山东大学 计算机科学与技术 学院

计算机视觉 课程实验报告

学号：201900161140	姓名： 张文浩	2021.11.5
实验题目：图像匹配一		

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。）

实验要求：

- 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。

实验步骤：

Harris 算法步骤：
①：利用 sobal 算子计算图像 x，y 方向上的梯度，Ix、Iy
②：计算梯度的协方差矩阵，Ixx，Iyy，Ixy

$$\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

③：利用高斯函数对 Ixx，Iyy，Ixy 进行滤波
④：计算出局部特征结果矩阵 M 的特征值和响应函数 C

$$\begin{aligned} R &= \det[M(\sigma_I,\sigma_D)] - \alpha[\text{trace}(M(\sigma_I,\sigma_D))]^2 \\ &= g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2)+g(I_y^2)]^2 \end{aligned}$$

⑤：将计算出响应函数的值 C 进行非极大值抑制，滤除一些不是角点的点，同时满足大于设定的阈值

第一步：

导入图像，转换为灰度图像。

```
img = imread("E:/计算机视觉/exp7/2.jpg");
resize(img, img, Size(300, 300));
imshow("原始图", img);
cvtColor(img, image_gray, COLOR_BGR2GRAY);
```

第二步：

利用 Sobal 算子计算梯度，并计算梯度的协方差

```
Sobel(image_gray, dx, CV_32F, 1, 0, 3);
Sobel(image_gray, dy, CV_32F, 0, 1, 3);
dxx = dx.mul(dx);
dxx = dx.mul(dx);
dyy = dy.mul(dy);
dxy = dx.mul(dy);
```

第三步：

对 dxx，dyy，dxy 进行高斯滤波

```
GaussianBlur(dxx, dxx_gauss, Size(7, 7), 0, 0);
GaussianBlur(dyy, dyy_gauss, Size(7, 7), 0, 0);
GaussianBlur(dxy, dxy_gauss, Size(7, 7), 0, 0);
```

第四步;

利用近似计算得到响应矩阵 C

```
C = dxx_gauss.mul(dyy_gauss) - dxy_gauss.mul(dxy_gauss) - k * (dxx_gauss +
dyy_gauss).mul(dxx_gauss + dyy_gauss);
```

第五步：

利用得到的响应矩阵 C 进行非极大值抑制，每次找到一个响应值大于阈值的像素点后，先遍历一遍邻域的八个像素，如果这八个像素中有响应值比自己还大的，这个点就被抑制，不画了。否则画出一个小圆标注出来。

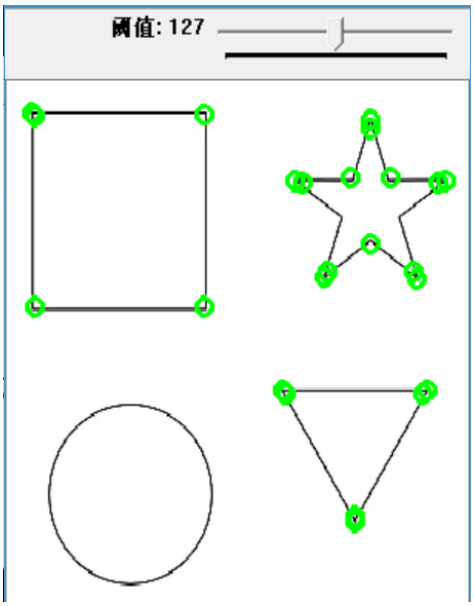
```
for (int i = 0; i < row; i++) {
    for (int j = 0; j < col; j++) {
        if (C.at<float>(i, j) > threshold1) {
            bool flag = true;
            for (int k = 0; k < 8; k++) {
                int tx = i + dir[k][0];
                int ty = j + dir[k][1];
                if (tx >= 0 && ty >= 0 && tx < row && ty < col && C.at<float>(i, j)
< C.at<float>(tx, ty)) {
                    flag = false;
                    break;
                }
            }
            if (!flag) continue;
            circle(image_my, Point(j, i), 5, Scalar(0, 255, 0), 2, 8, 0);
        }
    }
}
```

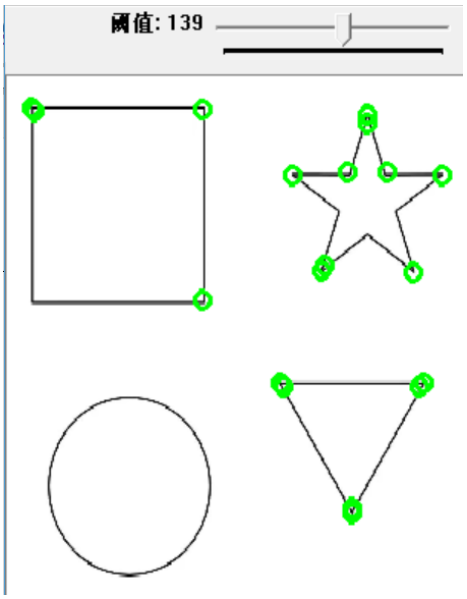
第六步：

与 opencv 的 cornerHarris 函数对比效果和速度。

实验结果及分析：

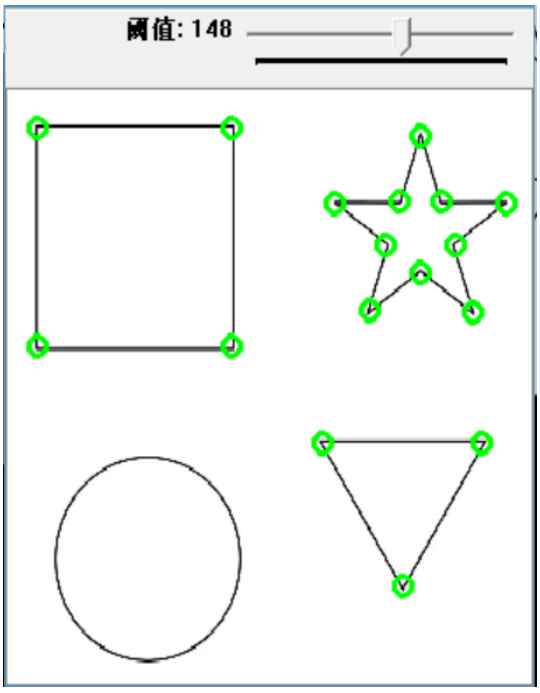
Threshold=127：　　Threshold=139





后来我发现调整第三步对梯度进行高斯滤波的高斯核的大小可以得到更好的效果。上面是高斯核大小为 3X3 的效果，发现不管怎么调整阈值效果都不是很好。
下面是将高斯核大小改为 7X7 后的效果，效果就很好了。

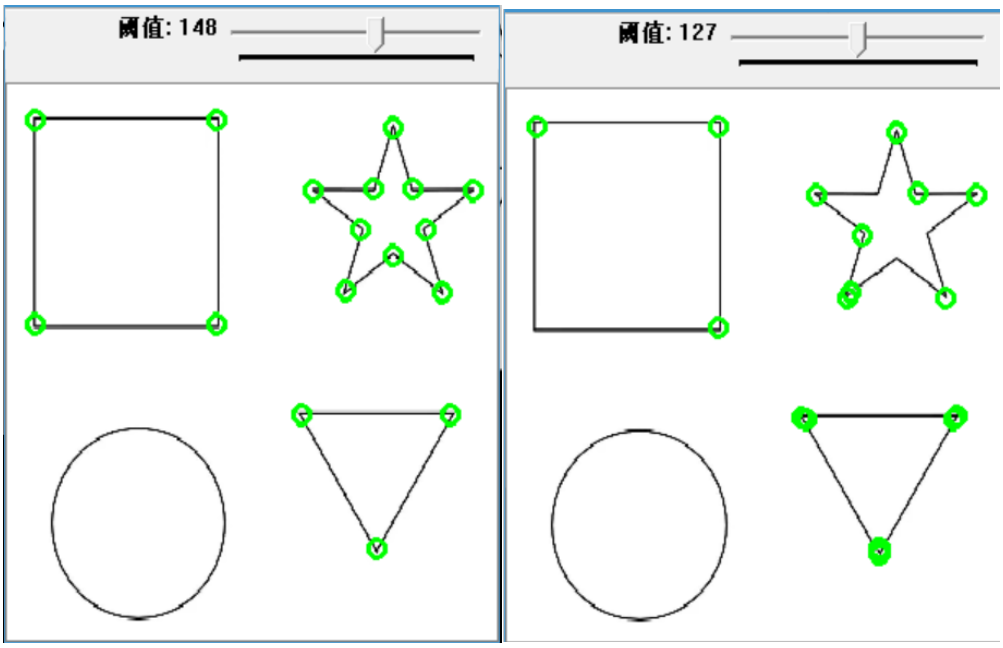
Threshold=148

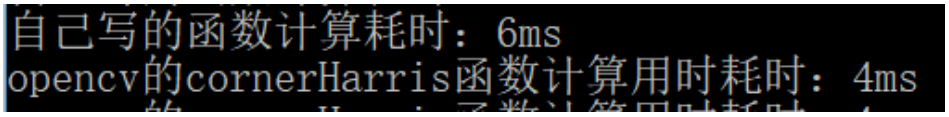


阈值调整为 148 的时候，效果非常不错。

与 opencv 的 cornerHarris 函数对比效果和速度。

My Harris　　opencv cornerHarris





发现在这个样例中，opencv 的 cornerHarris 函数的效果还不如自己的函数，但速度快了一点，不过速度上差距不是很明显。

结果分析与体会：

在本次实验中，我实现了 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。

发现通过增大第三步对梯度进行高斯滤波的高斯核的大小可以很有效地提升效果，并且最后惊讶的发现，在这个样例中，虽然速度上比 OpenCV 的 cornerHarris 函数稍慢一点，但效果却比 OpenCV 的 cornerHarris 函数好很多。分析其原因，我认为还是高斯滤波起到了很大作用。