**网络空间安全学院**

2020春季学期

《数字图像处理》课程设计报告

车牌自动识别算法的设计与实现

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名1： | 林泽枫 | 姓名2： | 林泽杰 |
| 学号1： | 201841404121 | 学号2： | 201841404122 |
|  |  |  |  |
| 日期： | 7.3 |  |  |

目录

[**一、设计任务说明** 1](#_Toc41913821)

[1.1 课程设计任务 1](#_Toc41913822)

[1.2 开发该系统软件环境及使用的技术说明 1](#_Toc41913823)

[1.3 开发计划 1](#_Toc41913824)

[**二、系统设计** 1](#_Toc41913825)

[2.1 概要设计 1](#_Toc41913826)

[2.2 详细设计 1](#_Toc41913827)

[**三、系统实施及结果** 1](#_Toc41913828)

[3.1 系统测试结果 1](#_Toc41913829)

[3.2 算法性能 1](#_Toc41913830)

[3.3 github提交日志 1](#_Toc41913831)

[**四、课程设计总结** 1](#_Toc41913832)

**一、设计任务说明**

1.1 课程设计任务

描述本课程设计的主要任务及要求

利用OpenCV库和其他开源工具，设计并实现车牌自动识别算法，基本功能要求：

1. 对给定的包含有汽车车牌的照片进行处理，利用图像分割算法将目标从背景中分离出来。
2. 对目标图像进行合适的处理，然后利用Tesseract库实现车牌号码的识别，将结果输出。
3. 要求提供比较友好的用户接口，可以对新的图片导入到系统中进行处理，并将结果返回给用户。
4. 要求处理过程的自动化，即输入图像，自动输出车牌信息，无需人去干预。

1.2 开发该系统软件环境及使用的技术说明

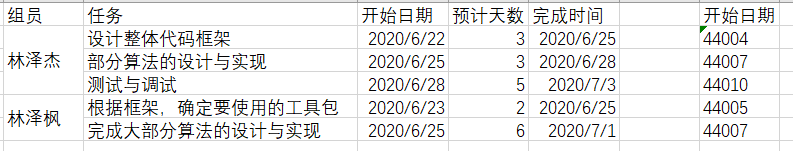
描述系统开发时所使用的开发环境及涉及到的技术

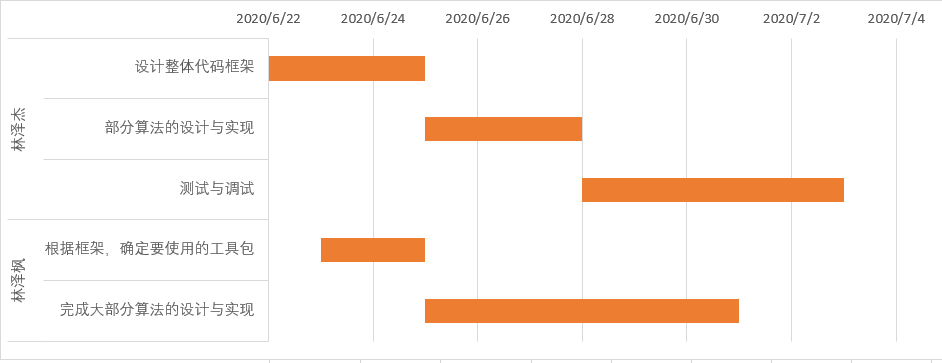
开发环境：使用Python 解释器 3.7.6 的Pycharm 集成开发环境

涉及到的技术：openCV、 numpy 、tkinter 、PIL

1.3 开发计划

描述组员之间的分工情况，画出甘特图





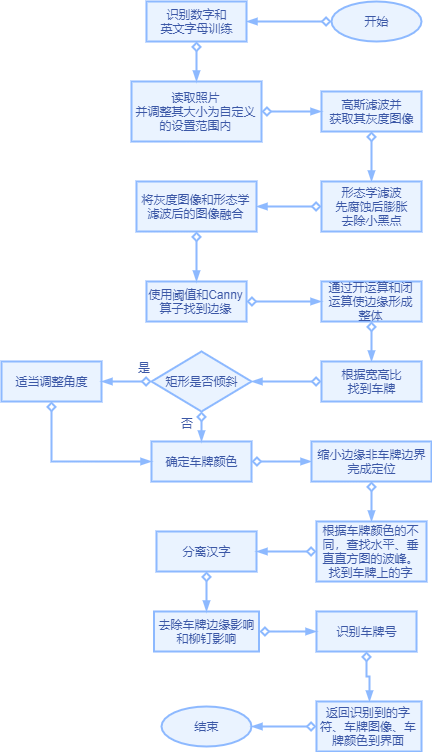
**二、系统设计**

应该包含核心代码，不要将所有代码都贴在这里

2.1 概要设计

系统整个工作功能框图 /流程图及文字描述

通过界面的摄像头捕获图像或上传文件形式，导入程序中。先进行模型训练，可识别数字和英文字母。读取文件，先高斯滤波去除噪点，再进行形态学滤波，滤波结果与原图像的灰度图像融合。找到边缘，使边缘形成一个个整体，根据自定义的车牌纵横比来判断整体中哪个为车牌。通过设定的颜色范围，判断车牌的颜色，查找水平、垂直直方图的波峰来定位字符，分离汉字和去除其他影响，利用训练结果来对这些字符进行识别，结束后将识别结果、车牌图像、车牌颜色返回到界面上。



2.2 详细设计

设计的具体的算法，基本数据结构设计，并发处理，相关代码

先使用图像边缘和车牌颜色定位车牌，再识别字符。车牌定位在predict方法中，车牌字符识别也在predict方法中，车牌字符识别使用的算法是opencv的SVM， opencv的SVM使用代码来自于opencv附带的sample，StatModel类和SVM类都是sample中的代码。

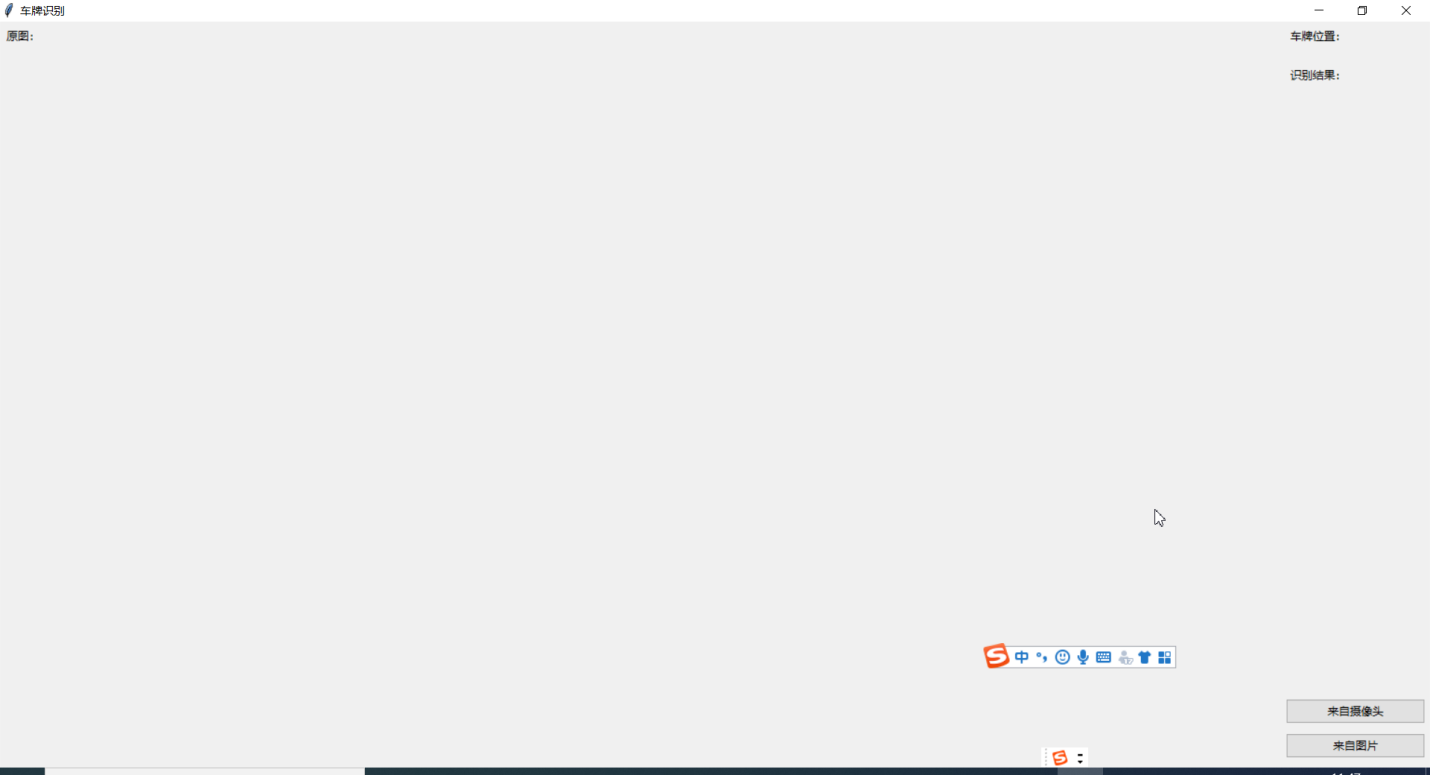
def predict(self, car\_pic):  
 if type(car\_pic) == type(""):  
 img = imreadex(car\_pic)  
 else:  
 img = car\_pic  
 pic\_hight, pic\_width = img.shape[:2]  
  
 if pic\_width > MAX\_WIDTH:  
 resize\_rate = MAX\_WIDTH / pic\_width  
 img = cv2.resize(img, (MAX\_WIDTH, int(pic\_hight\*resize\_rate)), interpolation=cv2.INTER\_AREA)  
   
 blur = self.cfg["blur"]  
 #高斯去噪  
 if blur > 0:  
 img = cv2.GaussianBlur(img, (blur, blur), 0)#图片分辨率调整  
 oldimg = img  
 img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 #equ = cv2.equalizeHist(img)  
 #img = np.hstack((img, equ))  
 #去掉图像中不会是车牌的区域  
 kernel = np.ones((20, 20), np.uint8)  
 img\_opening = cv2.morphologyEx(img, cv2.MORPH\_OPEN, kernel)  
 img\_opening = cv2.addWeighted(img, 1, img\_opening, -1, 0);  
  
 #找到图像边缘  
 ret, img\_thresh = cv2.threshold(img\_opening, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
 img\_edge = cv2.Canny(img\_thresh, 100, 200)  
 #使用开运算和闭运算让图像边缘成为一个整体  
 kernel = np.ones((self.cfg["morphologyr"], self.cfg["morphologyc"]), np.uint8)  
 img\_edge1 = cv2.morphologyEx(img\_edge, cv2.MORPH\_CLOSE, kernel)  
 img\_edge2 = cv2.morphologyEx(img\_edge1, cv2.MORPH\_OPEN, kernel)  
  
 #查找图像边缘整体形成的矩形区域，可能有很多，车牌就在其中一个矩形区域中  
 try:  
 contours, hierarchy = cv2.findContours(img\_edge2, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 except ValueError:  
 image, contours, hierarchy = cv2.findContours(img\_edge2, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 contours = [cnt for cnt in contours if cv2.contourArea(cnt) > Min\_Area]  
 print('len(contours)', len(contours))  
 #一一排除不是车牌的矩形区域  
 car\_contours = []  
 for cnt in contours:  
 rect = cv2.minAreaRect(cnt)  
 area\_width, area\_height = rect[1]  
 if area\_width < area\_height:  
 area\_width, area\_height = area\_height, area\_width  
 wh\_ratio = area\_width / area\_height  
 #print(wh\_ratio)  
 #要求矩形区域长宽比在2到5.5之间，2到5.5是车牌的长宽比，其余的矩形排除  
 if wh\_ratio > 2 and wh\_ratio < 5.5:  
 car\_contours.append(rect)  
 box = cv2.boxPoints(rect)  
 box = np.int0(box)  
 #oldimg = cv2.drawContours(oldimg, [box], 0, (0, 0, 255), 2)  
 #cv2.imshow("edge4", oldimg)  
 #print(rect)  
  
 print(len(car\_contours))  
  
 print("精确定位")  
 card\_imgs = []  
 #矩形区域可能是倾斜的矩形，需要矫正，以便使用颜色定位  
 for rect in car\_contours:  
 if rect[2] > -1 and rect[2] < 1:#创造角度，使得左、高、右、低拿到正确的值  
 angle = 1  
 else:  
 angle = rect[2]  
 rect = (rect[0], (rect[1][0]+5, rect[1][1]+5), angle)#扩大范围，避免车牌边缘被排除  
  
 box = cv2.boxPoints(rect)  
 heigth\_point = right\_point = [0, 0]  
 left\_point = low\_point = [pic\_width, pic\_hight]  
 for point in box:  
 if left\_point[0] > point[0]:  
 left\_point = point  
 if low\_point[1] > point[1]:  
 low\_point = point  
 if heigth\_point[1] < point[1]:  
 heigth\_point = point  
 if right\_point[0] < point[0]:  
 right\_point = point  
  
 if left\_point[1] <= right\_point[1]:#正角度  
 new\_right\_point = [right\_point[0], heigth\_point[1]]  
 pts2 = np.float32([left\_point, heigth\_point, new\_right\_point])#字符只是高度需要改变  
 pts1 = np.float32([left\_point, heigth\_point, right\_point])  
 M = cv2.getAffineTransform(pts1, pts2)  
 dst = cv2.warpAffine(oldimg, M, (pic\_width, pic\_hight))  
 point\_limit(new\_right\_point)  
 point\_limit(heigth\_point)  
 point\_limit(left\_point)  
 card\_img = dst[int(left\_point[1]):int(heigth\_point[1]), int(left\_point[0]):int(new\_right\_point[0])]  
 card\_imgs.append(card\_img)  
 #cv2.imshow("card", card\_img)  
 #cv2.waitKey(0)  
 elif left\_point[1] > right\_point[1]:#负角度  
   
 new\_left\_point = [left\_point[0], heigth\_point[1]]  
 pts2 = np.float32([new\_left\_point, heigth\_point, right\_point])#字符只是高度需要改变  
 pts1 = np.float32([left\_point, heigth\_point, right\_point])  
 M = cv2.getAffineTransform(pts1, pts2)  
 dst = cv2.warpAffine(oldimg, M, (pic\_width, pic\_hight))  
 point\_limit(right\_point)  
 point\_limit(heigth\_point)  
 point\_limit(new\_left\_point)  
 card\_img = dst[int(right\_point[1]):int(heigth\_point[1]), int(new\_left\_point[0]):int(right\_point[0])]  
 card\_imgs.append(card\_img)  
 #cv2.imshow("card", card\_img)  
 #cv2.waitKey(0)  
 #开始使用颜色定位，排除不是车牌的矩形，目前只识别蓝、绿、黄车牌  
 colors = []  
 for card\_index,card\_img in enumerate(card\_imgs):  
 green = yello = blue = black = white = 0  
 card\_img\_hsv = cv2.cvtColor(card\_img, cv2.COLOR\_BGR2HSV)  
 #有转换失败的可能，原因来自于上面矫正矩形出错  
 if card\_img\_hsv is None:  
 continue  
 row\_num, col\_num= card\_img\_hsv.shape[:2]  
 card\_img\_count = row\_num \* col\_num  
  
 for i in range(row\_num):  
 for j in range(col\_num):  
 H = card\_img\_hsv.item(i, j, 0)  
 S = card\_img\_hsv.item(i, j, 1)  
 V = card\_img\_hsv.item(i, j, 2)  
 if 11 < H <= 34 and S > 34:#图片分辨率调整  
 yello += 1  
 elif 35 < H <= 99 and S > 34:#图片分辨率调整  
 green += 1  
 elif 99 < H <= 124 and S > 34:#图片分辨率调整  
 blue += 1  
   
 if 0 < H <180 and 0 < S < 255 and 0 < V < 46:  
 black += 1  
 elif 0 < H <180 and 0 < S < 43 and 221 < V < 225:  
 white += 1  
 color = "no"  
  
 limit1 = limit2 = 0  
 if yello\*2 >= card\_img\_count:  
 color = "yello"  
 limit1 = 11  
 limit2 = 34#有的图片有色偏偏绿  
 elif green\*2 >= card\_img\_count:  
 color = "green"  
 limit1 = 35  
 limit2 = 99  
 elif blue\*2 >= card\_img\_count:  
 color = "blue"  
 limit1 = 100  
 limit2 = 124#有的图片有色偏偏紫  
 elif black + white >= card\_img\_count\*0.7:#TODO  
 color = "bw"  
 print(color)  
 colors.append(color)  
 print(blue, green, yello, black, white, card\_img\_count)  
 #cv2.imshow("color", card\_img)  
 #cv2.waitKey(0)  
 if limit1 == 0:  
 continue  
 #以上为确定车牌颜色  
 #以下为根据车牌颜色再定位，缩小边缘非车牌边界  
 xl, xr, yh, yl = self.accurate\_place(card\_img\_hsv, limit1, limit2, color)  
 if yl == yh and xl == xr:  
 continue  
 need\_accurate = False  
 if yl >= yh:  
 yl = 0  
 yh = row\_num  
 need\_accurate = True  
 if xl >= xr:  
 xl = 0  
 xr = col\_num  
 need\_accurate = True  
 card\_imgs[card\_index] = card\_img[yl:yh, xl:xr] if color != "green" or yl < (yh-yl)//4 else card\_img[yl-(yh-yl)//4:yh, xl:xr]  
 if need\_accurate:#可能x或y方向未缩小，需要再试一次  
 card\_img = card\_imgs[card\_index]  
 card\_img\_hsv = cv2.cvtColor(card\_img, cv2.COLOR\_BGR2HSV)  
 xl, xr, yh, yl = self.accurate\_place(card\_img\_hsv, limit1, limit2, color)  
 if yl == yh and xl == xr:  
 continue  
 if yl >= yh:  
 yl = 0  
 yh = row\_num  
 if xl >= xr:  
 xl = 0  
 xr = col\_num  
 card\_imgs[card\_index] = card\_img[yl:yh, xl:xr] if color != "green" or yl < (yh-yl)//4 else card\_img[yl-(yh-yl)//4:yh, xl:xr]  
 #以上为车牌定位  
 #以下为识别车牌中的字符  
 predict\_result = []  
 roi = None  
 card\_color = None  
 for i, color in enumerate(colors):  
 if color in ("blue", "yello", "green"):  
 card\_img = card\_imgs[i]  
 gray\_img = cv2.cvtColor(card\_img, cv2.COLOR\_BGR2GRAY)  
 #黄、绿车牌字符比背景暗、与蓝车牌刚好相反，所以黄、绿车牌需要反向  
 if color == "green" or color == "yello":  
 gray\_img = cv2.bitwise\_not(gray\_img)  
 ret, gray\_img = cv2.threshold(gray\_img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
 #查找水平直方图波峰  
 x\_histogram = np.sum(gray\_img, axis=1)  
 x\_min = np.min(x\_histogram)  
 x\_average = np.sum(x\_histogram)/x\_histogram.shape[0]  
 x\_threshold = (x\_min + x\_average)/2  
 wave\_peaks = find\_waves(x\_threshold, x\_histogram)  
 if len(wave\_peaks) == 0:  
 print("peak less 0:")  
 continue  
 #认为水平方向，最大的波峰为车牌区域  
 wave = max(wave\_peaks, key=lambda x:x[1]-x[0])  
 gray\_img = gray\_img[wave[0]:wave[1]]  
 #查找垂直直方图波峰  
 row\_num, col\_num= gray\_img.shape[:2]  
 #去掉车牌上下边缘1个像素，避免白边影响阈值判断  
 gray\_img = gray\_img[1:row\_num-1]  
 y\_histogram = np.sum(gray\_img, axis=0)  
 y\_min = np.min(y\_histogram)  
 y\_average = np.sum(y\_histogram)/y\_histogram.shape[0]  
 y\_threshold = (y\_min + y\_average)/5#U和0要求阈值偏小，否则U和0会被分成两半  
  
 wave\_peaks = find\_waves(y\_threshold, y\_histogram)  
  
 #for wave in wave\_peaks:  
 # cv2.line(card\_img, pt1=(wave[0], 5), pt2=(wave[1], 5), color=(0, 0, 255), thickness=2)   
 #车牌字符数应大于6  
 if len(wave\_peaks) <= 6:  
 print("peak less 1:", len(wave\_peaks))  
 continue  
   
 wave = max(wave\_peaks, key=lambda x:x[1]-x[0])  
 max\_wave\_dis = wave[1] - wave[0]  
 #判断是否是左侧车牌边缘  
 if wave\_peaks[0][1] - wave\_peaks[0][0] < max\_wave\_dis/3 and wave\_peaks[0][0] == 0:  
 wave\_peaks.pop(0)  
   
 #组合分离汉字  
 cur\_dis = 0  
 for i,wave in enumerate(wave\_peaks):  
 if wave[1] - wave[0] + cur\_dis > max\_wave\_dis \* 0.6:  
 break  
 else:  
 cur\_dis += wave[1] - wave[0]  
 if i > 0:  
 wave = (wave\_peaks[0][0], wave\_peaks[i][1])  
 wave\_peaks = wave\_peaks[i+1:]  
 wave\_peaks.insert(0, wave)  
   
 #去除车牌上的分隔点  
 point = wave\_peaks[2]  
 if point[1] - point[0] < max\_wave\_dis/3:  
 point\_img = gray\_img[:,point[0]:point[1]]  
 if np.mean(point\_img) < 255/5:  
 wave\_peaks.pop(2)  
   
 if len(wave\_peaks) <= 6:  
 print("peak less 2:", len(wave\_peaks))  
 continue  
 part\_cards = seperate\_card(gray\_img, wave\_peaks)  
 for i, part\_card in enumerate(part\_cards):  
 #可能是固定车牌的铆钉  
 if np.mean(part\_card) < 255/5:  
 print("a point")  
 continue  
 part\_card\_old = part\_card  
 w = abs(part\_card.shape[1] - SZ)//2  
   
 part\_card = cv2.copyMakeBorder(part\_card, 0, 0, w, w, cv2.BORDER\_CONSTANT, value = [0,0,0])  
 part\_card = cv2.resize(part\_card, (SZ, SZ), interpolation=cv2.INTER\_AREA)  
   
 #part\_card = deskew(part\_card)  
 part\_card = preprocess\_hog([part\_card])  
 if i == 0:  
 resp = self.modelchinese.predict(part\_card)  
 charactor = provinces[int(resp[0]) - PROVINCE\_START]  
 else:  
 resp = self.model.predict(part\_card)  
 charactor = chr(resp[0])  
 #判断最后一个数是否是车牌边缘，假设车牌边缘被认为是1  
 if charactor == "1" and i == len(part\_cards)-1:  
 if part\_card\_old.shape[0]/part\_card\_old.shape[1] >= 7:#1太细，认为是边缘  
 continue  
 predict\_result.append(charactor)  
 roi = card\_img  
 card\_color = color  
 break  
   
 return predict\_result, roi, card\_color#识别到的字符、定位的车牌图像、车牌颜色

**三、系统实施及结果**

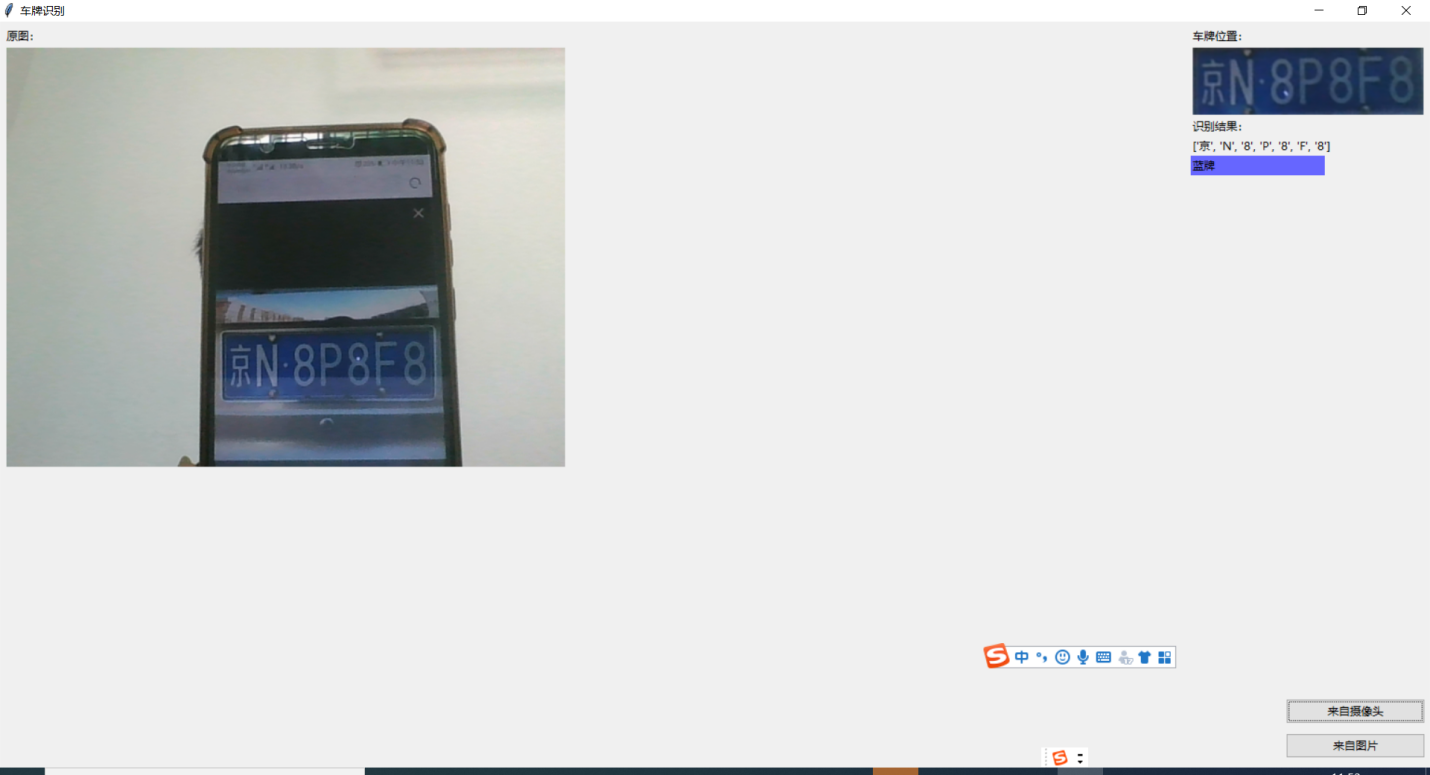
## 3.1 系统测试结果

系统测试步骤，文字描述加截图

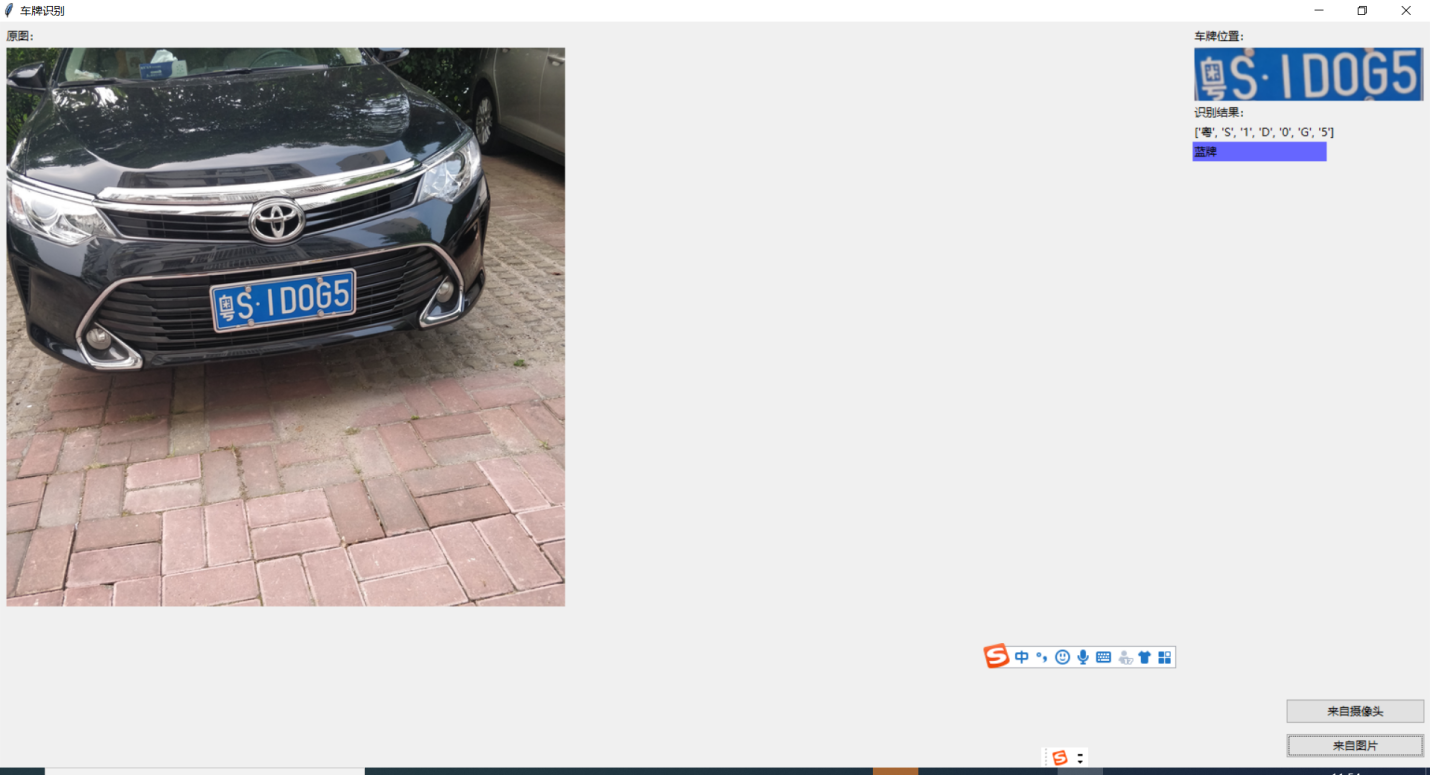
（1）运行进入系统窗口：

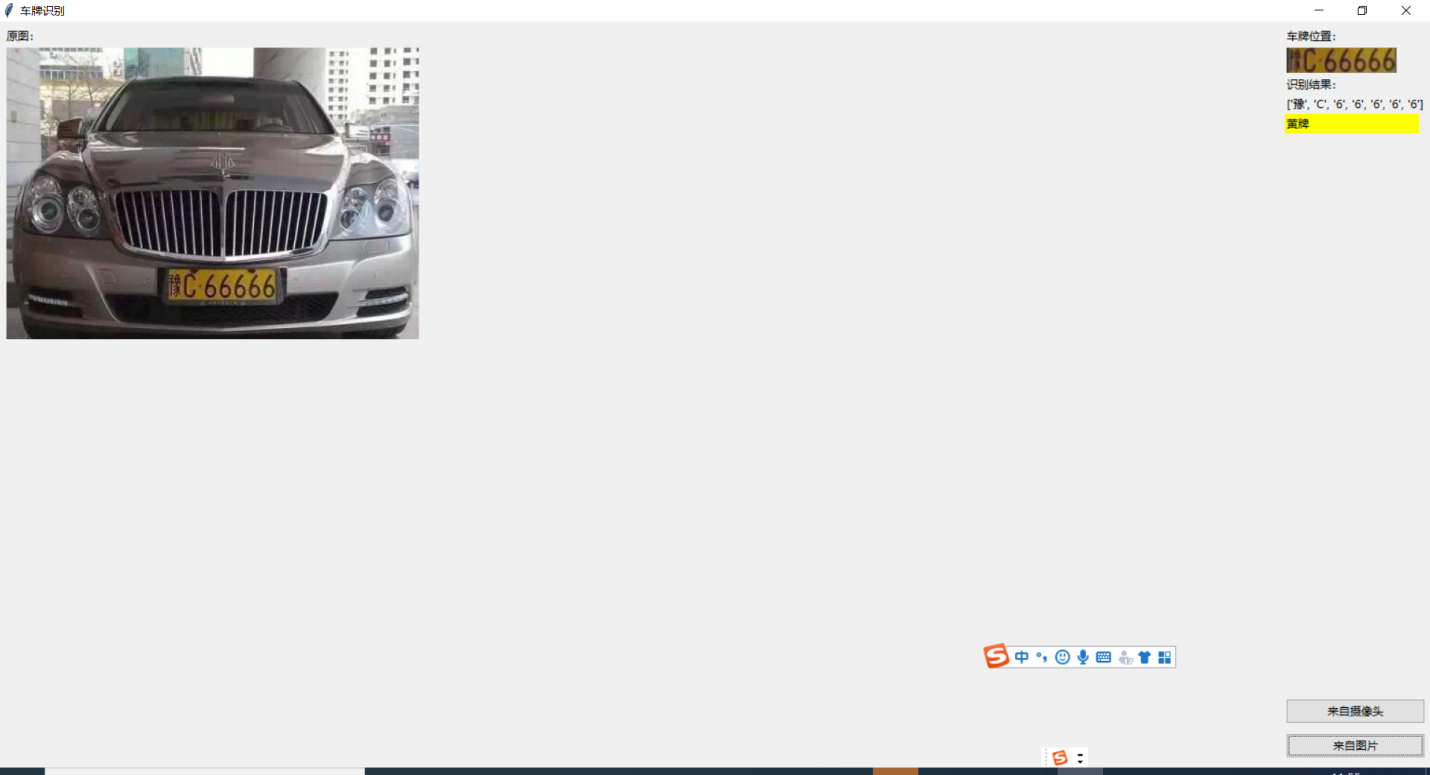


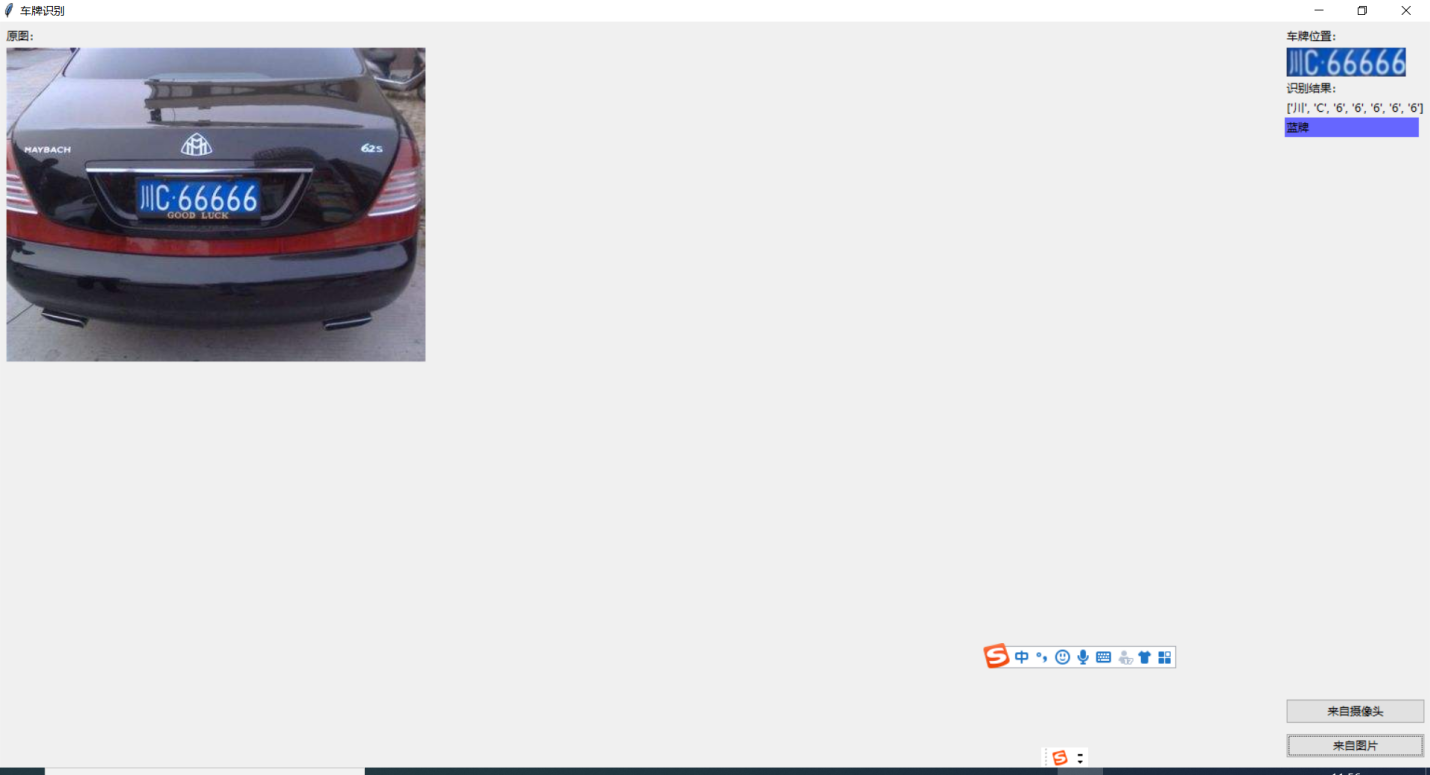
（2）选择“来自摄像头”

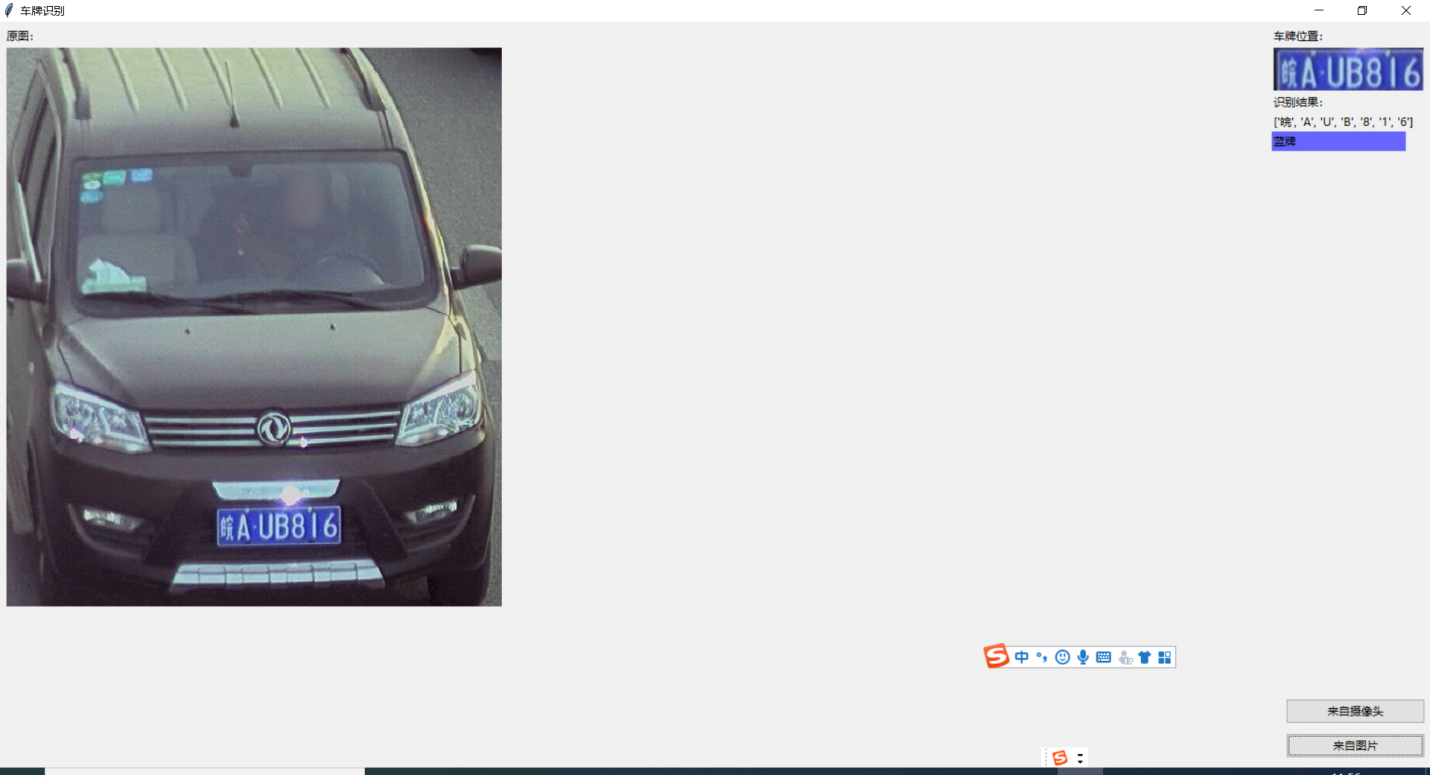


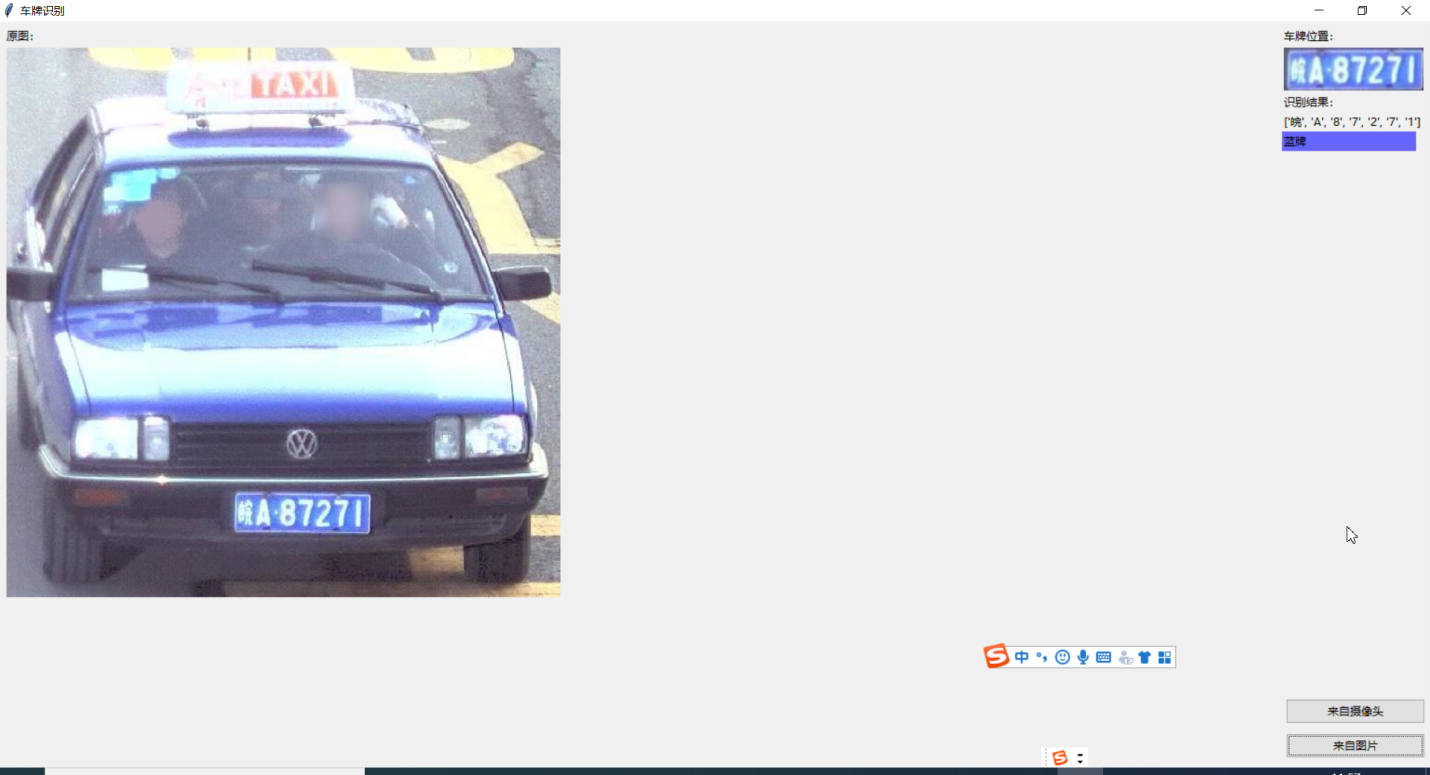
（3）选择“来自图片”

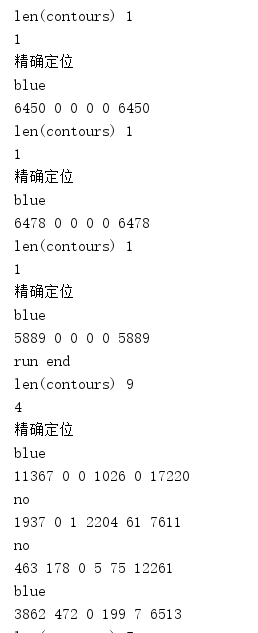












## 3.2 算法性能

给出算法处理的效率（每次处理所需要的时间）和算法的准确率的统计。

测试共22张图片，其中成功识别的数量为13，失败的数量为9。识别准确率为59%

（无法识别：6张（car1、car2、3、8、10、11），识别错误：3张（car4、car5、cAA662F））

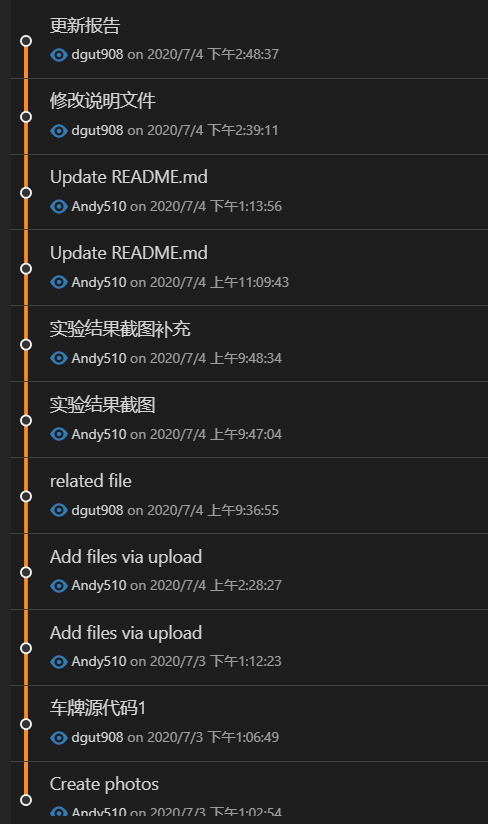
平均处理时间：0.5497s

## 3.3 github提交日志

截图显示各个成员在github提交的日志（必须提供）

dgut908:林泽枫

Andy510：林泽杰



**四、课程设计总结**

简要描述设计过程中遇到的问题，解决的方式

一开始的时候对图像处理的格式有些困惑，最终选择HSV颜色模型，根据HSV模型的特点，可去除光照对实际颜色的影响，有助于图像的分割以及车牌的识别。在初始识别车牌上的字符时，未考虑到车牌上的分割点和铆钉的影响，后根据它们的大小来进行排除。而车牌的边缘容易被认为是数字1，故进行判断，若是检测是1且太细，则认为是边缘，不存储为车牌字符。无法同时且自动识别，故在网上查询到可用tkinyer头文件，设置上传文件按钮，利用askopenfilename函数来打开本地图片文件，上传到程序中。

对于这次设实现来说，最难的点在于精确识别汉字、字母和数字，在测试过程中经常出现汉字识别错误，字母和数字识别混淆和错误，尤其是第一个中文字符出现的误差概率较大。这里我们在讨论和研究后，收集网上关于精确识别的的方法，一次次地调试传入参数，最终调成现在较为精确的状态。

车牌定位算法的参数受图像分辨率、色偏、车距影响，test目录下的车牌的像素都比较小，其他图片很可能因为像素等问题识别不了，识别其他像素的车牌需要修改config文件里面的参数

而最常见的问题就是就是传入参数变量书写错漏，这个问题我们利用了word的查询功能，比较轻易地修改好了相关变量名的错误。

有些逻辑上的问题，在我们两人都难以发现的时候，就主动请教其他同学来帮忙查看。我们先一行行介绍对应代码完成的功能，然后调试给帮手看，有时我们能通过这样发现比较大的逻辑缺陷，有时的缺陷会影响到下一步函数的操作。所以，在我们被坑了几次后，查找错误的时候都会从头开始，一步步精确确认错误所在，再解决问题。这也算是我们本次做比较复杂项目的一个好的经验和收获。