计算机科学与技术　　学院

计算机视觉 课程实验报告

学号：	姓名：	班级：
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：

（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。）

一、 图像直方图的获取与表示

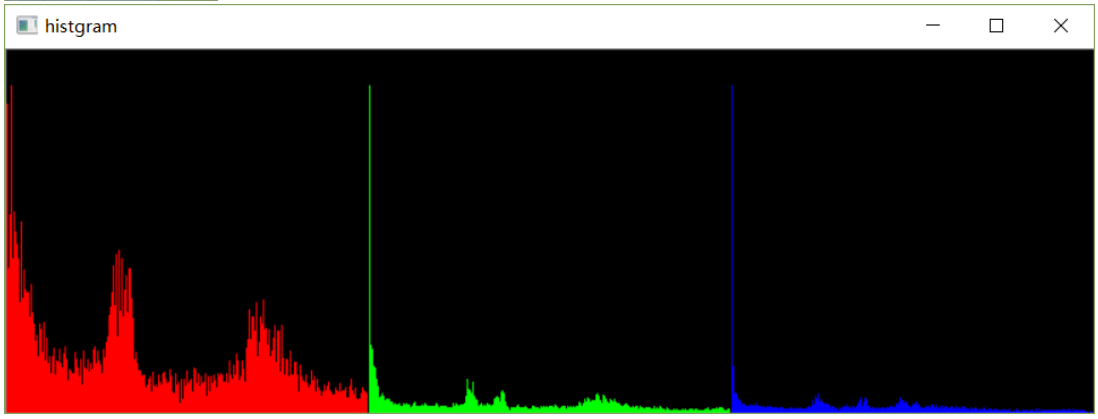
本来打算自己一个数据结构将直方图表示为 256*3（彩色图片）的向量，但在计算两个直方图的相似度时发现不好操作，而且精确度太低，于是采用了 opencv 提供的函数 cv::calcHist 来计算直方图，而后可以进行可视化，主要代码如下：

```
    //计算图像的直方图(红色通道部分)
    cv::calcHist(&rectImage, nimages, &channels[0], cv::Mat(), outputHist_red, dims,
&histSize[0], &ranges[0], uni, accum);
    //计算图像的直方图(绿色通道部分)
    cv::calcHist(&rectImage, nimages, &channels[1], cv::Mat(), outputHist_green,
dims, &histSize[1], &ranges[1], uni, accum);
    //计算图像的直方图(蓝色通道部分)
    cv::calcHist(&rectImage, nimages, &channels[2], cv::Mat(), outputHist_blue, dims,
&histSize[2], &ranges[2], uni, accum);

for (int i = 0; i < histSize[0]; i++)
    {
        float value_red = outputHist_red.at<float>(i);
        float value_green = outputHist_green.at<float>(i);
        float value_blue = outputHist_blue.at<float>(i);
        //分别画出直线
        cv::line(histPic, cv::Point(i*scale, histSize[0]), cv::Point(i*scale,
histSize[0] - value_red * rate_red), cv::Scalar(0, 0, 255));
        cv::line(histPic, cv::Point((i + 256)*scale, histSize[0]), cv::Point((i +
256)*scale, histSize[0] - value_green * rate_green), cv::Scalar(0, 255, 0));
```

```
        cv::line(histPic, cv::Point((i + 512)*scale, histSize[0]), cv::Point((i +
512)*scale, histSize[0] - value_blue * rate_blue), cv::Scalar(255, 0, 0));
    }
    cv::imshow("histgram", histPic);
```

效果如下：



二、 如何动态的通过交互来框出目标物体

主要利用的鼠标的回调函数 void onMouse 以及 Rect(Point, Point) 和画矩形函数 rectangle 来实现，先打开原视频，鼠标点击后暂停播放，记录矩形框的起点，鼠标抬起后记录第二个点，两个对角点围成的即为框出的目标图形：

```
    targetImage = image(Rect(originalPoint, processPoint));
                    //获取目标图像 targetImage
```

上诉部分通过 getStart 函数实现，然后采用 H-S 直方图进行处理，首先得配置直方图的参数，进行原图直方图的计算，然后归一化后传入计算，得到目标的可以进行相似度比较的直方图 MatND srcHist;

```
Mat srcHsvImage;
cvtColor(targetImage, srcHsvImage, CV_BGR2HSV);
//采用 H-S 直方图进行处理
//首先得配置直方图的参数
MatND srcHist;
//进行原图直方图的计算
calcHist(&srcHsvImage, 1, channels, Mat(), srcHist, 2, histSize,
ranges, true, false);
//归一化
normalize(srcHist, srcHist, 0, 1, NORM_MINMAX);
```

三、 如何对于下一帧的图片进行区域检索和直方图比较（重点）

首先由于之前的交互得到的矩形框的大小，而且视频的大小尺寸不会变，根据一般事实可知，物体一般的移动都是渐变的，即其未来出现的区域不

会离原来的矩形框区域太远，所以可以在矩形框周围选定适当的区域进行直方图相似度匹配搜索，经过调试后发现将区域改为（3*width）X（3*height）时效果会更好。

二狗在划定区域内保持矩形框大小不变进行左右上下的二重循环遍历，将“框”出的测试图片 compareImg 进行同样的操作而后得到其直方图，最后利用 opencv 的直方图比较函数 compHist，将其与目标进行比较，Opencv 提供的比较方法有四种：

Correlation 相关性比较

Chi-Square 卡方比较

Intersection 十字交叉性

Bhattacharyya distance 巴氏距离

巴氏距离计算公式：

$$d(H_1,H_2)=\sqrt{1-\frac{1}{\sqrt{\bar{H}_1\bar{H}_2N^2}}\sum_I\sqrt{H_1(I)\cdot H_2(I)}}$$

测试发现第四中巴氏距离效果要好一些，设计函数 compHist(const MatND srcHist,Mat compareImage)完成比较并返回秒速相似度的值，值越小表示越相似，为 0 时表示相同。

四、 效率问题

实验发现将矩形框进行加 1 平移，会大大增加计算量（视频会很卡顿），很难做到实时的对目标跟踪的效果，可以调整平移的步数，有两种方案，一是将左右和上下平移的步数设为矩形框的宽或高的若干分之一，或者将其设为较大一点的固定值，都能够增加效率，遍历区域不断比较直方图得到相似值 comnum，取最小的相似值时的检测图片，框出矩形框进行显示，另外对代码可以进一步重构优化。矩形框操作核心代码如下：（注意越界检查）

```
for (int Cy = Y1; Cy <= Y2; Cy += 10) {
        for (preStart.x = X1, preStart.y = Cy; preStart.x <= X2; preStart.x += 10) {
            if ((preStart.x + width) < image.cols)
                preEnd.x = preStart.x + width;
            else
                preEnd.x = image.cols - 1;
            if ((preStart.y + height) < image.rows)
                preEnd.y = preStart.y + height;
            else
                preEnd.y = image.rows - 1;
            Mat compareImg;
            compareImg = image(Rect(preStart, preEnd));
            double c = compHist(srcHist, compareImg);
            if (comnum > c) {
                get1 = preStart;
                get2 = preEnd;
                comnum = c;
            }
```

```
        }
    }
```

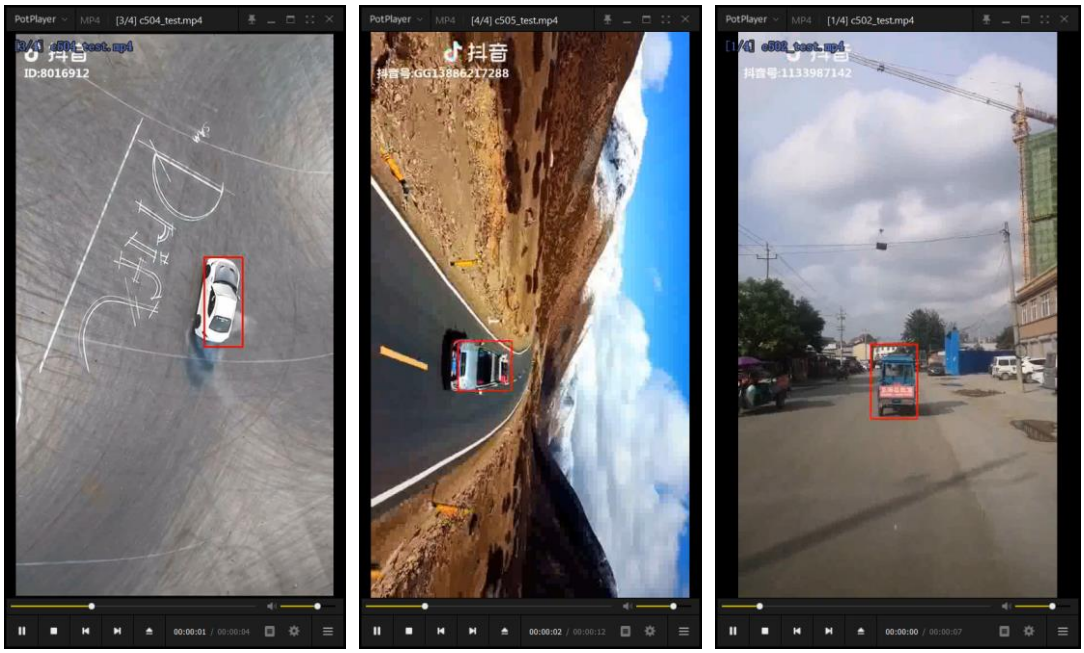
五、 标记显示和初始位置更新问题

由于物体是在运动的，所以需要更新计算区域的坐标位置，并且可以和直方图的比较相似度相结合，但检测区域和目标图像很接近时便更新矩形框到该位置，而且但相似度的值大于一定阀值后可断定目标被跟丢，不显示矩形框。代码如下：

```
//在原始视频图像上刷新矩形，只有当与目标直方图很相似时才更新起点搜索区域，满足目标进行移动的场景
        if (comnum < 0.15) {
            X1 = get1.x - width;
            X2 = get1.x + width;
            Y1 = get1.y - height;
            Y2 = get1.y + height;
            if (X1 < 0)
                X1 = 0;
            if (Y1 < 0)
                Y1 = 0;
        }
        if(comnum<0.5)
            rectangle(image, get1, get2, Scalar(0, 0, 255), 2);
```

六、 效果展示

处理好的视频截图如下：（可以观看上传的压缩包内的 c502_test.mp4、c503_test.mp4、c504_test.mp4、c505_test.mp4 视频）



经过测试后发现，对于目标姿态改变较小的并且在图像中大小变化不大的目标物体，能够保证有较高的识别追踪准确率。但是对于姿态变化和一些干扰较敏感。