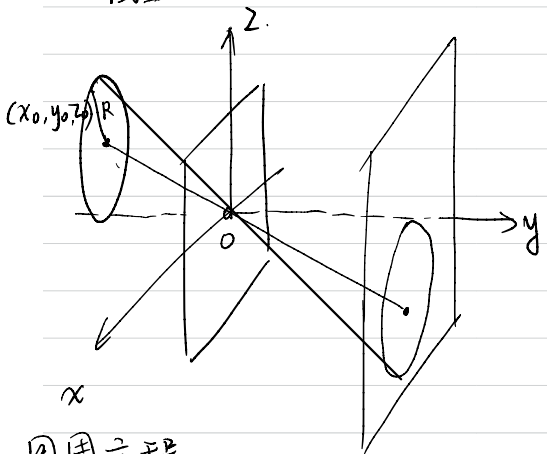
1) a. 圆盘



圆周方程：

$$(x-x_0)^2+(z-z_0)^2=R^2,\ y=y_0$$

或
$$\begin{cases} x = x_0 + R\cos\theta \\ z = z_0 + R\sin\theta \end{cases}$$

成像平面：

$$y=a$$

(x, y_0, z) 的像点为 (x', a, z')

$$\frac{x}{x'}=\frac{y_0}{a}=\frac{z}{z'}$$

$$\Rightarrow \begin{cases} x' = \frac{a}{y_0}x = \frac{a}{y_0}(x_0+R\cos\theta) \\ z' = \frac{a}{y_0}z = \frac{a}{y_0}(z_0+R\sin\theta) \end{cases}$$

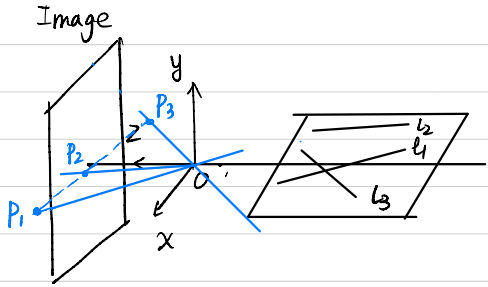
为原点在 $(\frac{ax_0}{y_0}, a, \frac{az_0}{y_0})$，半径为 $\frac{aR}{y_0}$ 的圆盘.

b.

Step 1. $A=C=D=0$, $B=1$

平面方程：$y=0$

Image



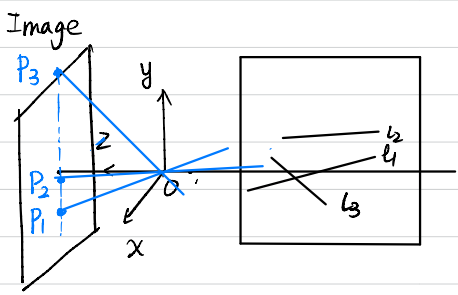
$\vec{l_1}=(x_1, 0, z_1)$ 为 l_1 的直线向量.

$P_1=(\frac{x_1 f}{z_1}, 0, f)$ 为 vanishing point

类似地，$\vec{l_2}=(x_2, 0, z_2)$，$\vec{l_3}=(x_3, 0, z_3)$

$P_2=(\frac{x_2 f}{z_2}, 0, f)$，$P_3=(\frac{x_3 f}{z_3}, 0, f)$

Image



Step 2. $A=1$, $B=C=D=0$

平面方程：$x=0$

$\vec{l_1}=(0, y_1, z_1)$ 为 l_1 的直线向量.

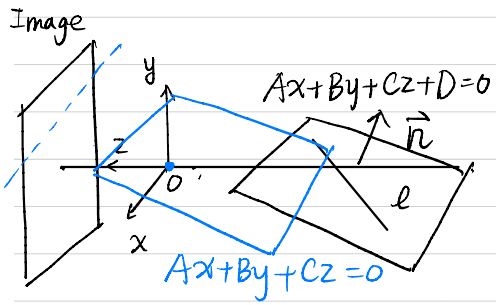
$P_1=(0, \frac{y_1 f}{z_1}, f)$ 为 vanishing point

类似地，$\vec{l_2}=(0, y_2, z_2)$，$\vec{l_3}=(0, y_3, z_3)$

$P_2=(0, \frac{y_2 f}{z_2}, f)$，$P_3=(0, \frac{y_3 f}{z_3}, f)$

c. 通过观察，特殊情况中，vanishing point 均落在物平面过原点的平行面与像平面的交线上，假设在一般情况下同样

成立，进行验证：



平面的法向量为 $\vec{n}=(A,B,C)$

l 在平面上，满足 $\vec{l}\cdot\vec{n}=0$

$\vec{l}=(x_0,y_0,z_0)$

$Ax_0+By_0+Cz_0=0$ ··· ①

过原点O作l的平行点，交像平面于P，即vanishing point.

(x_1,y_1,f)

$P=\left(\frac{x_0f}{z_0},\frac{y_0f}{z_0},f\right)$

计算两平面交线：

$$\begin{cases}Ax+By+Cz=0\\ z=f\end{cases}$$

$$\Rightarrow l'=\begin{cases}Ax+By+Cf=0\\ z=f\end{cases}$$

P代入l'方程左侧，代入①可得：

$$A\cdot\frac{x_0f}{z_0}+B\cdot\frac{y_0f}{z_0}+Cf$$

$$=(Ax_0+By_0)\frac{f}{z_0}+Cf$$

$$=-Cz_0\cdot\frac{f}{z_0}+Cf$$

$$=0$$

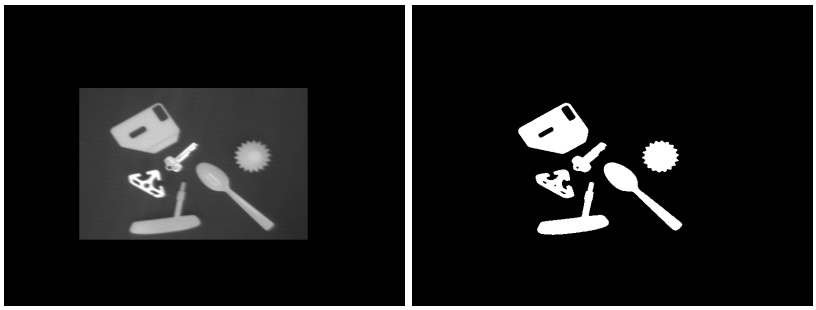
验证完毕，假设成立.

编程作业报告

作业一：物体分割

a. 将灰度图像处理为二值图像

原理较为简单，设定阈值为100，以 `many_objects_1.png` 为例，处理前后对比如下：(`many_objects_2.png` 结果见附录)

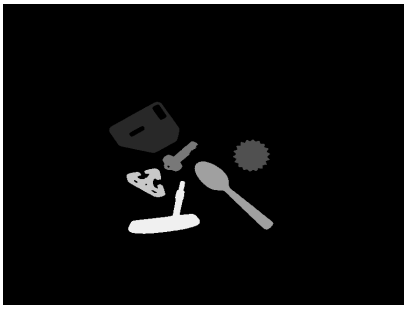


b. 提取图中强连通分量，灰度作为标记

1. 设置`label_distance = 40`，为不同标签之间的单位距离。
2. 遍历像素值为255的像素点，如果遇到未标记的像素点，则创建队列。
3. 加入该像素点至队列中。
4. 此后，若队不空，出队，标记为当前标签，加入上下左右像素为255的像素点。
5. 重复步骤4，直到队列空。更新标签 label+=label_distance，重复步骤2。

这种方法相较递归算法计算复杂度较低，可以在O(image_size)时间内完成。

处理结果如下：



c. 计算每个连通分量的质心位置、方位角、圆度

数学原理：

- 质心位置：连通分量里所有像素点坐标加权平均，方位角与圆度计算参考课程。

输出列表中第二个元素（完整见附录）：

```
{'x': 317,
 'y': 461,
 'orentation': -0.756879895721013,
```

```
'roundedness': 0.9875212700111777
}
```

输出角度为弧度，圆度为0-1之间的数值，越接近1越圆。与原图中的圆对应，其圆度为0.9875212700111777，接近1，说明圆度计算正确。

作业二：Hough 变换

a. 利用 Sobel 算子计算图像边缘

Sobel Kernel 的具体数值分别如下：

```
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
```

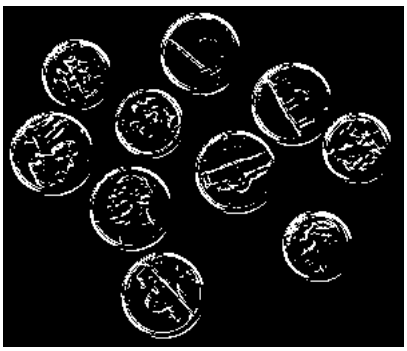
b. Hough 变换：把图像空间超过阈值的边缘点映射到参数空间

在离散的像素空间画图参考了维基百科-Midpoint_circle_algorithm

选定参数空间的范围：

- x, y: 不超过图像边界
- r: <=40

选定边缘阈值，超过阈值的边缘点才会通过过滤，不同阈值对应边缘结果不同，如下为 `threshold = 100` 时的结果：

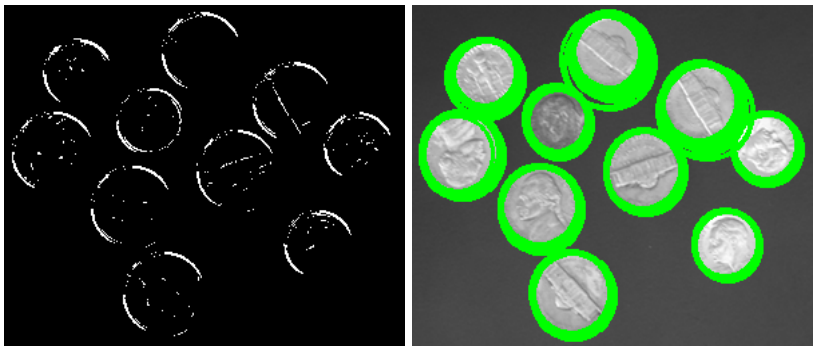


c. 投票法确定圆的中心位置与半径

[Test 1]

选取参数为：`edge_thresh_value = 200, vote_thresh_value = 40`

效果不理想：



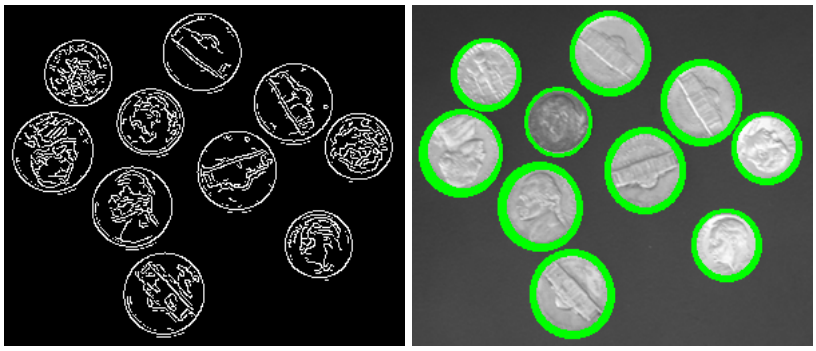
为确定问题，控制变量，进行Test 2如下。

[Test 2] 控制变量进行实验，确定问题

猜测可能是 Sobel 边缘检测不准确导致，使用 Canny 边缘检测：

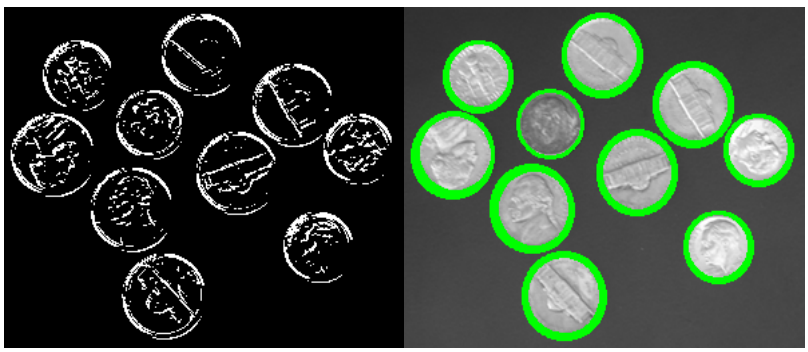
```
edge_image = cv2.Canny(gray_image, 75, 150)
```

选取参数为：`vote_thresh_value = 80`
效果较好：



[Test 3] 根据参数对阈值参数进行调试

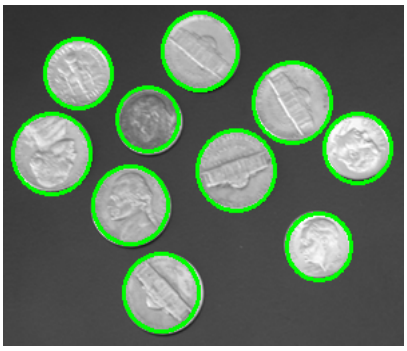
确定问题后，降低边缘阈值，使边缘像素点更多，提高投票阈值，参数如下：`edge_thresh_value = 100, vote_thresh_value = 80` 得到如下结果：



[Test 4] 对输出进行算法优化

观察输出发现，有些圆的边缘被检测为了多个圆，会被重复输出，但这些输出具有相同特征——半径相同，圆心之间距离较小。因此在筛选票数环节增加筛选，若已存在半径相同，圆心距离较小的圆，则不再添加该圆，

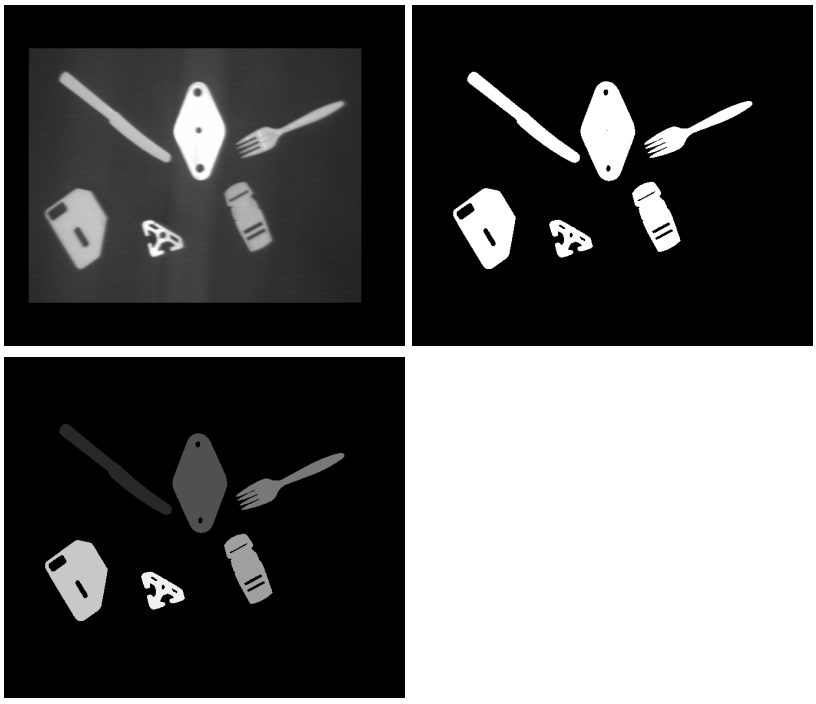
其余参数不变，结果如下：



输出：[(23, 82, 108), (24, 174, 236), (25, 49, 56), (25, 103, 265), (28, 144, 95), (28, 207, 118), (28, 33, 147), (29, 107, 36), (29, 119, 174), (29, 70, 216)] 恰好为位置不同的10个圆，图案较为清晰且没有重复。

附录

任务一 `many_objects_2.png`结果



输出：

```
[{'x': 178, 'y': 188, 'orentation': 0.9283523922633338, 'roundedness':
0.008793583672043125},
 {'x': 197, 'y': 332, 'orentation': 0.041169903748532084, 'roundedness':
0.3105344355149078},
 {'x': 196, 'y': 473, 'orentation': -1.168338272187214, 'roundedness':
0.0215800565346433},
 {'x': 331, 'y': 413, 'orentation': 0.455678905356984, 'roundedness':
0.18662301751973373},
 {'x': 347, 'y': 129, 'orentation': 0.17194384007168115, 'roundedness':
0.5497739192983098},
 {'x': 366, 'y': 266, 'orentation': 1.1024963334078555, 'roundedness':
0.47073699409167896}]
```

任务一 `many_objects_1.png` 完整输出


```
[{'x': 265, 'y': 266, 'orentation': -1.460754671970092, 'roundedness':
0.5399328967651852},
 {'x': 317, 'y': 461, 'orentation': -0.756879895721013, 'roundedness':
0.9875212700111777},
 {'x': 321, 'y': 326, 'orentation': -0.7998601259965585, 'roundedness':
0.14388076482916917},
 {'x': 390, 'y': 418, 'orentation': 0.7947532073590663, 'roundedness':
0.026382793555897142},
 {'x': 373, 'y': 268, 'orentation': 1.0405937556264675, 'roundedness':
0.47046065604465137},
 {'x': 451, 'y': 304, 'orentation': -1.112160429419226, 'roundedness':
0.29778576780976074}]
```