姓名

- 倪诗宇

学号

- 201900180065

实验日期

- 2021.11.01

实验题目

- 霍夫变换
- 实现基于霍夫变换的图像圆检测

实验过程中遇到的问题和解决办法

- 问题一：
  - 问题：在优化之前，使用循环遍历所有角度的方法。该方法的准确度比较高，但是最大的问题是该方法运行奇慢无比，根本无法拖动滑动条进行调参。
  - 解决方法：后来写了第二个版本，不用从 0 到 2π 遍历所有的角度，而是先将原图转化为灰度图，然后计算各个点的梯度对应的角度。为了保证精度，防止 Sobel 算子计算的梯度指向不准，采用在计算出来的角度上下 0.1 的浮动范围内进行检测的方法。这个方法速度相较于上一个方法有明显提升，拖动滑动条能够迅速得到结果，方便参数调节

    这里是投票函数的完整代码

    ```
    Sobel( binary_image, dx, CV_32F, 1, 0 , 3 );//得到 x 方向上的梯度
    Sobel( binary_image , dy, CV_32F, 0, 1 , 3 );//得到 y 方向上的梯度
    ```

    ```
    void vote(int x , int y){//输入的是坐标 (x,y)
        if(target_image.at<uchar>(y,x) != 0)
            for(int r = R_min ; r < R_max ; r++){
                for(double theta = -0.1 ; theta <= 0.1 ; theta = theta +
    0.05){
                    int a = int (x + r * cos(atan(dy.at<float>(y,x) /
    dx.at<float>(y,x)) + theta));
                    int b = int (y + r * sin(atan(dy.at<float>(y,x) /
    dx.at<float>(y,x)) + theta));
                    if(a >= 0 && a < target_image.cols && b >= 0 && b <
    target_image.rows){
                        H[r][a][b] += 1;
                    }
                }
            }
    }
    ```

- 问题二

- 问题：在识别圆的时候，半径都是对的，但是圆心明显不对
- 解决：

For every edge pixel (x,y) :

For each possible radius value r:

For each possible gradient direction θ:

// or use estimated gradient

$$a = x - r\cos(\theta)$$

$$b = y + r\sin(\theta)$$

$$H[a,b,r] \mathrel{+}= 1$$

end

end

这里的的 (x, y) 和 (a, b) 指的都是坐标。而我们在使用数组时，比如 $a[1][2]$ 访问的是数组的第二行的第三个元素，而这个元素的坐标应该是 $(2, 1)$ 。 找出的圆心不对是因为在访问数组元素时，错把索引当成坐标，也就是把 (y, x) 当成了 (x, y) 。

还需要注意的是，opencv中图像的访问比如

```
source_image.at<uchar>(x,y)
```

x 代表的是行信息， y代表的是列信息，这个点的坐标应该是 (y, x)

相反，$Point(x, y)$ 则指的是坐标为 (x, y) 的点

- 问题三：
  - 这里也不能说是问题，应该是需要注意的一个点。

$$a = x - r\cos(\theta)$$

  - PPT里的式子 这里的

$$b = y + r\sin(\theta)$$

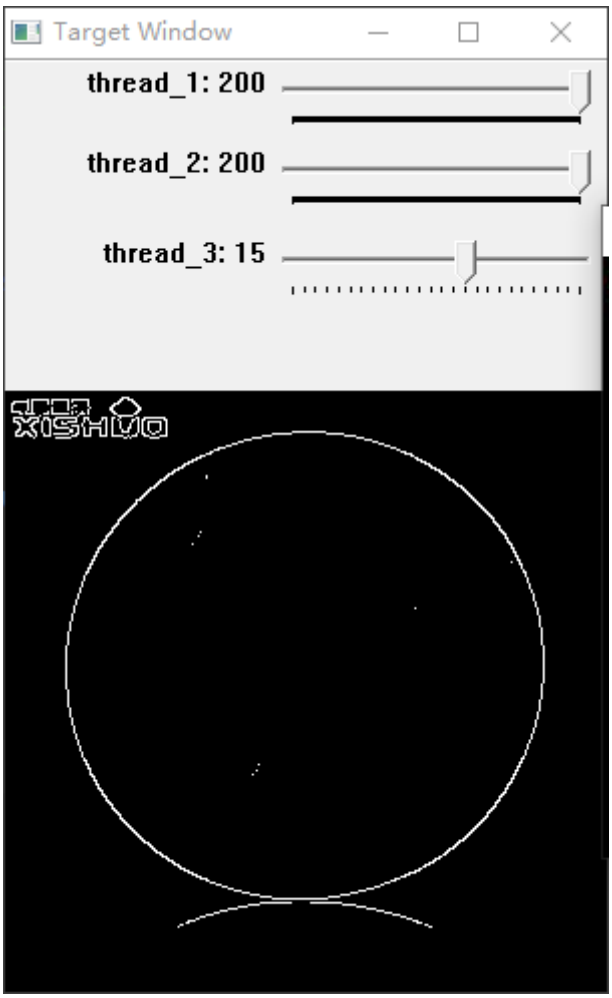
$a = x - r\cos(\theta)$ 是因为这里的 θ 是一条线与 x 轴负方向的夹角。而我们计算梯度的角度时大多采用 $\arctan$，$arccos$, $\arcsin$ 等形式，这里求的角度都是线与 x 轴正方向的夹角。因此，如果直接使用该夹角，公式应该为 $a = x + r\cos(\theta)$

结论分析与体会

循环遍历版本

- 效果很不错
- 只不过调参是个大问题，运行太慢了


Target Window
thread_1: 94
thread_2: 165
thread_3: 17



梯度改进版本

- 一共五个参数
  - 前两个是Canny函数的两个参数，用于控制边缘检测

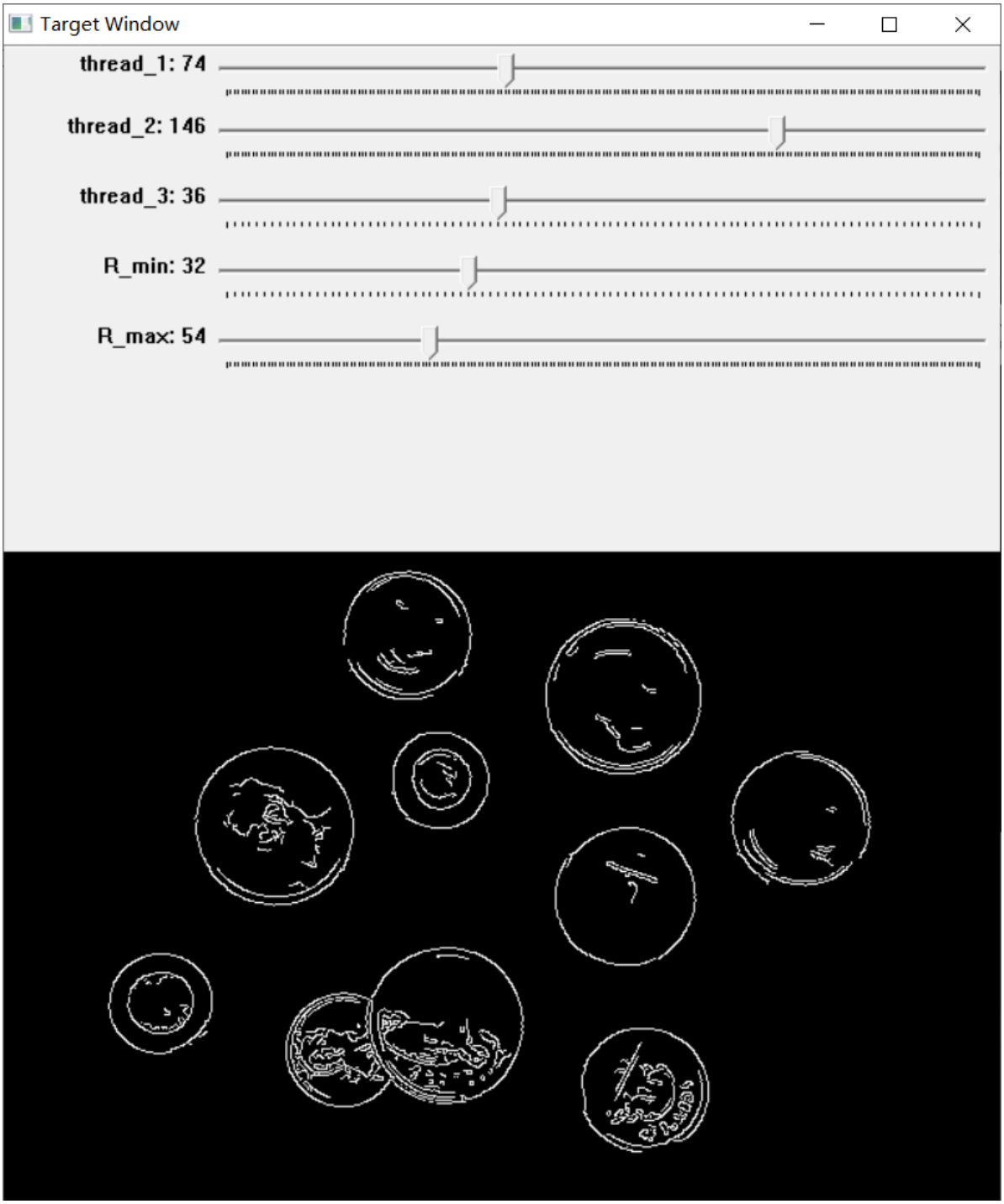
```
Canny(binary_image , target_image , thread_1 , thread_2);
```

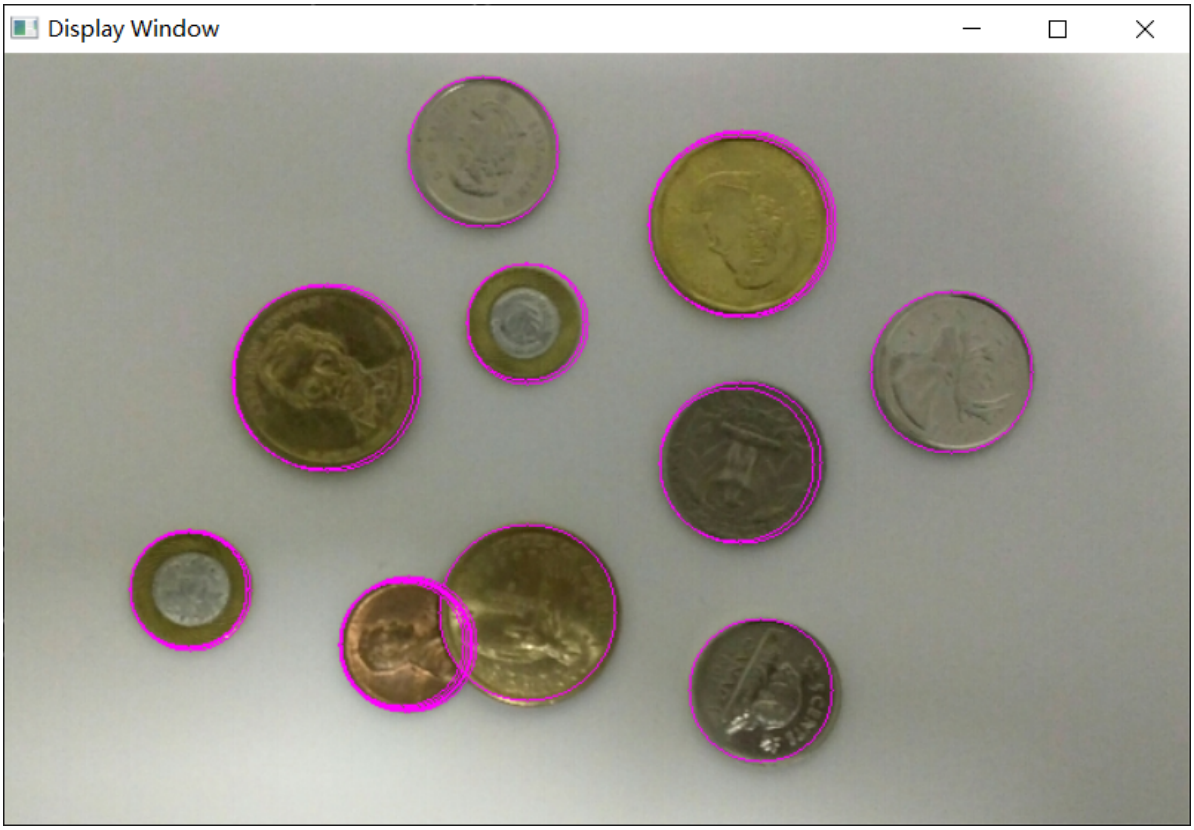
- 第三个是投票得分的阈值，控制圆是否输出

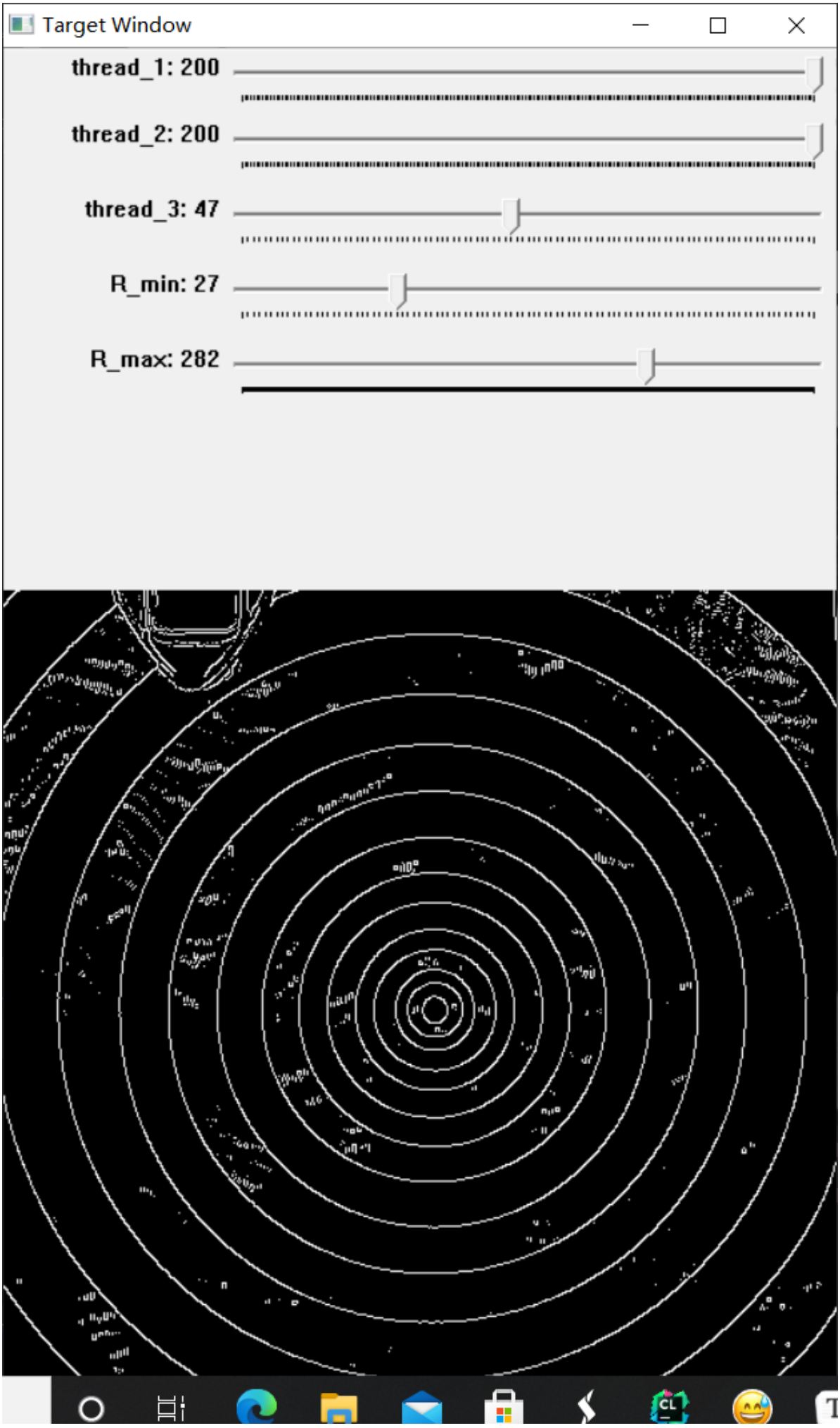
```
circles.clear();
    for(int r = R_min ; r < R_max ; r++)
        for(int i = 0 ; i < target_image.rows ; i++)
            for(int j = 0 ; j < target_image.cols ; j++)
                if(H[r][j][i] >= thread_3){
                    //cout << r << endl;
                   circles.push_back(Vec3f(r,j,i));
                }

    for(int i = 0 ; i < circles.size() ; i++){
        circle(source_image , Point(circles[i][1] , circles[i][2]) ,
circles[i][0] , Scalar(0 , 255 , 0));
    }
```

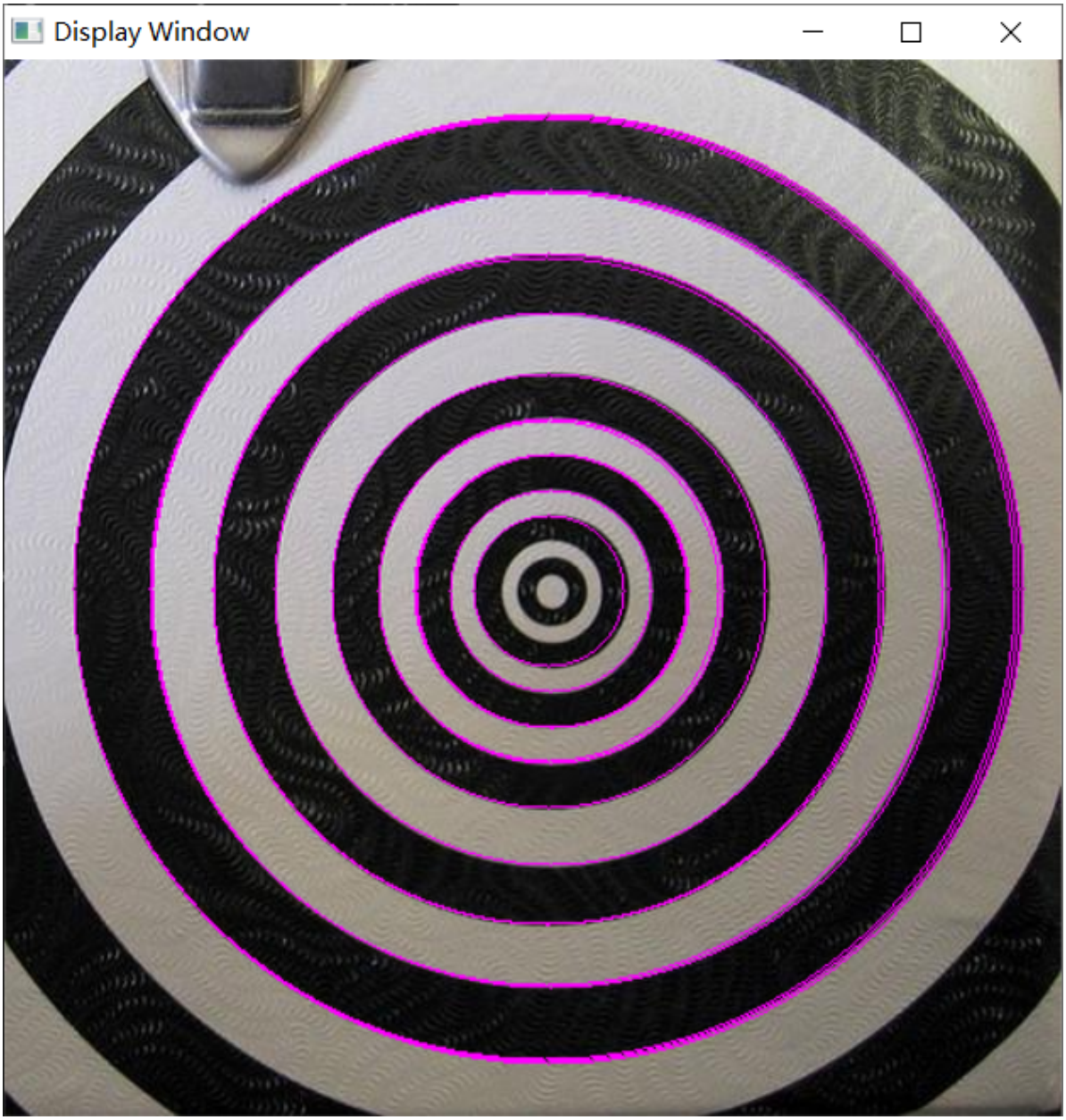
- 后两个参数是进行投票时遍历半径的范围

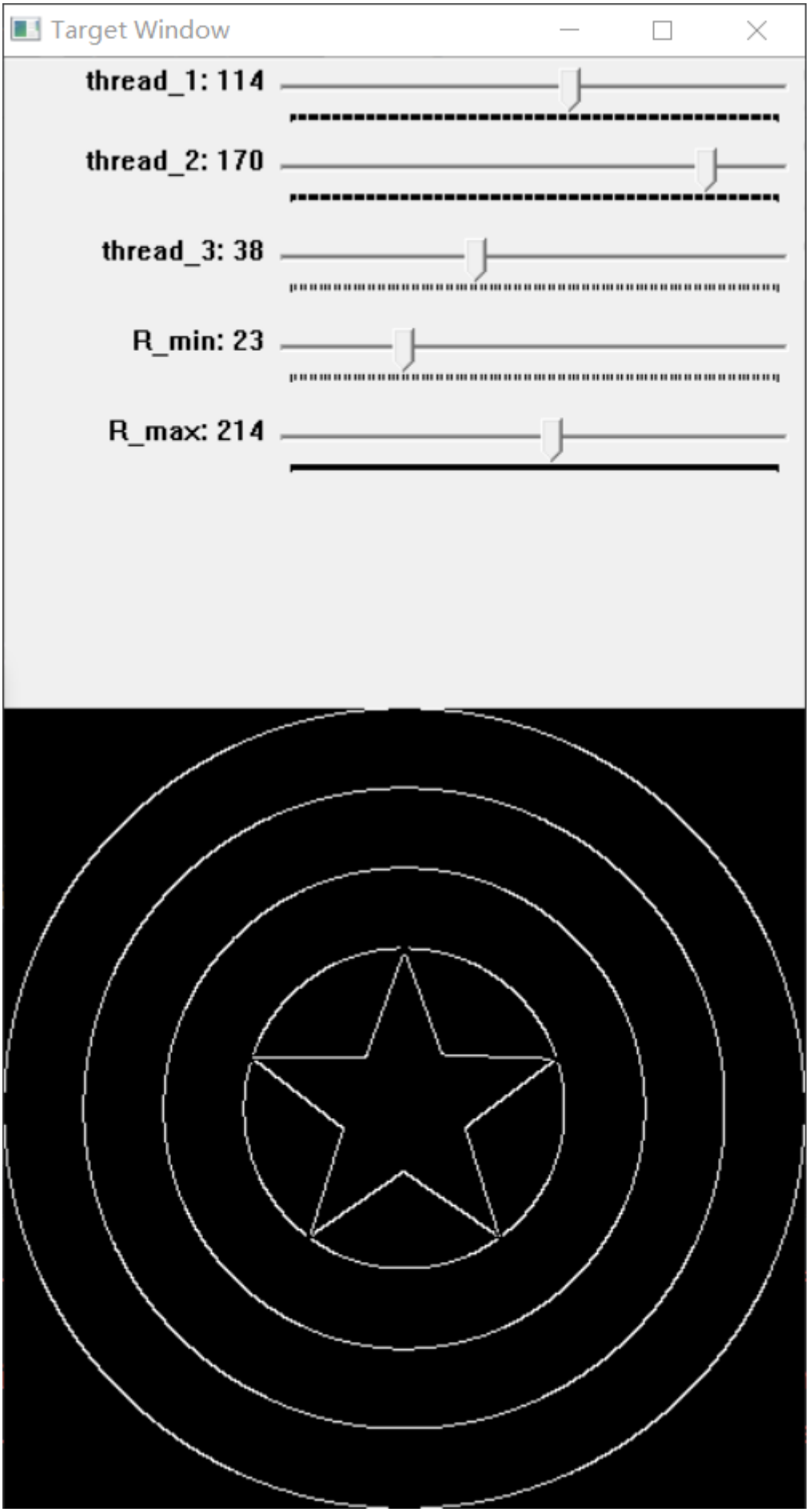
- 运行速度很快，方便参数调节
- 调参后，效果很好

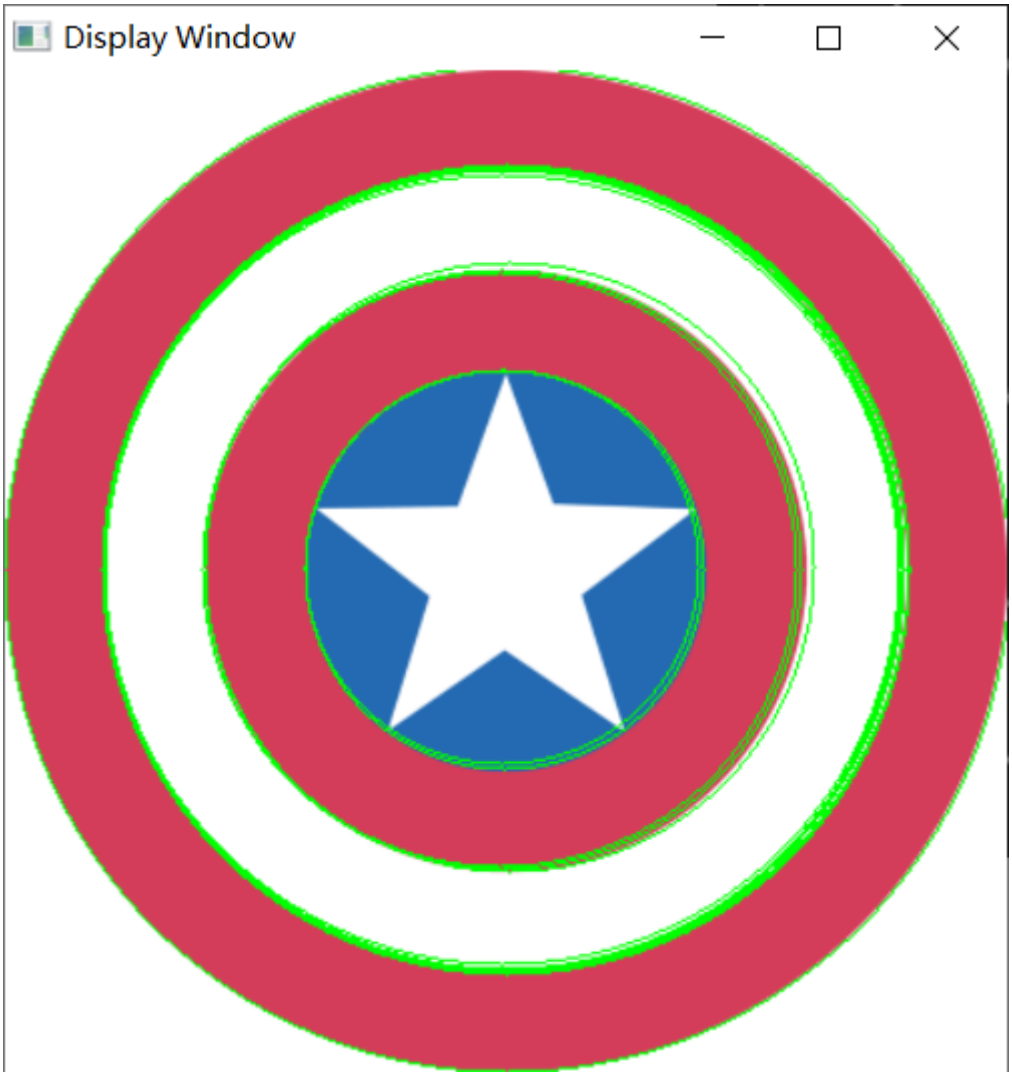

Target Window
thread_1: 74
thread_2: 146
thread_3: 36
R_min: 32
R_max: 54


Display Window


Target Window
thread_1: 200
thread_2: 200
thread_3: 47
R_min: 27
R_max: 282


Display Window


Target Window
thread_1: 114
thread_2: 170
thread_3: 38
R_min: 23
R_max: 214


Display Window