浙江大学

本科实验报告

椭圆拟合

课程名称：	计算机视觉
姓名：	夏豪诚
学院：	计算机科学与技术学院
专业：	信息安全
学号：	3170102492
指导老师：	宋明黎

2019 年 12 月 1 日

浙江大学实验报告

专业：信息安全
姓名：夏豪诚
学号：3170102492
日期：2019 年 12 月 1 日
地点：无

课程名称：计算机视觉　指导老师：宋明黎　成绩：
实验名称：椭圆拟合　实验类型：综合实验　同组学生姓名：无

一、 实验环境

表 1: 测试环境

item	detail
CPU	Intel® Core™ i7-6700HQ CPU 2.60GHz
RAM	16.0GB DDR4 2133MHz
hard disk	SSD 256GB
OS	Windows 10 Pro 64-bit
IDE	Microsoft Visual Studio Community 2019
OpenCV	3.4.5

二、 实验目的和要求

1. 实验目的

简单尝试使用 opencv 的库函数进行轮廓检测和曲线拟合，在本次实验中即表现为椭圆拟合。

2. 基本要求

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

三、 实验内容和步骤

1. 实验内容

实验内容主要为符合实验基本要求的编程实现。具体可以分为以下几个部分：

(1) 图片的读入；

(2) 图像二值化；

(3) 检测轮廓点；

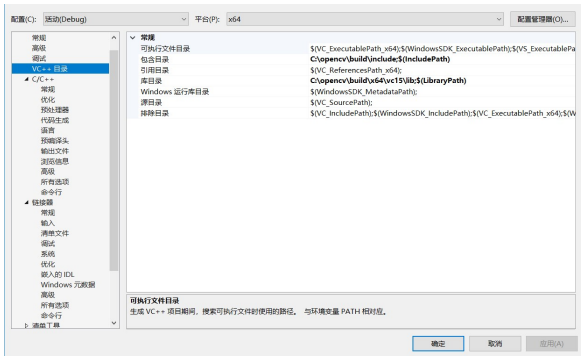
(4) 椭圆拟合与绘制

针对这四个部分的具体代码实现将在实验步骤中进行详细说明。

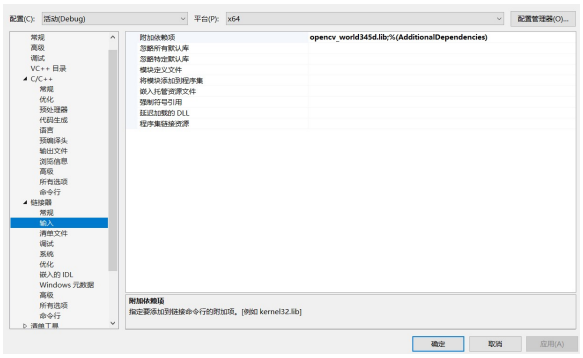
2. 实验步骤

3.2.0 实验环境配置

环境配置与上一次实验相同，在 Windows 10 操作系统的环境下，我们可以在官网下载 exe 后缀的可执行文件，运行后可以将编译好的对应版本 opencv 解压到制定的目录下，在本次实验中我选择的版本为 3.4.5，解压完成后为了可以在 VS 中调用 opencv 库函数我们需要修改 VC++ 目录下的包含目录和库目录添加 opencv 的解压位置，而后在链接器-> 输入-> 附加依赖项中添加 lib 文件。



(a) 配置图 1.



(b) 配置图 2.

图 1: VS 配置界面

3.2.1 图片的读入

本次实验输入相对比较简单，采用直接指定输入图像路径的方式实现图像的读入，具体代码如下：

```
if (argc == 2) {
  if (argv[1] == "--help") cout << "Usage:" << endl << "EllipseFitting.exe [your input
      filepath]" << endl;
  else filename = argv[1];
}
else {
  cout << "EllipseFitting.exe --help to see the usage." << endl;
  return 0;
}


// read img
Mat gray_img = imread(filename, IMREAD_GRAYSCALE);
Mat result = imread(filename);
```

值得注意的是在读入图像的同时，我们使用IMREAD_GRAYSCALE这一参数来获得了一幅灰度图像，这是为了给之后的图像二值化处理提供方便。

3.2.2 图像二值化

图像二值化是通过检测灰度图像中不同像素的亮度，根据我们设定的阈值来进行区分，亮度高于阈值的我们将其对应像素值的亮度设置为最大值，即255，否则设置为0，这样整幅图像中的像素点也就变成了不是最小值 0 就是最大值 255 的情形，这样可以为之后的检测轮廓点提供遍历。具体代码如下：

```
// set a resonable threhold
 Mat binary_img = gray_img;
 double sum = 0;
 double size = 0;
 for (int row = 0; row < gray_img.rows; row++) {
   for (int col = 0; col < gray_img.cols; col++) {
     uchar& m = gray_img.at<uchar>(row, col);
     sum += m;
     size++;
   }
 }
 int thresh;
 thresh =(int)(sum*1.11 / size);
 cout << "thresh" << thresh << endl;

 // get binary img
 for (int row = 0; row < gray_img.rows; row++) {
   for (int col = 0; col < gray_img.cols; col++) {
     uchar& m = gray_img.at<uchar>(row, col);
     uchar& dest_m = binary_img.at<uchar>(row, col);
     if (m >= thresh) {
       dest_m = 255;
     }
     else dest_m = 0;
   }
 }
```

通过代码可以看到在设置阈值的时候，我选择的测策略是先对整幅灰度图的所有像素的亮度取一个平均值，而后在直接使用的这个平均值来进行处理的时候，我发现效果并不令人满意，之后再查看了二值图图像后选择对这个平均值乘上一个系数后作为最后的阈值，如上代码所示这个系数为 1.25，在乘上这个系数后，相对而言处理的结果有了比较明显的提升。

3.2.3 检测轮廓点

检测二值化图像轮廓点的函数为findContours()，contours变量用于记录得到的轮廓点。第三个参数RETR_LIST表示检测所有的轮廓，包括内围、外围轮廓，但是检测到的轮廓不建立等级关系，彼此之间独立，没有等级关系，而第四个参数CHAIN_APPROX_NONE保存物体边界上所有连续的轮廓点到 contours 向量内。换言之，在 contours 向量内的每一个元素都是一个单独的轮廓。

```
// get contours
 vector<vector<Point>> contours;
 findContours(binary_img, contours, RETR_LIST, CHAIN_APPROX_NONE);
```

3.2.4 椭圆拟合与绘制

在进行拟合时由于椭圆的拟合至少需要 6 个点，我们会把把少于 6 个点的检测结果直接丢弃。接下来通过对椭圆拟合原理的阐述说明这一点。对于椭圆总有一般方程：

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{1}$$

在拟合时就是通过获得的点集数据代入，求解这些参数，显然六个未知量的方程我们需要 6 个点为我们提供六个方程的方程组才能够求解。回到本次实验的要求，是调用 CvBox2D cvFitEllipse2(const CvArr* points) 实现椭圆拟合。根据在之前的代码中我们得到的 contours 向量，我们需要根据其中的每个独立的轮廓中的轮廓点集，使用 **CvBox2D** 记录最小面积的包围矩形，但是 **CvBox2D** 为 opencv 中的 c 函数，在 c++ 对应其的为 **RotateRect**，同样的用于真正实现轮廓点返回这个最小面积的包围矩形的 **cvFitEllipse2**，在 c++ 中的代替函数为 **FitEllipse**，得到了这个最小面积的包围矩形后我们就可以直接使用ellipse函数将椭圆轮廓绘制在原图上。

```
// fitellipse and draw on the pic
  for (int i = 0; i < contours.size(); i++)
  {
    if (contours[i].size() >= 6)
    {
      RotatedRect box = fitEllipse(contours[i]);
      ellipse(result, box, Scalar(0, 0, 255), 1, LINE_4);
    }
  }
```

四、 主要仪器设备

计算机，Visual Studio 2019

五、 实验结果

1. 编译运行

在 VS2019 中编译成功后，在控制台运行可执行程序，并指定图片路径，运行效果如图所示：

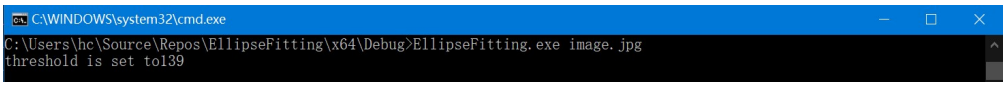


图 2: 控制台运行效果

2. 文件变化

我们首先查看目录下的文件，可以看到出现了运行的结果，二值图binary.jpg和拟合结果result.jpg。

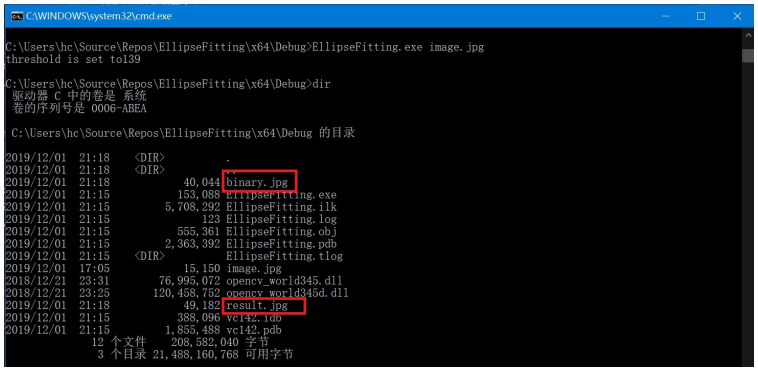


图 3: 文件变化

3. 运行结果

拟合使用的原图即为课件中给出的图像image.jpg：

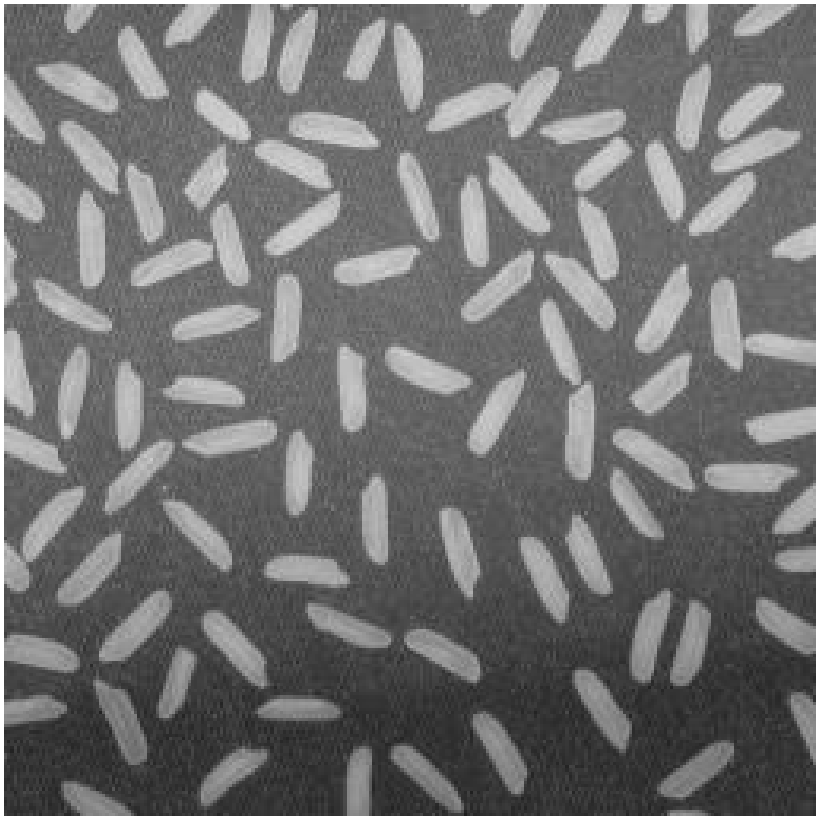
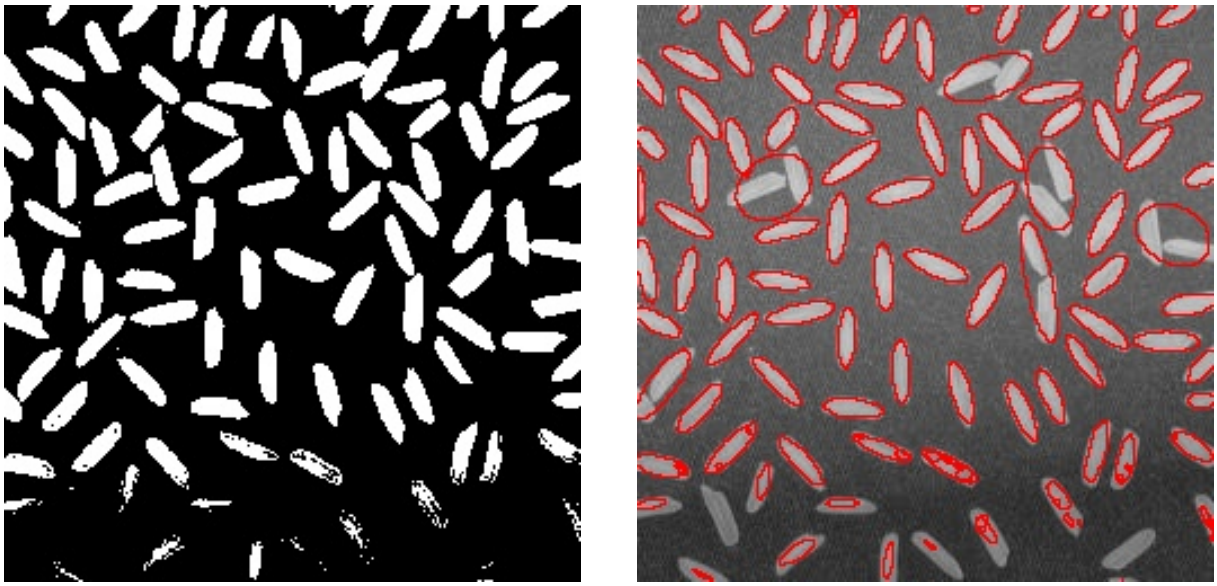


图 4: 原图

得到的二值图binary.jpg和拟合结果result.jpg如下：



(a) 二值图 (binary.jpg)

(b) 拟合图 (result.jpg)

图 5: 运行结果

六、 实验结果分析

在本次实验中总体而言没有遇到很多困难，但最后的结果还是不够令人满意，在对比了不同阈值下的二值图和结果后，我总结了以下问题，并对解决方案提出了一些设想，希望能够随着之后课程的进行，设计更好的方案来解决这些问题。

(1) 由于全图的光照强度不一致，在测试样例中我们可以明显看到图片的下部偏暗，这导致一旦阈值相对偏大，下方的椭圆图案就会在二值化时直接消去；

(2) 上部中存在着不同的椭圆图案之间距离过近的问题，它们之间较小空隙中的像素实际上的亮度值也是相对较高的，这直接导致了在阈值设置不够高的情况下，这些非椭圆图案内的点也会被判定为二值化后亮度为 255 的点，为之后进行椭圆拟合带来了困难。

通过分析后我们可以明显发现，这两个问题的本质实质上都指向了图片的整体亮度不均，但是局部区分相对较为明显这一状况。针对这个问题，我想到的解决方案是在二值化图像时使用局部阈值，对该点周围合理区域内的像素的亮度值进行采样，而后得到阈值，这样就可以从根源上解决图片的整体亮度不均的问题。本次实验很大程度上激发了我对问题的探索意识和解决问题的意识，尽管最后没有在提交的代码中很好解决这个问题，但相关的思考还是极大地激发了我对计算机视觉的好奇与兴趣，希望能在之后的课程中学到更多的知识。