姓名

- 倪诗宇

学号

- 201900180065

实验日期

- 2021.11.07

实验题目

- Harris角点检测
- 实现Harris角点检测算法，并与 OpenCV 的 cornerHarris 函数的结果和计算速度进行比较

实验过程中遇到的问题和解决办法

- 问题一：
  - 问题：计算出来的 R 值比较大，导致创建的阈值拖动条特别长
  - 解决：原本的 R 使用的是 double 类型的二维数组

```
double R[1000][1000];

for(int i = 0 ; i < source_image.rows ; i++)
    for(int j = 0 ; j < source_image.cols ; j++)
        if(R[i][j] > thread){
            position_x.push_back(i);
            position_y.push_back(j);
        }
```

改为使用 Mat 类型数据，然后进行归一化，将数值改为 0----1 之间的值，方便设置阈值

- target 矩阵用于存储每个位置的 R 值
- 调用 自定义 calcuM 函数，计算 R 的值并存储在 target 中
- 然后将 target 中的值进行归一化

```
Mat target = Mat::zeros(dx.rows , dx.cols , dx.type());
calcuM(k2 , target , size2);
normalize(target , target, 0, 1, NORM_MINMAX);
```

- 遍历 target 矩阵，将大于阈值的点画出来

```
for(int i = 0 ; i < target.rows ; i++)
    for(int j = 0 ; j < target.cols ; j++)
        if(target.at<float>(i,j) > thread2)
            circle(source_image , Point(j,i) , 2 , Scalar(0 , 255 , 0));
```

- 问题二：
  - 问题：createTrackbar 中的值只能设置为整数，没法表示浮点数，但是我们需要对浮点型的数据进行调节
  - 解决：createTrackbar 中的值仍然使用整数，在函数中使用 double 类型的值来进行替代

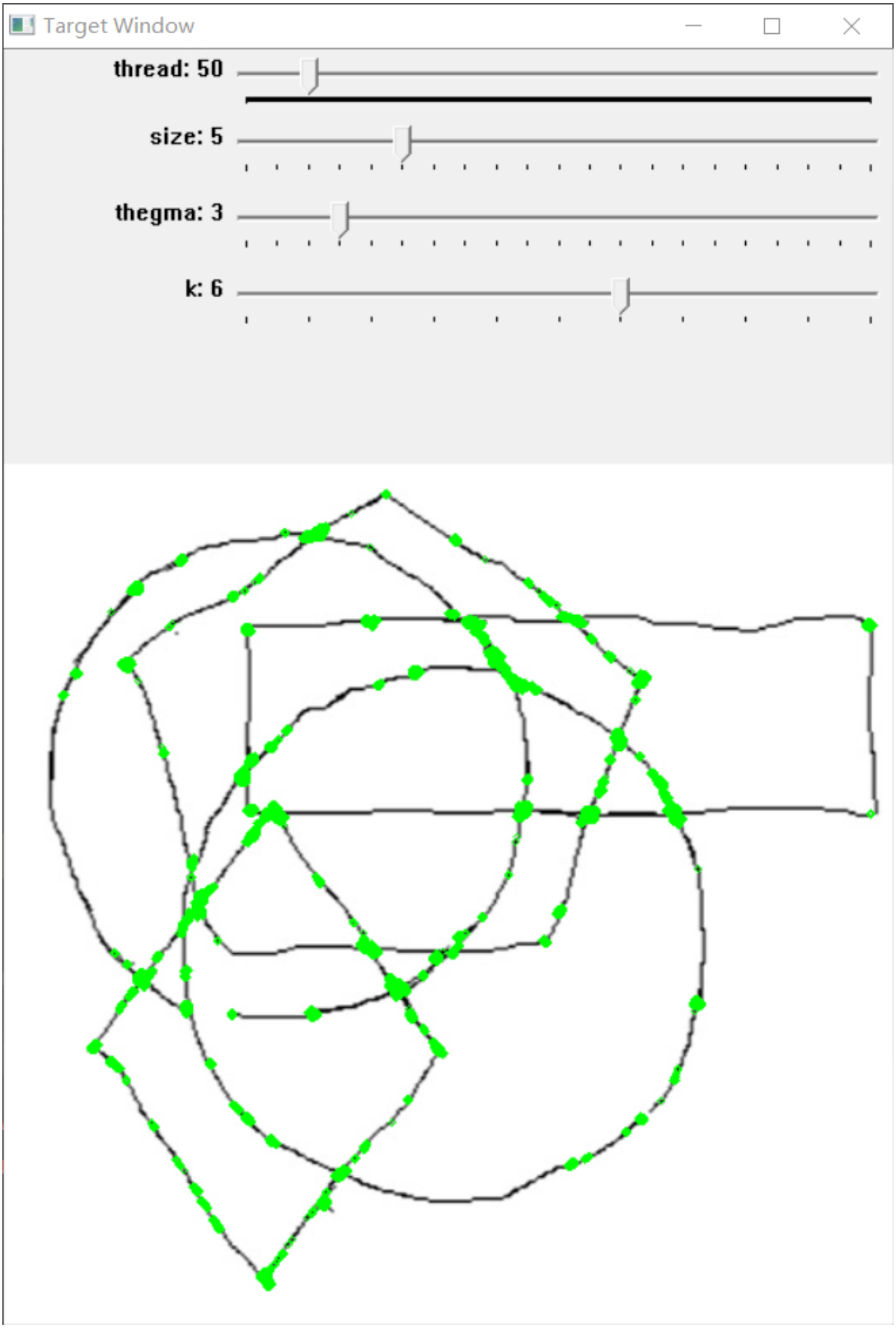
```
int thread = 50;
int thegma = 3;
int k_int = 6;
int size = 5;

createTrackbar("thread", "Target Window", &thread, 500, changeThread ,
0);
createTrackbar("size", "Target Window", &size, 20, changeThread , 0);
createTrackbar("thegma", "Target Window", &thegma, 20, changeThread ,
0);
createTrackbar("k", "Target Window", &k_int, 10, changeThread , 0);
```

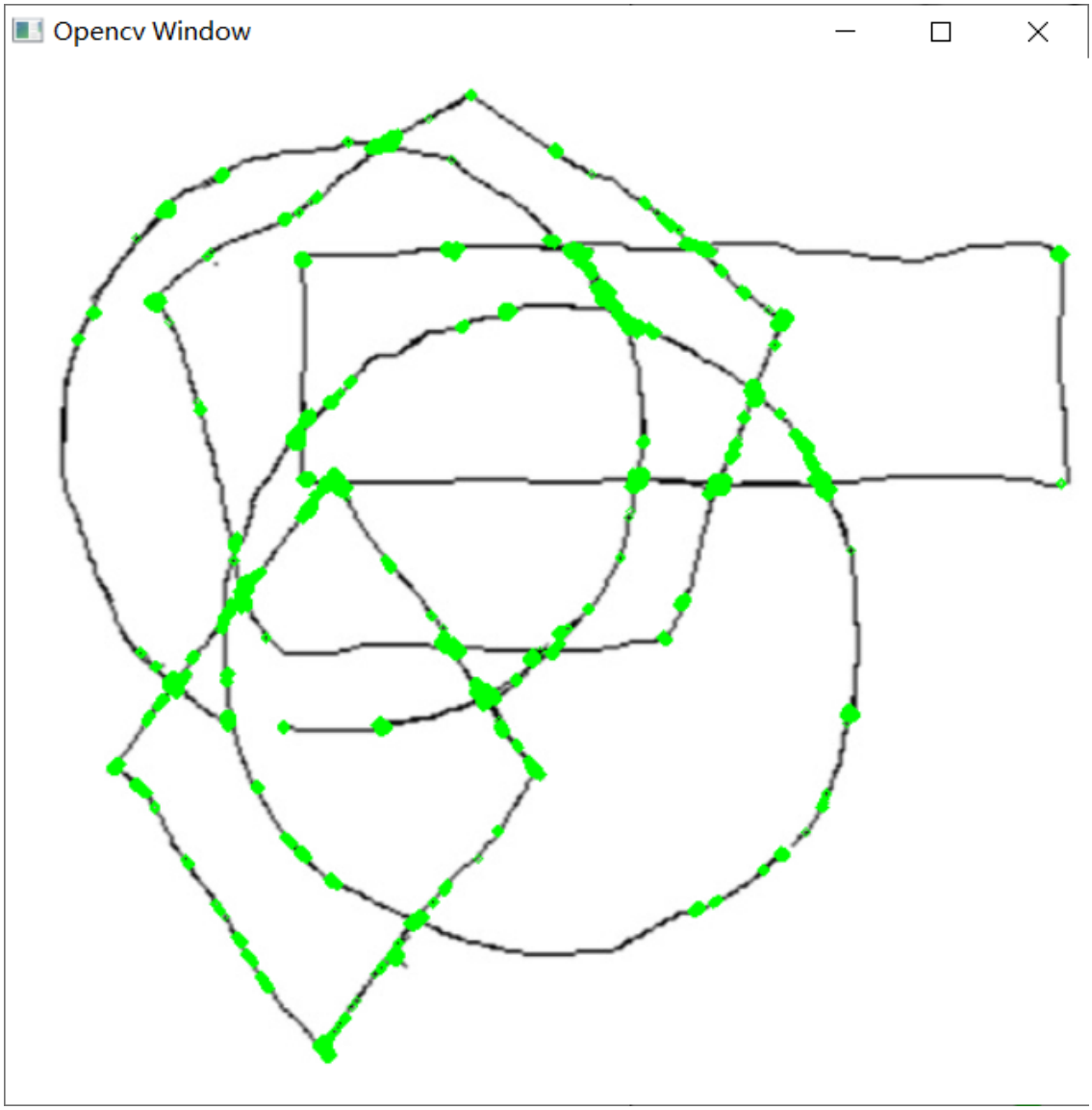
在函数中使用 k2 来替代 k， 使用 thread2 来替代 thread

```
float thread2 = thread * 1.0 / 100.0;
double k2 = k_int * 1.0 / 100.0;
```

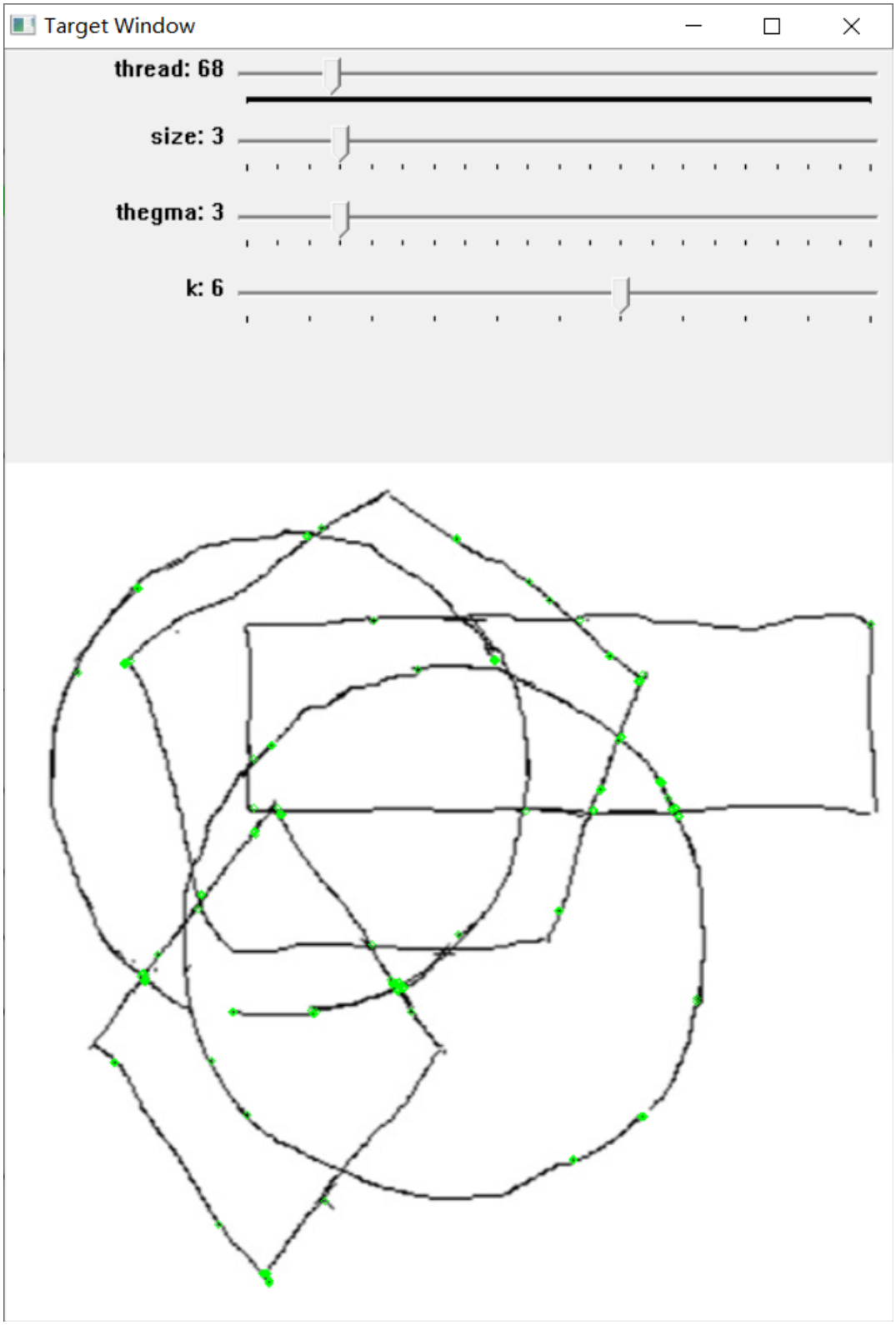
结论分析与体会

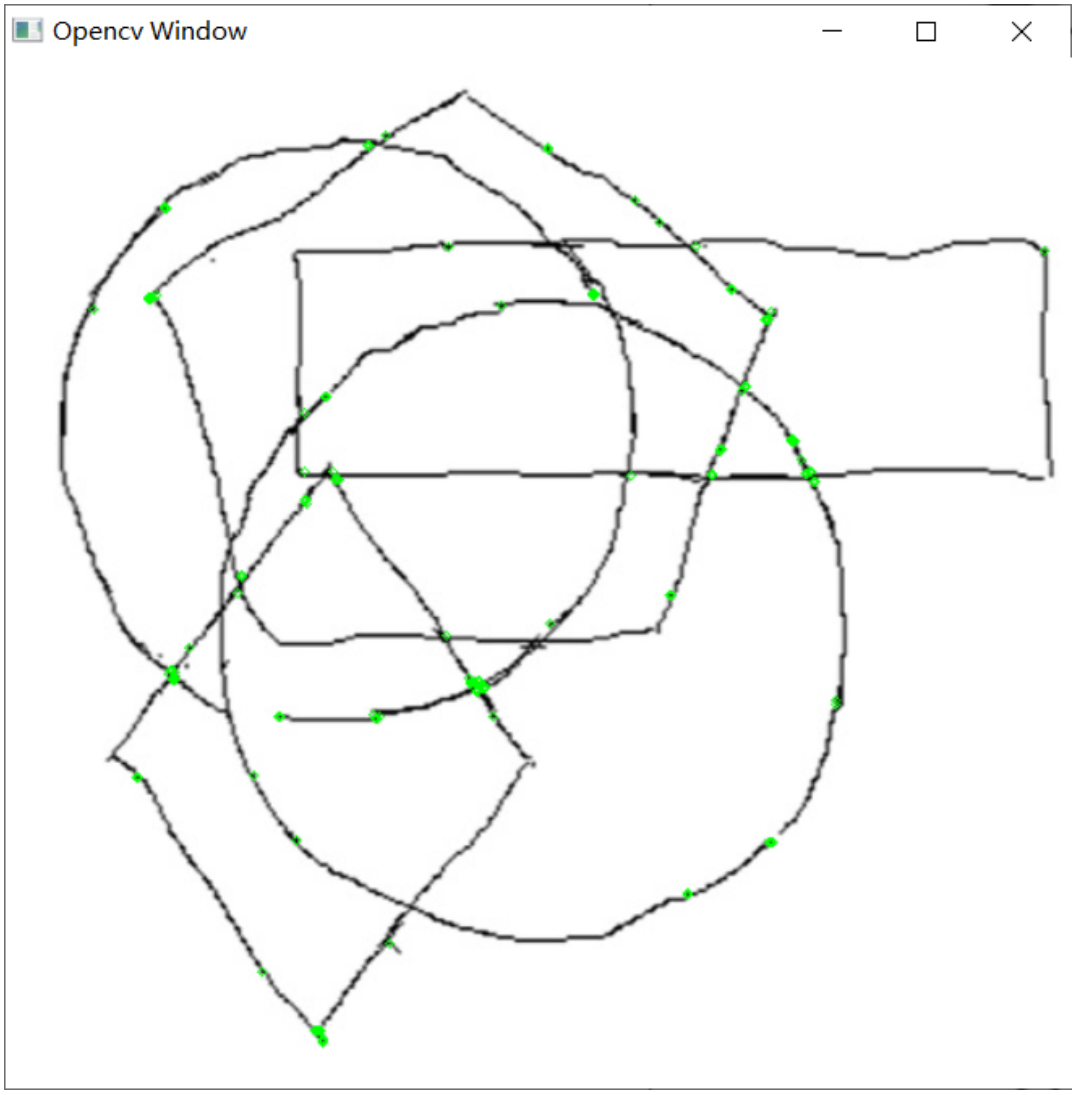
- 经测试 $\sigma = 3$ 时与 opencv 自带函数实现效果基本一致
- 样例一：
  - 第一张是自己实现的角点检测
  - 第二张是opencv自带函数cornerHarris实现的角点检测


Target Window
thread: 50
size: 5
thegma: 3
k: 6

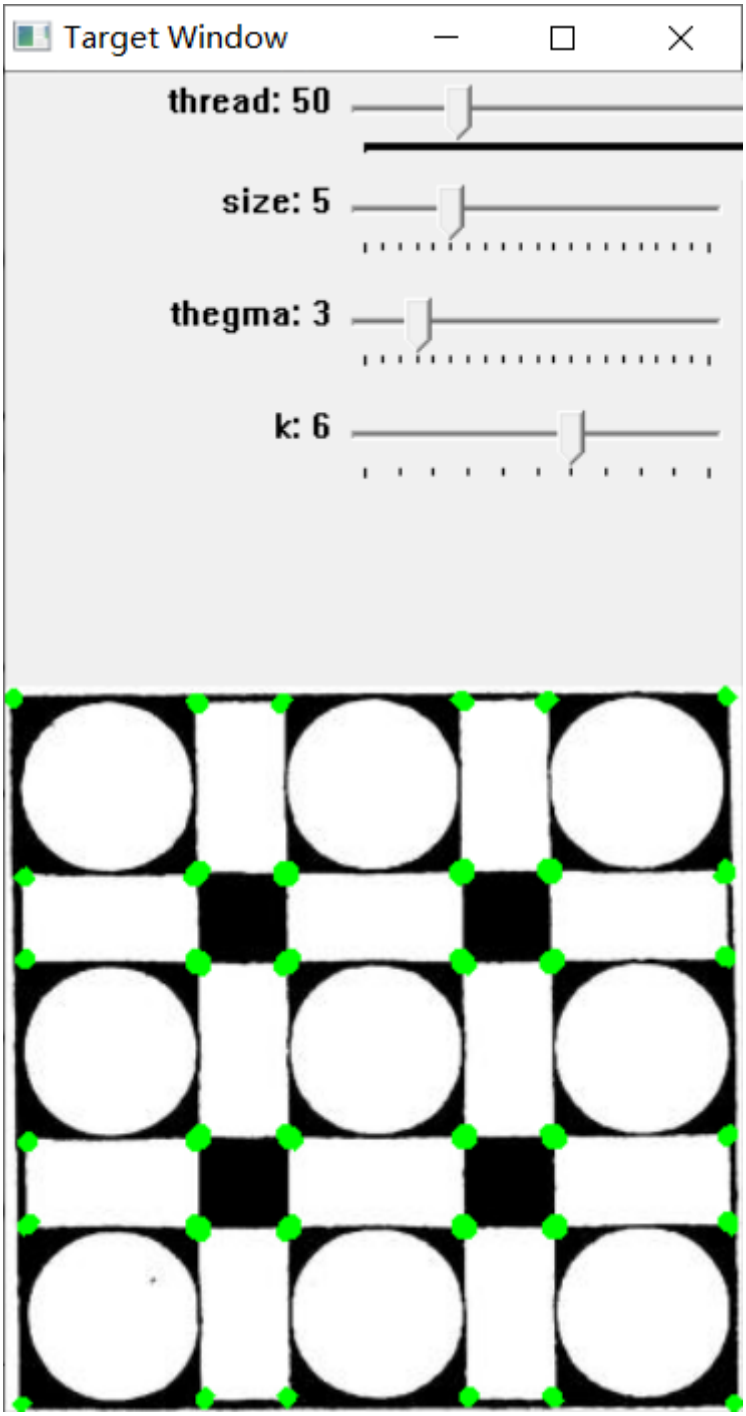


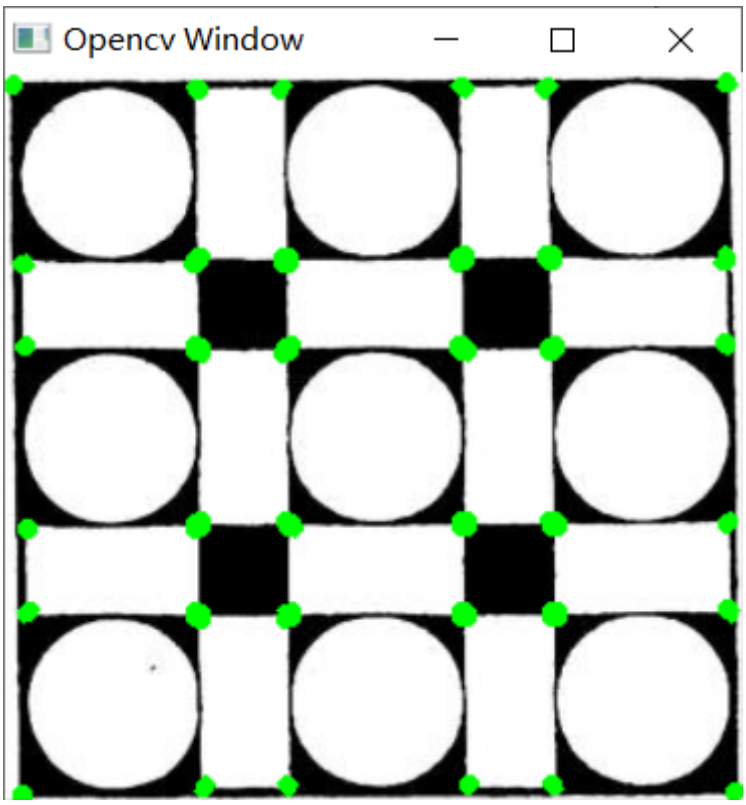
- 样例二：
  - 第一张是自己实现的角点检测
  - 第二张是opencv自带函数cornerHarris实现的角点检测


Target Window
thread: 68
size: 3
thegma: 3
k: 6

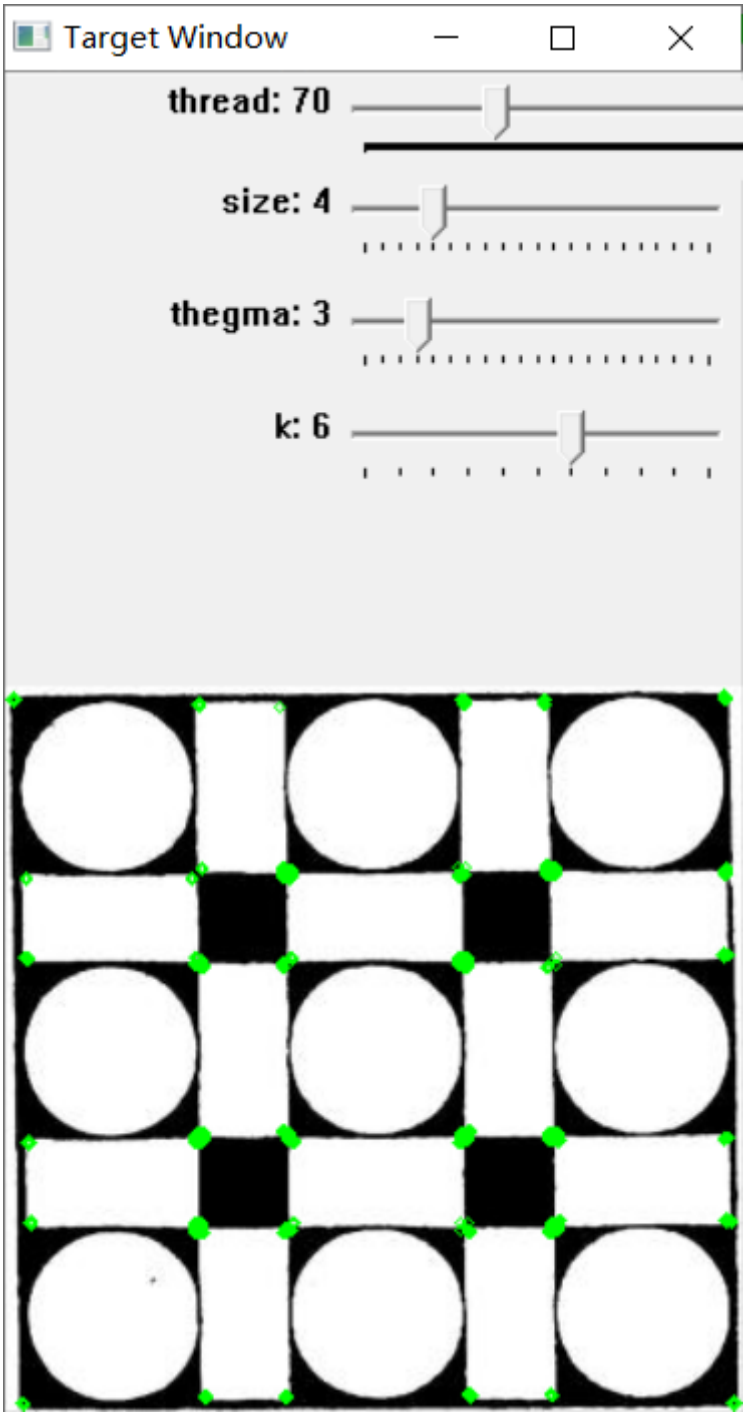


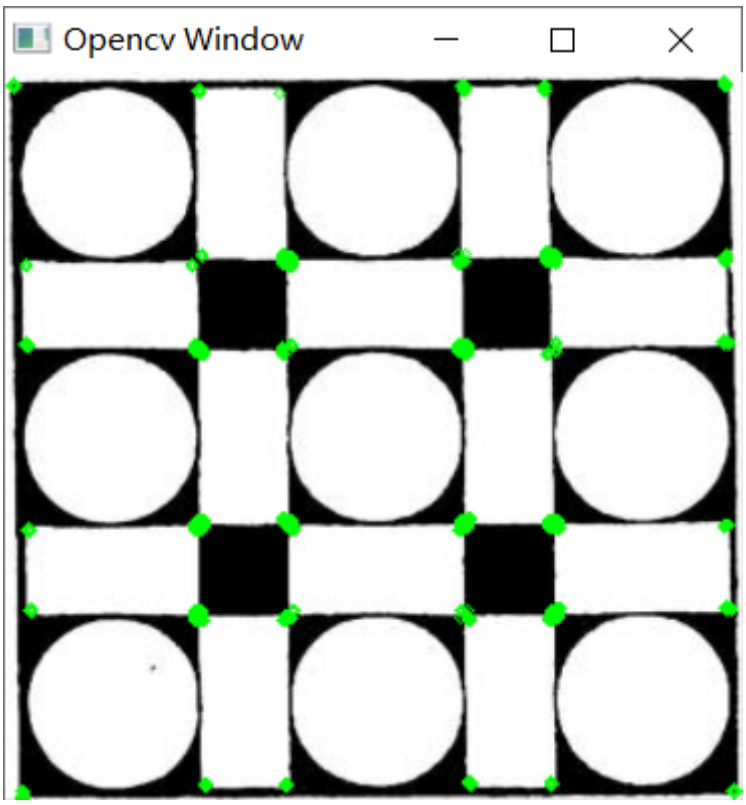
- 样例三
  - 第一张是自己实现的角点检测
  - 第二张是opencv自带函数cornerHarris实现的角点检测


Target Window
thread: 50
size: 5
thegma: 3
k: 6



- 样例四
  - 第一张是自己实现的角点检测
  - 第二张是opencv自带函数cornerHarris实现的角点检测


Target Window
thread: 70
size: 4
thegma: 3
k: 6



- 速度对比
  - 有时候我的快
  - 有时候opencv的快
  - 不过opencv快的次数比较多，可能因为我用的是高斯，没有利用其行列可分离性。opencv利用的是进行行列可分离的boxFilter

```
own function time(s): 0.0031056
opencv function time(s): 0.0013919
own function time(s): 0.0011902
opencv function time(s): 0.0013156
own function time(s): 0.0011894
opencv function time(s): 0.001336
own function time(s): 0.0014366
opencv function time(s): 0.0012864
own function time(s): 0.0031248
opencv function time(s): 0.0013711
own function time(s): 0.0028112
opencv function time(s): 0.0013518
own function time(s): 0.0033923
opencv function time(s): 0.0013665
own function time(s): 0.0029715
opencv function time(s): 0.0014992
own function time(s): 0.0075409
opencv function time(s): 0.0014343
own function time(s): 0.0028767
opencv function time(s): 0.0013894
own function time(s): 0.0028292
opencv function time(s): 0.0013899
own function time(s): 0.0028647
opencv function time(s): 0.0014941
own function time(s): 0.0027958
opencv function time(s): 0.0014959
own function time(s): 0.0030896
opencv function time(s): 0.0013717
```

```
own function time(s): 0.0028244
opencv function time(s): 0.0014009
own function time(s): 0.0028159
opencv function time(s): 0.0013732
own function time(s): 0.0029305
opencv function time(s): 0.0015511
own function time(s): 0.0027989
opencv function time(s): 0.0013706
own function time(s): 0.0028266
opencv function time(s): 0.0013806
own function time(s): 0.0028454
opencv function time(s): 0.0013639
own function time(s): 0.0031369
opencv function time(s): 0.0013675
own function time(s): 0.0029519
opencv function time(s): 0.0013921
own function time(s): 0.0030495
opencv function time(s): 0.0012913
own function time(s): 0.0028713
opencv function time(s): 0.0015948
own function time(s): 0.0028117
opencv function time(s): 0.0015284
own function time(s): 0.0028443
opencv function time(s): 0.0012593
own function time(s): 0.0032038
opencv function time(s): 0.0013671
own function time(s): 0.0027903
opencv function time(s): 0.0013535
own function time(s): 0.0030259
opencv function time(s): 0.0013752
own function time(s): 0.0028695
opencv function time(s): 0.0012521
own function time(s): 0.0028241
opencv function time(s): 0.0013966
own function time(s): 0.0028379
opencv function time(s): 0.0014702
own function time(s): 0.0030046
opencv function time(s): 0.0012622
own function time(s): 0.0028452
opencv function time(s): 0.001346
own function time(s): 0.003057
opencv function time(s): 0.0013646
own function time(s): 0.0100911
opencv function time(s): 0.0014081
```