专业：求是科学班（计算机）
姓名：蒋仕彪
学号：3170102587
日期：2019/11/27

浙江大学实验报告

课程名称：计算机视觉　　指导老师：宋明黎　　成绩：_____
实验名称：HW#2: 实现椭圆拟合

一、实验目的和要求

调用 CvBox2D cvFitEllipse2(const CvArr* points)实现椭圆拟合。

二、实验内容和原理

2.1 用 canny 算法构出图片轮廓

Canny 算法能有效地构出一张图片里的所有“轮廓”（颜色变化剧烈的“边”）。流程如下：

算法步骤：
1. 用高斯滤波器平滑图像.
2. 用一阶偏导有限差分计算梯度幅值和方向.
3. 对梯度幅值进行非极大值抑制（NMS）.
4. 用双阈值算法检测和连接边缘.

在 opencv 中我们不必重复造轮子，有一个现成的函数可以调用：

```
cvCanny ( CvArr* src,
          CvArr* dst,
          double threshold1,
          double threshold2,
          int aperture_size=3 )
```

这四个参数分别表示原始图片、目标图片、两个阈值。

这里涉及到两个阈值 th1 和 th2（th1<th2），是用来解决噪声的。Canny 使用了双阈值的方法，若一个像素值 > th2 称为 Strong edge，< th1 称为 Weak edge. **强边缘的像素点会直接被确定为边缘，弱边缘的像素点会直接被否决；对于位于中间的像素点来说，如果它和边缘的点相邻，则它也会被确定为边缘。**

```
Mat myCanny(Mat picOriginal, int th1, int th2) {
    Mat picEdge;
    Canny(picOriginal, picEdge, th1, th2);
    return picEdge;
}
```

2.2 提取连通块和椭圆拟合

OpenCV 里用

void findContours(InputOutputArray image,
OutputArrayOfArrays* contours,
int mode,
int method,
CvPoint offset=cvPoint())

来提取连通块。其中 image 表示用来处理的图片，结果储存在 contours 里（在 C++里我们可以直接传入 vector<vector<Point>> ）。Mode = CV_RETR_EXTERNAL 表示只检测最外围轮廓。CV_CHAIN_APPROX_NONE 会保存物体边界上所有连续的轮廓点到 contours 向量内。

执行了 findContours(image, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE) 后，我们会得到边缘图像 image 每个连通块里的所有点（存在一个 vector<vector<Point>> 里）。

接着我们要对每一个连通块执行椭圆拟合。

cvFitEllipse2 一般用于 C，所以我采用了等价的 RotatedRect fitEllipse(InputArray points) 来实现椭圆拟合。fitEllipse 支持传入嵌套的 vector，我们可以很方便地把之前的轮廓直接传进去。

注意调用椭圆拟合函数 fitEllipse 或 cvFitEllipse2 时，传入的点数至少要是 6 个。

还有一个基于视觉效果的考量：如何评估拟合结果呢？我想到的方法是，**在原来的连通块提取的结果上直接把拟合的椭圆画上去**，这样用户就能直观地感受拟合结果了。有个问题是: **ellipse（画椭圆函数）似乎把像素点的颜色叠加上去的，如果直接在边缘图像上画椭圆，会得到奇怪的复合颜色**。为了解决这个问题，我先新建了一个黑色的空画布，用

void drawContours(InputOutputArray image,
InputArrayOfArrays contours,
int contourIdx,
const Scalar& color)

把连通块提取结果给画上去，然后再把椭圆拟合的结果画上去。这样就能清晰地看到原来的边缘和红色的拟合结果。

具体代码如下：

```
void redraw(int, void*)
{
    Mat picEdge = myCanny(picOriginal, th1, th2);
    vector<vector<Point> > contours;
    findContours(picEdge, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);
    Mat ans = Mat::zeros(picEdge.rows, picEdge.cols, CV_8UC3);
    for (int i = 0; i < contours.size(); i++) {
        if (contours[i].size() < 6) continue;
        RotatedRect elipseResult = fitEllipse(contours[i]);
        drawContours(ans, contours, i, Scalar(255, 255, 255));
        ellipse(ans, elipseResult, Scalar(0, 0, 255), 1, CV_AA);
    }
    imshow("RESULT", ans);
}
```

2.3 增加滚动条

如何设置 Canny 算法的两个阈值很让我伤脑筋，因为不同的图可能需要不同的取值。对此我学习了 opencv 里设置滚动条的方法：

int createTrackbar(const string& trackbarname,
const string& winname,
int* value,
int count,
TrackbarCallback onChange = 0,
void* userdata = 0);

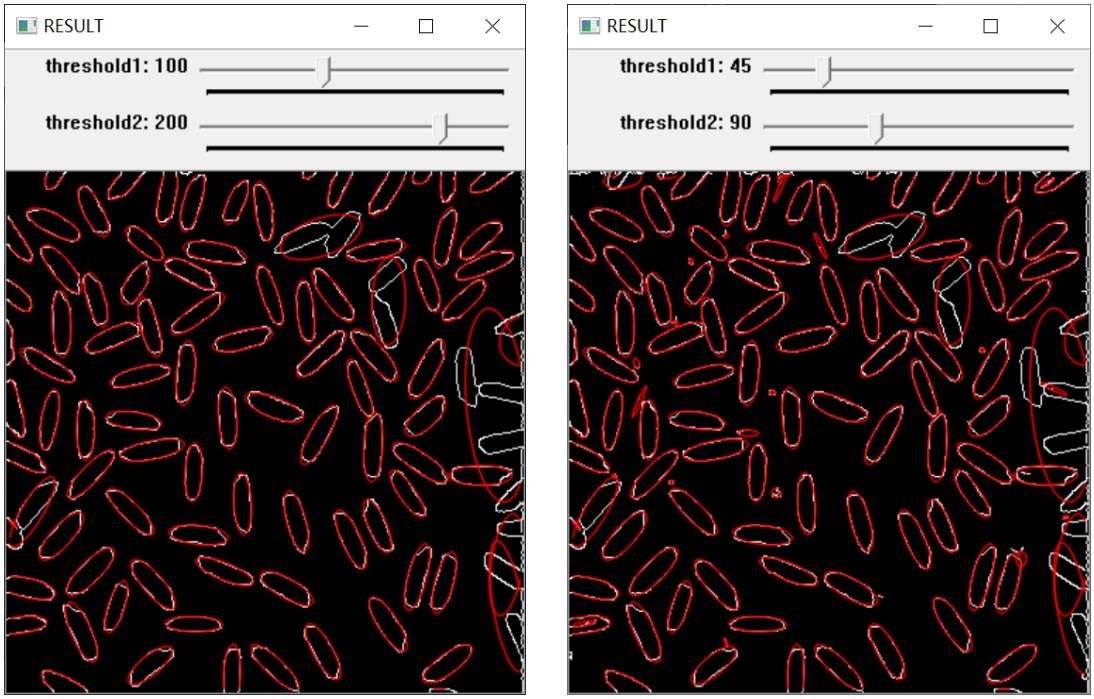
trackbarname 表示这个滚动条的名字，winname 表示窗口名字（我们要专门新建一个窗口，然后把这个滑动条以传入窗口名字的方式附在窗口上面），value 表示接受到的数值，count 表示滑动条上界，onChange 表示回调函数（用户移动了进度条后触发的函数事件）。

因为有两个阈值，我就分别开了两个进度条，用来接受 th1 和 th2 的结果。

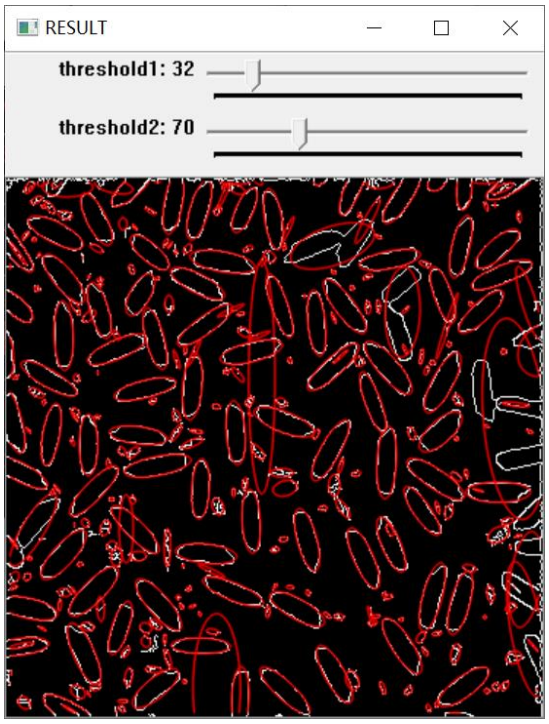
代码如下：

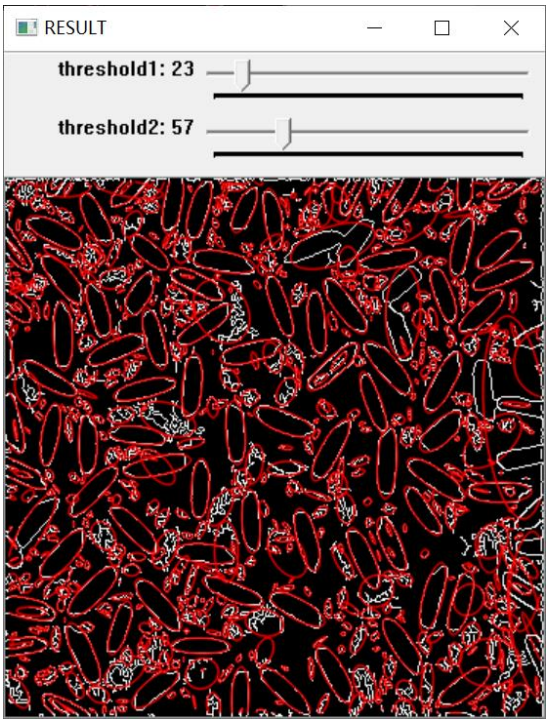
```
int th1 = 100, th2 = 200;
namedWindow("RESULT");
createTrackbar("threshold1", "RESULT", &th1, maxcolor, redraw);
createTrackbar("threshold2", "RESULT", &th2, maxcolor, redraw);
redraw(th1, 0);
redraw(th2, 0);
waitKey();
```

三、成果展示

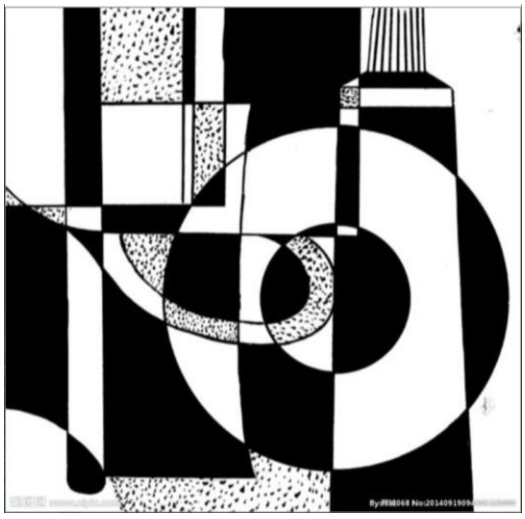


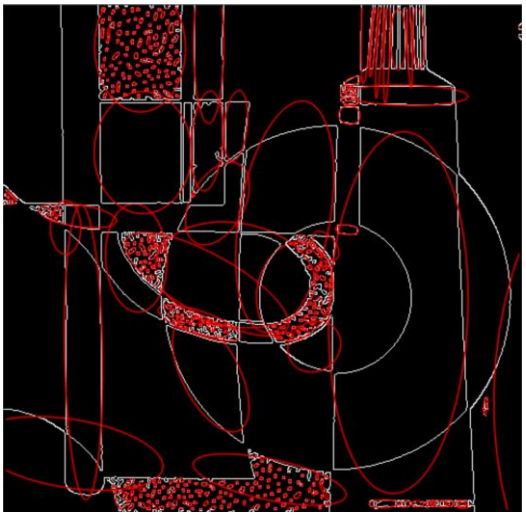
不同阈值下 sample 的结果





小阈值下 sample 的结果





测了一组其他图片

四、感想

本次实验步骤很简单，但是很有趣(#^.^#)。

我有一个需求是“在原图上勾出红色的椭圆”，之前这一步一直不成功，搞了半天才发现是 drawContours 函数的问题， 调了好久>_<。最后是直接在黑色画布上重画的。

椭圆拟合本身算法比较固定，我觉得影响结果的主要因素就是勾边勾的好不好。在跑 PPT 给出的例子时，我发现右边中间有几个图形在勾边的时候连在了一起，所以分连通块的时候也没分进去，直接影响了最后拟合的效果。我想过一些解决方法，比如手动剖开一些“快要分离”的图形（其实这个表述一点都不形式化，我不知道具体要怎么搞）；或者在每次椭圆拟合之后，算一下椭圆和原连通块的面积交和总面积的比例，判断此次拟合是不是成功。但是这些方法都不具有普适性，我最后就没有实现。