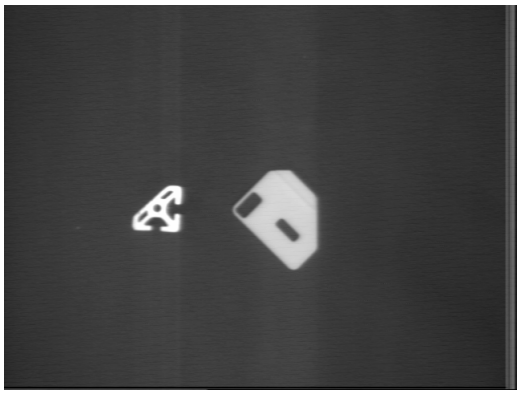
CV_HW1

520030910379 刘宜 rong

1. 编程作业一

接下来我们将对这张图片进行一系列的处理



1.1. 对图像进行二值化

首先我们对原 RGB 图像进行灰度化，图像如下。对图像的二值化，思路比较简单，我们只需要

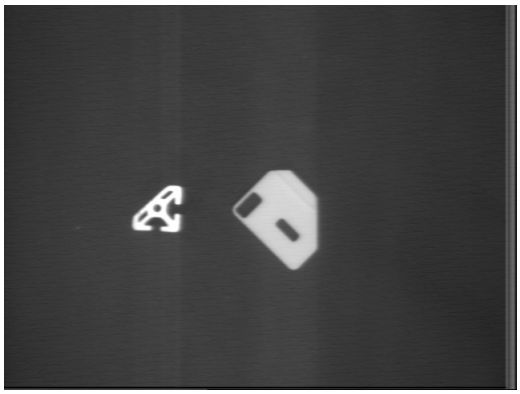


图 1: 灰度图像

遍历整张灰度照片，对每一个像素进行判断，阈值为 128，将超过阈值的点设为白色，低于阈值的点设为黑色即可，处理后的照片如下。

1.2. 对图片进行标签

接下来我们要对图像进行标签，我的思路是采取上课所讲的 Sequential Labeling Algorithm 的

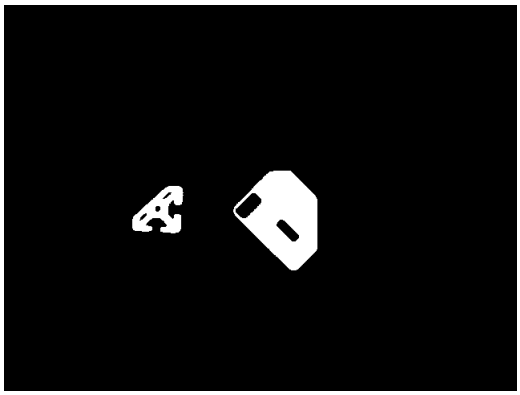


图 2: 二值化图像

方法，首先先对图像进行第一遍的遍历，第一遍遍历时，对每一个非背景点打上标签，这一遍的遍历只去处理标签，也就是说我们会对每一个未被标签的点打上标签而不管这个标签和某些其他标签的关系。这里我的遍历规则是，对于 ABCD 的 2×2 滑动窗，如果 BCD 都为背景，则给 A 打新的标签，如果 BCD 不全为背景点，则 A 的标签和其中任意一个不为背景的点的标签一致。因此在第一遍处理后，我得到一张图像，并且其中每一个非背景点都有标签。

第二遍的遍历我想要处理标签的关系，也就是说有一些不同的标签其实是标记了同一个物体，因此我们要从这一类里找一个标签代表物体的标签，这和等价类的思想非常相似，因此我采取了类似等价类的思想，第二遍遍历时，若 BC 均为非背景点，但是他们的标签不一样，那就把这两个标签归为同一等价类。最终在每一个等价类中取一个根节点作为整个物体的标签即可。在这一层里我嵌套了一些比较低效的循环，我认为这里可以达成一些优化，使效果更好。这里我还遇到了标签超过 255 的情况，我选择先使用 int 类型的数组，最后将其归一化到 0-255 上，增强了算法的鲁棒性。

这个问题我也考虑过用 dfs 的方法进行实现。但是那样不满足 two pass 的要求。

当然以上两种方法都没法解决同学上课提到

的那种特殊情况，但是在本次任务中没有这样的情况，因此没有予以考虑。

最终由于 label 可能比较相近，而且太靠近 0 的 label 可视性不好，因此这里展示的图像是对每个图像的 label 乘以一个系数后的图像。

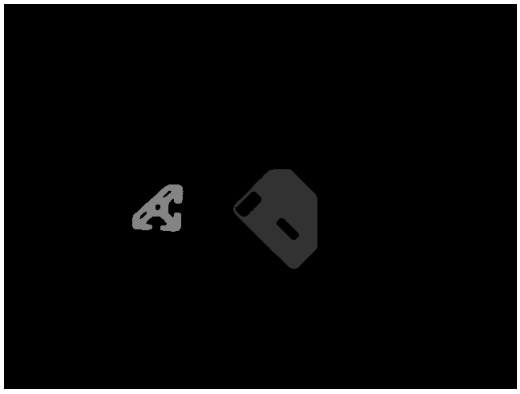
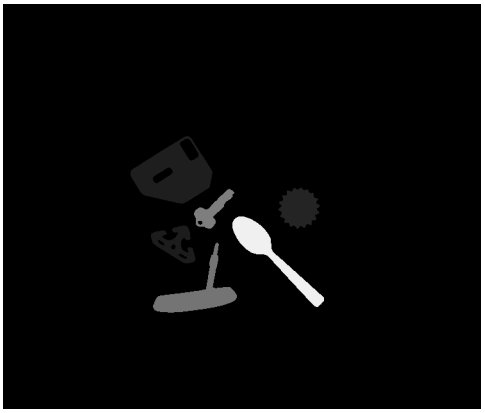


图 3: 标签后图片

其他的图片我也进行了处理，图像如下。



1.3. 计算位置方向和圆度

对这些信息的计算非常的基础，将每一个标签对应的像素点的坐标存入在两个数组里，然后分别计算 x 的平均值，y 的平均值，这即为位置。然后计算方向中的参数 abc，并套用公式求出 θ，然后将 θ 带入求圆度的公式里进行计算即可。(在做到这里时发现 ppt 上计算方向的角度的公式打错了)。

最终数据依次存在一个字典里。下面是一些结果（其中 1 代表这个物体的 label 是 1)。

表 1: *many_objects_2* 的结果

position_x_1	178.09966856060606
position_y_1	188.3515625
orientation_1	-0.6431420831724862
roundness_1	0.007633528961638766

2. Hough 方法找到图像中的圆圈

我们将对这张图片进行处理

2.1. 计算边缘点

采用 sobel 边缘检测的方法，使用的核为课件上的 3×3 核，因此我先对图片进行了一圈 padding，之后计算 x 和 y 方向的梯度，最后平方求和并且归一化到 0-255 的区间即可。

处理后的图像如下。

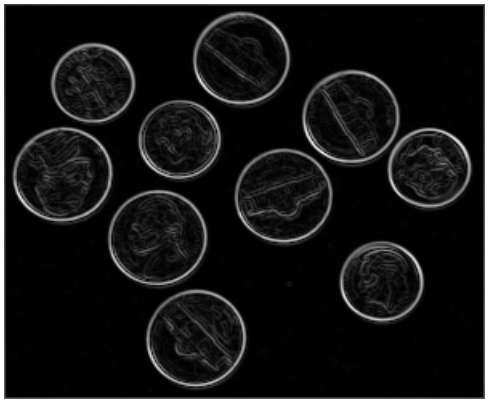


图 4: 边缘点图像

2.2. 阈值处理图像与投票

首先先用阈值处理得到的边缘点图像，这里我选取的阈值为 80，方法和任务一中的的第一个小任务一样。处理后的图像如下。之后我们要对本

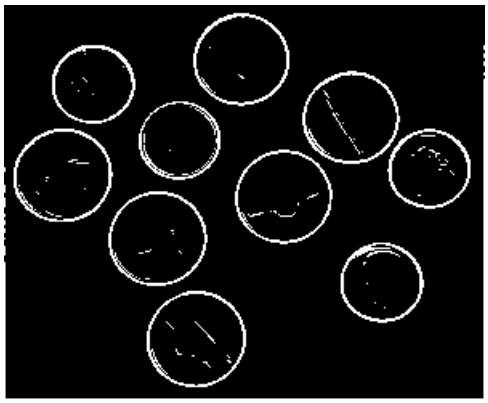


图 5: 阈值处理后图像

图片进行投票来找到可能的圆，我采用了如下的方法。

首先初始化投票数组。之后固定一个 r，遍历整张图片，每一个非背景像素点，会对以自己为圆心，半径为 r 的圆上的所有点进行一次投票，遍历完后，对下一个 r 进行遍历。这样就得到了最后的 accum_array，为一张大的投票表。

这种方法有一个比较不精确的地方，就是在计算圆上的点时，会产生量化的问题，因为我们计算出的点很有可能不为整数，因此我采用了就近取点的方式，这样造成了一些误差，对于改进的方法，我想到了可以设置一个误差区间，在区间范围里的都应该予以收入投票结果中。

2.3. 画出圆

首先设置阈值，多次尝试后我采用了 45 进行处理，因此处理得到一个 list，其中存储了所有满足条件的圆的信息。因为量化问题的存在，这个 list 中的很多圆圆心非常的靠近，这在物理意义上其实就是一个圆，因此我又对这个 list 处理，剔除多余的圆，并取其中最小半径的圆画在图像中。

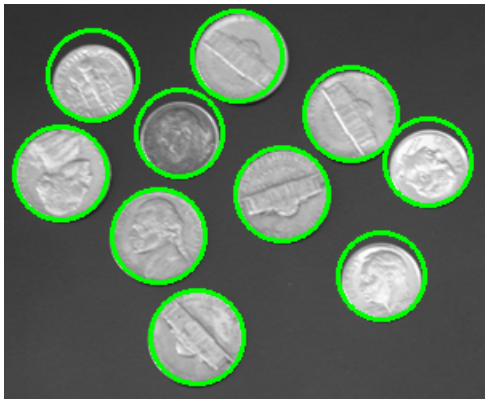


图 6: 框选出圆的图像

最后发现有几个圆没有很好地框住硬币，我调了一些参数但是仍然不能框住，对这个效果的不佳，我认为是因为我设计的方法本身存在的一些缺陷，比如在最右下角的那个硬币，本身的轮廓就不是很圆，而且在上侧边缘上还带有一些干扰点（来自硬币本身的图案），这些干扰点会在投票时投出很多票，使得圆的结果向上偏移，造成了没有框中的结果。此外我个人认为我在计算投票点的时候采取的量化方式也有一些小的问题，这是可以改进的地方。