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　　：201600181073	姓名：　唐超	班级：　智能 16

实验题目：基于直方图的目标跟踪

实验内容：

- 实现基于直方图的目标跟踪：已知第 t 帧目标的包围矩形，计算第 t+1 帧目标的矩形区域。
- 选择适当的测试视频进行测试：给定第 1 帧目标的矩形框，计算其它帧中的目标区域。

实验过程中遇到和解决的问题：

1. 读取视频并逐帧播放

用 videocapture 类读取视频，事先声明 Mat 类变量 frame，用 while 循环将每一帧读取至 frame 并显示。

```
Mat frame; //帧
int count = 0;
VideoCapture cap("C:\\Users\\hp\\Desktop\\CV实验\\E5\\车流一组.avi");
namedWindow("frame");
while (count<2000)
{
    cap >> frame;  //读取帧
    imshow("frame", frame);
    waitKey(100);
    count++;
 }
```

2. 用鼠标交互确定跟踪目标矩形框

由于第一帧图像需要用鼠标画出矩形框，以及单独统计框中图像像素分布的直方图，因此将第一帧从循环中取出单独处理。第一帧等待 5 秒以画出矩形框。

声明 Rect 型变量 rect，Point 型变量 tl，int 型变量 xx，yy 为全局变量。在显示第一帧时，设置回调函数 setMouseCallback，以监测显示窗口的鼠标动作。 定义函数 mouseEvent，当鼠标左键按下并拖拽时，tl 会记录鼠标第一次点击的位置坐标，xx 和 yy 记录鼠标鼠标松开时的位置坐标，根据以上信息，将矩形框的位置及大小赋给 rect。

在主函数中用 renctangle 函数进行渲染得到含有矩形框的第一帧图像。

```
void mouseEvent(int event, int x, int y, int flags, void* userdata)
{
 if (event == EVENT_LBUTTONDOWN)
```

```
        { //每次鼠标按下记录一次
                tl = { x,y };
        }
        else if (event == EVENT_MOUSEMOVE && flags == EVENT_FLAG_LBUTTON)
        {  //左键按下拖动拉框
                rect = { tl.x,tl.y,x - tl.x,y - tl.y };
                xx = x;
                yy = y;  //将鼠标左键松开前的坐标值（矩形框右下角的坐标）赋给xx，yy
        }
       }
```

3. 统计目标矩形框内图像的像素分布直方图，用二维数组 a 表示

在 2 中对 rect 渲染前，需要对矩形框内图像统计像素值分布直方图，若在渲染后再统计，会将渲染的矩形框的像素信息也统计进去，造成干扰。

遍历矩形框内的每一点，将像素信息存入二维数组 a[256][3].

```
int a[256][3] = { 0 };
for (int k = 0; k < 3; k++)
{
for (int i = tl.y; i < yy; i++)
    for (int j = tl.x; j < xx; j++)
    {
        a[frame.at<Vec3b>(i, j)[k]][k]++;
    }
 }
```

4. 对下一帧，用相同大小矩形框遍历在原矩形框上下左右 30 个像素点的区域，求得每个矩形框内图像的统计直方图，用二维数组 b 表示。

5. 求向量 a 和 b 的夹角余弦值，若大于阈值 0.7，则作为新一帧的目标矩形框。

```
        double  c1 = 0, c2 = 0, d = 0;
        for (int m = 0; m < 256; m++)
        {
            for (int n = 0; n < 3; n++)
            {
                d += b[m][n]*a[m][n];
                c1 += a[m][n] * a[m][n];
                c2 += b[m][n] * b[m][n];
            }
        }
```

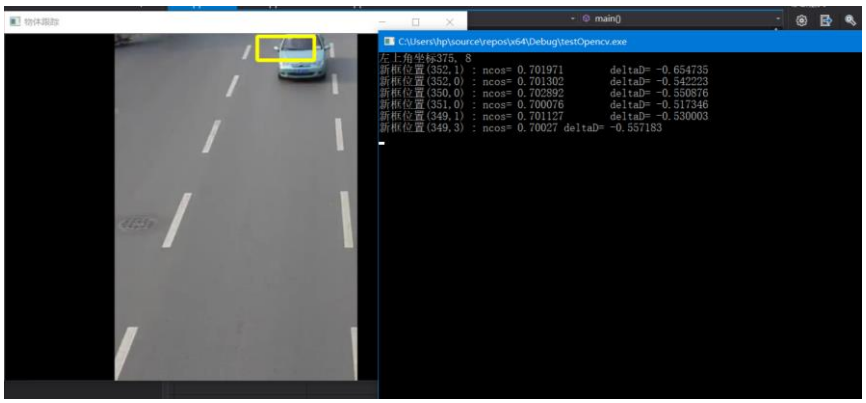
```
                double d1 = sqrt(c1), d2 = sqrt(c2);
                double ncos = d / (d1*d2);
                double deltaD = (d1 - d2) / d1;
                if (ncos > 0.8)
                {
                    x00 = xxx;
                    y00 = yyy;
                    cout << "新框位置("<<xxx << "," << yyy << ")" << " : " << "ncos=
" << ncos <<"\t"<<"deltaD= "<<deltaD<< endl;
                    goto paint;
                }
```

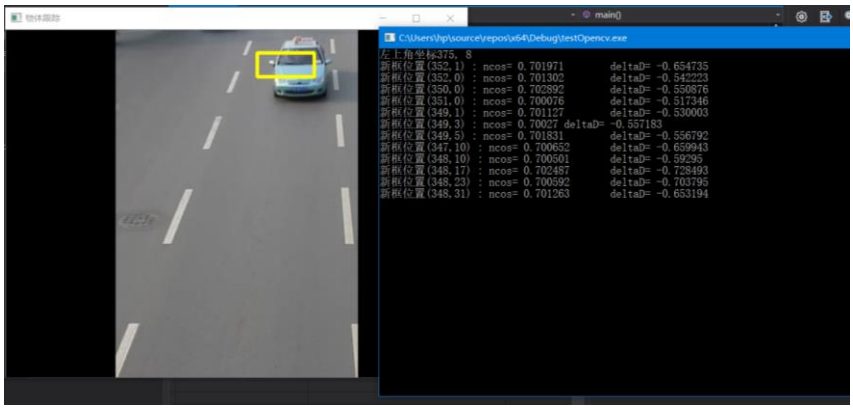
6. 画出目标矩形框

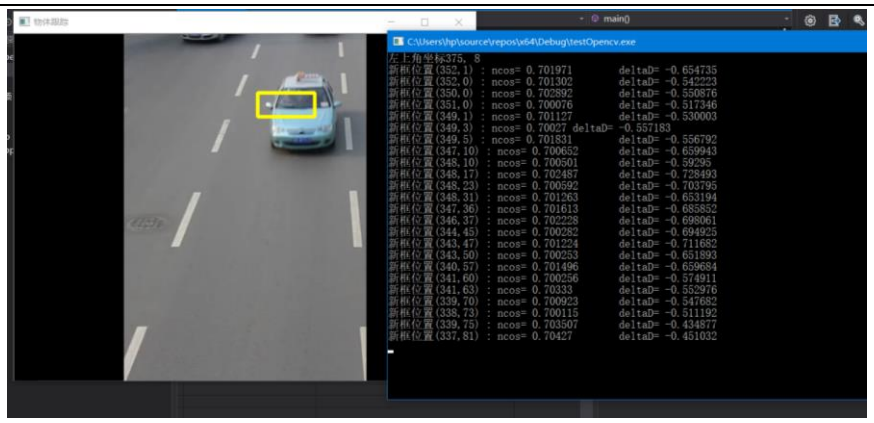
判断目标矩形框左上角坐标是否满足 x00<frame.cols-w&&y00<frame.rows-h,若满足则用 rectangle 函数画出矩形框，否则物体一般已经跑出图像外，结束程序。

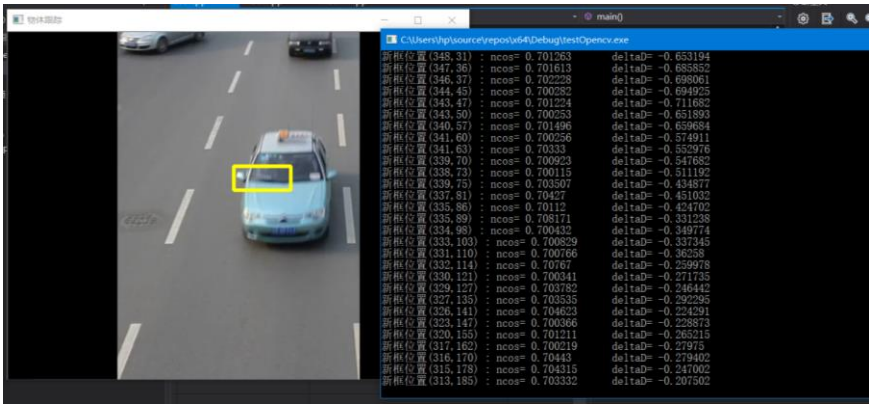
对每一帧重复以上步骤 4、5、6。

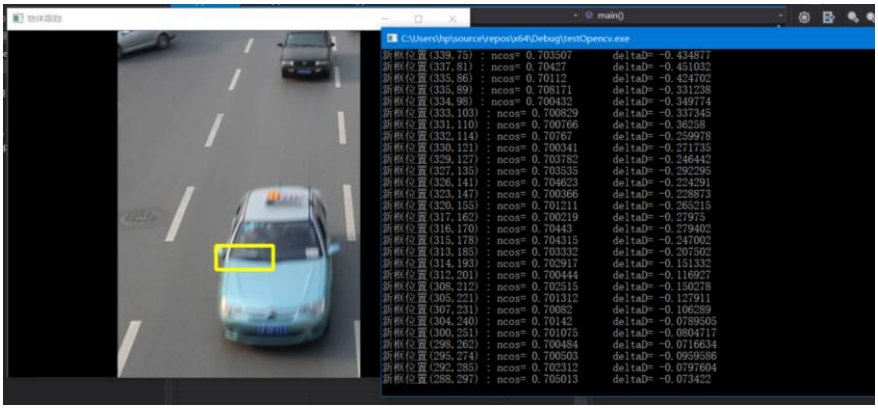
实验截图如下：











结论分析与体会：

确定每一帧目标物体矩形框位置时，采用了阈值筛选的方法，另外可以用最大夹角余弦的矩形框作为新的目标矩形框，效果应该会更好。实验视频采用的是道路监控，车辆由远及近在图像中会变大，造成往后的跟踪下降，这种方法更适用于跟踪图像在视频中大小变化较小的情况。

通过这次实验，我学会了对视频进行简单的处理，以及对一张图的像素值做直方图统计并衡量与另一张图的统计直方图的相似度。