山东大学 计算机科学与技术 学院

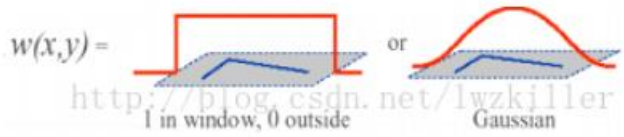
计算机视觉 课程实验报告

学号：201900130151	姓名：莫甫龙	
实验题目： 图像匹配 1		

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。）

当窗口发生[u,v]移动时，那么滑动前与滑动后对应的窗口中的像素点灰度变化描述如下：



通过泰勒展开，E(u,v)可以化为如下形式：

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Windowing function - computing a weighted sum (simplest case, w=1)

Note: these are just products of components of the gradient, Ix, Iy

因为 M 是一个二次型矩阵，所以对于角点响应，我们可以用如下公式来进行计算度量：

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

其中： detM=Ix*Iy-Ixy*Ixy

traceM=Ix+Iy，k 一般取值在 0.04—0.06

对此，所有理论部分已经清楚，开始进行实验：

首先将图片进行处理，转化为二值图像：

```
Mat imag = imread( filename: "C:\\Users\\15339\\Desktop\\pictures\\1234.png");
Mat input;
cvtColor( src: imag,  dst: input,  code: COLOR_RGB2GRAY);
input.convertTo( m: input, CV_64F);
```

然后，通过 Sobel 函数求出梯度，再通过 mul 求出二次梯度，接着进行高斯滤波（即使用高斯窗口）。

```
Sobel( src: input,  dst: Ix,  ddepth: -1,  dx: 1,  dy: 0);
Sobel( src: input,  dst: Iy,  ddepth: -1,  dx: 0,  dy: 1);
Ixx = Ix.mul( m: Ix);
Ixy = Ix.mul( m: Iy);
Iyy = Iy.mul( m: Iy);
GaussianBlur( src: Ixx,  dst: Ixx,  ksize: Size( _width: 11,  _height: 11),  sigmaX: 1,  sigmaY: 1);
GaussianBlur( src: Ixy,  dst: Ixy,  ksize: Size( _width: 11,  _height: 11),  sigmaX: 1,  sigmaY: 1);
GaussianBlur( src: Iyy,  dst: Iyy,  ksize: Size( _width: 11,  _height: 11),  sigmaX: 1,  sigmaY: 1);
```

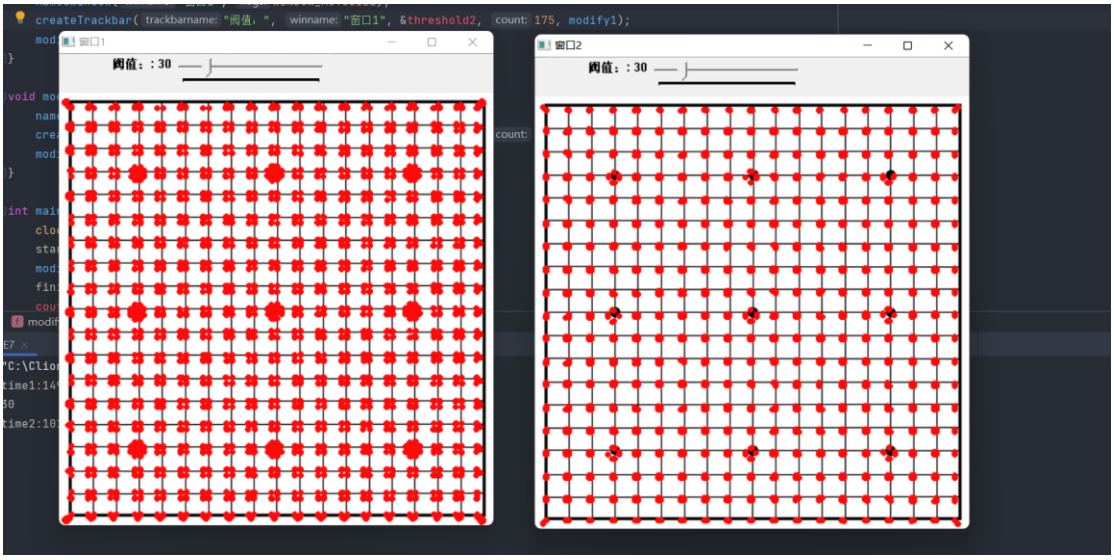
接着计算度量值：

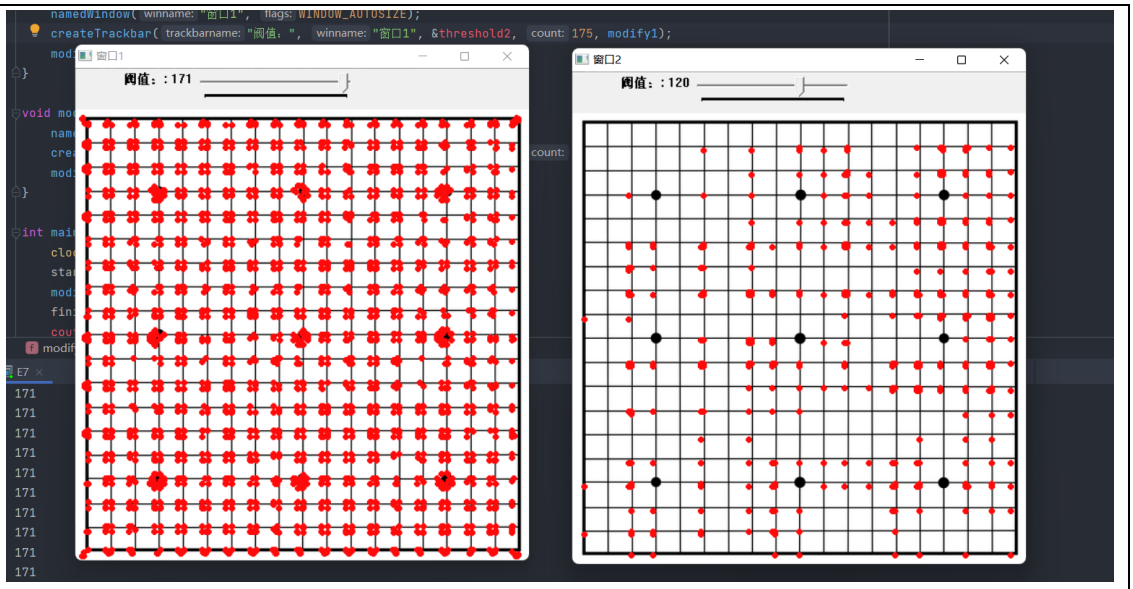
```
for (int x = 0; x < output.rows; x++) {
    for (int y = 0; y < output.cols; y++) {
        double det_M = Ixx.at<double>(x, y) * Iyy.at<double>(x, y) - Ixy.at<double>(x, y) * Ixy.at<double>(x, y);
        double trace_M = Ixx.at<double>(x, y) + Iyy.at<double>(x, y);
        double r = det_M - 0.05 * trace_M * trace_M;
```

接着将大于阈值的点在原图上画出来，通过设置不同的阈值大小，来比较不同的效果。

```
        if ((int) r > (threshold2*2000000)) {
            circle( img: imag,  center: Point(x, y),  radius: 2,  color: Scalar( v0: 10,  v1: 10,  v2: 255),  thickness: 2);
        }
    }
}1
imshow( winname: "demo",  mat: imag);
```

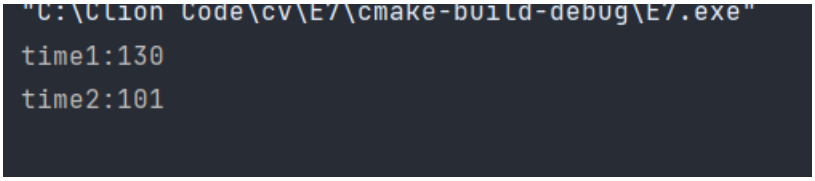
还调用了 opencv 自带的 cornerHarris 函数，两者效果图如下：





可以看出，当阈值变大，角点就会变少，但是我自己写的函数却很难体现这一点，只能看到角点变少了一点点，后来我查看了一下计算出的度量角点的值 r，发现这个值都特别大，所以阈值不怎么好设置。

两者的运行结果如下：



结果分析与体会：

这一个实验地难点主要在于如何区分出角点和非角点，Harris 方法很好地将这个问题转化为了数学问题，将实现简单化。