计算机视觉大作业

小组成员：

陈婧心、荆家振、张源娣、杨宣慧

一、功能简介

1、 检测出输入图片中大小相近的 10 个瓶盖（同色或不同色）的姿态（正、反、立）
2、 在原图中框选出瓶盖，同时可以显示截取出的单个瓶盖的图片
3、 识别瓶盖颜色
4、 排除图片中不是瓶盖的干扰物

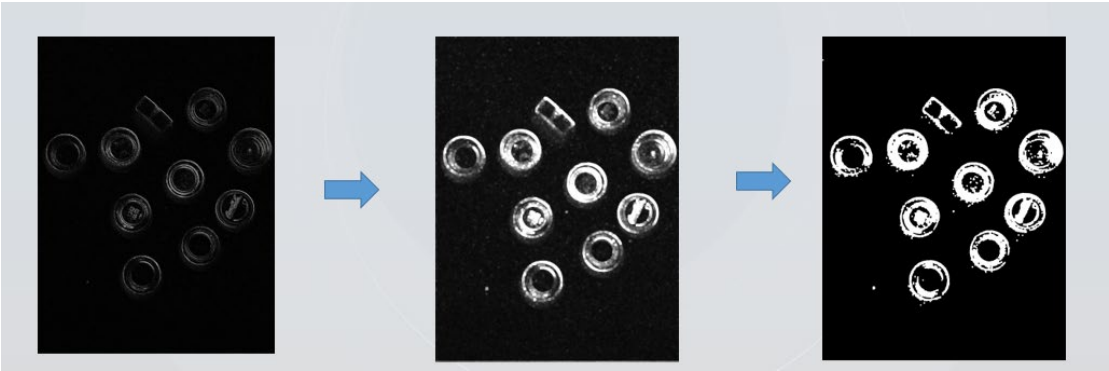
二、实现方法

1、 提取瓶盖

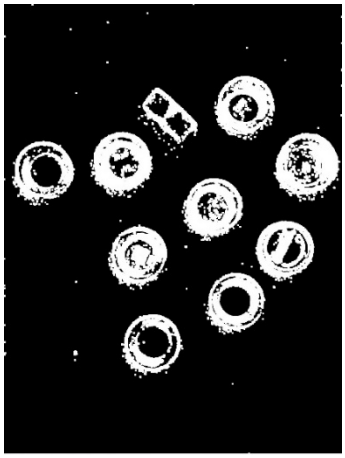
将瓶盖框选出的步骤可细分为以下四个：

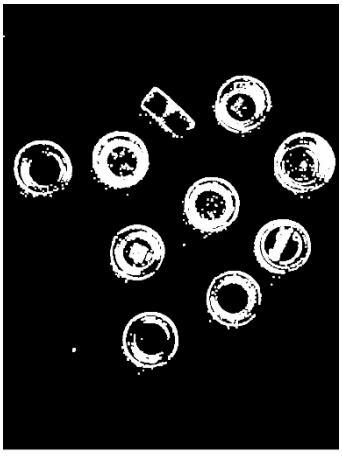
1.1 边缘提取

用 Sobel 算子边缘提取，再进行图像增强和膨胀操作，使得图片中物体外边缘明显且连接。二值化边缘识别完成后的图片。



1.2 降噪



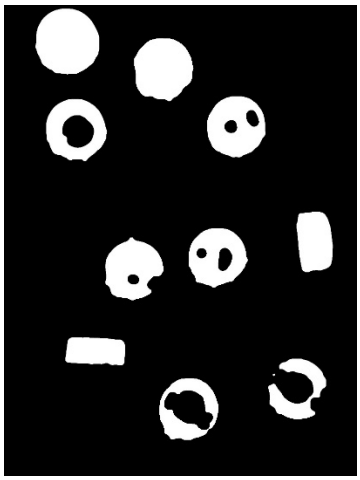


发现仅进行边缘区别后的图片由于纸张带有纹理/杂质，拍照时瓶盖自身有阴影，图片背景及瓶盖周围有多多少少的噪点，为了在之后框选时更加准确，进行了去噪操作。

在进行了开操作（处理小噪点）和模糊化重新二值化（处理较大噪点）后，图像清晰度提升，瓶盖边缘更加明显。

1.3 图形明显化

过滤掉噪声后，用大型 kernel 对图片进行闭操作，尽量将瓶盖区域填白，在下一步框选时获取较大准确率。



1.4 瓶盖分离

因为瓶盖无论平放还是立放，瓶盖展示半径不变，故用最小包围圆框选出瓶盖：

根据实际拍照实验，将半径控制在 240~300 时，瓶盖能够都被框选，且杂物及图片处理误差不会进行干扰。

被框选出的瓶盖再用最小矩形包围：

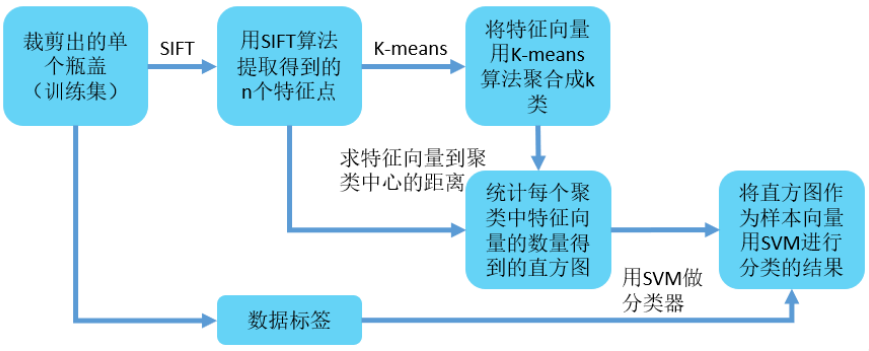
计算出矩形长宽，长宽差过大则瓶盖立放，长宽接近则瓶盖平放。

经过上述操作后，瓶盖提取的正确率大于 90%，对于实验的 10 张图片正确率为 100%

2、 正反识别

方法一：基于 SIFT 特征和 SVM 的图像分类

训练数据流如下：



（1） 将剪裁好的单个瓶盖分类作为训练集（正反分开）

（2） 用 SIFT 算法检测图片最多 n 个特征点，得到最多 n 个 128 维的特征向量，存储特征向量数据之后用来训练

（3） 将同一类型的所有训练图片提取到的特征向量用 K-means 算法进行聚类，聚合成 K 个聚类，得到 K 个聚类的中心点的向量表示，将结果存储

（4） 遍历同类中所有图片，对每张图片的所有特征向量，将其与每个聚类中心求距离，选择最近的中心点。统计出分别属于 K 个聚类的特征向量的数量做直方图。

（5） 统计出的直方图为 K 维向量，作为样本向量，结合种类（正、反）的标签训练 SVM 分类器，将训练结果保存，用来做之后的

方法二：使用 VGG 或 ResNet 模型进行分类。

流程：

（1） 从 pytorch、torchvison 库导入基础模型和优化器。

（2） 人工编写 annotation.txt，对训练/测试集进行数据标注。

（3） 使用自定义的 DataLoader 读入训练/测试数据。

（4） 使用选定的模型和优化器进行训练。

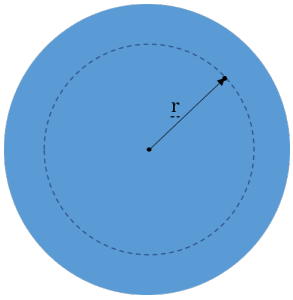
（5） 使用训练后的模型，对不在测试集内图片进行检测测试。

结果：

对 VGG 和 ResNet 模型，我们都进行了 0.001 的 learning rate 的 400 次迭代训练，最终都能在测试集上达成 98%以上的正确率。训练采用了人工分类的 300 张瓶盖图片作为训练集，使用约 100 张人工分类的瓶盖图片作为测试集（约 60 张同时出现在训练集）。此时使用不在测试集内的图片进行测试，偶尔会出现测试不准确的问题，我们认为是数据集过小导致过拟合的结果。相比较而言，VGG 的效果要更好一些，我们认为因为 VGG 模型的参数更少，导致过拟合程度比 ResNet 模型要小一些。

最终，因为数据集太小，人工标注以及训练太过耗时以及在笔记本电脑上运行时速度过慢等原因，我们选择不再使用机器学习方式，转而使用方法一进行检测。

3、 颜色识别



（1） 根据图片大小获得中心点位置，以中心点为圆心，取附近的 rgb 值，根据规定好的图片大小，距离大小 r 取 70 至中心点范围，每次循环递增 1

（2） 对于每一个 r，依据定好的判断规则对附近点的 rgb 值做判断，返回规定好的所属颜色标志值

(3) 对于每一个 r，将附近点的对应颜色标志值求和，取和最大的颜色，如果该颜色占所有颜色比重的 60%以上则返回该颜色值标志值，否则返回-1。

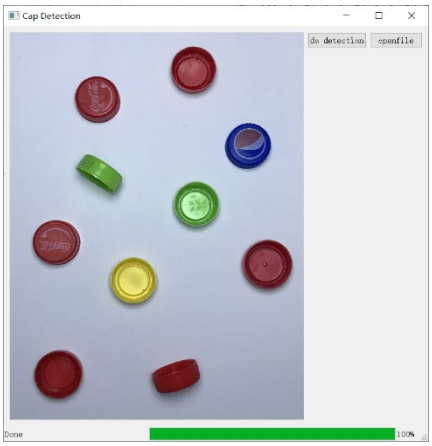
(4) 当 r 递增了 5 次，即取了 5 次附近值，判断是否存在颜色判断失败或者最后两次颜色结果在变化的情况，如果有那么继续进行循环，直至颜色归到一定值，返回判断出来的颜色结果

三、UI 界面

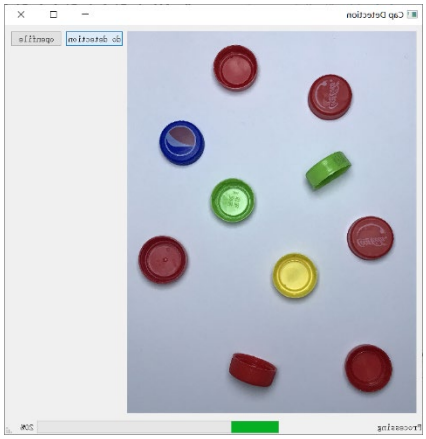
1. 使用指南：

(1) 运行 main.py 之后，弹出主界面：

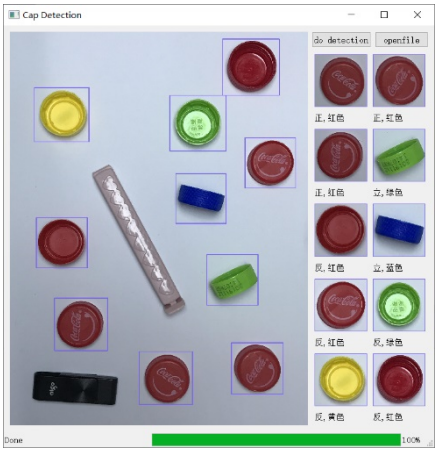
(2) 点击“openfile”按钮，并在弹出的选择框中选择图片文件，打开文件后会将图片显示在左侧：



（3）此时点击 do detection，可以进行瓶盖颜色、状态检测，下方进度条会提示进度：



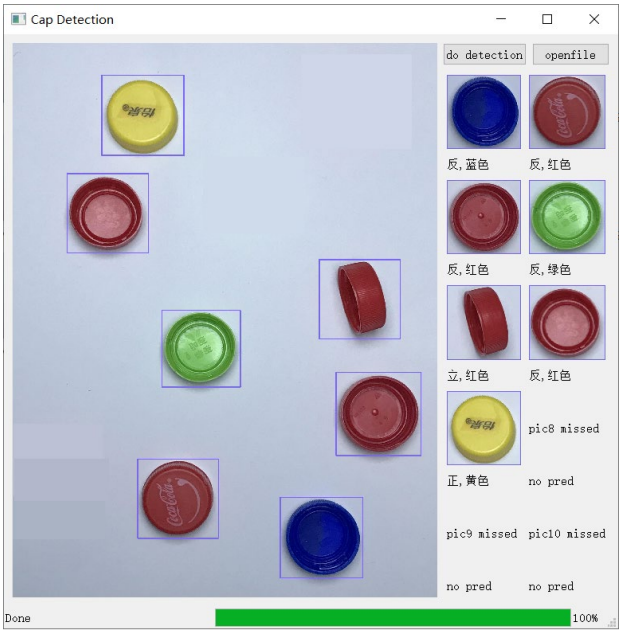
（4）检测完成后，将在右侧显示检测到的瓶盖和检测结果：



2. 错误检测：

（1）不打开图片直接点 do detection 时，可以提示错误：未打开文件

（2） 当图片中瓶盖数量不足 10 个时，将只显示检测到的瓶盖



四、总结

1. 瓶盖提取时黄色的瓶盖灰度值接近背景灰度，故二值化时需实验出合理的分度值，以及在边缘再处理时合理使用开闭操作，选择 kernel 大小，使处理后图片识别出的物体边缘闭合。
2. 识别正反时发现训练集大小、内容、训练时参数选取都对结果有明显影响。所以分类时针对被分类的对象的总体特点进行训练。非常泛化的分类（指对任意的瓶盖进行正反面分类）是比较难做到的。
3. 在做机器学习的过程中，遇到过的问题主要有：
   （1）标注数据集繁琐
   （2）使用 CPU 训练则太慢，使用 GPU 训练则因为显存只有 3GB，必须使用很小的 batch_size（VGG-4, ResNet-2），导致训练中 loss 很不稳定，收敛较慢。
   虽然最终我们放弃了机器学习方式，但是通过这次作业，我们对机器学习有了初步的了解，也认识到了一些机器学习的不足。
4. 在颜色判断的时候由于黄色与红色的 rgb 值非常接近，很容易出现判断错误的情况，所以对于黄色做了特殊判断处理，更加明确的区分红色与黄色，提高了精度。
5. 在瓶盖图片的拍照过程中，刚开始的时候瓶盖的影子很严重，没有控制好光照，于是我们调整了环境光照，使拍出来的阴影很少。由于采用的背景白纸太小，一开始没有控制好距离，导致边缘扭曲比较严重，影响了拍摄效果，在后来的拍摄中注意了这个问题。反光比较严重的部分对于检测结果也有影响，我们后来的拍摄也注意调整了光照，尽量减少了反光。
6. UI 方面，我们先是直接使用 Designer 以及 PyQt Tools 生成的 py 文件作为主文件，导致代码结构较为混乱，后来使用 UI、分类函数以及主界面逻辑分离的方式组织代码，从而使得代码结构清晰了很多。