

SHANGHAI JIAO TONG UNIVERSITY

**瓶盖状态识别程序报告**

项目名称： 瓶盖状态识别程序

组员姓名： 刘千禧、唐毓瑾

谢佳航、王新哲

任课教师： 孙焱

日期: 2020 年1月4日

目录

[SHANGHAI JIAO TONG UNIVERSITY 1](#_Toc29070820)

[**瓶盖状态识别程序报告** 1](#_Toc29070821)

[一、 综述 2](#_Toc29070822)

[1. 概述 2](#_Toc29070823)

[2. 小组分工 2](#_Toc29070824)

[二、 技术实现细节 2](#_Toc29070825)

[1. 软件架构 2](#_Toc29070826)

[2. 物理环境 3](#_Toc29070827)

[3. 前端UI界面与本地服务器 3](#_Toc29070828)

[4. 图片模型 3](#_Toc29070829)

[5. 识别模型 4](#_Toc29070830)

[三、 运行使用说明 6](#_Toc29070831)

[1. 前端环境 6](#_Toc29070832)

[2. 后端环境 6](#_Toc29070833)

[3. 其他 6](#_Toc29070834)

1. 综述
   1. 概述

该项目是2019~2020大三下学期，计算机视觉课程（SE342）的瓶盖状态识别小组大作业，要求编写一组程序，识别一副图片中十个瓶盖的位置及正、反、侧三种状态，并在识别出侧面瓶盖时准确描述其方向。

* 1. 小组分工

|  |  |
| --- | --- |
| 姓名 | 分工 |
| 刘千禧 | 软件架构设计，UI界面，图片有效部分裁剪 |
| 唐毓瑾 | 正反面瓶盖与侧面瓶盖识别 |
| 谢佳航 | 正面瓶盖与反面平盖识别 |
| 王新哲 | 正面瓶盖与反面平盖识别 |

1. 技术实现细节
   1. 软件架构
      1. 软件一共分为4个主要部分，前端，本地服务器，图片模型，识别模型。
      2. 软件的数据流及架构如图所示。



* 1. 物理环境
     1. 使用纸盒限制外界光照的影响。
     2. 整个拍摄过程均使用IPhoneX拍摄，只使用IPhoneX自带闪光灯进行照明。
     3. 将所有的瓶盖置于一张白纸上。
  2. 前端UI界面与本地服务器
     1. 前端UI界面基于react框架，使用JavaScript编写。
     2. 本地服务器使用tornado包，使用Python3编写。
     3. 前端框架将用户选择的图片以base64字符串的形式发送给本地服务器，本地服务器将base64字符串发送给图片模型。
     4. 获取结果时，本地服务器从图片模型获取base64字符串形式的图片。
  3. 图片模型
     1. 图片模型使用Python3编写，在接收图片后，图片模型将其转化为pilImage，然后对其进行裁剪处理，截取其有效部分，在识别完成后，再根据识别信息标记图片。
     2. 识别前处理
        1. 图片缩放

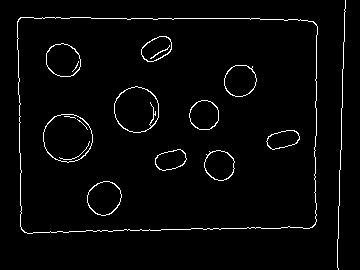
对图片进行缩放，将图片大小改变为400\*300。因为分辨图片中的有效部分并不需要很高的分辨率，缩放后可以加快程序的运行速度。

裁剪前图片如下：



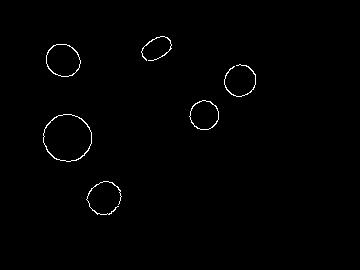
* + - 1. 得到边缘图

使用大卷积盒的均值滤波去除噪声，使用低阈值的canny算法提取边缘，得到下图：



* + - 1. 得到瓶盖图

使用大卷积盒的高斯滤波过滤缩放后的图片，使用高阈值的canny算法提取边缘，此时大部分的瓶盖都将被提取，而A4白纸边缘和其他背景将不被提取，如图所示：



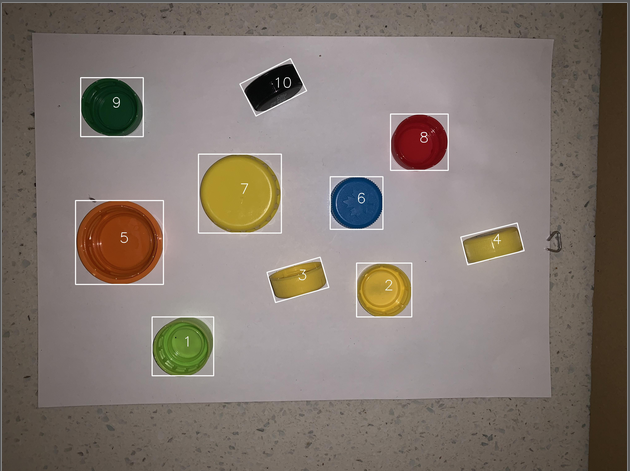
* + - 1. 得到有效边

结合边缘图和瓶盖图，在边缘图中选择较长的且与边缘接近平行的直线，计数瓶盖图中有多少点落在边缘图中的两条直线内，通过计数值最高的四条直线（上下左右各一条，也可能不存在有效边）可以将图片裁剪，再根据比例映射到原图（缩放前），得到下图：



* + 1. 识别后处理
       1. 标记图片

根据识别信息对原图进行标记，如图所示：



* 1. 识别模型
     1. 识别模型使用Python3编写，接手图片模型裁剪后的图片，识别瓶盖的状态，并使用一个列表返回。
     2. 正反面与侧面识别
        1. 图片缩放

对图片进行resize，将图片大小改变为400\*300。原因有两个，第一，分辨瓶盖的正反面与侧面并不需要很高的分辨率，resize后可以加快程序的运行速度。第二，在后面识别时会涉及圈出瓶盖的矩形框的长度和宽度，需要保证图中的瓶盖大小。

* + - 1. 均值滤波

对缩小后的图片进行均值滤波，达到去噪的效果。

* + - 1. 边缘检测

然后再用canny算子进行边缘检测，同时对图像进行二值化处理。

* + - 1. 轮廓检测

使用OpenCV中的findContours函数来查找检测图片中所有的轮廓，对获得的轮廓进行遍历，使用boundingRect函数获得将找到的形状包起来，函数返回矩形左上角坐标(x,y)和矩形的宽和高。为排除其他非瓶盖物体对检测结果的影响，仅当矩形的宽和高在[10,100]之间时判定矩形内是瓶盖。

考虑到正面和反面的瓶盖基本都是圆形的，圈住正面和反面瓶盖的矩形长宽比接近1，但由于拍摄角度的原因，图片上正面和反面放置的瓶盖并不都是圆，所以设置的筛选条件是当矩形的宽与矩形的高的比值在(0.85,1)或(1,1.15)之间,或为1时，判定矩形内瓶盖是正面或者反面。

使用minAreaRect函数找到包围物体的最小旋转矩形，当矩形的宽和矩形的高的比值大于1.2或者小于0.8时，判定矩形内瓶盖是侧面。

* + - 1. 返回值存储

使用五个列表来存储符合条件的矩形的四个顶点坐标与一个表示瓶盖方向的向量(x,y,z)。当符合筛选条件时，将矩形的对应参数加到list中。当矩形内为正面或反面时，x,y为0，z=1或-1，当矩形内为侧面时，z=0,x,y分别表示矩形边与x轴方向所成的较小角的余弦值与正弦值。

* + 1. 正面与反面识别
       1. 定义瓶盖口朝下为正面，口朝上为反面。瓶盖正面与反面的一个主要区别是：反面的瓶盖有封口圈，而正面的瓶盖表面没有那么多圆形边缘。基于这个特性，我们选择opencv的霍夫变换工具来检测瓶盖中含有圆的数量来判断瓶盖的正反。
       2. 预处理

先对图像使用双边滤波做降噪处理，因为双边滤波具有局部性，可以在消除噪声的同时尽可能保留边缘信息，为后面检测螺纹提供便利。相比之下均值、高斯等滤波器虽然降噪效果更好，但是会暴力地破坏边缘信息，导致最终正面反面长一样。双边滤波的降噪能力不及前面这些滤波器，但是它滤不掉的往往是瓶盖颗粒表面被闪光灯照出的小亮点，这些小噪声可以通过形态学开操作去除。

之后为了进一步凸显边缘信息，我们尝试了增加对比度和图片锐化两种方法。其中增加对比度对明暗差距明显、饱和度高的瓶盖效果明显，但对明暗差距小的瓶盖效果不佳。对于黄色这种灰度值较小的瓶盖，增加对比度甚至会导致颜色“瓦楞化”，引进原本不存在的边缘，导致后面的操作误判。最终我们选择对各种颜色瓶盖一视同仁的锐化操作来增加边缘信息。为了不引进噪声，锐化的力度不是很大，最终的效果可能不是很明显，但至少保证了不会引入新的误差

* + - 1. 圆检测

由于螺纹不是连续的圆，从而霍夫变换检测圆时圆心累加器的阈值不能设置太高，否则圆会检测不出来。同时由于不可避免的噪声存在，最终边缘检测出来的圆形的边往往比较厚，所以我们要设置略大的圆心最小距离来避免一个圆重复检测。（当螺纹处于瓶盖正中心时又有可能少检测圆，但实践表明大部分情况下还是能把内外的圆都检测出来）

* + - 1. 圆筛选

由于有些瓶盖正面有文字或图案，而前面的圆检测又比较宽松，这可能会导致检测出奇怪的圆。为了筛去这些离谱的圆，我们根据圆心位置来筛选这些圆：过于偏离圆心的圆会被筛去。具体筛选条件为：圆心距离图片中心不得超出图片宽的1/10

1. 运行使用说明
   1. 前端环境
      1. 搭建
         1. 下载node.js 参考网站：http://nodejs.cn/download/
      2. 运行
         1. >> cd forntend
         2. >> npm start
   2. 后端环境
      1. 搭建
         1. 建议使用Anaconda导入python环境

>> conda env create -f environment.yaml

* + - 1. 或搭建python3.6环境，使用的opencv版本不同可能会导致识别出错

>> pip install tornado==4.5.3

>> pip install numpy==1.14.2

>> pip install opencv-python==4.1.2.30

>> pip install pillow==5.0.0

* + 1. 运行
       1. 直接运行即可

>> cd backend

>> python3 main.py

* 1. 其他

项目github地址为<https://github.com/1000happiness/bottleCapIdentification>，在image文件夹中存储了若干样例图片