

**视听觉信号处理实验报告**

实验 二

题 目 实验A：找到枣子和橘子的数量

学 院 未来技术学院

专 业 人工智能（视听觉信息处理）

学 号 7203610121

学 生 刘天瑞

任 课 教 师 姚鸿勋

哈尔滨工业大学计算机科学与技术学院

2022年秋季

说明

实验报告一般包含以下几个部分：实验内容，实验目的，实验设计、算法和流程，实验结果，结论，参考文献。

注意：不要把所有代码都粘贴到实验报告中，如确有必要，请粘贴少量关键代码即可，源代码单独提交！

Notes: Experimental report usually includes following sections: experiment content, experiment purpose, experiment design、algorithm and procedure， experimental results, conclusion, reference.

Do not copy and paste all source code into the report, if necessary, please paste some key codes. At the same time, please submit the total source code and the executive program.

**实验一 报告**

1. **实验内容（contents）**

1. 选择合适的图像处理算法找到图1.（源文件见附件）中的橘子和枣子的数量。  
2. 在1.的基础上确定每个水果的外边界，并使用边界线或者mask将属于水果的像素点标注出来。



图1 橘子们和枣子们

1. **实验目的（purposes）**

综合运用图像处理中的知识解决实际问题，以及可能出现的多种多样的情况。

1. **实验设计、算法和流程(Design, algorithm and procedure)**

**3.1算法介绍**

1. **二值化（特征增强）**

思路：实验指导书中这一步要做的就是将水果和白板背景大致地区别出来，因为整体的思路是通过计算轮廓面积的范围来区分这些，为了能让轮廓更加分明和清晰，此时就需要二值化水果和背景白板。这两个部分越冲突（差距越明显）对后续轮廓勾勒处理就越有益。

所以在主函数里读取图像时使用灰度图。（参数1为原图，其后续画轮廓时使用，参数0为灰度图）

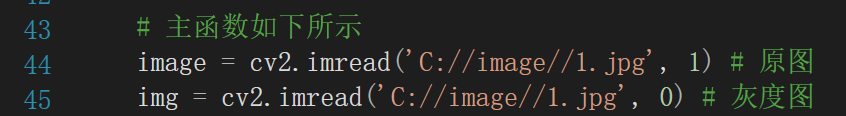


图2 读取图像（原图和灰度图）

之后遍历一次该灰度图，设定一个阈值进行二值化处理，使得小于该阈值（即颜色偏暗）的像素用0来重新替代该灰度值（直接变成黑色），使得大于这个阈值（即颜色偏亮）的像素用255来替代该灰度值（直接变成白色），如此就可以很好地得到一个只有两个灰度的新图片（虽然实际会有很多其他问题，这部分只是理想的构思，后续报告会有解决新问题的方法）。

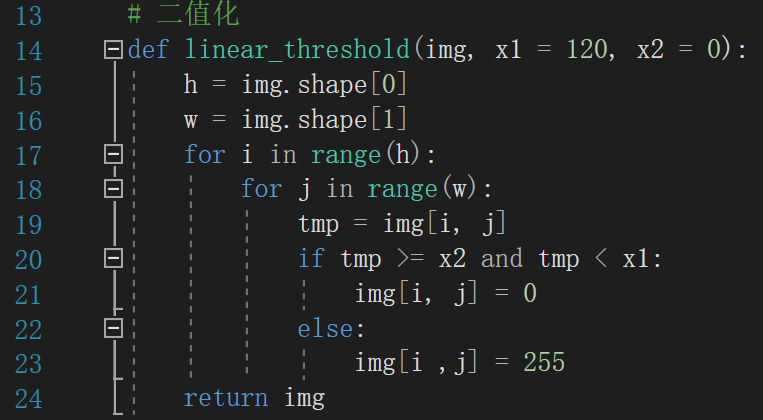


图3 灰度图二值化处理（设置双阈值）

我在一些博客查询资料后决定采用opencv库里的findContours()函数来找到高对比度图像的轮廓，而轮廓面积可以使用contourArea()函数进行计算，接下来就是将图像转化成这两个函数满足的接口规格。而通过像素值对图像进行灰度处理，经过多次尝试后选择了120这个灰度值，可以尽量小化阴影对轮廓识别的影响，也能够避免橘子内部轮廓被识别成枣子。二值化部分我认为最重要的是判断水果或是背景的边界灰度阈值。如果灰度选的过小，虽然可以避免一些阴影的影响，但是会将反光的部分也变得很大。最后在面积判断时反光的面积大于枣子的面积将导致反光部分判断为枣子，而枣子反而可能会被判断为反光；而相反如果阈值选的过大，虽然反光部分处理得很好，但是会把很多阴影放到水果部分，这样勾勒出来的轮廓也不是很准确。

阴影问题我认为一直是本次实验需要完善的地方，虽然在后面的开操作部分理论上可以优化，但是实际操作仍有不足，所以有一个优化的突破口就是在一开始二值化时直接把阴影当成白色桌面背景。仔细观察可以发现，阴影的灰度一般会比正常的水果低，只需再设置一个阈值，然后对小于该阈值的像素直接用255替代，这样我们就可以得到合适的滤去阴影的效果。所以不妨设置阴影阈值为0，这样能提前筛去一部分的阴影。

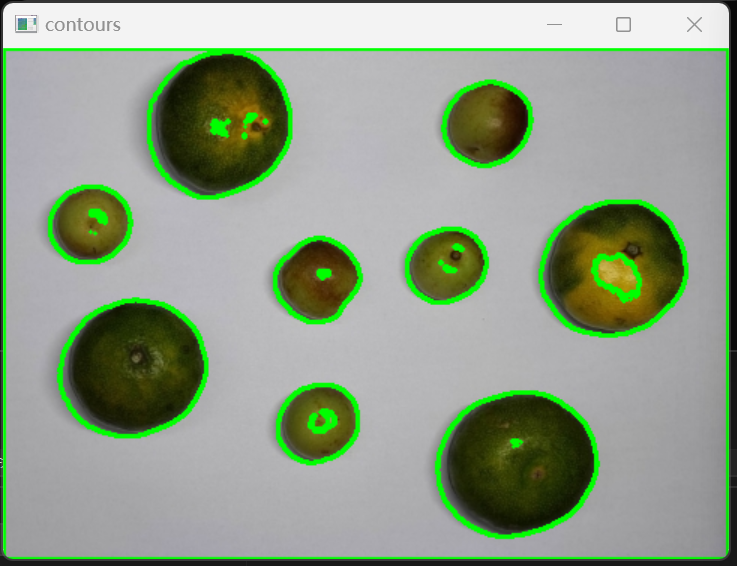


图4 原始图像描轮廓处理（阈值合适，效果较好）

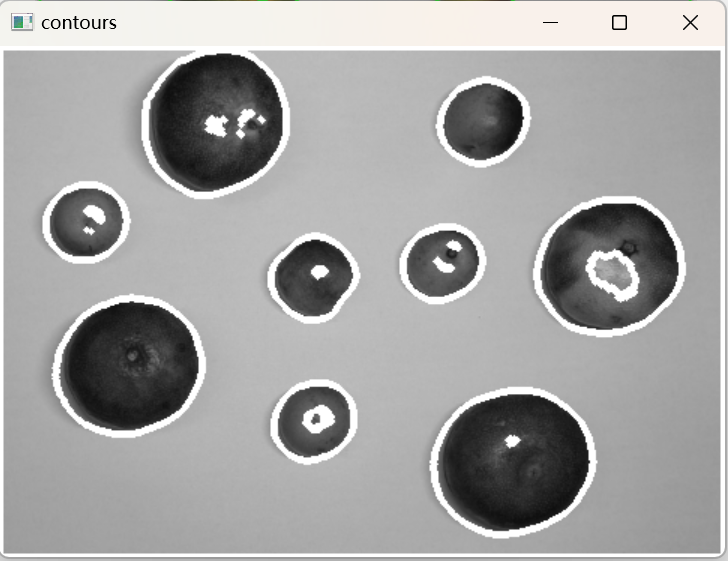


图5 灰度图像描轮廓处理（阈值合适，效果较好）

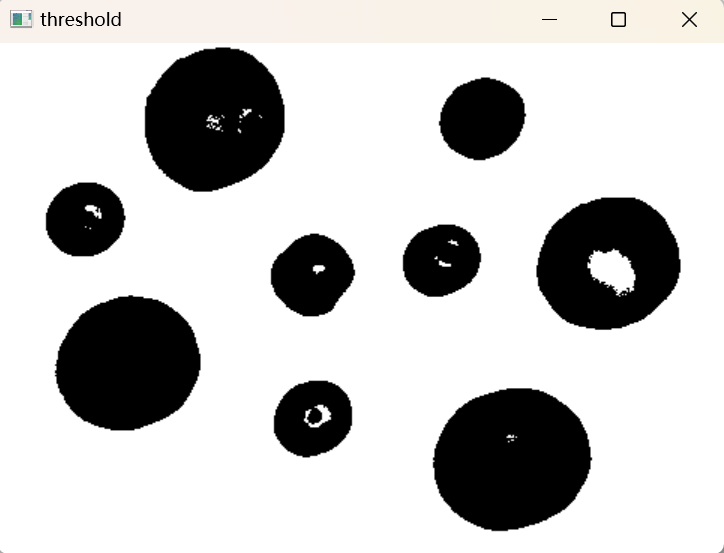


图6 灰度图像二值化处理后效果



图7 阈值过大情况下会导致将很多阴影划进去

1. **膨胀腐蚀**

在二值化的过程中遇到的第一个问题就是阴影部分很难解决，首先有的阴影像素的灰度值肯定比水果的灰度小，而有的阴影像素的灰度比水果的大，所以不能简单地再设置一个阈值然后把阴影筛去。所以我想到的一个办法就是形态学里面的腐蚀膨胀，这种开运算可以让我们把一些和图片略有脱离的阴影去掉，还有就是腐蚀掉一些反光的地方，如下图部分：

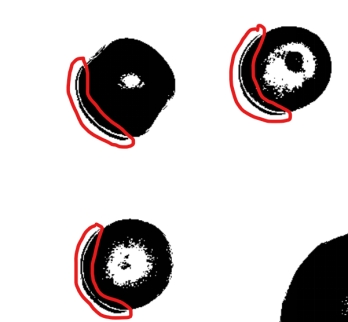
 

图8 执行开操作之前

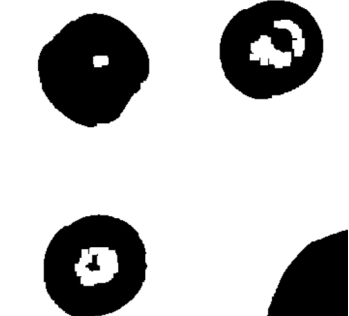
 

图9 执行开操作之后

腐蚀膨胀的算法较为简单，直接可以调用opencv库里的现有函数。设置好参数就可以解决。

1. **轮廓选择和面积计算**

对于如何区分枣子和橘子，我想到最简单的方法就是找出轮廓并且通过计算轮廓面积来区分不同水果，最后可以通过面积设定一个阈值来得到枣子和橘子的个数。

找轮廓可以调用opencv库里的函数：

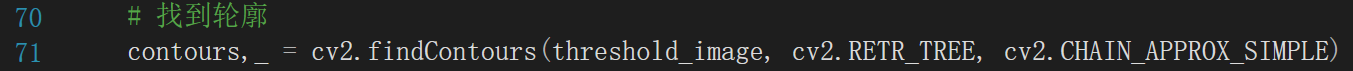
****

图10 寻找轮廓的findContours()函数

计算面积也可以调用opencv库里的函数：



图11 计算轮廓面积的contourArea()函数

**3.2算法实现**

1. **二值化（特征增强）**

首先遍历整个图像，确定图片的高度h和宽度w，然后进行2维循环，遍历的空间复杂度为O(n2)。

随后设定阈值，这里我通过多次实验，找到了阈值为120时效果不错，这种情况下既不会将太多阴影认为是水果，也不会将太多反光认为是桌面背景（虽然还是会有，但也不可避免）。

当然，这部分还适当进行了优化处理：某些阴影灰度过低，于是可以将这些特别低的阴影像素查找出来，然后直接以灰度值为255来替代，让它变为白色背景。

1. **腐蚀膨胀（开操作）**

这一部分不再赘述，直接调用库函数即可。

详细讲一下getStructuringElement和morphologyEx的参数和返回值：

getStructuringElement()的第一个参数表示内核的形状，有三种形状可以选择，矩形MORPH\_RECT交叉形；MORPH\_CROSS椭圆形以及MORPH\_ELLIPSE。第二和第三个参数分别是内核的尺寸以及锚点的位置。一般在调用腐蚀膨胀运算之前先定义一个Mat类型的变量来获得getStructuringElement()函数的返回值。对于锚点的位置，有默认值Point（-1,-1），表示锚点位于中心点。element形状唯一依赖锚点位置，其他情况下，锚点只是影响了形态学运算结果的偏移。返回的参数是指定形状和尺寸的结构元素。morphologyEx()的返回参数是目标图像。第一个参数是源图像；第二个参数是int类型的op，表示形态学运算的类型，具体如下所示：

* MORPH\_OPEN – 开运算（Opening operation）
* MORPH\_CLOSE – 闭运算（Closing operation）
* MORPH\_GRADIENT -形态学梯度（Morphological gradient）
* MORPH\_TOPHAT - “顶帽”（“Top hat”）
* MORPH\_BLACKHAT - “黑帽”（“Black hat“）

第三个参数就是输入图像的内核，即形态学运算的内核。若为null的时候，表示的是使用参考点位于中心3\*3的核。我们一般使用函数getStructuringElement配合这个参数的使用。getStructuringElement函数会返回指定形状和尺寸的结构元素（内核矩阵）。

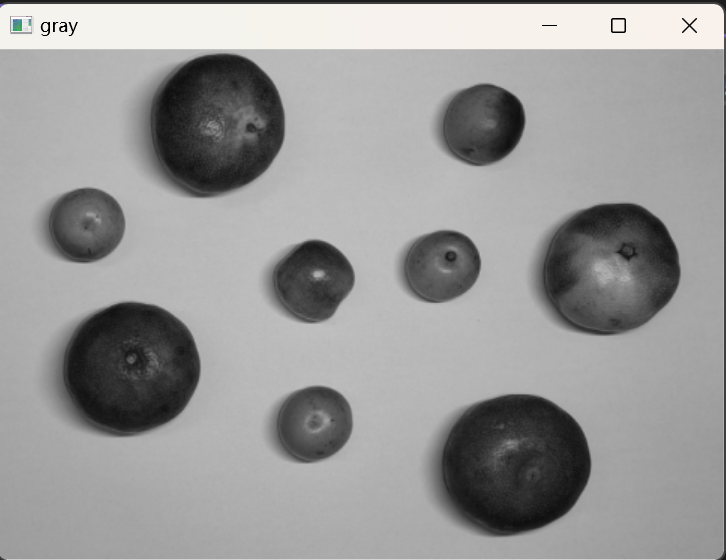


图12 灰度图

1. **轮廓选择和面积计算**

在这个过程中需要对开操作后的图像的边界，通过这个边界得到所需要的轮廓，最后对这个轮廓进行面积计算。其实仅需要对不同面积进行一个筛选，然后对不同范围的面积贴标签（是枣子或是橘子）。这样就能统计出枣子的个数和橘子的个数。

来详细讲讲findContours()函数。findContours()的第一个参数是image，即输入的原始图像，一般是二值化后的图像，黑色为背景，白色为目标。第二个参数是method，例如cv2.RETR\_CCOMP参数表明建立一个等级树结构的轮廓。当然还有别的参数例如cv2.RETR\_EXTERNAL只检测外轮廓。第三个参数是CV\_CHAIN\_APPROX\_SIMPLE， 表明仅保存轮廓的拐点信息，把所有轮廓拐点处的点保存入contours。findContours()返回值contours是轮廓点。列表格式，每一个元素位一个3维数组（其形状为(n,1,2)，其中n表示轮廓点的个数，2表示像素点坐标）表示一个轮廓。还有一个返回值hierarchy是轮廓间的层次关系（本次实验还未用到），为三维数组，形状为(1,n,4)，其中n表示轮廓总个数，4指的是用4个数表示各轮廓间的相互关系。第一个数表示同级轮廓的下一个轮廓编号，第二个数表示同级轮廓的上一个轮廓的编号，第三个数表示该轮廓下一级轮廓的编号，第四个数表示该轮廓的上一级轮廓的编号。然后就是找到轮廓后通过一个面积大小的滤波器，将面估计过小的轮廓（光影）删去，将较小的轮廓记为枣子，将较大的轮廓记为橘子，将最大的轮廓（图片边界）删去。

所以要调用另一个计算面积的库函数就是contourArea()。该函数输入是一个列表格式的contours，返回值是面积。首先遍历每一个轮廓（contours），并且计算这个轮廓的面积。根据已得到的经验：一些为0的面积是独个像素点导致的，还有一些较小的面积则是识别出了水果表面的纹路，如果面积小于1000则说明是反光，如果面积在1000到3000之间，说明是枣子，面积在6000到9000之间说明是橘子，而超过9000则只可能是整张图像。最后符合面积在1000到9000的水果再让其进入一个新列表，并且重点关注下分别对枣子和橘子进行计数，具体实现过滤器的代码如下图所示：

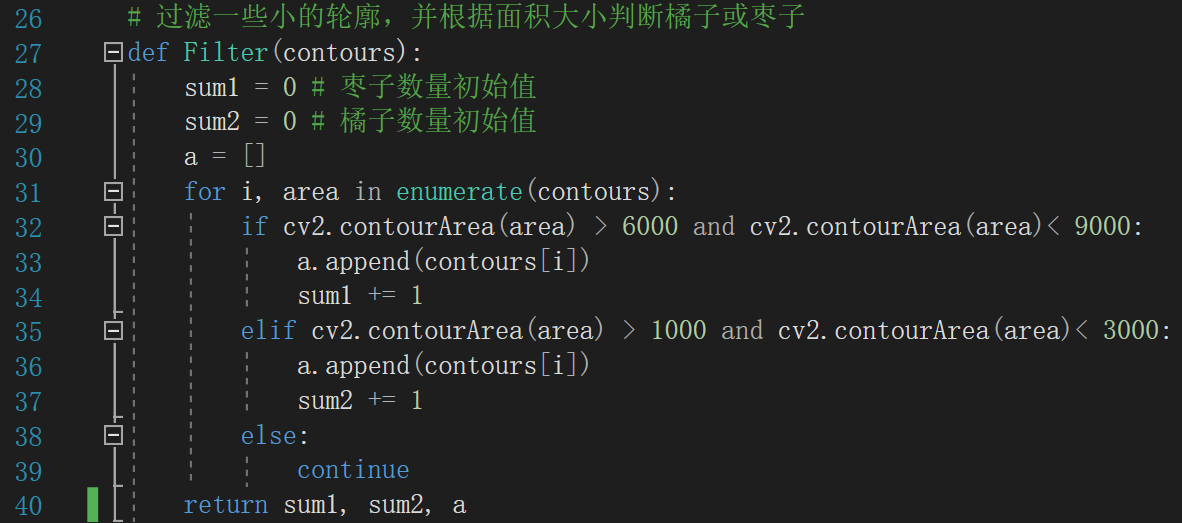


图13 选择轮廓的过滤器Filter()函数

1. **在原图上勾勒出轮廓**

记得在上一步中得到了contours这个列表元组，在这个过程中需要将contours在原图上画出来，可以调用以下库函数如下图所示：

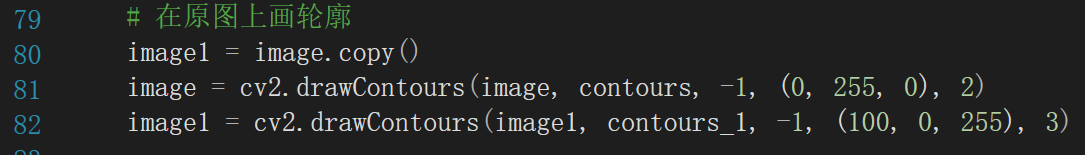


图14 绘制轮廓的drawContours()函数

drawContours()函数的第一个参数是背景图，第二个参数是需要描点的列表，第三个参数是需要绘制轮廓的编号，如果是-1，则绘制所有的轮廓，第四个参数是画线的颜色，通过RGB三通道表示，最后一个参数是像素点的大小（线的粗细）。

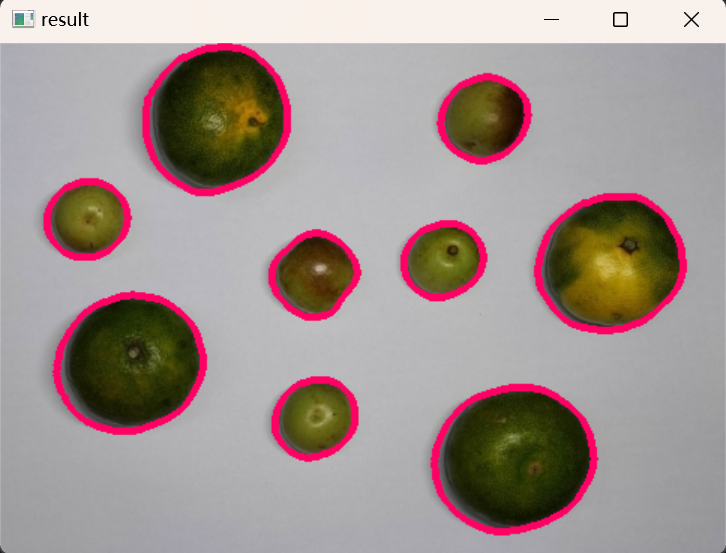


图15 勾勒出水果轮廓的效果

1. **Canny算子**

Canny算子用于轮廓边缘检测。这里简单介绍一下具体过程（非实验重点）：

1.计算图像梯度，得到可能边缘。因为梯度是灰度变化明显的地方，而边缘也是灰度变化明显的地方。因为灰度变化的部分很难判断，而首先就是得到了所有可能为边缘的集合；

2.非极大值抑制。通常灰度变化的地方都比较集中，将局部范围内的梯度方向上，灰度变化最大的保留下来，其他不保留，这样可以剔除掉绝大部分点。将有多个像素宽的边缘变成一个单像素宽的边缘。即“胖边缘”变为“瘦边缘”；

3．双阈值筛选。通过非极大值抑制后，仍然有很多可能的边缘点，进一步设置一个双阈值，即低阈值和高阈值。灰度变化大于高阈值的，设置为强边缘像素，而小于低阈值的直接剔除。在低阈值和高阈值之间设置弱边缘。进一步判断如果其领域内有强边缘像素，则保留，否则剔除。这样做的目的只是保留强边缘轮廓，但是有些边缘可能不闭合，从而需要满足低阈值和高阈值之间的点进行补充，使得边缘尽可能闭合。

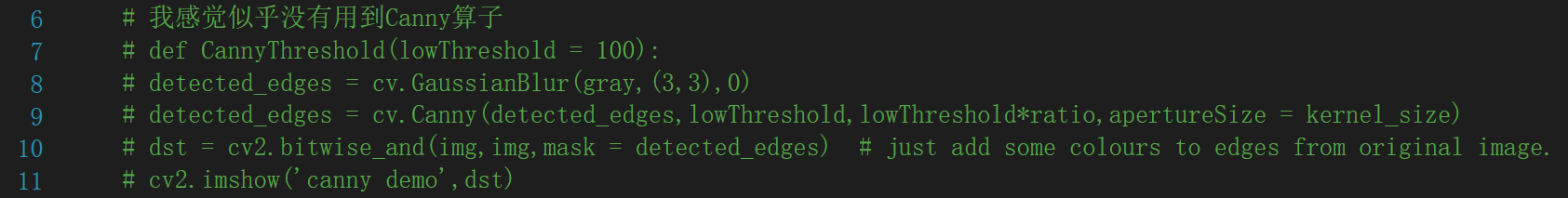


图16 似乎没有用到的Canny算子

1. **实验结果(results)**

发现能够正确识别不同水果大小，但是轮廓边缘存在问题，原因主要是将图像转换为灰度图进行处理导致的，若采用RGB原始图像处理可能识别效果更好。

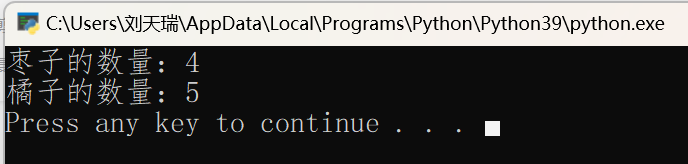


图17 分类计算结果

1. **结论(conclusion)**

我认为视听觉的第二次实验与第一次实验相比更为方便易懂，因为本次实验允许引入图像处理opencv库从而使得解决思路清晰直接。而实验输出结果成功的关键在于了解每个调用的库函数其输入参数含义和输出值的接口规范，合理地调用库会使得代码封装性更强，也通俗易懂了很多。最后，我还做了如下总结：

1.了解并掌握了什么是二值化、腐蚀膨胀、轮廓等图像处理的方法或函数；

2.了解并掌握了Canny算子的原理和应用；

3.从整个实验过程来说图像处理的方法有很多，并且也有很多库函数可以调用，但是选择什么样的处理方法来解决和优化图像很重要，有的方法效果一般，没有必要使用（比如上面实验中没有必要用到Canny算子来描绘边缘轮廓，简单地找轮廓函数即可实现）；

4.了解双阈值在图像处理中的重要作用。一个是在二值化中可以过滤阴影和反光，另一个是在Canny算子，高阈值是将要提取轮廓的物体与背景区分开来，就像阈值分割的参数一样，是决定目标与背景对比度的；而低阈值用来平滑边缘的轮廓，有时高阈值设置过大可能导致边缘轮廓不连续或者不够平滑，通过低阈值来平滑轮廓线，或者使不连续的部分连接起来。

1. **参考文献(reference)**

[1] [Python - OpenCV 图像二值化处理\_小小晓晓阳的博客-CSDN博客\_python opencv 二值化](https://blog.csdn.net/bugang4663/article/details/109589177?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166730375716800182113571%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166730375716800182113571&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-4-109589177-null-null.142%5ev62%5eopensearch_v2,201%5ev3%5eadd_ask,213%5ev1%5et3_esquery_v1&utm_term=opencv%E4%BA%8C%E5%80%BC%E5%8C%96&spm=1018.2226.3001.4187)

[https://blog.csdn.net/bugang4663/article/details/109589177?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522166730375716800182113571%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=166730375716800182113571&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~top\_click~default-4-109589177-null-null.142^v62^opensearch\_v2,201^v3^add\_ask,213^v1^t3\_esquery\_v1&utm\_term=opencv%E4%BA%8C%E5%80%BC%E5%8C%96&spm=1018.2226.3001.4187](https://blog.csdn.net/bugang4663/article/details/109589177?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166730375716800182113571%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166730375716800182113571&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-4-109589177-null-null.142%5ev62%5eopensearch_v2,201%5ev3%5eadd_ask,213%5ev1%5et3_esquery_v1&utm_term=opencv%E4%BA%8C%E5%80%BC%E5%8C%96&spm=1018.2226.3001.4187)

[2] [Imgproc.findContours函数 - chen2013 - 博客园 (cnblogs.com)](https://www.cnblogs.com/chenjianxiang/p/4225782.html)

<https://www.cnblogs.com/chenjianxiang/p/4225782.html>

[3] [OpenCV-计算轮廓面积cv::contourArea\_翟天保Steven的博客-CSDN博客\_cv.contourarea](https://blog.csdn.net/zhaitianbao/article/details/116256678?ops_request_misc=&request_id=&biz_id=102&utm_term=opencv%E7%9A%84contourarea&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-116256678.142%5ev62%5eopensearch_v2,201%5ev3%5eadd_ask,213%5ev1%5et3_esquery_v1&spm=1018.2226.3001.4187)

[https://blog.csdn.net/zhaitianbao/article/details/116256678?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=opencv%E7%9A%84contourarea&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-0-116256678.142^v62^opensearch\_v2,201^v3^add\_ask,213^v1^t3\_esquery\_v1&spm=1018.2226.3001.4187](https://blog.csdn.net/zhaitianbao/article/details/116256678?ops_request_misc=&request_id=&biz_id=102&utm_term=opencv%E7%9A%84contourarea&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-116256678.142%5ev62%5eopensearch_v2,201%5ev3%5eadd_ask,213%5ev1%5et3_esquery_v1&spm=1018.2226.3001.4187)

**注意：正文内容为小四，1.5倍行距**