作业七验报告及测试文档

数据科学与计算机学院 梁育诚 16340133

一、实验内容

第一部分：

输入图像：普通A4打印纸，上面可能有手写笔迹或者打印内容，但是拍照时可能角度不正。（数据采用作业3的数据）。

输出：已矫正好的标准普通A4纸（长宽比为A4纸的比例），并裁掉无用的其他内容，只保留完整A4纸张。

只能采用图像分割的办法获取A4纸的边缘，并对A4纸做矫正，并对比前面用Canny算子获取图像边缘的算法，分析两者的优劣。

第二部分：

用Adaboost实现手写体数字的检测器；然后针对测试数据进行测试（含手写输入图片）

二、实验原理

第一部分：

图像分割是指将数字图像细分为多个图像子区域（亚像素）的过程。图像分割的目的是简化或者改变图像的表达形式，通过将图像分成互不重叠的区域，使得我们能够提取出感兴趣的目标。在整个计算机视觉处理的过程中，图像分割处于中层视觉地位，它是目标表达的基础，使得更高层的图像分析和理解成为可能。分割的方式有多种，这里我用到了Otsu阈值法。

Otsu法按图像的灰度特性将图像分为背景和前景（目标）两部分，背景和目标之间的类间方差越大，说明构成整个图像的两部分的差别越大。因此，使得类间方差最大的分割意味着错分概率最小，也说明该分割越准确。

Otsu法的基本步骤如下：

1. 记t为目标与背景的分割阈值，这个是我们要求的。目标点数占图像比例为w0，平均灰度为u0；背景点数占图像比例为w1，平均灰度为u1。则图像的总平均灰度u = w0\*u0+w1\*u1。

2. 从最小灰度值到最大灰度值遍历t，我们要找到一个t，使得  
最大。此时t就称为该分割的最佳阈值。

3. 应用时为了计算方便，使用的是等价公式。

完成图像分割后，我们就可以进行边缘检测了。图像分割将目标很明显地分割出来，因此检测边缘的时候，只需要对图像求梯度。这里用到了sobel算子对图像求梯度。最后求出的每个像素点的梯度（包括x方向和y方向）。然后在梯度大于一定阈值的地方就是边缘了。在技术上，sobel算子是离散型的差分算子，用于求图像亮度的梯度近似值，因此它是基于一阶导数的算子。它主要的好处就是对像素周围的位置进行了加权计算。

接着是检测A4纸的边缘。这里用了之前实现的霍夫变换，原来不再赘述。

最后是A4纸矫正。A4纸的矫正实际上是一个透视变换，是一块区域到另一块区域的变换。透视变换能够将变形的A4纸恢复到正常形状的A4纸，主要难点在于求解出透视变换的变换矩阵，这是计算部分的内容，在后面介绍。

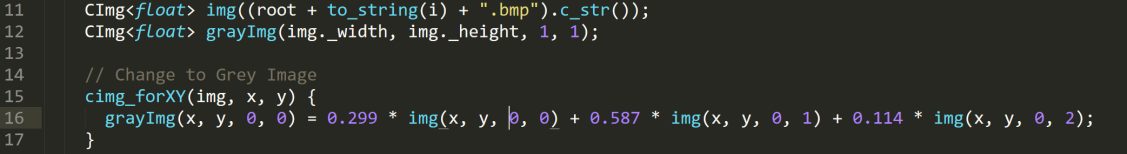
第二部分：

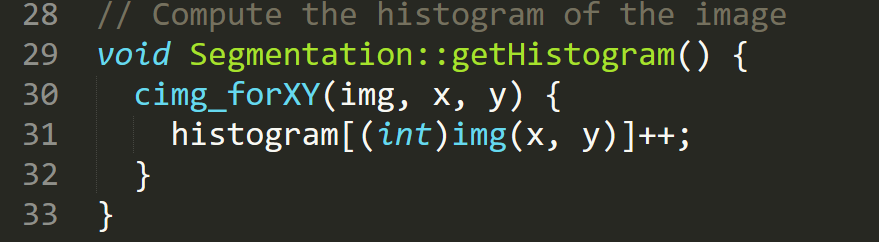
Adaboost算法是一种典型的Boosting算法，其核心思想很简单，针对同一个训练集构建不同的弱分类器，然后按照一定方法来将这些弱分类器集合起来，构成一个强分类器。训练数据有两个，一个是图片数据，一个是与之对应的标签数据。每一轮学习，根据每个样本的分类正确与否，来更新每个样本的权重，正确的样本权重减少，错误的样本权重增加。最后就能够得到一个具有不错识别能力的强分类器了。

三、实验过程

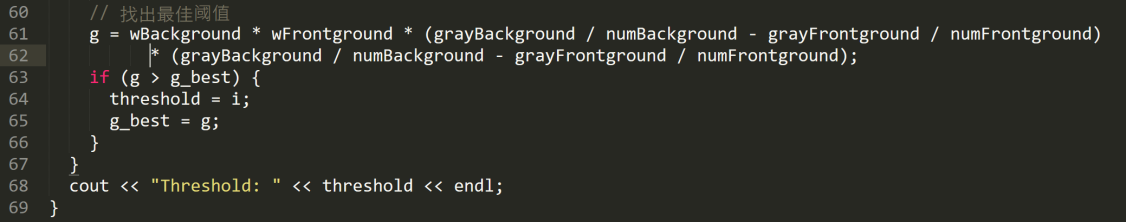
第一部分：

1. 首先实现图像分割。使用Otsu阈值法，需要先将图片进行灰度化处理，然后获取图像的灰度直方图，根据灰度直方图的分布找出最佳分割阈值。

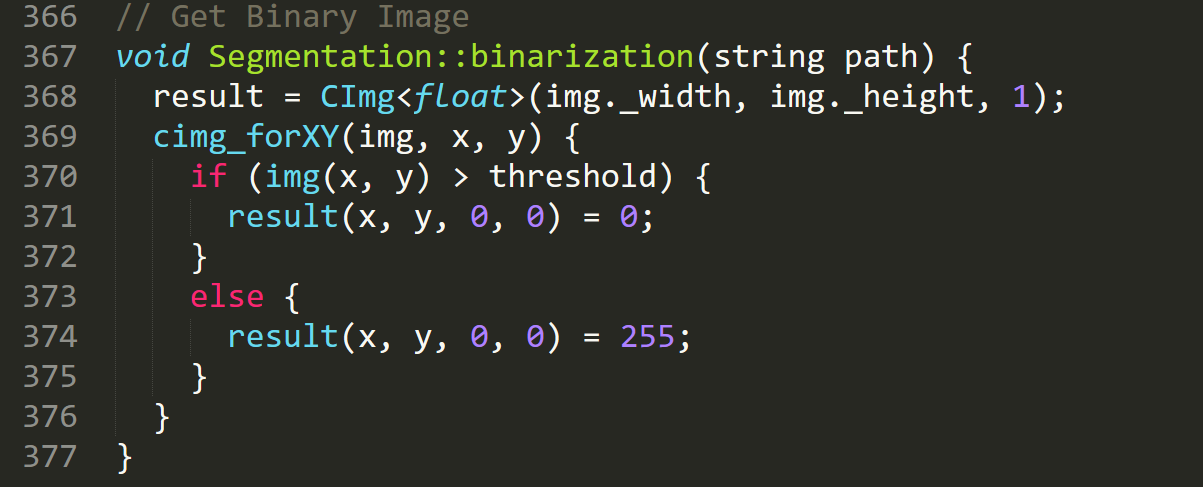




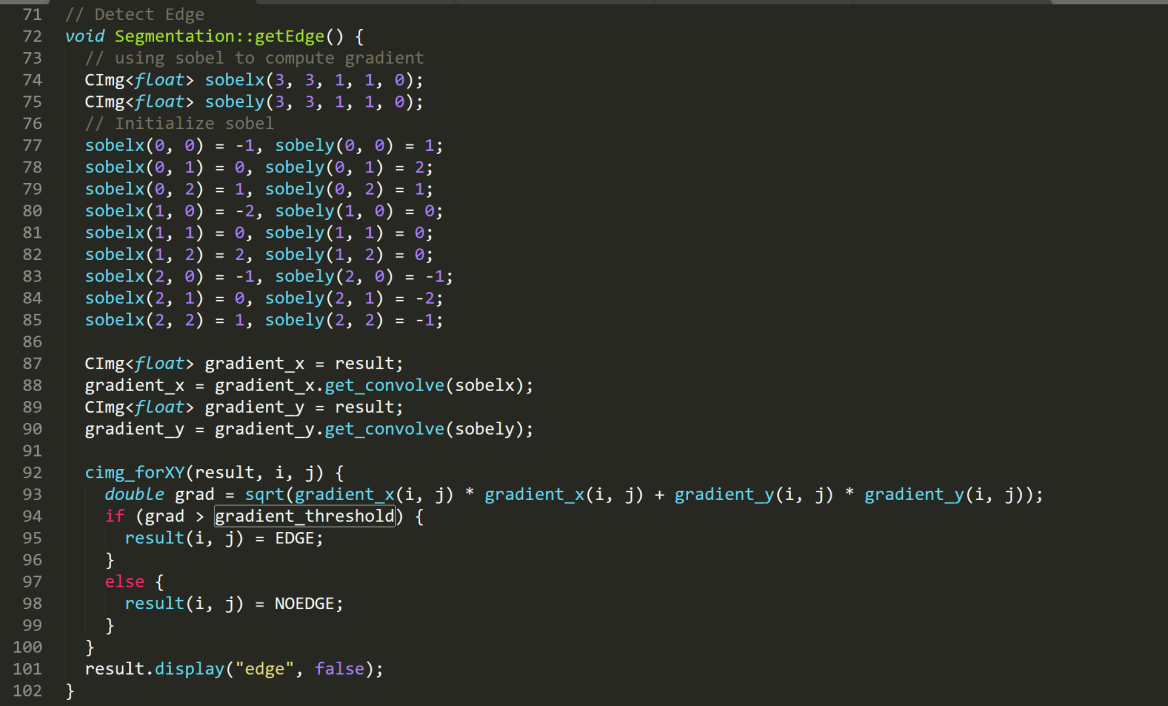




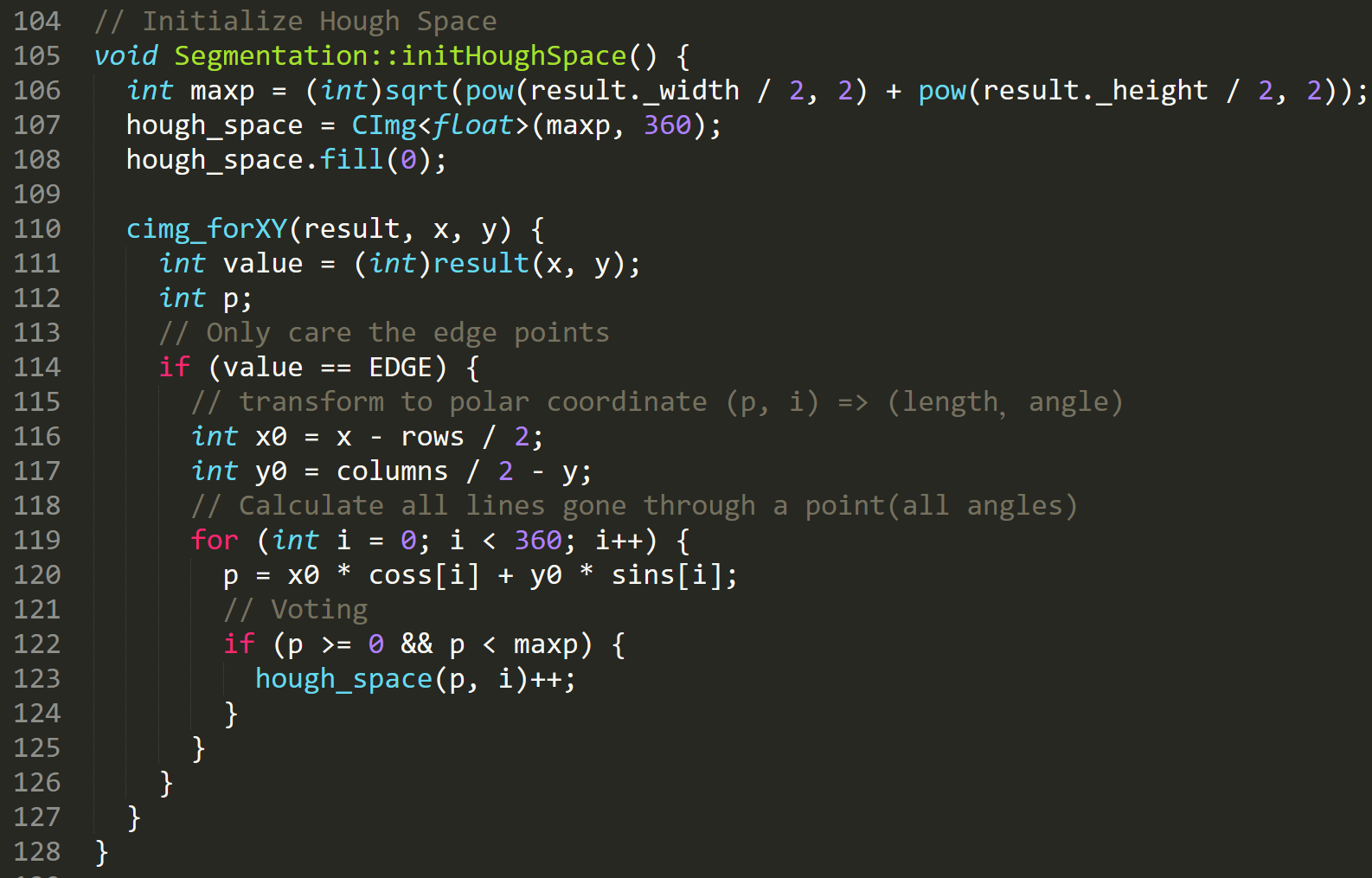
2. 得到最佳分割阈值后，遍历整个图像，将小于阈值的像素设为255（白色），大于阈值的像素设为0（黑色）。



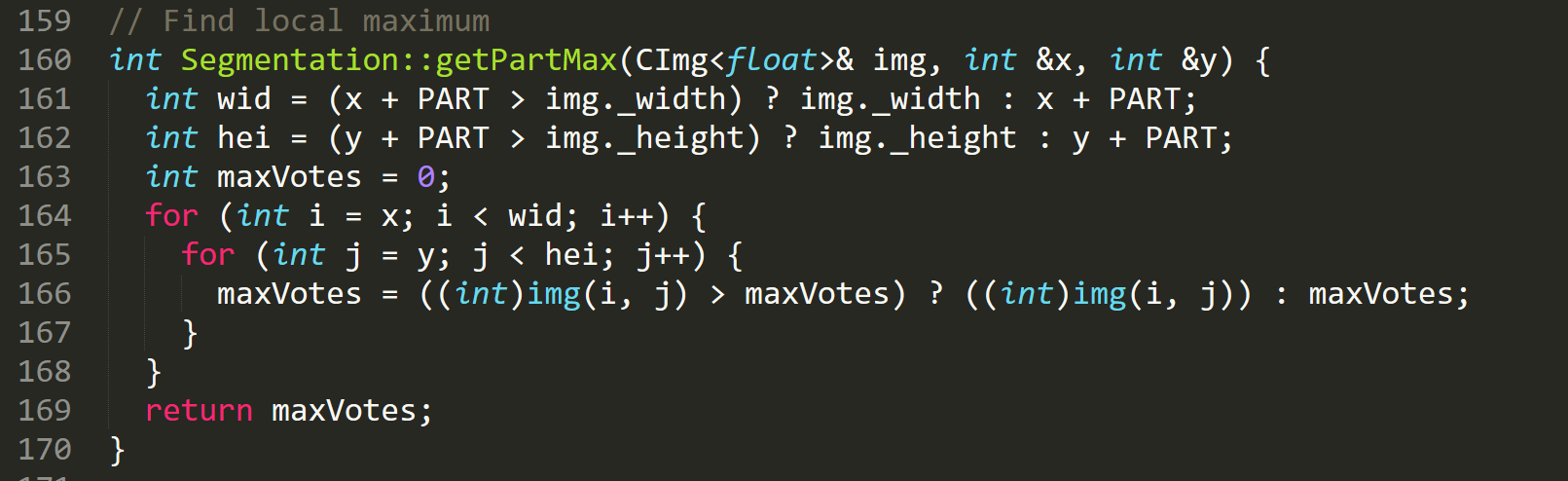
3. 然后对图像计算梯度。利用sobel算子求出该图像在x方向和y方向上的梯度。这里可以用CImg库提供的get\_convolve函数实现（其实就是卷积）。然后对于梯度空间，同样使用一个阈值进行过滤，梯度高于该阈值的点就认为是边缘，梯度低于该阈值的点就认为是背景。，因为边缘的一边是黑色，一边是白色，所以我们认为只有边缘的点梯度才是最大的。如果以0为分界，则会包含过多的噪声，因此这里的阈值手动设为30。

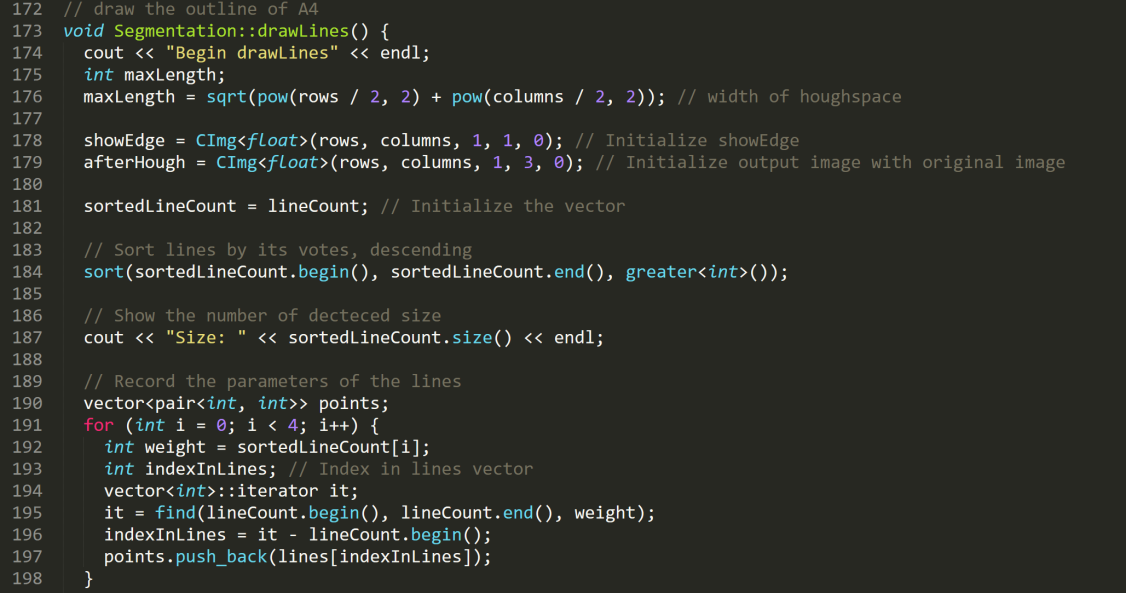


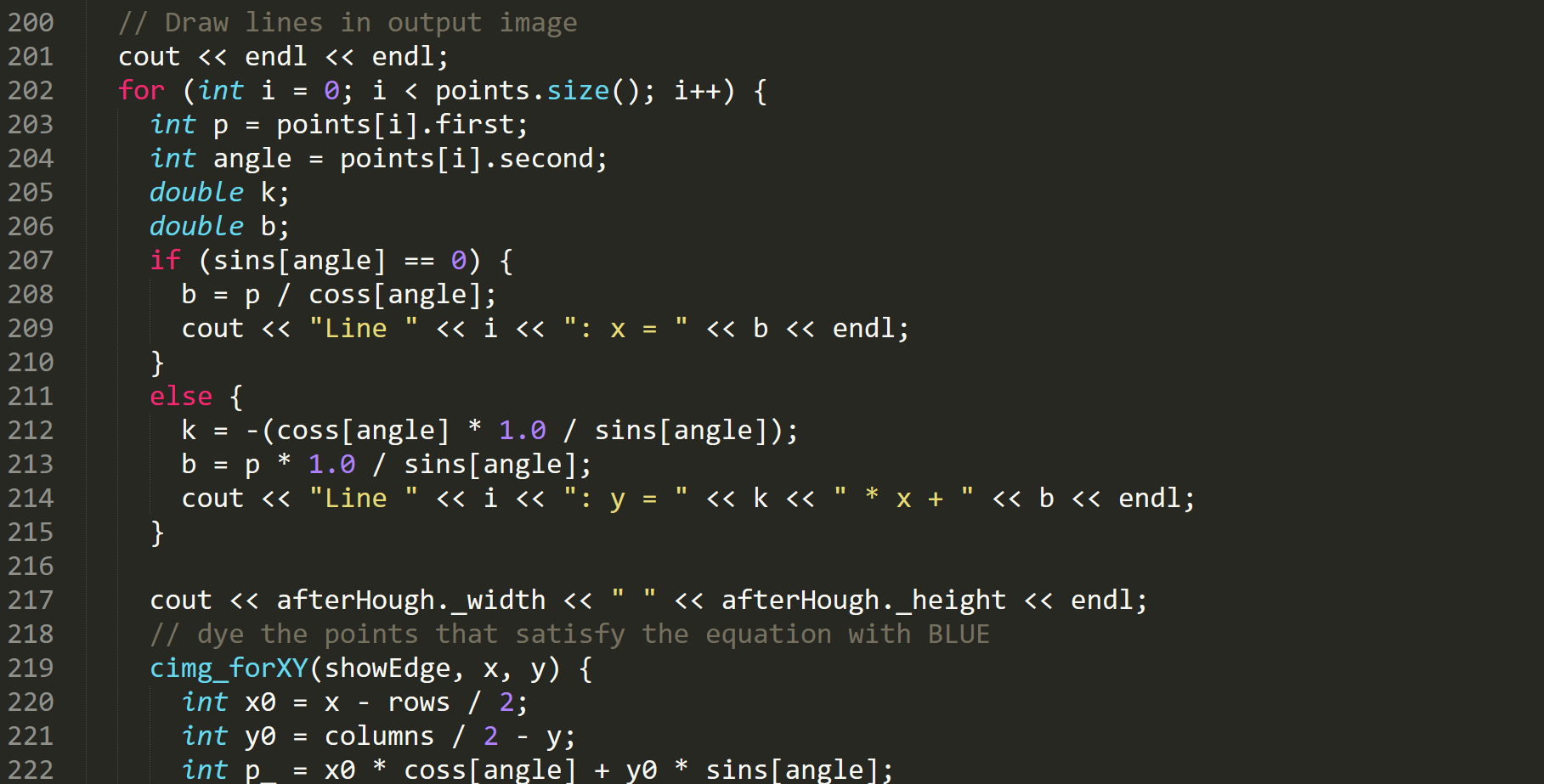
4. 然后就是基于图像分割后的图像进行A4纸检测，使用霍夫变换，代码用回之前作业自己写的，效果不错。修改部分参数，邻域大小设置为60，因为不需要在原图显示检测出来的直线，因此可以删减部分代码。注意画线的时候要重新用另一个CImg，原来的CImg是灰度图，无法画彩色的线，不然会越界报错。

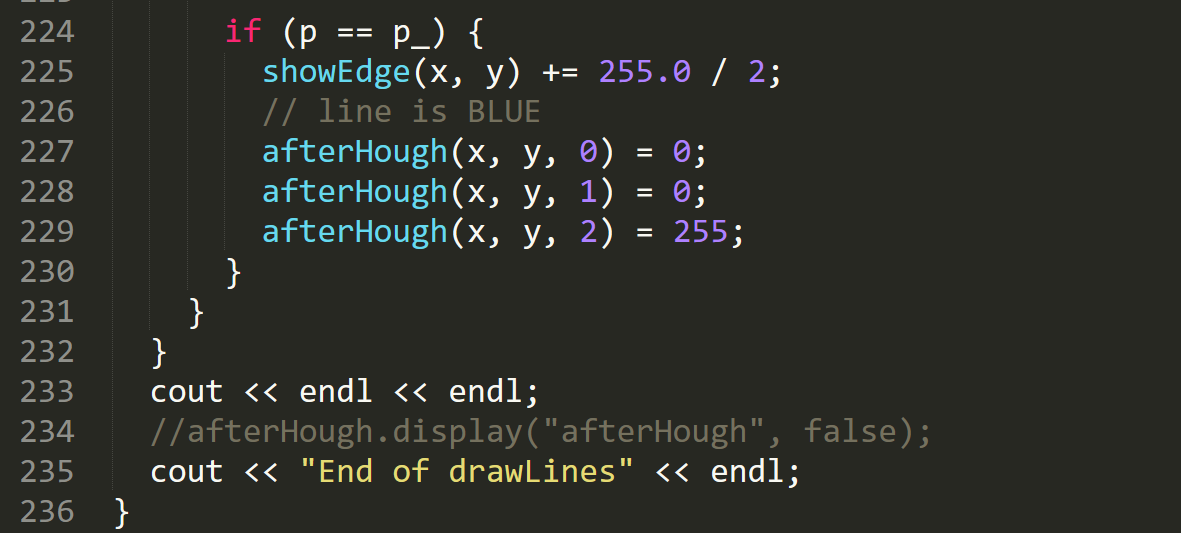




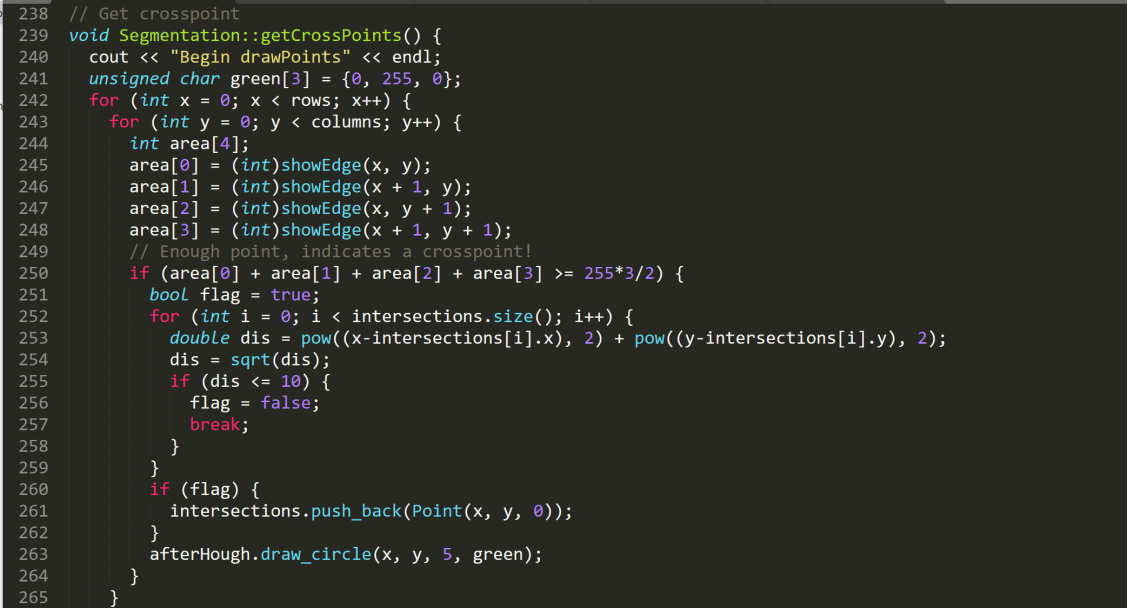






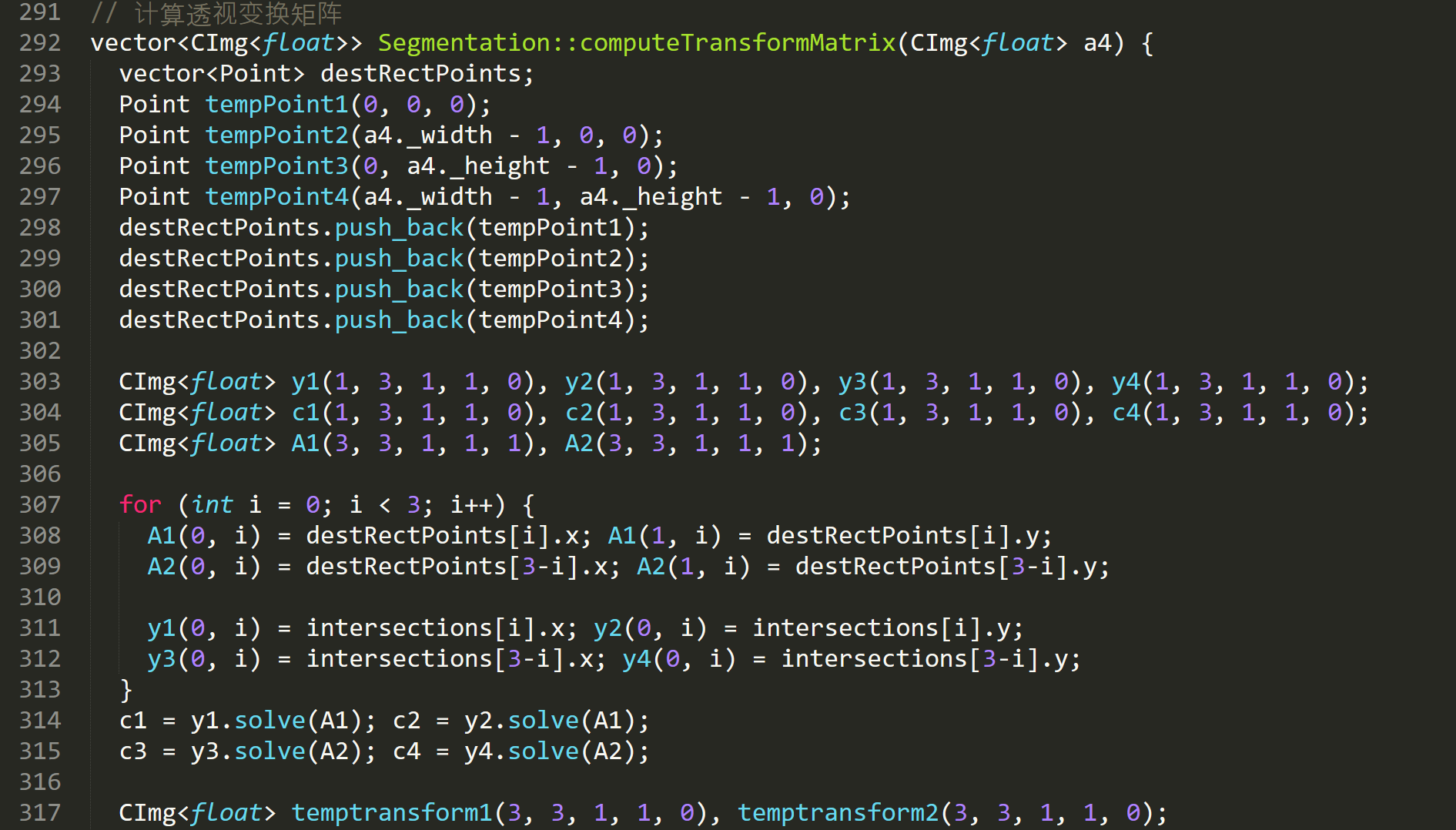


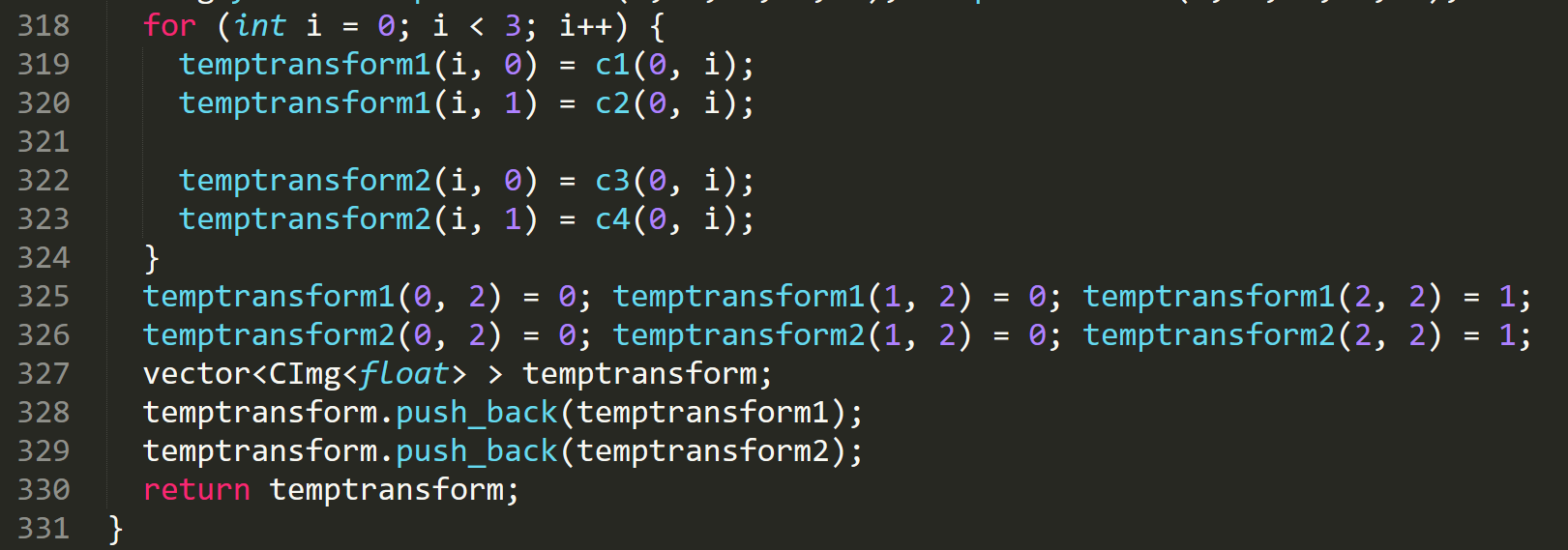
5. 检测出A4纸的四个角点。因为之前用的是邻域的方法，所以在靠近角点位置会有很多个点都满足情况。之前的作业只需要画出效果，因此这样做没有问题，但是这次需要获取出A4纸非常准确的四个点，不能少也不能多。思考了一下之后发现只需要在原来的基础上加一点条件就好了，判断在当前点与集合中的点的距离，如果该距离小于某一个值，那么我就认为他们是聚在一起的，也就不可能是两个不同的角点了，就不添加进去。这个距离我设置了10，足够检测出绝大多数的A4纸了。还有一个问题就是，后面需要用到这四个点进行透视变换，但是这个集合是无序的，而变换的时候需要四个点是按左上->右上->左下->右下顺序排列的。因此在最后还需要对点集合进行排序，先对y排序，然后分为两部分，前两个为一部分，后两个为一部分，每部分再对x进行排序，最后的结果就是有序的。



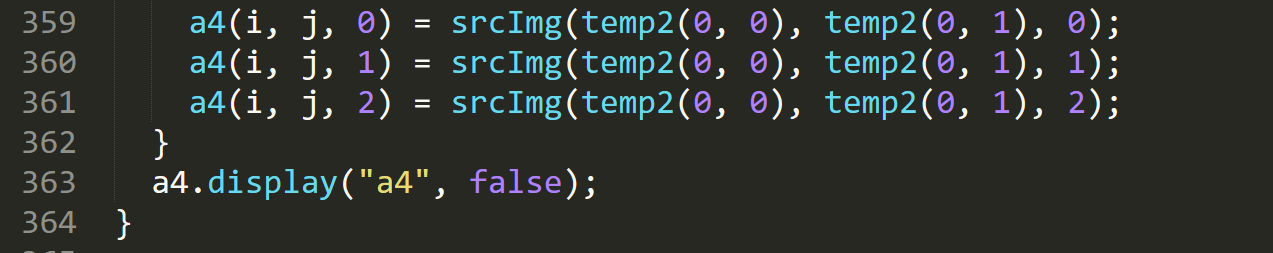


6. 然后根据提取到的A4纸四个角点的坐标，将A4纸作warping，矫正为一张正的A4纸图像。变换的部分参考网上一些快速算法即可，大多都是利用矩阵的性质减少运算量，但其实求解的方程是一样的。









第二部分：

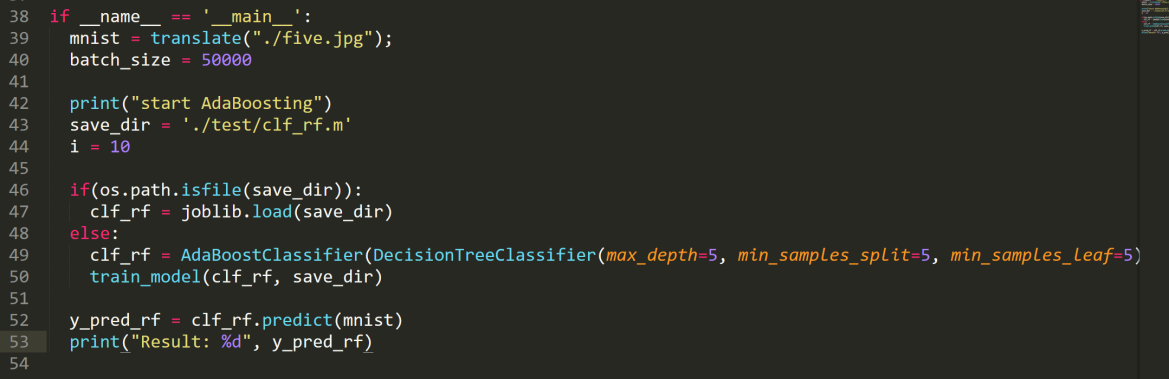
1. 训练模型。Python中有sklearn库提供了AdaBoost算法的分类器，我们只需要确定参数，然后训练出模型就可以了。数据提取自MNIST数据集，这里用了tensorflow来导入数据，它可以自动提取缺失的数据，在解析的时候也比较方便。数据分为训练数据和测试数据，每个数据又分为图片和标签。使用训练数据训练模型，然后用测试数据对模型进行测试，得出一个准确率。因为训练一个模型用时较长，因此在训练后要保存模型，便于后面的测试。参数主要是弱分类器的数量、学习率、分类算法等，这里我参考了网上的一些例子，分类器数量设在200-400，学习率为0.05，分类算法是SAMME.R。





2. 测试手写数据。读入一张手写数字的图片，然后使用训练出来的模型对这个数据进行检测，查看结果是否正确。读入图片需要转换成MNIST数据，这里是上网找了教程做的格式转换。



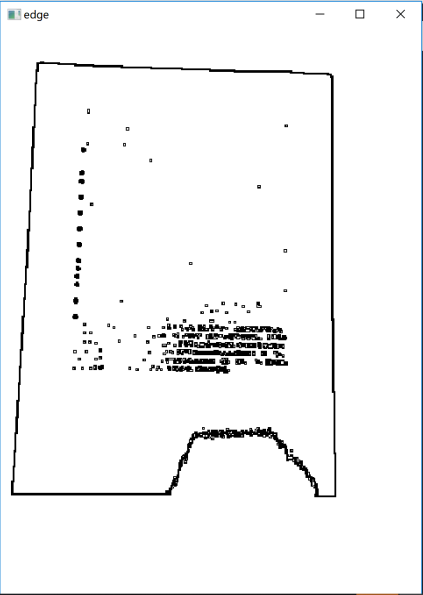
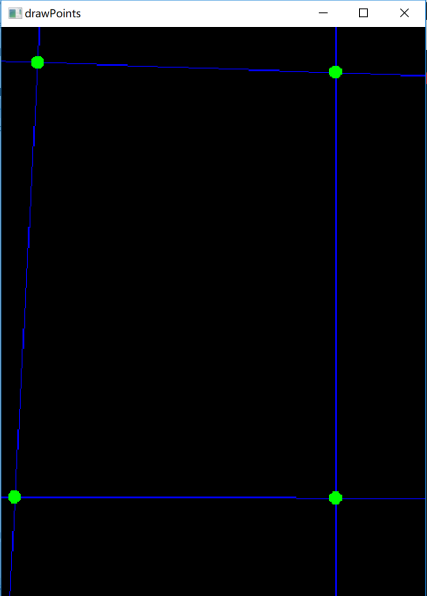


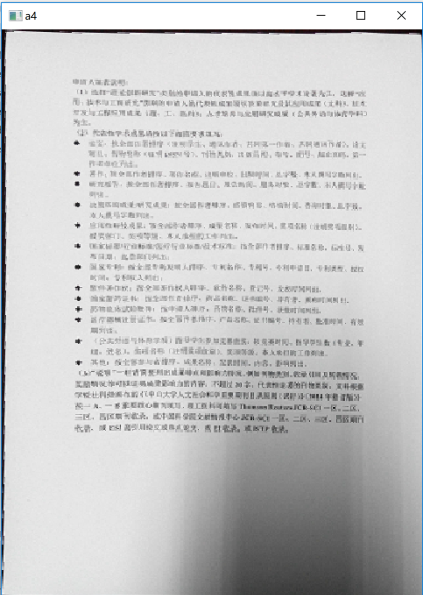
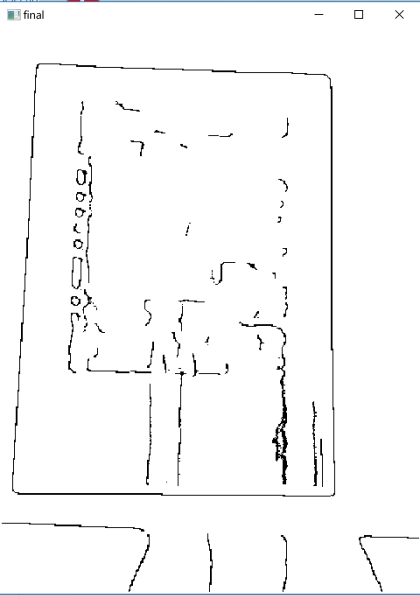
四、测试与分析

第一部分：

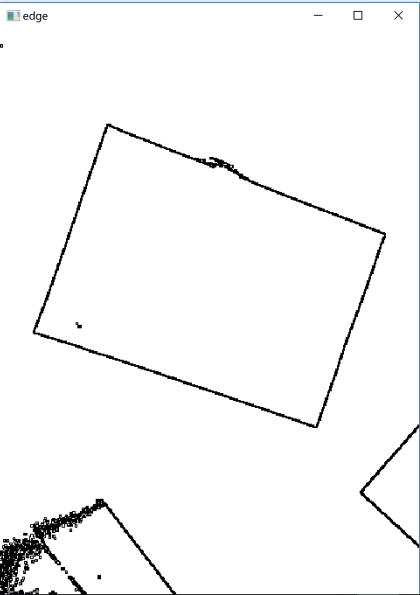
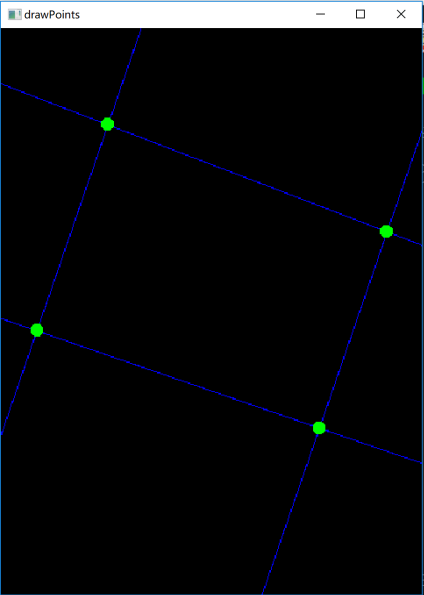
说明：测试数据使用作业三的A4纸数据，每张图片展示图像分割后提取边缘后的图、霍夫变换后检测出的图和最后矫正后的A4纸图。最后再给出一张使用canny算子获取边缘的效果图进行对比。

①

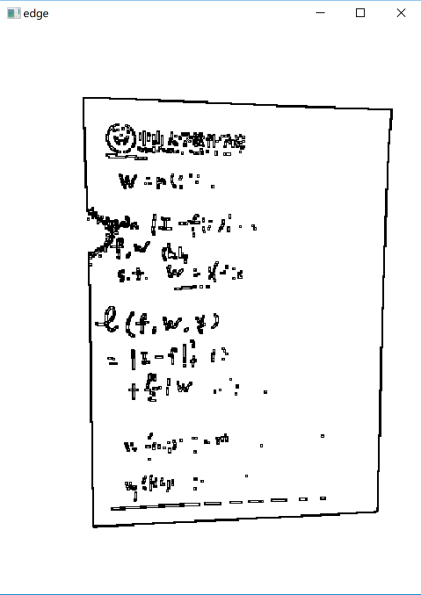
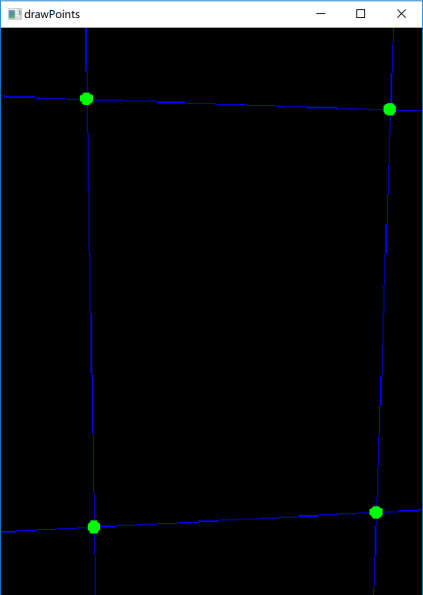
 

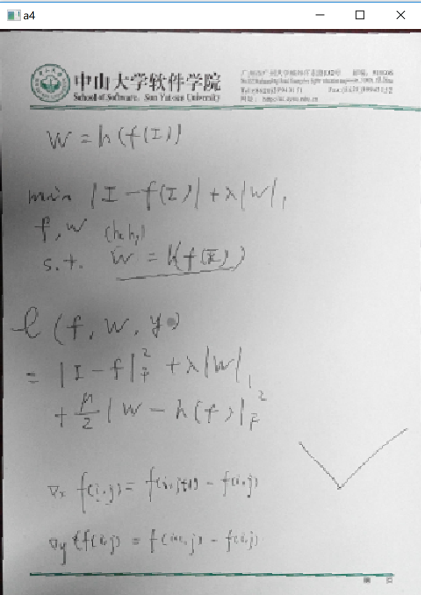
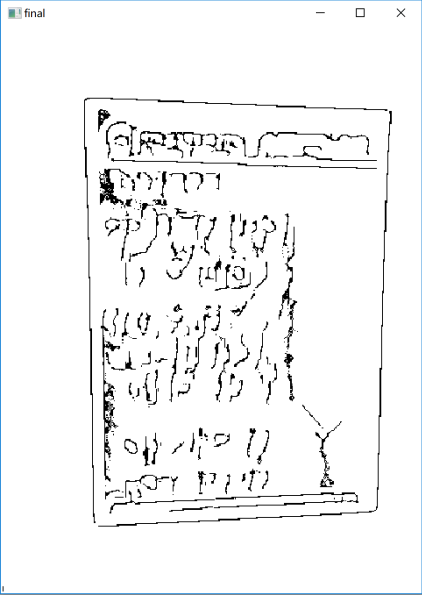
②

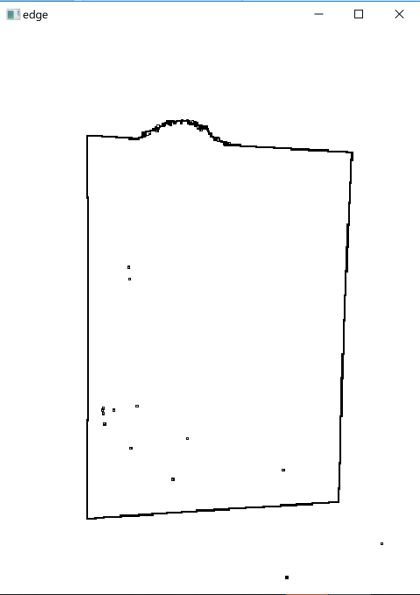
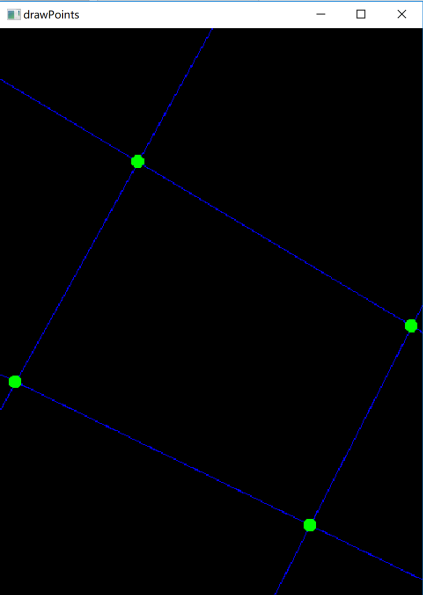
 

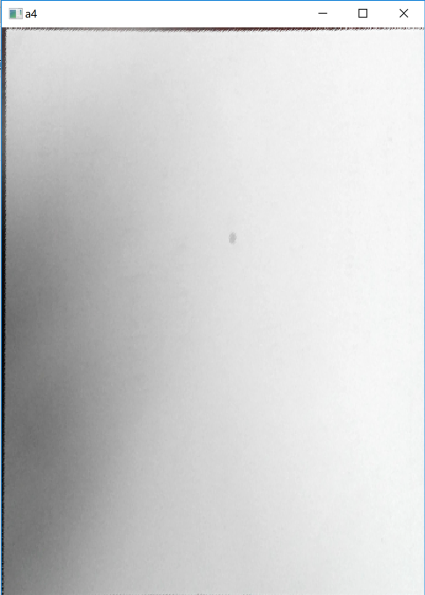
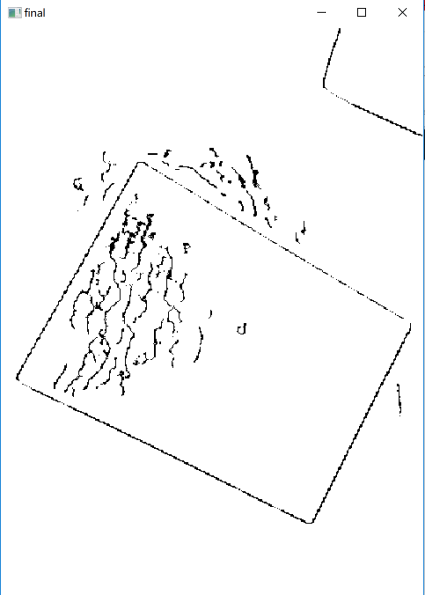
③

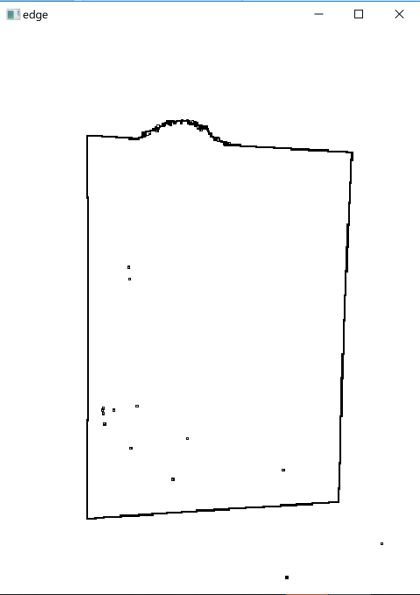
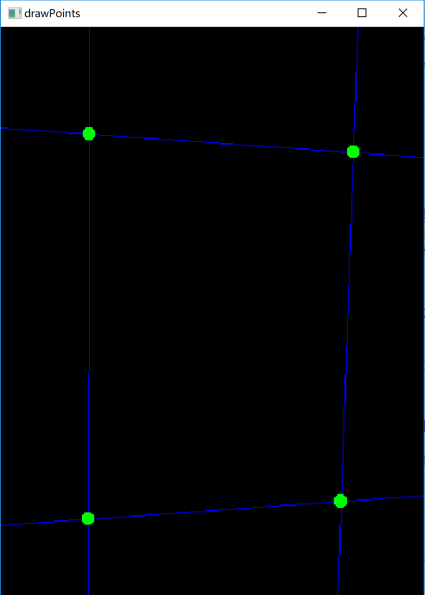
 

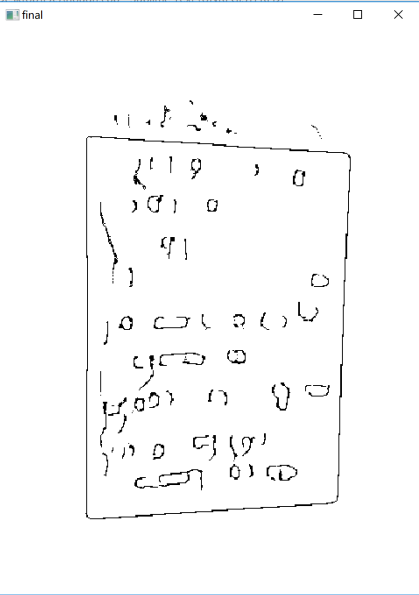
④

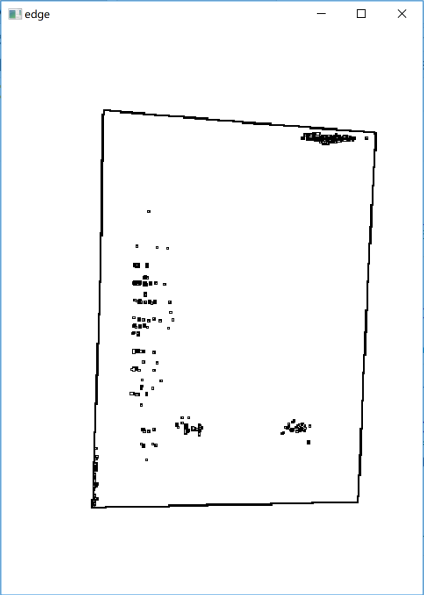
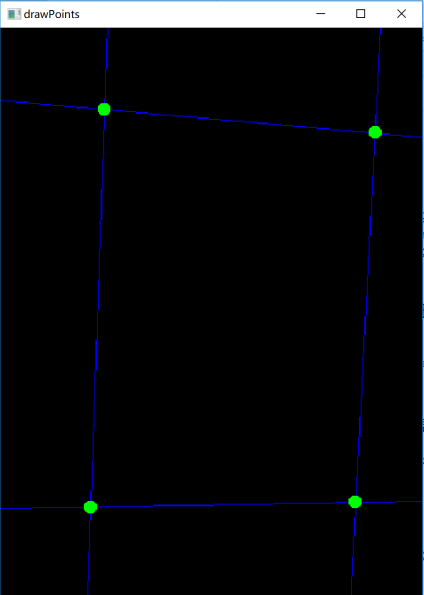
 

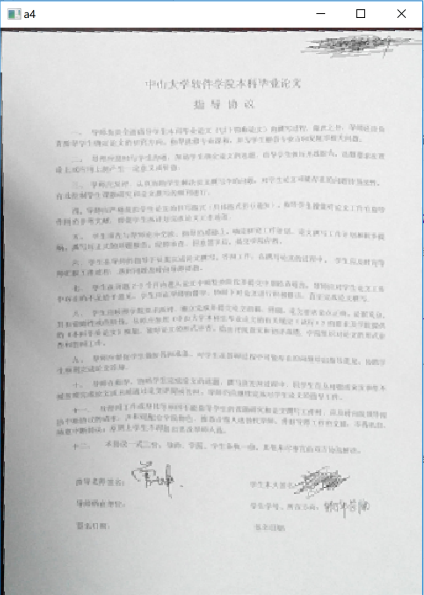
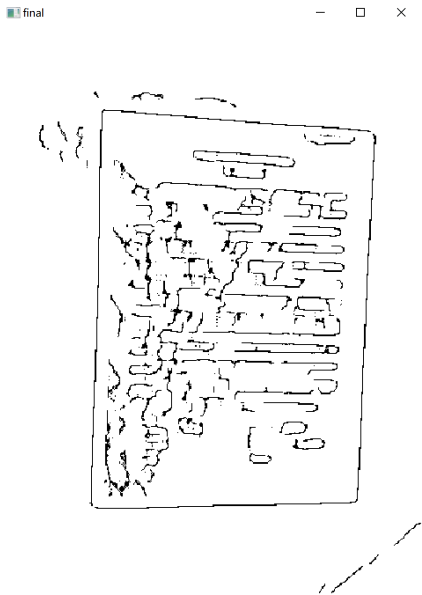
⑤

⑥

分析：从矫正的角度来看，算法是比较成功的，六张图片都能很好地将A4纸矫正输出，都很准确地检测到了A4纸的边缘，在丢弃无用信息的同时，很好地保留了有用信息。

对比使用canny提取边缘和图像分割提取边缘，我认为它们各有优劣。Canny算子对于图像中的细节保留的较多，而图像分割对于目标的细节保留的较少。以本数据集为例，在有光线影响的情况下，canny算子依然能够很好地提取到边缘，但使用图像分割则会出现缺失。之所以最后能够正确检测，是因为霍夫变换很好地投票出四条直线。因此我认为，对于目标与背景易于区分、对细节要求较少的数据，可以采用图像分割，而对于受光照影响较大、边缘细节丰富的数据，应该使用canny算法。

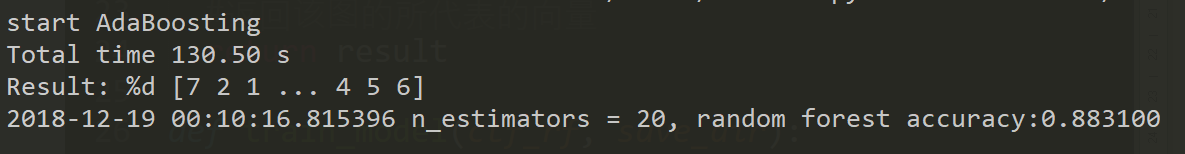
第二部分：

首先是训练模型，使用MNIST的测试数据集测试：

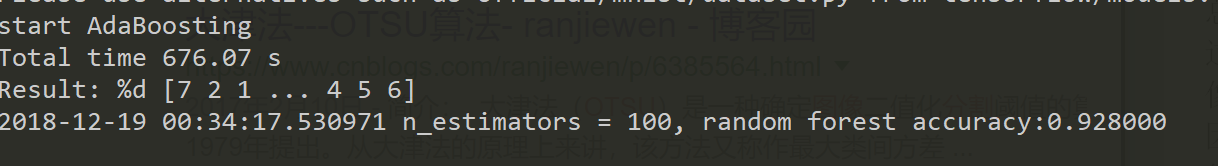
Estimators = 10



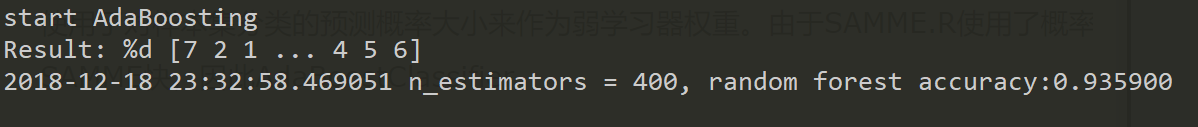
Estimators = 20



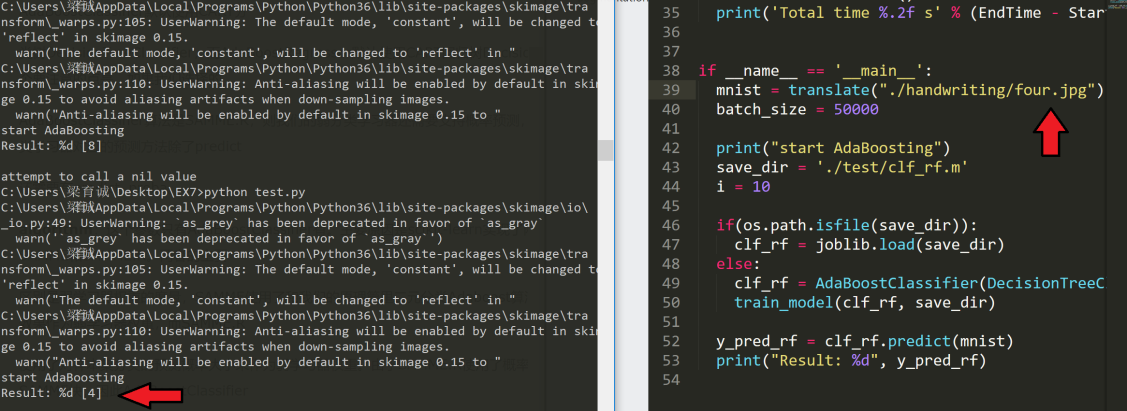
Estimators = 100

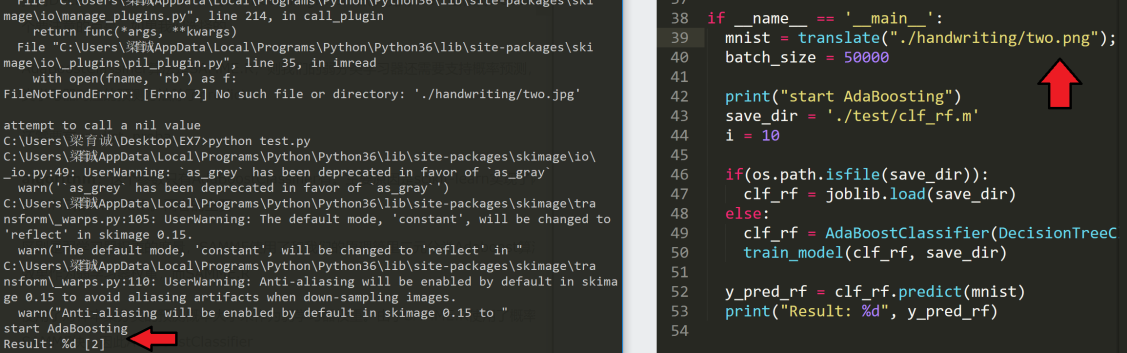


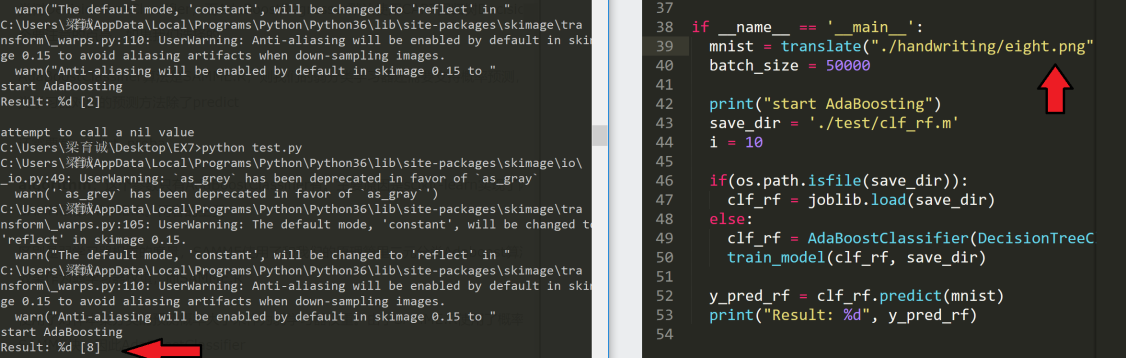
Estimators = 400（因为训练时间过长，之前已经训练好了，这里无训练时间，大概是1个小时）



然后是自己输入手写的数字图片进行识别：







分析：弱分离器数量越多，模型准确率越高，当然耗时也就越长。训练模型后，使用MNIST数据集测试，最好的准确率接近93.6%。然后输入自己手写的数字图片，基本能识别出数字，但是对于部分数字，或者写的不工整的，模型可能会识别错误，但总体来说正确率还是很高的。

五、难点、思考、总结与体会

难点：1. 在做图像分割的时候，计算类间方差的公式经常报错，一直卡在公式那里，于是我想可能是做除法的时候除以0了。输出信息，发现确实是这样，检查算法没有错，最后发现原来是类型错误，除法的时候要使用小数，否则就会截取整数部分。

2. 确定梯度阈值。在计算梯度提取边缘的时候，一开始我判断的条件是以0为界限，不等于0的都认为是EDGE。但是发现这样做会有很多点，噪声很大，因此我觉得要设置一个一定大小的阈值，用于去除噪声。通过输出数据，找到了一个大概的值30，跑了几次效果还不错，就确定用这个了。

3. 霍夫变换。霍夫变换的代码是用回之前自己写的，但是在画线的时候就报错了，因为是在cimg\_forXY中，因此我认为是图像越界问题，但是x、y都没有越界，只可能是图像色道问题。查看变量名后，发现在Segmentation整个类中我们操作的都是main传入的灰度图，因此不能有彩色。新增一个变量用于存储彩色图即可。

4. 透视变换。透视变换本身不难，之前做人脸变形已经研究过。这次难点在于如何拿到四个角点。一开始的问题是检测到过多的角点，这是因为我使用了邻域的方法来判断。解决办法是计算两点之间的距离，过小则认为他们是同一个点，这样实现后角点的数量就是正确的4个，但是顺序不对。我们需要按序排好才能应用透视变换，上网查了一下，其实很简单。先对y排序，然后分成两组（前两个一组，后两个一组），再对每组的x进行排序。

5. Adaboost检测自己手写图片的时候，因为要将图片转化为MNIST类型，所以在网上找了一下代码。检测了几张发现结果全部错误，我认为模型应该是没问题的，可能是测试数据处理的不对。上网看了MNIST解析后，发现测试数据转化的时候不需要取反，因为1是黑，0是白，修改过后就能正确识别大多数的数字了。

总结：这次实验用到了之前学过的知识——霍夫变换、边缘检测等，在写代码的过程中，也发现自己对于计算机视觉的一些概念都比较熟悉了，基本知道一些图像处理的思路与方法，使用边缘检测算子的时候也比较熟练地运用了库的函数。因为霍夫变换是自己写的代码，因此理解起来也没花多少时间，稍微修改一下就能用上，节省了很多时间。在透视变换上还是花了一点时间理解，但是网上找到了不错的资料，因此也比较快就写出代码了。至于AdaBoost算法，因为是第一次接触到了学习算法，在谷歌一搜基本都是python的版本，因此也就用了python来实现。经过这次实验，我大致知道了如何使用数据集去训练一个模型，然后再用这个模型来对测试数据进行检测。我们要在理解算法原理的基础上，对参数进行调试，从而得出一个比较好的模型。这次训练出来的模型准确率还算不错，对于大部分手写数字都能识别，但有时候还是会识别错误。总而言之，这次作业收获很大，既有对以前知识的回顾，也接触到了全新的东西。

六、参考资料

1. 四顶点校正透视变换的线性方程求解：<https://www.cnblogs.com/faith0217/articles/5027490.html>

2. AdaBoost实现MNIST手写体数字识别：<https://blog.csdn.net/Barry_J/article/details/81950336>

3. 手写数字图片预处理：<https://blog.csdn.net/qq_40358998/article/details/79281936>

4. sobel算子：<https://blog.csdn.net/Mahabharata_/article/details/69099136>

5. 图像分割OTSU：<https://blog.csdn.net/liyuanbhu/article/details/49387483>