计算机视觉大作业——瓶盖检测

一、题目要求

在自制的UI界面上输入一张有瓶盖的图片，将图片中不同姿态的瓶盖染成不同颜色，并且把瓶盖中心点的坐标显示出来。

二、开发工具

语言：C++

开发软件：Visual Studio，Qt

1. 算法介绍

**整体流程：**先将图片转成灰度图，并对其进行Gaussian平滑处理，使用区域分割保证图中的每一个瓶盖都能被找到，再使用HoughCircles对灰度图进行霍夫变换圆检测，并得到正反瓶盖的圆心及其半径，并使用多边形逼近找到侧面瓶盖，（正侧检测另一种方式是先区分连通域，再通过外接圆判别是否是正或反的瓶盖，并通过多边形拟合找到侧面瓶盖），再通过检测瓶盖上螺纹的圈数区分瓶盖的正反。

**3.1分割区域**

1. 目的：将图片的每个瓶盖区域用其外接矩形分割出来，便于后续检测瓶盖的轮廓。
2. 流程：
3. 二值处理图像：

由于图像的背景为黑色，先将图像二值化，阈值取得较低（尽量使瓶盖区域都是白色的）。

1. 分割图像：

先将图像进行简单的高速滤波滤去噪点。

创建一个大小等同于图片的二维int数组（以下用a表示），用于标记不同的瓶盖；初始值为0。

用count记录数到第几个瓶盖，初始值为1。

对图像进行一次循环遍历，若其a不为0则不处理（说明已经遍历过），否则继续处理。

若像素值为0，则将a记为-1；否则为发现了像素值为255的点，将其a记为count，并从a开始，进行BFS遍历，将与其连通的所有像素值为255的点的a记为count，将count加一，说明这是个合法的瓶盖域，然后继续循环遍历。遍历结束，得到各个瓶盖的区域，count就是瓶盖区域的数量。

1. 优化：

以下是对BFS算法进行的一些优化：

1. 在每次BFS中，记录这些点的数量，若数量小于一定值（说明是由于光照或瓶盖边缘等因素产生的干扰），则忽视这个域，将其a记为-1，像素值记为0（同化为背景），并不增加count。
2. 为了提升算法效率，BFS采用的是八连通算法。但是当中心点的四连通个数不足2时，说明这个点处于边缘，若不特殊对待这些点，可能会使本来分开的两个瓶盖由于其间隔较小，受不稳定因素连在一起，从而影响之后的判断。于是当某点的四联通个数不足2时，算法不把这个点的邻点放置于BFS的队列中。后来发现，在实际测试中，这是一个比较大的优化，不仅使瓶盖能正确分割，还对瓶盖的边缘进行了平滑处理，一定程度上优化了后续的步骤。
3. 在BFS中，记录每个区域的最大、最小的x\y值，确定其外接矩形，以便之后用矩形划分图像区域，并记录该矩形的大小。

不足：无法区别颜色相近且相连的瓶盖。

1. 效果：

经过测试和上述优化，对于约30张测试图片，每张无相连瓶盖的图片都可以做到10个瓶盖域。

对于每个瓶盖域，用其瓶盖的实际大小（即白像素点个数，记为size）除以其外接矩形的面积（记为area)。由于圆是自对称图案，在给定一定误差范围后，圆瓶盖的size/area约等于0.78（π/4）。虽然矩形由于角度不同，它size/area理论值在0.5-1之间浮动，但由于形状简单，在之后的判断中较简单；有瓶盖相连的区域，由于其连接处必定下凹，且总体形状不规则，size/area较低，area又大，也可以区分出来。

对于每一个区域，由于上述分割确保了区域中都有瓶盖，所以当参数的拟合度较高而无法检测出矩形或者圆形时，可逐步降低拟合度，直到得到一个矩形或者圆形为止。

**3.2正侧检测**

整体思路：正上方拍照时，正面瓶盖形似圆行，侧面虽然会有圆弧，但更接近矩形，且四个角点角度接近直角。

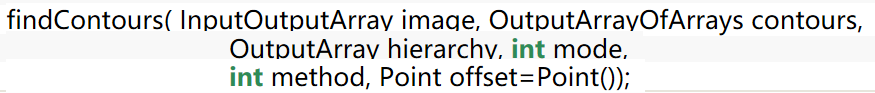
**3.2.1检测圆**

1、方法一

* 1. 思路：对图像中每个瓶盖形成的连通域进行分析，识别出圆。
  2. 流程：

1. 区分连通域：首先，利用开操作及适合的滤波，将原图片转化为边缘较为清楚的二值图像，通过opencv内置函数findContours()找出连通域外边缘，存放在数组contours中。边缘如图1。

附上findContours()函数说明。



Image：单通道图像矩阵，可以是灰度图，但更常用的是二值图像，一般是经过Canny、拉普拉斯等边缘检测算子处理过的二值图像。

Contours：定义为“vector<vector<Point>> contours”，是一个向量，并且是一个双重向量，向量内每个元素保存了一组由连续的Point点构成的点的集合的向量，每一组Point点集就是一个轮廓。有多少轮廓，向量contours就有多少元素。

Hierarchy：不重要，用不到，是Contours的一种索引。

Mode：定义轮廓的检索模式：

取值一：CV\_RETR\_EXTERNAL只检测最外围轮廓，包含在外围轮廓内的内围轮廓被忽略。

取值二：CV\_RETR\_LIST 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，这就意味着这个检索模式下不存在父轮廓或内嵌轮廓。

取值三：CV\_RETR\_CCOMP 检测所有的轮廓，但所有轮廓只建立两个等级关系，外围为顶层，若外围内的内围轮廓还包含了其他的轮廓信息，则内围内的所有轮廓均归属于顶层。

取值四：CV\_RETR\_TREE，检测所有轮廓，所有轮廓建立一个等级树结构。外层轮廓包含内层轮廓，内层轮廓还可以继续包含内嵌轮廓。

Method：定义轮廓的近似方法：

取值一：CV\_CHAIN\_APPROX\_NONE 保存物体边界上所有连续的轮廓点到contours向量内。

取值二：CV\_CHAIN\_APPROX\_SIMPLE 仅保存轮廓的拐点信息，把所有轮廓拐点处的点保存入contours向量内，拐点与拐点之间直线段上的信息点不予保留。

取值三和四：CV\_CHAIN\_APPROX\_TC89\_L1，CV\_CHAIN\_APPROX\_TC89\_KCOS使用teh-Chinl chain 近似算法。

Point偏移量，所有的轮廓信息相对于原始图像对应点的偏移量，相当于在每一个检测出的轮廓点上加上该偏移量，并且Point还可以是负值。

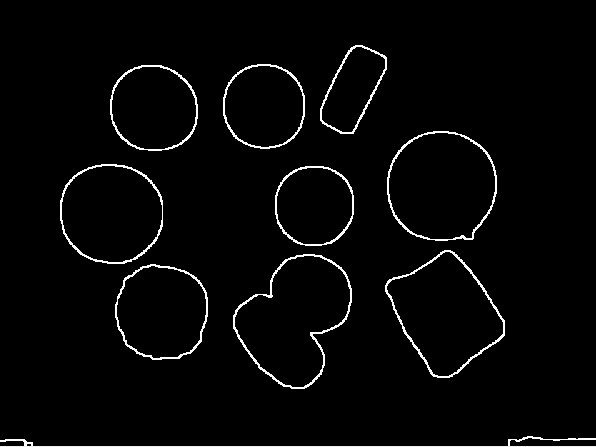


图1

1. 判别圆：对于连通域里的图形，选择contours中三点P1、P2、P3，计算P1P2P3的外接圆，判断contours中是否其余点是否都在圆上。

但是由于考虑到多瓶盖相连的情况，如图中红框所示。为了保证选取到的三点属于同一个瓶盖，将contours中的所有点按照y轴坐标由小到大排序，y坐标最大的点即为连通域的最上方一点，记为P1(x1,ymax)。然后令y = ymax-h（相当于水平线向下h进行扫描），扫描到的最靠近P1的左右两点为P2、P3。由此，可以计算出外接圆的圆心和半径信息。

图2

P3

P2

P1

判断contours上是否有超过3\*r个点在圆上，若有，说明有°的圆是存在的，则这个圆成立。否则，可能为矩形，可能为多瓶盖形成的其他连通域。

图3

1. 方法二
2. 思路：通过霍夫变换进行圆的检测
3. 流程：
4. 首先将原始图片转化为灰度图，对灰度图进行高斯平滑处理，过滤噪音，防止检测出多余的圆
5. 使用HoughCircles对灰度图进行霍夫变换圆检测。

HoughCircles函数的原型为：void HoughCircles(InputArray image,OutputArray circles, int method, double dp, double minDist, double param1=100, double param2=100, int minRadius=0,int maxRadius=0 )

image为输入的灰度图像。

circles为输出圆向量，每个向量包括三个浮点型的元素——圆心横坐标，圆心纵坐标和圆半径。

method为使用霍夫变换圆检测的算法，在Opencv2.4.9中只实现了2-1霍夫变换，它的参数是CV\_HOUGH\_GRADIENT。

dp为第一阶段所使用的霍夫空间的分辨率，设dp为1，表示霍夫空间与输入图像空间的大小一致。

minDist为圆心之间的最小距离，如果检测到的两个圆心之间距离小于该值，则认为它们是同一个圆心。

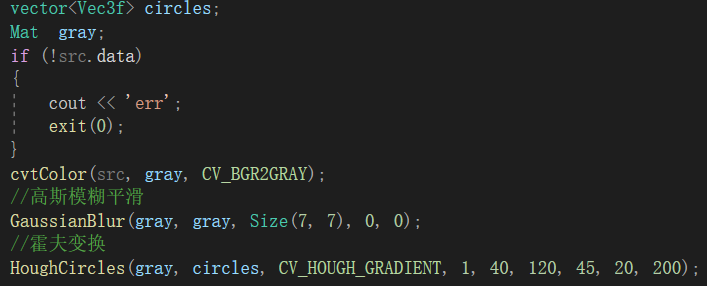
param1为表示传递给canny边缘检测算子的高阈值。

param2为检测阶段圆心的累加器阈值。越小表示对检测的圆规范性要求越低，更易检测出更多的不规范的圆。

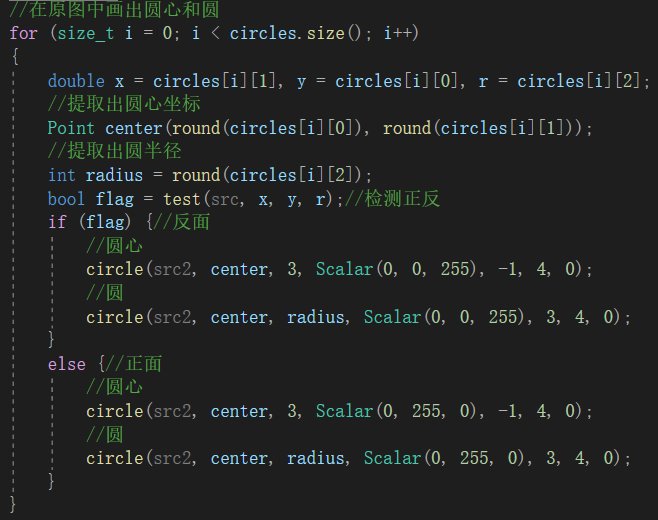
minRadius和maxRadius为所检测到的圆半径的最小值和最大值。

根据我们拍摄的图片，将参数设置为：



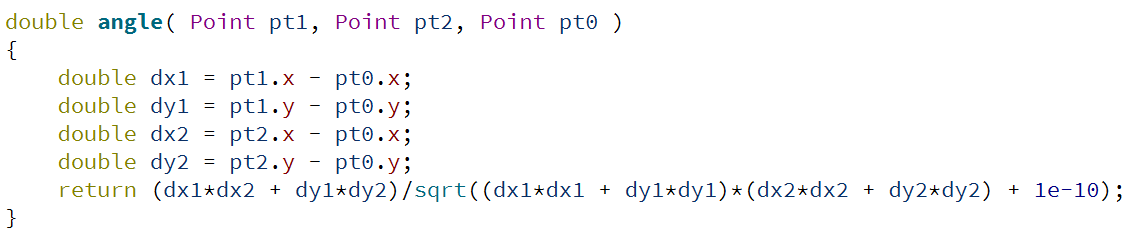


1. 根据circles中的圆心和半径，在原始图片中用标出检测的圆的圆心和半径。



**3.2.2检测矩形**

1. 思路：对图像中每个瓶盖形成的连通域进行分析，识别出矩形。
2. 流程：
3. 对于某一连通域，利用多边形拟合函数opencv::approxPolyDP()将边缘拟合为多边形。（往下有approxPolyDP()函数说明）
4. 如果拟合之后的多边形是凸多边形，且有条边，那么说明它可能是一个矩形，接下来用angle函数分析它的四边对应夹角对应的余弦值。



将该四边形的最长边以及四个角中角度最小（余弦值最大）的角筛选出来，若是该角足够接近直角，认为该四边形为矩形，即可当成是侧面瓶盖。

下面附上approxPolyDP()函数说明：



功能：该函数是以指定的精度逼近一条多边形曲线

说明：该函数是用一条具有较少顶点的曲线/多边形去逼近另一条具有较多顶点的曲线或多边形，并且它们之间的距离小于等于指定的精度，其中算法采用Douglas-Peucker algorithm。

参数：

curve：存储在std::vector或Mat中的2D曲线点集

approxCurve：逼近的曲线，其类型需要与输入曲线的类型匹配

epsilon：用于指定逼近精度的参数，为原始曲线与逼近曲线的最大距离

closed：曲线是否闭合的开关（若为真，则逼近曲线为闭合曲线）

1. 考虑到类似上图连通域中包含多个瓶盖的情况，考虑将连通域中正放瓶盖剔除。

检测圆中方法二利用霍夫圆较精确地检测出所有正反放瓶盖，并且将圆的圆心与半径信息记录在circles结构中，则可以利用circles信息，将连通域中在circles中任一圆上的点剔除掉。

如下文函数所示，contours[i]中存储的是所有边缘像素信息，circles中存储了所有检测出的圆心点，利用OnCircle()函数判断contours[i]中每个点是否在圆上，若在，则删除contours[i]中此点，在如图的例子中，最终contours[i]边缘中留下的是图4中的蓝色区域（由于光照及开操作等影响，蓝色边缘在实际瓶盖的周围，这并不影响结果）。经过多边形拟合后，得到的就是图3中的绿色区域。

图3

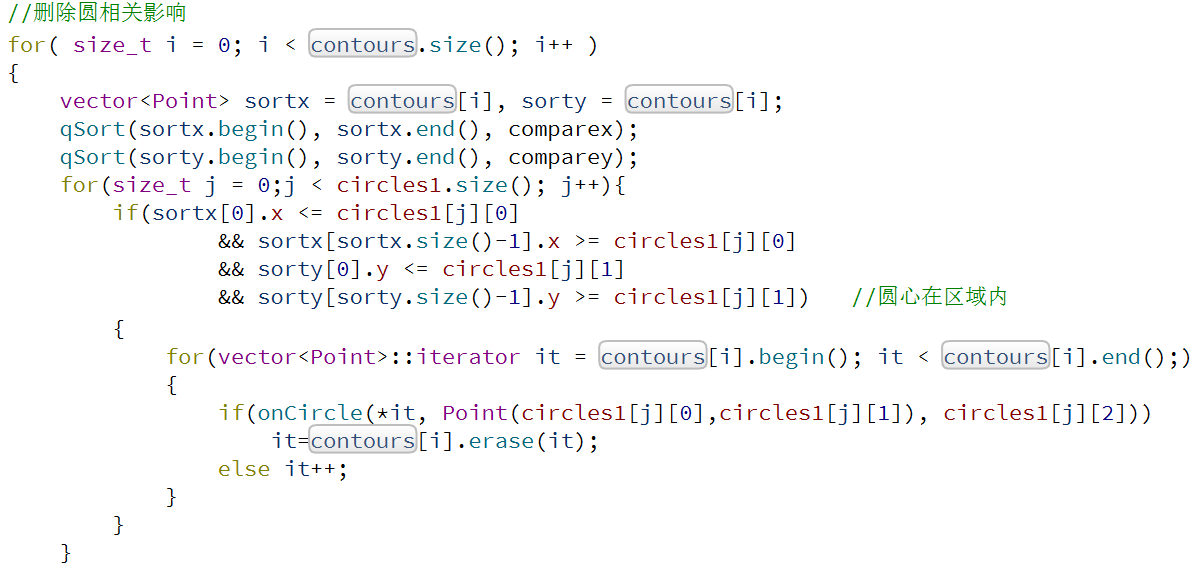


图4

**3.3正反检测**

（1）思路：对于瓶盖的正反，较为明显的区别是，反面的瓶盖在盖内总会有几圈圆环，因此从该方面着手，正面的瓶盖外圈基本只有一个圆，但是反面的瓶盖有两个或以上的圆。我们认为通过检测圆的个数可以区别正反。

（2）流程：

a. 对输入的原始图片通过Gaussian滤波后进行canny边缘检测，得到图像中所有轮廓

b. 由之前的正侧检测得到正反瓶盖的圆心以及它们的半径，通过检测同心圆的存在与否来判断是不是反面瓶盖。

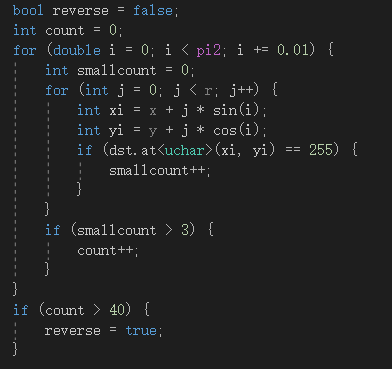
c. 对于所有的圆，在每一个圆心位置处，在以这个点为圆心，半径为r/2的圆内，将所有像素点数值设为0

d. 我们再创建一个初始为0的数count，对于一个圆心，选取一个方向，在这个方向上辐射出一条长度为这个圆的半径的线段，检查这条线段上白点的个数，如果大于3，则count++

e. 对于所有方向均做此处理后，检查count的值，如果count的值大于40，那么判断这个圆是瓶盖的反面，否则是正面

重要代码展示：

白点计数：



1. 函数调用说明

* 分割区域

void Widget::split()

* 正侧检测
* 检测圆

void Widget::searchCircles()

* 检测矩形

void Widget::searchSquares()

* 正反检测

bool Widget::test(Mat& src, int x, int y, int r)

此函数在searchCircles()中被调用，即在识别出圆后对每一个圆进行正反检测。

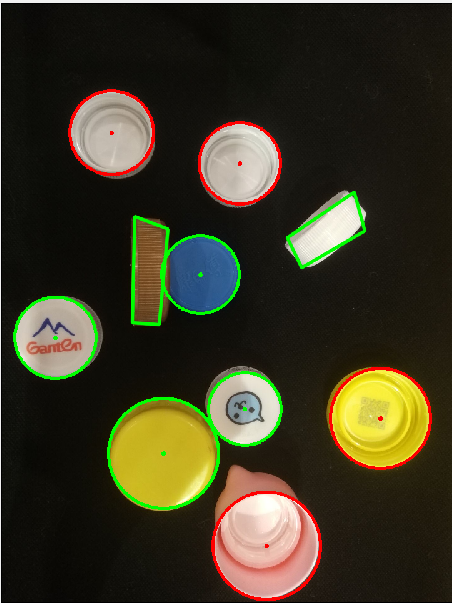
1. 效果展示

标识说明：

红色圆：反面；

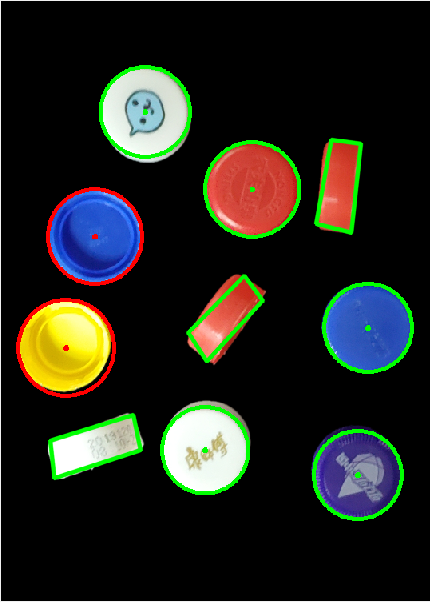
绿色圆：正面；

绿色框：侧面。

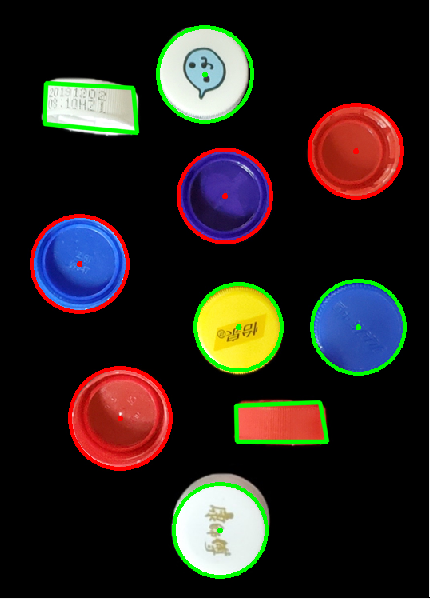
（图一）

（图二）

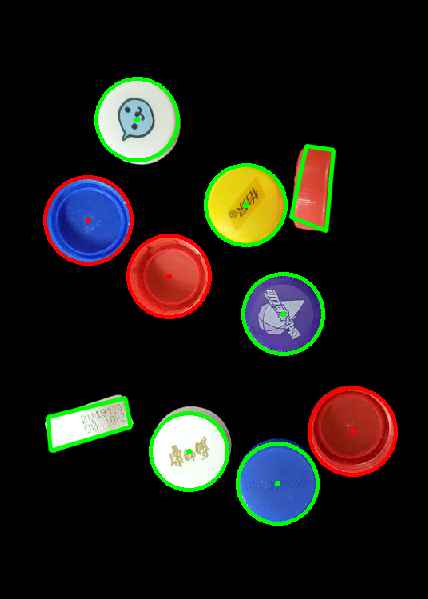
（图三）

（图四）

（图五）

（图六）

（图七）

（图八）

（图九）

（图十）

六、成员分工

|  |  |  |
| --- | --- | --- |
| 小组成员 | 完成工作 | 贡献度 |
| 黄雨瑒 | 正侧检测 | 25% |
| 周一凡 | 分割区域 UI | 25% |
| 董彦君 | 正侧检测 | 25% |
| 李思洋 | 正反检测 | 25% |