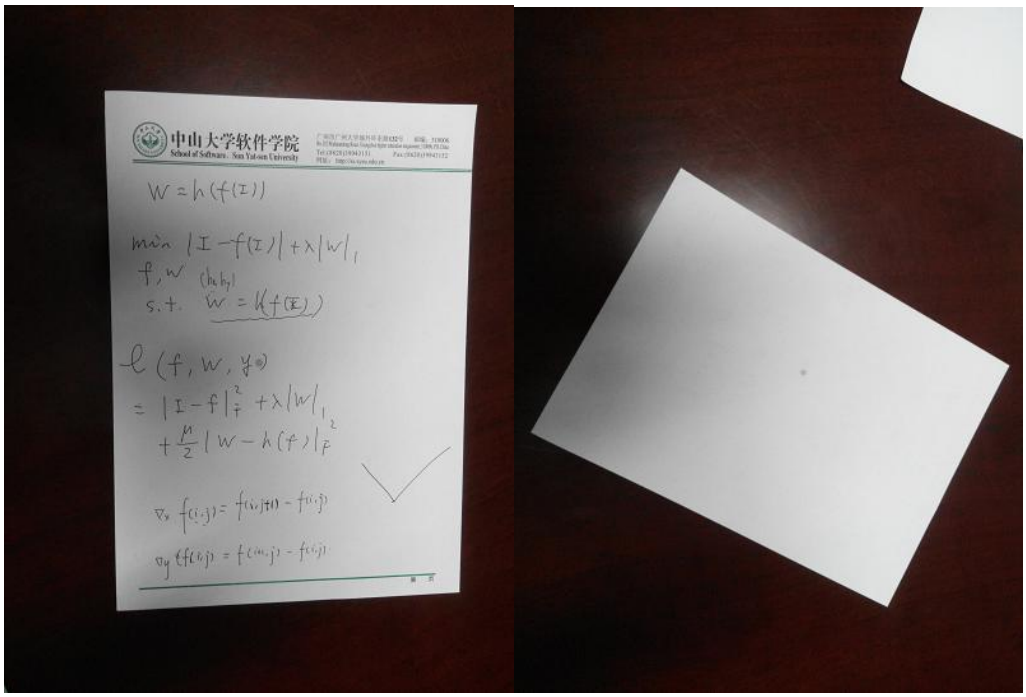
实验三的测试环境为 Windows10 操作系统，使用 MinGW64 进行编译.

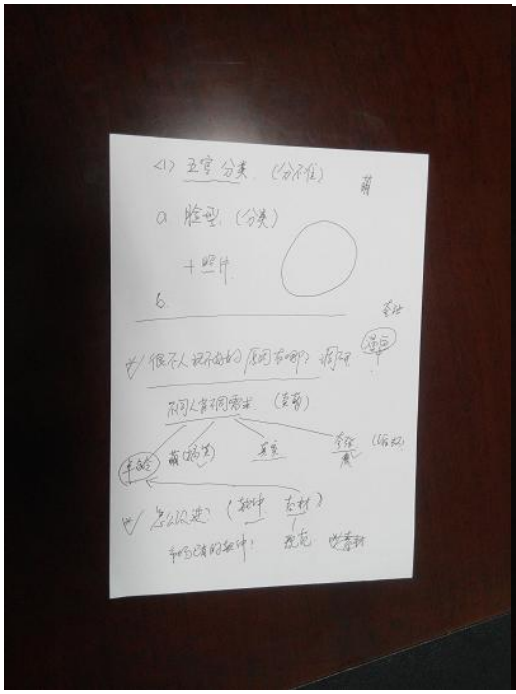
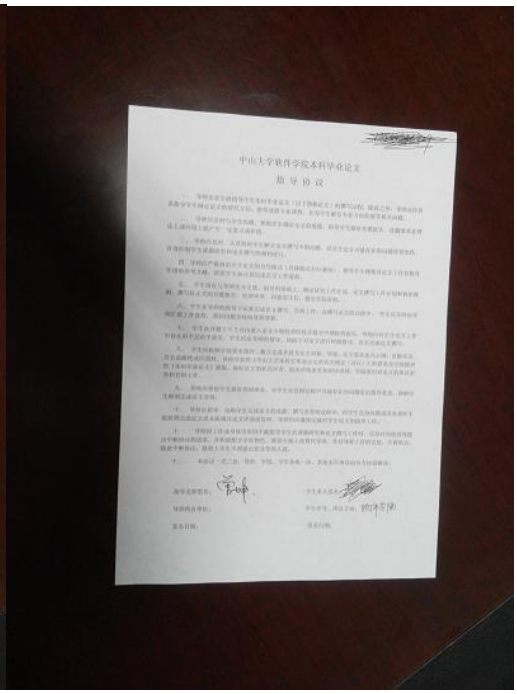
三、测试数据

实验 3-1 的测试数据为 Dataset1 文件夹中的 6 张图片，实验 3-2 的测试数据为 Dataset2 文件夹中的 6 张图片，转换为.bmp 文件后，分别存放于文件夹 Test1 和 Test2。测试数据的图片如下所示：

Dataset1（分别为 Line1-Line6）：





Dataset2（分别为 circle1-circle6）：



四、测试说明

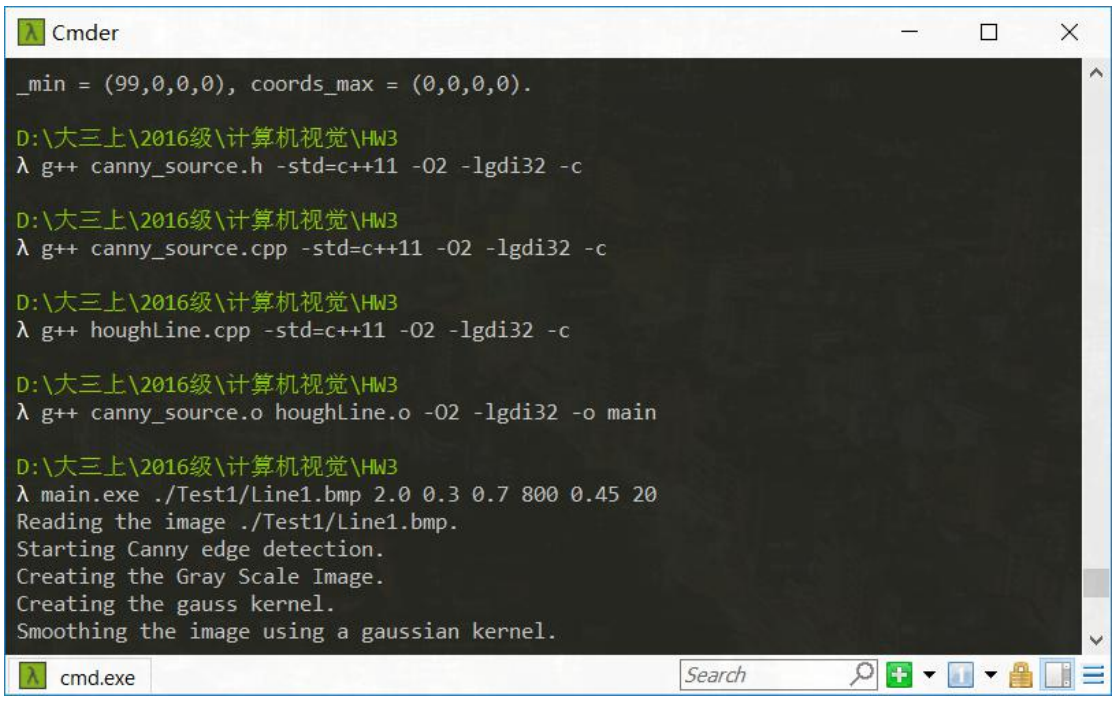
本次测试的文件主要有 5 个，分别为：

CImg.h
canny_source.h
canny_source.cpp
houghLine.cpp
houghCircle.cpp

实验可以之间按下列提示参数运行 mainLine.exe 和 mainCircle.exe.

实验也可分两次编译，分别进行 Ex3-1 和 Ex3-2，打开命令行，进入文件所在的目录后，分别进行下面的编译，即可分别进行 Ex3-1 和 Ex3-2 的实验测试（本次实验为防止图片过大而造成的栈溢出或者时间过长等问题，将.bmp 图片缩小尺寸后进行实验，实验的图片均保存在 Test1 和 Test2 文件夹）：

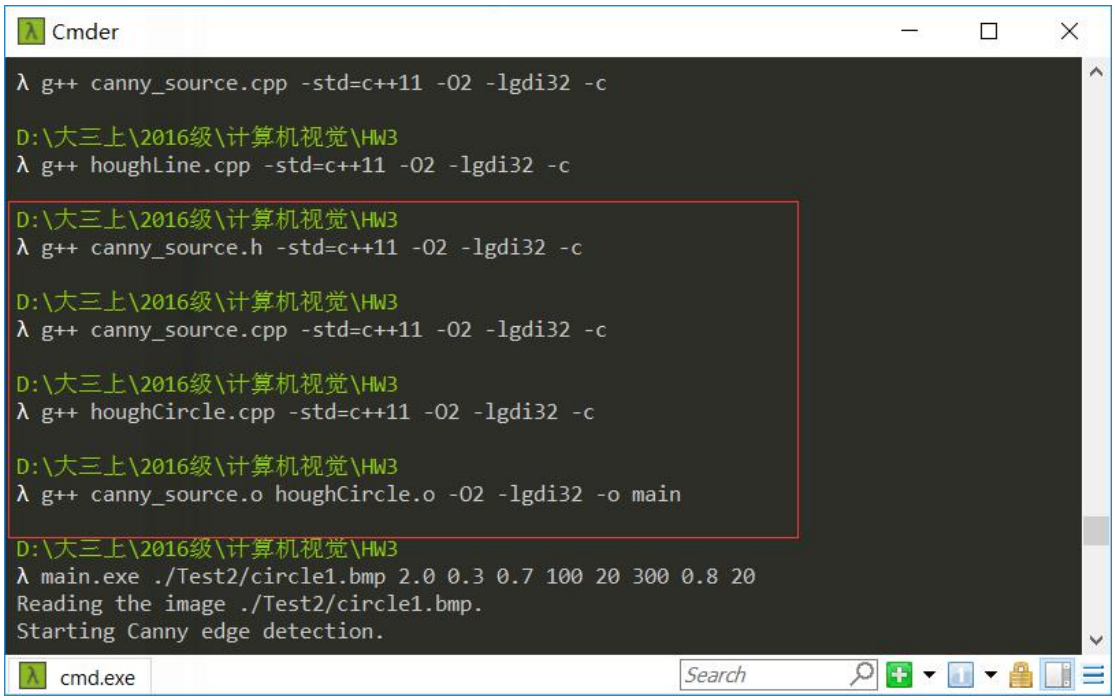
Ex3-1 的实验：



上图是测试是测试 Line1.bmp 的命令，测试其他的直线，只需要将上面 main.exe 后面的 Line1.bmp 改为 Line2.bmp 等即可，为了达到最好的测试效果，建议测试 Line1-Line6 的参数如下所示：

Line1: main.exe ./Test1/Line1.bmp 2.0 0.3 0.7 800 0.45 20
Line2: main.exe ./Test1/Line2.bmp 2.0 0.3 0.7 800 0.45 20
Line3: main.exe ./Test1/Line3.bmp 2.0 0.3 0.7 800 0.45 20
Line4: main.exe ./Test1/Line4.bmp 2.0 0.3 0.7 800 0.45 20
Line5: main.exe ./Test1/Line5.bmp 2.0 0.3 0.7 800 0.70 20
Line6: main.exe ./Test1/Line6.bmp 2.0 0.3 0.7 800 0.45 20

Ex3-2 的实验：



上图是测试是测试 circle1.bmp 的命令，测试其他的直线，只需要将上面 main.exe 后面的 circle1.bmp 改为 circle2.bmp 等即可，为了达到最好的测试效果，建议测试 circle1-circle6 的参数如下所示：

Circle1: main.exe ./Test2/circle1.bmp 3.0 0.3 0.7 500 50 300 0.8 30
Circle2: main.exe ./Test2/circle2.bmp 3.0 0.3 0.7 500 50 300 0.8 30
Circle3: main.exe ./Test2/circle3.bmp 3.0 0.3 0.7 180 40 300 0.7 30
Circle4: main.exe ./Test2/circle4.bmp 3.0 0.3 0.7 200 50 300 0.8 30
Circle5: main.exe ./Test2/circle5.bmp 3.0 0.5 0.7 200 40 300 0.8 30
Circle6: main.exe ./Test2/circle6.bmp 3.0 0.3 0.7 200 40 300 0.7 30

Ex3-1 参数说明（总共输入 7 个参数，下面依次说明 7 个输入参数）：
1）测试图片的路径（要求为.bmp 文件）
2）高斯模糊的方差
3）滞后处理的低阈值（范围 0-1）
4）滞后处理的高阈值（范围 0-1）
5）需要删除边缘点少于某个范围的范围值
6）投票数的阈值需要大于的值因子（范围 0-1）

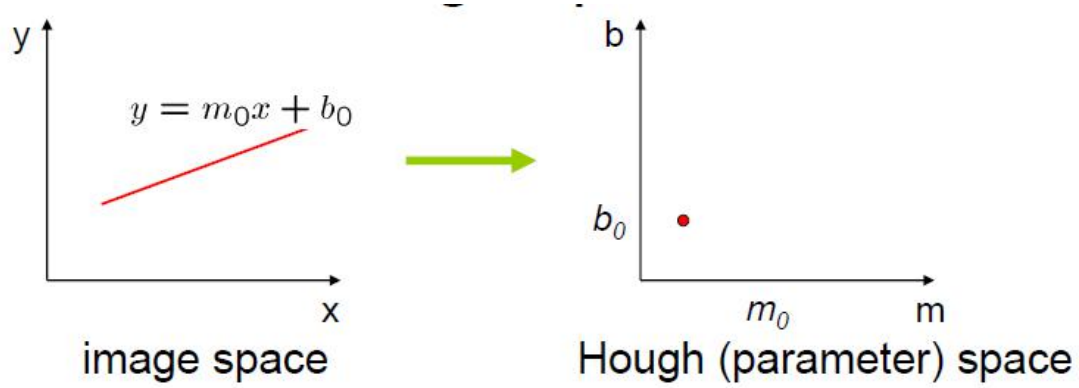
7）判断霍夫空间中是否为同一个点的距离

Ex3-2 参数说明（总共输入 9 个参数，下面依次说明 9 个输入参数）：
1）测试图片的路径（要求为.bmp 文件）
2）高斯模糊的方差
3）滞后处理的低阈值（范围 0-1）
4）滞后处理的高阈值（范围 0-1）
5）需要删除边缘点少于某个范围的范围值
6）空间霍夫空间圆半径的最小值
7）空间霍夫空间圆半径的最大值
8）投票数的阈值需要大于的值因子（范围 0-1）
9）判断霍夫空间中是否为同一个点的距离

五、实验原理

实验的基本原理是霍夫变换，霍夫变换是一种根据边缘来对可能的图形的位置进行“投票”的方法，在其最初的形式化中，每一个边缘点为通过它的所有可能直线进行投票，然后检查那些对应着最高累加器或者区间的相应图形来寻找可能的匹配模型。

在直线检测的过程中，给定一系列的数据点(x_i, y_i)，确定直线模型$y = mx + b$中的参数(b, m)，则由$b = y - mx$可以确定参数空间里面的点(b_i, m_i)，然后将该直线上的参数投一票，在所有的点投票完毕后，投票数最多的就是最佳的参数模型，这就是直线霍夫变换实验的原理。



在圆的霍夫变换中，原理和直线的基本相同，但是根据圆的参数方程：

$$\begin{cases} x = a + r\cos\theta \\ y = b + r\cos\theta \end{cases}$$

得：

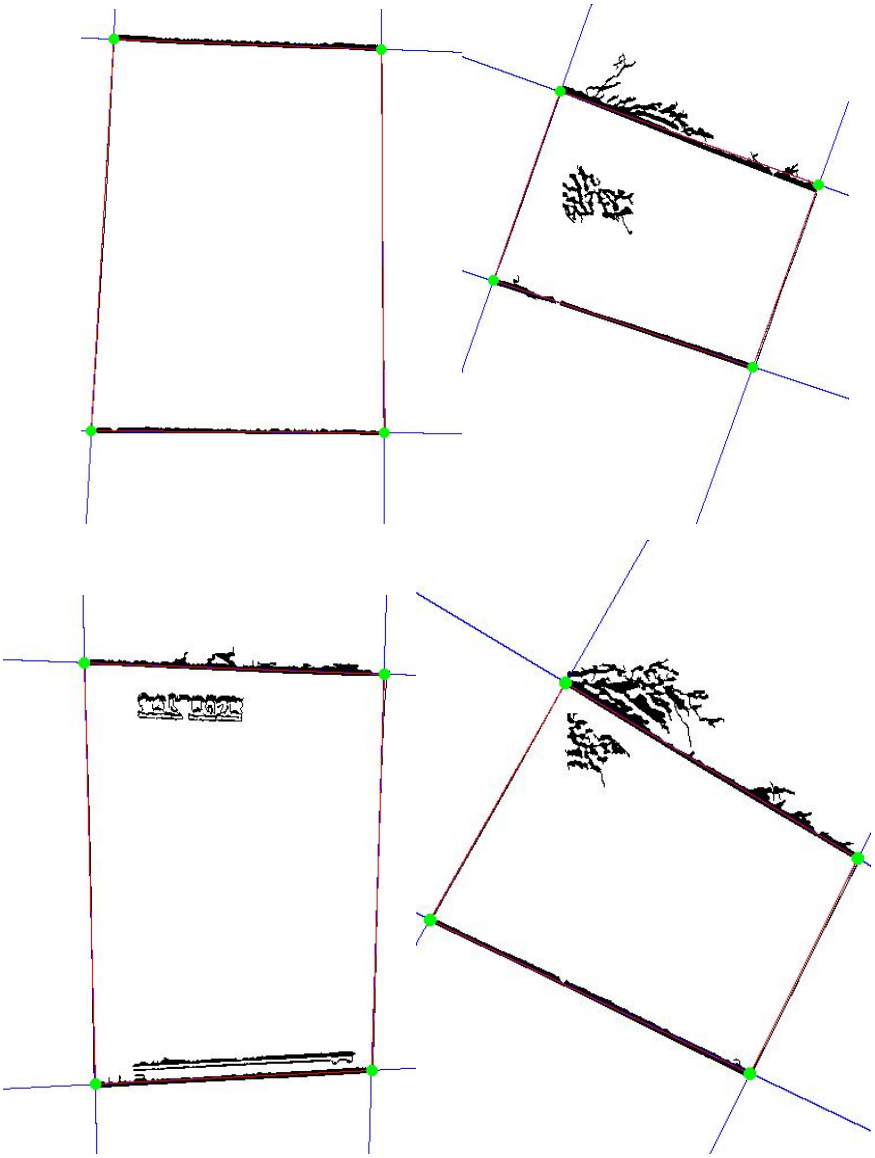
$$\begin{cases} a = x - r\cos\theta \\ b = y - r\cos\theta \end{cases}$$

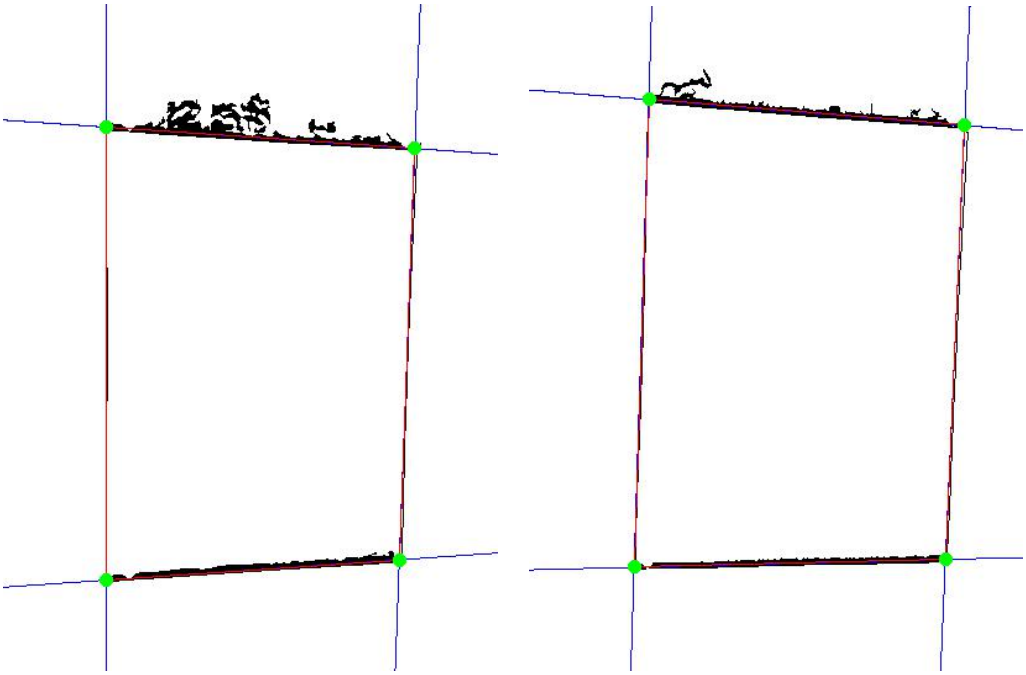
知圆的霍夫空间为一个三维空间，因此需要设置一个三维空间进行投票，其余的原理和直线的霍夫检测相同。其余要做的就是设置一定的阈值进行边缘点的过滤和投票数最大的点进行过滤，还需要先对图像进行 canny 边缘检测，canny 边缘检测的算法已经在上一次实验中完成，因此本次实验直接在上一次实验的基础上进行。

六、实验结果

实验结果分为两个部分，分别是 Ex3-1 和 Ex3-2，在下面实验结果的展示中，主要展示实验的最终过程（即 Ex3-1 的 I_4 和 Ex3-2 的 I_3）.

Ex3-1 实验结果如下所示（从左往右，从上往下分别为 Line1-Line6）：





Line1-Line6 的直线方程分别如下图所示：

Line1：

```
Line Equation: -0.037690x + 0.999289y = 30.270832
Line Equation: 0.998402x + 0.056519y = 35.883796
Line Equation: -0.999980x + 0.006283y = -314.305886
Line Equation: -0.006283x + 0.999980y = 441.738061
Intersection0 : x = 34, y = 31
Intersection1 : x = 314, y = 42
Intersection2 : x = 10, y = 441
Intersection3 : x = 317, y = 443
```

Line2：

```
Line Equation: 0.942991x + 0.332820y = 126.126565
Line Equation: -0.338738x + 0.940881y = 51.349343
Line Equation: -0.314987x + 0.949096y = 262.309559
Line Equation: 0.940881x + 0.338738y = 410.296654
Intersection0 : x = 101, y = 90
Intersection1 : x = 32, y = 286
Intersection2 : x = 300, y = 376
Intersection3 : x = 368, y = 187
```

Line3：

```
Line Equation: -0.037690x + 0.999289y = 62.270832
Line Equation: -0.999684x + 0.025130y = -77.176842
Line Equation: 0.999507x + 0.031411y = 370.350806
Line Equation: 0.050244x + 0.998737y = 475.709105
Intersection0 : x = 78, y = 65
Intersection1 : x = 368, y = 76
Intersection2 : x = 89, y = 471
Intersection3 : x = 356, y = 458
```

Line4：

```
Line Equation: 0.867071x + 0.498185y = 176.425934
Line Equation: -0.514440x + 0.857527y = 39.786764
Line Equation: -0.431456x + 0.902134y = 295.382826
Line Equation: 0.893841x + 0.448383y = 472.383370
Intersection0 : x = 131, y = 125
Intersection1 : x = 12, y = 333
Intersection2 : x = 293, y = 468
Intersection3 : x = 388, y = 279
```

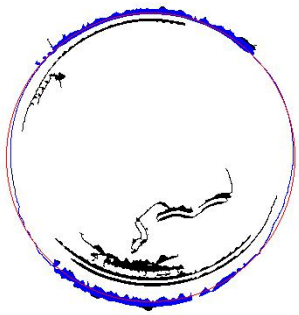
Line5：

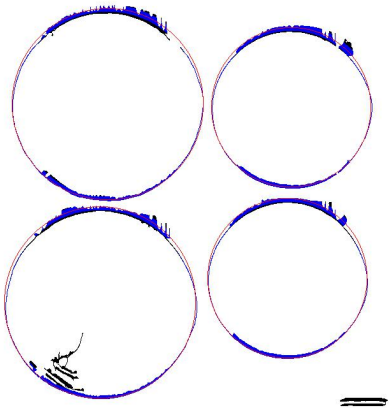
```
Line Equation: -0.069060x + 0.997613y = 93.545759
Line Equation: 1.000000x + 0.000000y = 83.000000
Line Equation: 0.999289x + 0.037690y = 336.996554
Line Equation: 0.069060x + 0.997613y = 470.169769
Intersection0 : x = 83, y = 99
Intersection1 : x = 332, y = 116
Intersection2 : x = 83, y = 465
Intersection3 : x = 320, y = 449
```

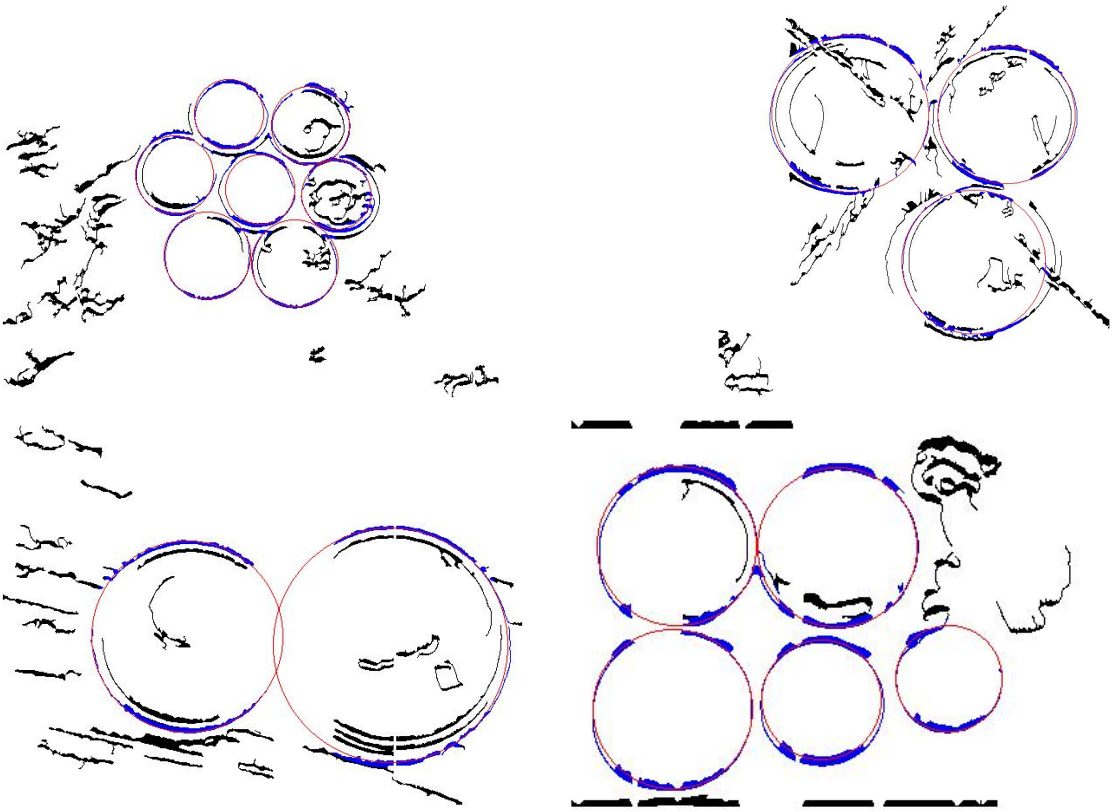
Line6：

```
Line Equation: -0.081591x + 0.996666y = 67.785012
Line Equation: 0.999507x + 0.031411y = 99.350806
Line Equation: 0.999033x + 0.043968y = 356.634011
Line Equation: 0.025130x + 0.999684y = 456.941066
Intersection0 : x = 97, y = 75
Intersection1 : x = 352, y = 96
Intersection2 : x = 85, y = 454
Intersection3 : x = 337, y = 448
```

Ex3-2 实验结果如下所示（从左往右，从上往下分别为 circle1-circle6）：



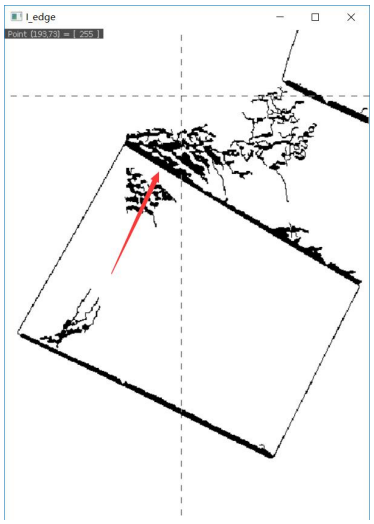




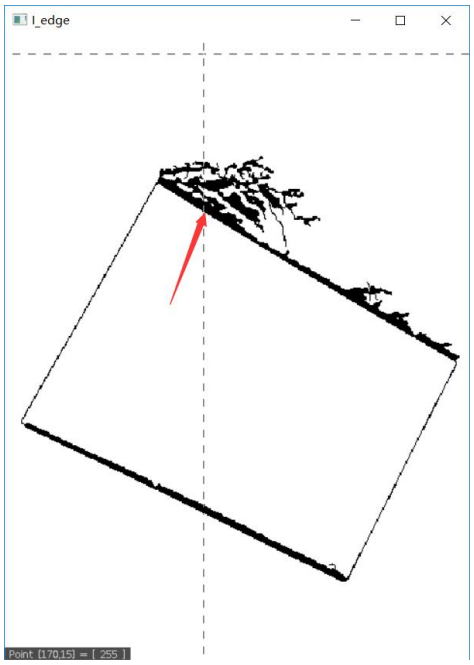
Ex3-2 中输出硬币的数目分别为：1，4，7，3，2，5（输出过程有专门程序实现）

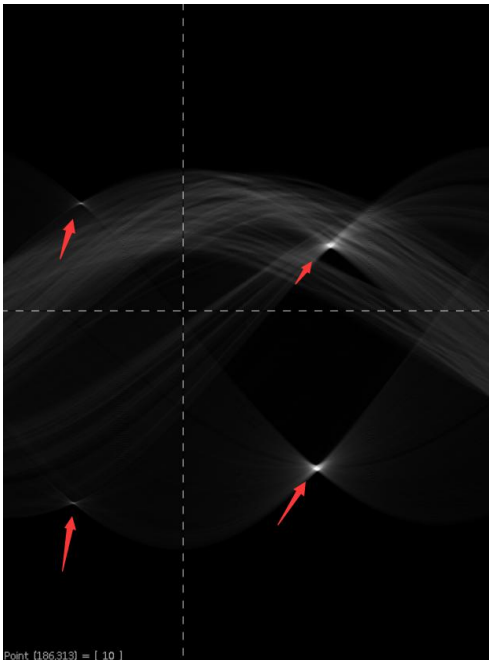
七、实验分析

Ex3-1 中，通过删除相关的干扰因素，可以使得直线检测的效果提高很多，而且准确度非常高，但是一些光照因素会给 A4 纸的边缘检测造成一些干扰，比如 Line4（左）中的光照部分会造成 Line4（右）边缘检测中的相关干扰，



为了使得这部分的干扰淡化，可以调整 sigma 的值模糊这部分，或者通过 canny 算法后调整删除边的大小将这部分删除，通过参数的调整（减小），可以达到下面效果，从而提升后序直线检测的准确度，调整后的图如下（左）所示：

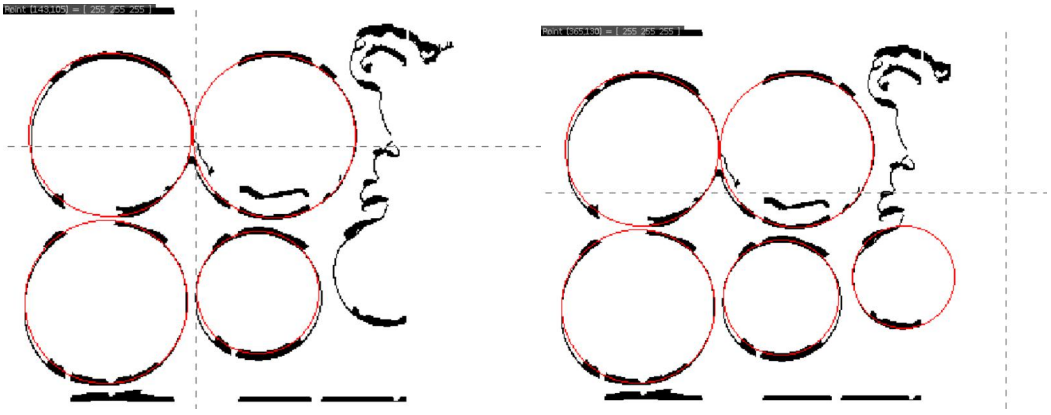




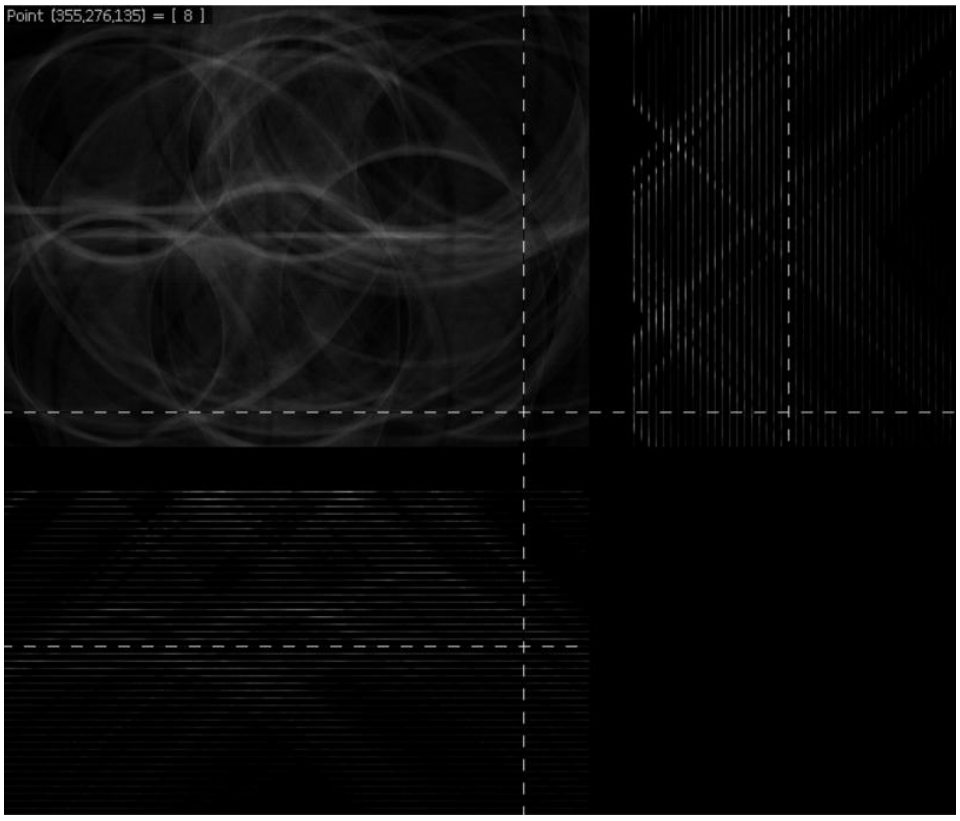
其次可以观察生成的直线的霍夫空间的图像，是一系列正弦函数，而这些正弦函数又相互相交，从而产生投票点较多的点，这些点对应的参数就是直线的参数，然后再通过找这些区域亮点的局部最大值就能够找出直线的方程，霍夫空间如上（右）所示（Line4）。

直线霍夫检测的生成时间是比较短的，因为只是一个对应到二维空间中，但是圆的霍夫检测的生成时间就比较长了，因为要对应到三维的霍夫空间，而且半径的遍历会造成很大程度的时间浪费，因此限制了半径的搜索范围来优化时间，但是依然是比较慢的。

至于圆的霍夫变换，分析的过程和直线类似，但是需要注意的是，要适当调整投票数的阈值大小，否则会导致某些半径搜索的丢失，以 circle6 为例，若设置参数为 3.0 0.5 0.7 200 40 300 0.8 30，则会少了一个圆（下左），但是只需要将阈值修改为 0.7，则会编程下右的图。



圆的三维霍夫空间如下所示：



总结而言，sigma 的值对于去除光照影响等因素有影响，一般而言 sigma 的值越大，噪音留下痕迹就约明显，但是又不是一个单调的过程。lenTodelete 的值也需要根据具体的图像来进行调整，否则容易把一些有用的边删除掉。rmin 和 rmax 的功能主要是优化三维霍夫空间的生成时间，但是不能过于夸张，否则会导致某些半径的圆被省略掉。至于投票阈值的设定则比较稳定，这里采取输入一个影响因子 t，由 $(\max Vote - \sqrt{\max Vote}) * t$ 计算得出阈值，在具体的测试中表明 t 的取值在 0.7-0.8 之间的效果会比较好，而寻找局部最大值的阈值则设为 20 比较好.

八、实验思考

1）如何在保证精度的结果条件下加快运行速度.

答：对于直线的霍夫检测，由于图片中大量的点都不是边缘点，因此在遍历的过程中会消耗大量的时间，因此可以考虑统计票数的过程中，只存储具有票数的边缘点，在最后找极值的过程中会省略大量的遍历时间，有点类似于创建一个稀疏矩阵的想法。这种做法在维数较高的情况下，对算法效率的改进是非常明显的。

2）如何提升圆的霍夫变换的速度.

答：圆的霍夫变换，在本次实验中主要用到的是遍历 $[r\min, r\max]$ 范围内的半径值，从而进行统计，但是这种做法会消耗大量时间。一种改进的方法是利用霍夫梯度法进行启发式优化，来缩短遍历的时间。