

**创新综合实践**

项目工作总结报告

|  |  |
| --- | --- |
| 项目名称： | 车牌识别系统 |
| 团队名称： | 第一组 |
| 院（系）： | 计算机学院 |
| 专业： | 软件工程 |
| 班级： | 软件1806 |
| 指导教师： | 王文浪 |
| 起止时间： | 2021年8月30日至2021年12月17日 |
| 团队成员： |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **姓名** | **角色** | **分工** |
| 04183180 | 白子煜 | 组长 | 管理、协调、推进整个团队工作、算法设计、程序编码、文档撰写 |
| 04183173 | 高浩 | 设计师 | 架构设计与部署、算法设计、程序测试、资料查找、文档撰写 |
| 04183167 | 阿卜莱提 | 主程序员 | 算法设计、程序编码、文档撰写、程序测试、数据管理、数据整合 |

项目工作总结报告

**1. 项目概述**

## 1.1 项目背景

随着计算机的飞速发展和数字图像技术的日趋成熟，信息来源的自动检测、图像识别技术越来越受到人们的重视，车辆牌照识别技术作为交通管理自动化的重要手段，对于交通车辆管理、停车场管理方面有着十分重要的意义，车辆牌照系统广泛应用在高速公路电子收费站、交通车辆出入控制、公路流量监控、失窃车辆查询、停车场车辆管理、监测黑牌机动车、监控违章车辆的电子警察等需要车牌认证的重要场合，尤其在高速公路收费系统中提高了公路系统的运行效率。如果研制出一种能在公众场合迅速准确地对汽车牌照进行自动定位识别的系统(CPR)，那么这将是一件非常有意义的工作，并将极大地提高汽车的安全管理水平及管理效率。

车辆牌照定位与识别是计算机视觉与模式识别技术在智能交通领域应用的重要研究课题之一。该技术应用范围非常广泛, 其中包括:

1. 交通流量检测；
2. 交通控制与诱导；
3. 机场、港口等出入口车辆管理；
4. 小区车辆管理；
5. 闯红灯等违章车辆监控；
6. 不停车自动收费；
7. 道口检查站车辆监控；
8. 公共停车场安全防盗管理；
9. 计算出行时间；

车辆安全防盗、查堵指定车辆等。其潜在市场应用价值极大，有能力产生巨大的社会效益和经济效益。

## 1.2 国内外研究进展

车牌自动识别技术的研究国外起步地较早。早在20世纪80年代，便有一些零零散散的图像处理方法用于车牌自动识别的某些具体应用。在这个阶段，车牌自动识别技术的研究还没有形成完整的系统体系，一般采用简单的图像处理方法来解决某些具体问题，并且最终结果通常需要人工干预。

进入20世纪90年代后，车牌自动识别的系统化研究开始起步。典型的如Segment)、特征提取(Feature Extraction)和模板构造(Template Formation)、字符识别(CharacterRecognition)等三个部分，完成车牌的自动识别。 CharacterRecognition Technology)分析所获得的图像[1]，首先在二值化图像中找到车牌，然后用边界跟踪技术提取字符特征[2]，再利用统计最邻近分类器(Statistical NearestNeighbor Classifying System)与字符库中的字符比较，得出一个或几个车牌候选号码，再对这些号码进行核实检查，确定是否为该车牌号码，最终确定车牌号码。这个时期的应用在识别正确率方面有所突破，但还没有考虑识别实时性的要求，识别速度有待进一步提高。

由于我国的车牌格式和国外有较大的差异[3]，所以对于国外的车牌识别系统的研究技术[4]，我们只能进行参考，不能进行直接应用。虽然我国车牌的识别需要识别汉字，但是对于英文字母和数字的识别，我们可以借鉴国外的研究技术。例如 基于人车识别的智能车库系统[5],牌照自动识别系统的混合方法[6],基于物联网的利用车牌识别和人工神经网络打开安全门的系统[7], 智能交通系统中车牌识别框架的系统[8]从80年代中期开始，ARGUS英国Alphatech公司就开始了名为RAUS的车牌自动识别系统的研制。ARGUS的车牌识别系统的识别时间约为100毫秒，通过ARGUS的车速可高达每小时100英里。还有Hi-Tech公司的See/Car system, 新加坡Optasia公司的VLPRS等。另外日本、加拿大、德国等发达国家都有适用于本国的车牌识别系统[9]。

国内在90年代也开始了自己的车牌识别系统的研究。车牌识别的技术也是多种 多样，例如在车牌 识别中人工智能技术的运用[10]，低光照下车牌识别关键技术研究[11],基于视觉的车牌与车辆的检测、识别与追踪技术研究[12],目前比较成熟的产品有中科院自动化研究所汉王公司的汉王眼，香港亚洲视觉科技有限公司的慧光车牌号码识别系统等等。另外西安交通大学的图像处理和识别实验室、上海交通大学的计算机科学和工程系、清华大学人工智能国家重点实验室、浙江大学自动化系等都做过类似的研究。虽然这些车牌识别系统的识别率大多都比较高，如95%，甚至97%、98%，但是这些车牌识别系统的识别检测结果大多数是在简单受限制的场景下取得的，在实际的交通场合以及更加复杂的背景环境的情况下，这些车牌识别系统的识别率一般都达不到90%，甚至更低。

可以预见，未来越来越多的细分场景都必须要使用场景专用智能车牌识别一体机，才能让车牌识别技术为产业革命带来更多的发展方向，帮助企业实现大数据管理，从而完成行业改革进步。

## 1.3 本项目工作内容

本项目利用Python开发了一款基于OpenCV（计算机视觉库）的客户端车牌识别软件，支持上传本地或远端汽车图片，并能快速给出车牌的识别结果。

本项目支持蓝、绿、黄三种颜色车牌的车牌识别。

**参考文献**

1. 曹青; 任燕. 智能车牌识别专利技术综述[J]. 中国科技信息,2021,12,01
2. 鲁小平,陈阿林. 基于MATLAB的车牌识别[J]. 测绘通报,2019,10,25
3. 莎可（A.A Sasikala Sewwandi）. 车牌识别系统的设计与实现[J]. 沈阳师范大学,2020,03,23
4. 刘亮. 车牌识别技术研究[J]. 合肥工业大学,2010,04,01
5. Xing Wang;Gongfa Li;Ying Liu;Juntong Yun. Intelligent garage system based on human and vehicle identification[J]. International Journal of Wireless and Mobile Computing,2021,01(6)
6. Baldev Raj Marwah;Nitin Sharma;Pawan Kumar Dahiya. A Hybrid Approach for Automatic Licence Plate Recognition System[J]. International Journal of Sensors, Wireless Communications and Control,2021,01(01)
7. Md Yeasir Arafat;Anis Salwa Mohd Khairuddin;Uswah Khairuddin;Raveendran Paramesran. Systematic review on vehicular licence plate recognition framework in intelligent transport systems[J]. IET Intelligent Transport Systems,2019,05(05)
8. Kevin William George Cowdrey;Reza Malekian. Home automation - an IoT based system to open security gates using number plate recognition and artificial neural networks[J]. Multimedia Tools and Applications,2018,08(15)
9. 刘澍. Research on Key Technologies of License Plate Character Recognition System[J]. 华中师范大学,2019,06,01
10. 罗雅丽. 车牌识别中人工智能技术的应用[J]. 电脑编程技巧与维护,2021,07,18
11. 尹鹏程. 低光照下车牌识别关键技术研究[J]. 北京交通大学,2021,06,30
12. 徐振博. 基于视觉的车牌与车辆的检测、识别与追踪技术研究[J]. 中国科学技术大学,2021,05,26

**2. 系统设计**

## 2.1 需求分析

对本项目进行需求分析。

本次项目主要针对社区各车辆入口建设。

所有的出入口全部采用车牌自动抓拍识别，通过注册园区内部免费车辆、月票车辆、园区外部临时车辆、黑名单车辆进行有效的分类管理。

车牌识别：系统支持按各种类别进行车辆的分类注册，园区内部免费车辆识别自动放行；月票车辆在有效期内自动放行，超期后按临时车辆计费；园区外部临时车辆按时收费放行；黑名单车辆禁止进入园区。

界面优化：支持界面对于切割后的车牌图片进行展示以及存储，并且支持对产出结果的字符展示。对于项目支持打包成为客户端可运行的程序。

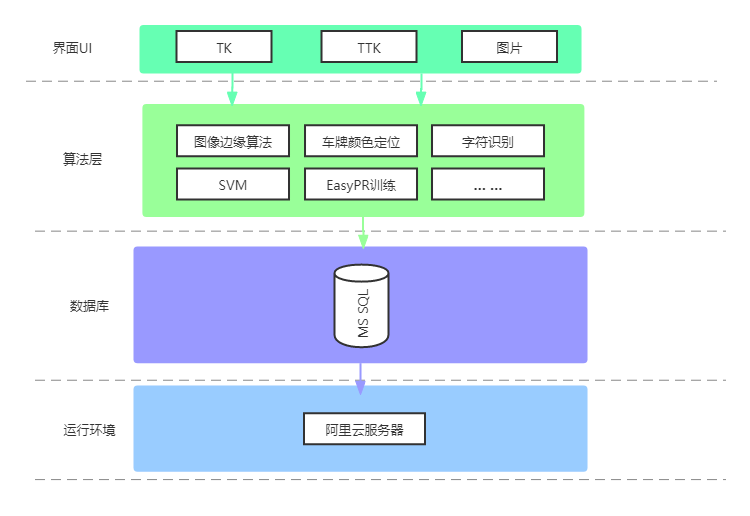
其主要特点如下：

1. 识别系统对环境的依赖性降低至最低限度，可以实现全天候正常工作，并且识别率保持较高水平。
2. 基于LPR识别系统提高了识别的速度和准确性
3. 可识别的最小号牌宽度为140个像素
4. 适应复杂的气候及光照条件，如阴天、雨天、晚上仍可保证高识别率
5. 具有极高的处理能力，对车辆行进过程中的所有图像都能进行识别和处理，不局限于单张图片，有效提高设备对于复杂环境的适应能力。

车辆牌照识别主要是需要利用技术手段实现对车辆抓拍图片进行识别以及得到车牌信息。

车牌识别主要需要实现定位车牌以及识别字符两个步骤，并且对于定位车牌的过程进行优化以及错误处理。对于车牌识别的干扰项有一定的处理能力和修复能力。

## 2.2 系统架构



**图 1系统架构图**

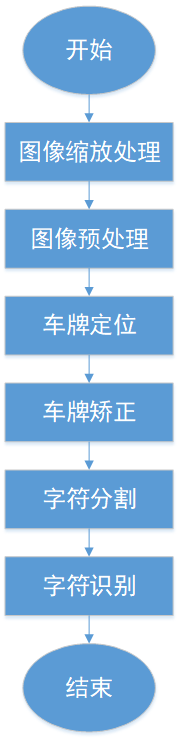
如上图所示为本项目的系统架构图，主要分为界面UI、算法层、数据库层以及对应的运行环境。

界面UI：主要使用Python GUI编程库TK（Tkiner）进行实现，并且支持对于需要分析的车牌图片的展示和车牌切割部分图片的展示。

算法层：主要是用于图片预处理、字符识别等核心算法的实现。

数据库：数据库主要是采用Mysql，包含测试的过程和训练图片的存储地址。

运行环境：数据库的运行环境使阿里云的ECS云服务器。



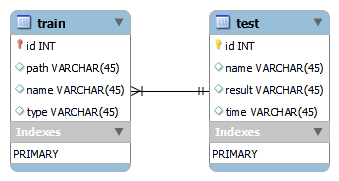
**图 2 汽车车牌识别的步骤**

整个车牌识别的流程分为图像的缩放处理、图像的预处理、车牌定位、车牌矫正、字符分割、字符识别以及最终的结果展示。其中的图像的预处理、车牌定位、字符识别的过程是识别过程的核心算法。

## 2.3 数据库设计

数据来源：数据库主要存储OpenCV的SVM训练数据，主要获取途径是github的EasyPR训练数据；

数据库E-R图设计如下：



**图 3 数据库ER图**

通过上面的ER图可以了解到整个项目使用了连个数据库，分别是统计训练数据的存储表train以及测试数据的收集存储表test：

**表 1 test表设计信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **列名** | **类型** | **默认值** | **数据类型** | **字段说明** |
| **1** | id | int | NULL | int | 测试图片的ID |
| **2** | name | varchar(45) | NULL | varchar | 测试图片名称 |
| **3** | result | varchar(45) | NULL | varchar | 识别结果 |
| **4** | time | varchar(45) | NULL | varchar | 测试时间 |

表1为test表的详细设计信息，由测试图片的相关信息以及识别结果组成，在EasyPR的测试库中可以获取对应的数据并导入。

**表 2 train 表设计详情**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **类型** | **默认值** | **数据类型** | **字段说明** |
| **id** | int | NULL | int | 训练的ID |
| **name** | varchar(45) | NULL | varchar | 训练图片名称 |
| **path** | varchar(45) | NULL | varchar | 训练的内容路径 |
| **type** | int | 0 | int | 训练类型 |

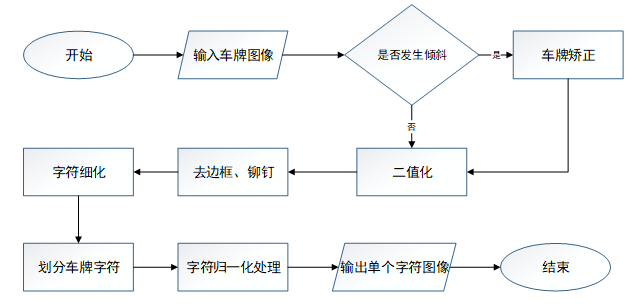
表2是train表的相信设计信息，由训练结果的相关信息以及支持中文的训练类型组成，在中文支持中国的车牌省份缩写，其中使用type的训练类型使用1代表是中文字符训练，使用0代表是数字字符训练。

**3. 算法设计**

车辆牌照识别整个系统主要是由车牌定位和车牌字符识别两部分组成。

其中车牌定位又可以分为图像矫正、边缘取处理、牌照的定位、车牌分割等模块；

字符识别可以分为字符分割、单个字符识别、车牌颜色处理、汉字组合分离等模块。



**图 4 识别整体流程图**

如上图所示，识别整体流程为利用输入车牌的图像，进行车牌预处理以及车牌角度矫正和判断，对于识别过程中的影响因素铆钉和扁阔进行去除和字符的精细化处理，然后对于处理后的字符进行识别和匹配，最后输出匹配的车牌号。

## 3.1 数据获取与预处理

1. 数据获取

车牌获取：考虑到现有牌照的字符与背景的颜色搭配一般有蓝底白字、黄底黑字、白底红字、绿底白字和黑底白字等几种，利用不同的色彩通道就可以将区域与背景明显地区分出来，例如，对蓝底白字这种最常见的牌照，采用蓝色B 通道时牌照区域为一亮的矩形，而牌照字符在区域中并不呈现。因为蓝色（255，0，0）与白色（255，255，255）在B通道中并无区分，而在G、R 通道或是灰度图象中并无此便利。同理对白底黑字的牌照可用R 通道，绿底白字的牌照可以用G 通道就可以明显呈现出牌照区域的位置，便于后续处理。

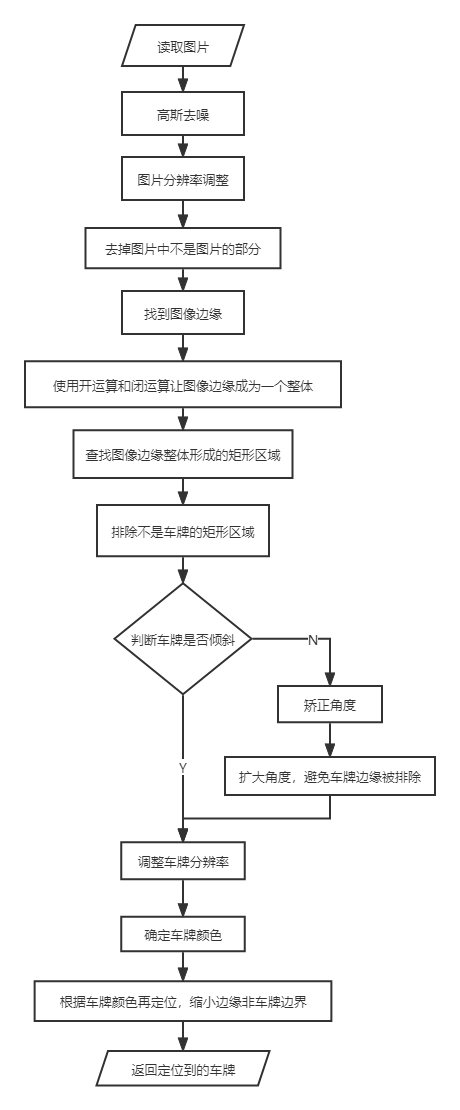
def preprocess\_hog(digits):  
 samples = []  
 for img in digits:  
 gx = cv2.Sobel(img, cv2.CV\_32F, 1, 0)  
 gy = cv2.Sobel(img, cv2.CV\_32F, 0, 1)  
 mag, ang = cv2.cartToPolar(gx, gy)  
 bin\_n = 16  
 bin = np.int32(bin\_n\*ang/(2\*np.pi))  
 bin\_cells = bin[:10,:10], bin[10:,:10], bin[:10,10:], bin[10:,10:]  
 mag\_cells = mag[:10,:10], mag[10:,:10], mag[:10,10:], mag[10:,10:]  
 hists = [np.bincount(b.ravel(), m.ravel(), bin\_n) for b, m in zip(bin\_cells, mag\_cells)]  
 hist = np.hstack(hists)  
 # transform to Hellinger kernel  
 eps = 1e-7  
 hist /= hist.sum() + eps  
 hist = np.sqrt(hist)  
 hist /= norm(hist) + eps  
 samples.append(hist)  
 return np.float32(samples)

通过上述代码结合openCV的sample训练内容。

class SVM(StatModel):  
 def \_\_init\_\_(self, C = 1, gamma = 0.5):  
 self.model = cv2.ml.SVM\_create()  
 self.model.setGamma(gamma)  
 self.model.setC(C)  
 self.model.setKernel(cv2.ml.SVM\_RBF)  
 self.model.setType(cv2.ml.SVM\_C\_SVC)  
 def train(self, samples, responses):  
 self.model.train(samples, cv2.ml.ROW\_SAMPLE, responses)  
 def predict(self, samples):  
 r = self.model.predict(samples)  
 return r[1].ravel()

用上述代码SVM训练车牌判断模型，基于EasyPR进行训练

1. 预处理(车牌定位)



**图 5 车牌预处理流程图**

定位车牌过程如上图所示：

获取采集到的图像之后，第一步需要对图像进行缩放处理。由于车牌采集设备采集到的图像像素点非常密集、数量庞大，这么密集庞大的像素点会严重拖慢车牌定位的速度，所以在车牌定位之前对图像进行缩放处理以减少像素点的数量，这对实时性要求较高的车牌识别系统是非常重要的。接下来要对图像进行高斯平滑和中值滤波以减少采集图像中的噪声点以便提高车牌定位算法的识别率。去除孤立噪声点之后，需要对图像进行灰度化操作，灰度化是指将图片的像素值规定到[0，255]区间上，根据人的眼睛对颜色的敏感程度分配蓝色、绿色和红色的权重，根据式(2-1)对RGB三个分量按照分配的权重进行加权求和可以得到较为准确的灰度图像像素值。

首先对于获取到的车牌进行高斯去噪并同时调整图片的分辨率，降低其他因素对于车牌识别结果的影响。

if blur > 0:  
 img = cv2.GaussianBlur(img, (blur, blur), 0)  
oldimg = img  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

上面的代码通过调整图片的分辨率，然后对于图片进行高斯去噪，以此来实现降低图片噪点，便于进行下一步操作。

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);

然后通过上面的代码去掉图像中不会是车牌的区域，进一步缩小车牌位置范围。

但在大多数情况下，采集到的车牌图像是不能保证正常曝光的，因此需要预设一个阈值以保证不在正常曝光下的车牌区域也能被定位到。计算图像的每个像素点的RGB值与正常曝光下的车牌蓝色RGB值之间的差值，若在阈值范围之内，则认定是车牌区域；若超过阈值，则认定不是车牌区域。依据该方法，可以从采集到的输入图像中获取疑似车牌的区域，该区域可能存在多个，许多符合条件的非车牌区域像素点也有可能存在其中，接下来需要用形态学的方法进一步处理定位出的车牌以获取最终的牌照区域。

通过去掉车牌中不属于车牌的区域，然后便于进行图片的边缘搜寻。

ret, img\_thresh = cv2.threshold(img\_opening, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
img\_edge = cv2.Canny(img\_thresh, 100, 200)  
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)

上面的代码利用处理后的图像找到图像边缘，并且让图像边缘成为整体，便于处理。

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))  
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  
 if wh\_ratio > 2 and wh\_ratio < 5.5:  
 car\_contours.append(rect)  
 box = cv2.boxPoints(rect)  
 box = np.int0(box)

上面的代码查找图像边缘整体形成的矩形区域，在一个个矩形区域中查找车牌，并且对于矩形区域长宽比为2到5.5之间的车牌进行检索。

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)  
 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])  
 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])  
 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)

上面的代码判断是否是倾斜的矩形区域，对于倾斜的部分进行角度矫正，然后使用颜色定位，排除不属于车牌的矩形，并识别对应的蓝、绿、黄三种颜色的车牌。

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

在此期间，需要使用上面的代码不断进行分辨率调整来提高准确率以及降低影响因素。

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"

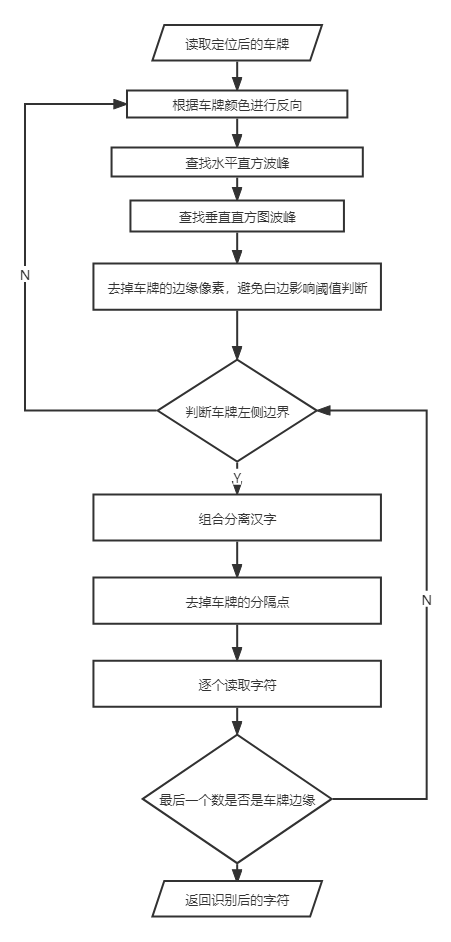
然后利用上面的代码对于颜色偏差的图片进行颜色调整。

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]

最后利用上述代码对车牌颜色进行再次定位，缩小边缘非车牌边界，并且对于X方向、Y方向未缩小的情况再次尝试。

## 3.2 特征计算与分析

车牌字符读取以及直方图分析



**图 6 字符识别计算分析过程**

读取车牌字符的过程如上图所示：

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)

首先通过上面的代码对黄、绿车牌字符的特点对这两种车牌进行反向，避免北京较暗造成识别结果的不准确。

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  
wave = max(wave\_peaks, key=lambda x:x[1]-x[0])  
gray\_img = gray\_img[wave[0]:wave[1]]  
row\_num, col\_num= gray\_img.shape[:2]  
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  
wave\_peaks = find\_waves(y\_threshold, y\_histogram)

其次通过上述代码水平、垂直方向的直方图波峰，并根据波峰最大获取到车牌区域，然后分别去除车牌边缘上下一个像素降低影响阈值。

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)

然后通过上面的代码对车牌左侧边界的判断来进行分离汉字以及车牌分隔点的去除。

并且在这其中也需要考虑对铆钉的处理和车牌边缘的判断：

if np.mean(part\_card) < 255/5:  
 print("a point")  
 continue  
part\_card\_old = part\_card  
w = part\_card.shape[1] // 3  
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 = 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(int(resp[0]))

通过上述的代码实现对锚点的影响排除和去除。

最后将处理后的结果返回给对应的展示界面进行结果的展示。

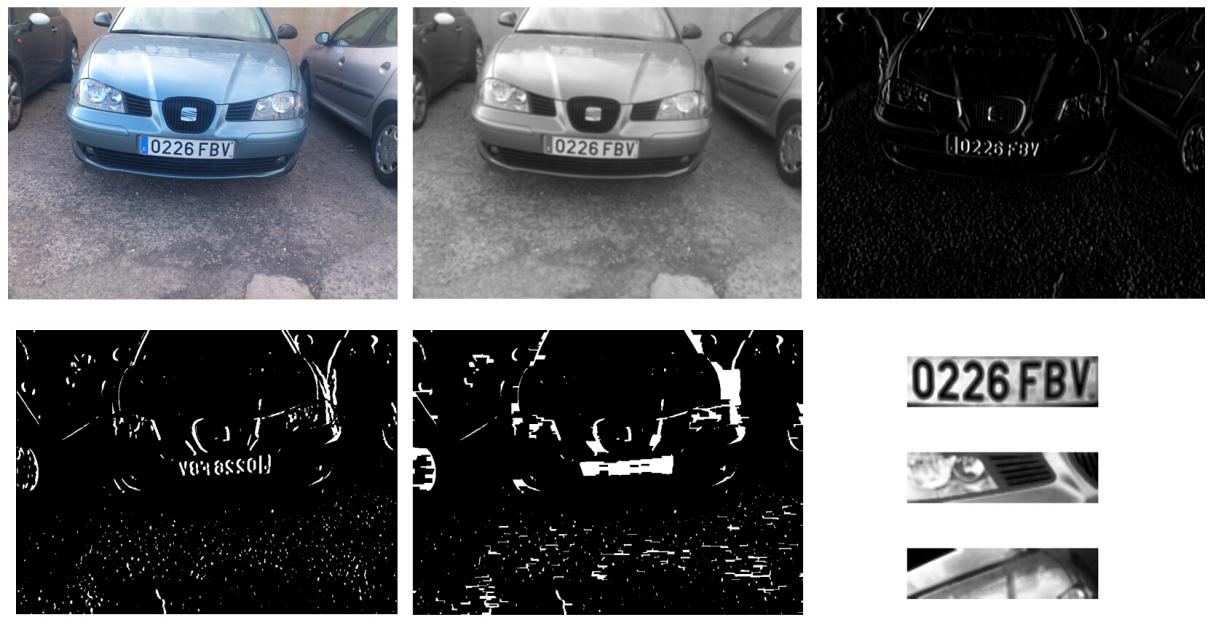
if charactor == "1" and i == len(part\_cards)-1:  
 if part\_card\_old.shape[0]/part\_card\_old.shape[1] >= 8:#1太细，认为是边缘  
 print(part\_card\_old.shape)  
 continue  
predict\_result.append(charactor)

然后利用上述代码进行车牌边缘的识别用来判断是否读取完毕车牌。

## 3.3 分类与预测

SVM 算法应用广泛：不仅仅支持线性和非线性的分类任务，还支持线性和非线性的回归任务。

正如上面叙述的，车牌识别有两个主要步骤，即检测与识别。其中车牌检测的目标是在图像或视频帧中检测到车牌的位置。在完成这一步后，进行识别部分，这里使用OCR算法来识别车牌上的字符，其中有数字，也包含字母。



**图 7 OCR识别过程**

上图中有各个图像的预处理步骤的结果：

其中图1为原彩色图像；

图2将彩色图像转化为灰度图，并采用5\*5模版对图像进行高斯滤波，去除环境噪声；

图3是使用Sobel滤波器求一阶水平方向导数，输出垂直边缘的结果；

图4使用Otsu自适应阈值算法获得图像二值化的阈值，从而得到二值图像；

图5采用闭操作，去除每个垂直边缘线之间的空白空格，并连接所有包含 大量边缘的区域（这步过后，我们将有许多包含车牌的候选区域）；

图6显示了执行SVM分类器后得到的正负样本，此时需要对这些正负样本（其实是分割后的图像块）进行分类。

## 3.4 结果可视化



**图 8 结果展示界面**

结果展示界面采用Python GUI包实现，利用TK提供的功能组件实现界面的展示。

布局实现，界面整体采用上下两栏布局，展示选取图片以及分析结果采取左右两栏布局。

header\_label = tk.Label(  
 win,  
 text='基于OpenCV的SVM算法实现的车牌识别',  
 bg='#0088FF',  
 font=('Arial', 16),  
 fg="#fff", height=2  
)  
header\_label.pack()

顶部蓝色标题展示：利用Python的GUI库Tkinter中包含的Label方法实现，代码实现如上。

def get\_imgtk(self, img\_bgr):  
 img = cv2.cvtColor(img\_bgr, cv2.COLOR\_BGR2RGB)  
 im = Image.fromarray(img)  
 imgtk = ImageTk.PhotoImage(image=im)  
 wide = imgtk.width()  
 high = imgtk.height()  
 if wide > self.view\_width or high > self.view\_height:  
 wide\_factor = self.view\_width / wide  
 high\_factor = self.view\_height / high  
 factor = min(wide\_factor, high\_factor)  
  
 wide = int(wide \* factor)  
 if wide <= 0: wide = 1  
 high = int(high \* factor)  
 if high <= 0: high = 1  
 im = im.resize((wide, high), Image.ANTIALIAS)  
 imgtk = ImageTk.PhotoImage(image=im)  
 return imgtk

左侧布局实现如上代码，需要进行车牌提取的图片展示是利用ImageTk结合GUI的grid布局进行展示。

def show\_roi(self, r, roi, color):  
 if r:  
 roi = cv2.cvtColor(roi, cv2.COLOR\_BGR2RGB)  
 roi = Image.fromarray(roi)  
 self.imgtk\_roi = ImageTk.PhotoImage(image=roi)  
 self.roi\_ctl.configure(image=self.imgtk\_roi, state='enable')  
 # 控制车牌号的展示  
 self.r\_ctl.configure(text="".join(r))  
 self.update\_time = time.time()  
 # 控制结果展示以及背景色控制  
 try:  
 c = self.color\_transform[color]  
 self.color\_ctl.configure(text=c[0], background=c[1], state='enable')  
 except:  
 self.color\_ctl.configure(state='disabled')  
 elif self.update\_time + 8 < time.time():  
 self.roi\_ctl.configure(state='disabled')  
 self.r\_ctl.configure(text="")  
 self.color\_ctl.configure(state='disabled')

结果展示实现如上，对于分析得到的车牌字符列表进行join方法拼接并且添加对于不同的车牌结果进行背景色的控制展示。

from\_pic\_ctl = tk.Button(  
 analysis\_res\_data,  
 text="选择图片",  
 width=20,  
 height=2,  
 activebackground="#fff",  
 activeforeground="#0088FF",  
 font=('Arial', 16),  
 bg="#fff",  
 command=self.from\_pic  
)

侧底部的选择图片按钮实现按钮如上，使用了GUI库中的Button(按钮元件)。

## 数据安全与隐私设计

Github 私有仓库管理：

利用私有github的特点，避免仓库代码被其他人访问的同时，也保证了测试训练数据的安全。

测试数据加密压缩处理：

对于测试的数据和训练的数据进行加密压缩来确保数据的安全。

**4. 系统实现**

## 4.1 开发环境

以下是本次系统研发中选择的开发技术，开发工具及其介绍。

Python 3.10.1：

Python语言包含了计算机视觉库OpenCV，3.10.1版本中的模式匹配大大增加了控制流的清晰度和表达能力，提供了更好的类型检查方法和更加人性的错误提示。

OpenCV：

OpenCV是Intel开源计算机免费视觉库，可以运行在[Linux](https://baike.baidu.com/item/Linux/27050" \t "https://baike.baidu.com/item/opencv/_blank)、[Windows](https://baike.baidu.com/item/Windows/165458" \t "https://baike.baidu.com/item/opencv/_blank)、[Android](https://baike.baidu.com/item/Android/60243" \t "https://baike.baidu.com/item/opencv/_blank)和[Mac OS](https://baike.baidu.com/item/Mac OS/2840867" \t "https://baike.baidu.com/item/opencv/_blank)操作系统上，提供了Python语言的接口，实现了[图像处理](https://baike.baidu.com/item/%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86/294902" \t "https://baike.baidu.com/item/opencv/_blank)和计算机视觉方面的很多通用算法和常用的特征提取与匹配方法。

Tkinter：

Tkinter模块是[Python](https://baike.baidu.com/item/Python/407313" \t "https://baike.baidu.com/item/_blank)的标准Tk GUI工具包的接口，tkinter提供了更丰富的窗口组件。

2021.3(Community Edition)：

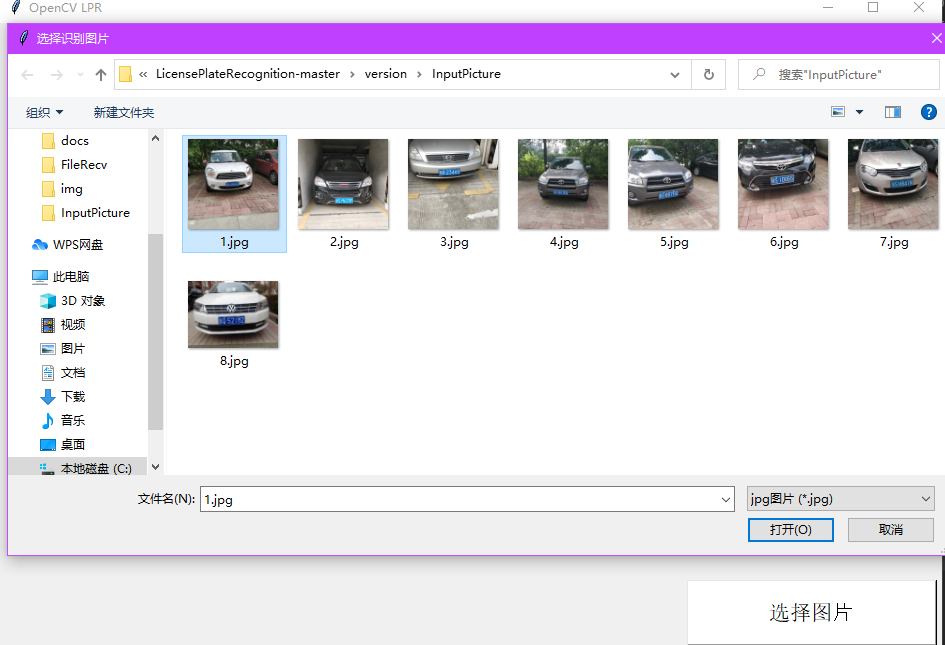
PyCharm是一种[Python](https://baike.baidu.com/item/Python/407313" \t "https://baike.baidu.com/item/PyCharm/_blank) [IDE](https://baike.baidu.com/item/IDE/8232086" \t "https://baike.baidu.com/item/PyCharm/_blank)（集成开发环境），具备了Python 软件包的共享索引 ，功能多，各种提示超强，每个文件都有其输出窗口，可以实现更顺畅的代码完成。

## 4.2 系统功能及实现



**图 9 系统的主界面**

上图为车牌识别系统的主界面，当前为没有选择分析图片的界面，可以点击选择图片然后开始下一步选择图片：



**图 10 选择车牌图片界面**

上图为选择车牌图片的界面，可以选择设备本地保存的车牌图片，并且支持黄、绿色车牌的选择和识别。点击了对应的图片后就开始对图片的车牌识别。



**图 11 成功识别蓝色车牌信息**



**图 12 成功识别蓝色车牌信息**

上面两张图为车牌识别的结果，其中就包含两张不同位置汽车的车牌进行识别，并且都可以正确的识别。除此之外，我们还支持实现了黄、绿色的车牌识别。



**图 13 成功识别绿色车牌信息**

上图为对于新能源汽车，绿色车牌的识别结果，并且支持对于不同颜色的车牌进行对应颜色的结果展。



**图 14 识别黄色车牌信息**

上图为成功识别黄色的车牌，并且通过对于黄绿色车牌的颜色反转避免了车牌识别过程中对于暗色车牌识别错误。

但是在识别过程中依旧存在对于中文字符训练次数不足的情况，导致在中文字符的笔画复杂时，训练的结果不能够准确的匹配到中文的字符和边界，导致采用了默认的“川”作为车牌的省份区别。

当然，除了对于中文字符的识别问题，我们还发现了在某些特定情况下，对于曝光度过高或者照片亮度过高的车牌会出现高斯模糊后直接无法判定车牌所在位置和车牌的信息的情况。



**图 15 车牌颜色和号码都检测错误**

上图黑车的车牌识别错误，并且出现了对于车牌截取错误的图片，在这里我们经过反复验证得到的结果是在强光的作用下，高斯模糊后图片的噪点依旧很高，无法准确的获取到对应的车牌区域，因此也难以对于车牌的内容进行识别。

在这里我们思考后发现需要通过对于EasyPR最新的训练结果进行导入和处理，除此之外，还需要获取除EasyPR之外的训练结果来增加最终结果的准确性。



**图 16 车牌颜色成功检测，车牌号码检测失败结果**

上图是因为在判断边界的过程中，没有对于边界的不可展示部分进行处理，在进行了展示结果位数限制后可以解决这一问题。

## 5. 项目总结

## 5.1 项目小结

工作内容方面：

本项目利用Python语言开发了一款基于OpenCV（计算机视觉库）的客户端车牌识别软件，Python语言包含了计算机视觉库OpenCV，提供了更好的类型检查方法， OpenCV是Intel开源计算机免费视觉库，可以运行在[Linux](https://baike.baidu.com/item/Linux/27050" \t "https://baike.baidu.com/item/opencv/_blank)、[Windows](https://baike.baidu.com/item/Windows/165458" \t "https://baike.baidu.com/item/opencv/_blank)、[Android](https://baike.baidu.com/item/Android/60243" \t "https://baike.baidu.com/item/opencv/_blank)操作系统上，提供了Python语言的接口，实现了[图像处理](https://baike.baidu.com/item/%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86/294902" \t "https://baike.baidu.com/item/opencv/_blank)和计算机视觉方面的很多通用算法和常用的特征提取与匹配方法，项目界面用Python自带的tkinte GUI工具包的接口来实现，tkinter提供了更丰富的窗口组件，开发工具用PyCharm，它具备了Python 软件包的共享索引 ，功能多，各种提示超强，每个文件都有其输出窗口，可以实现更顺畅的代码完成，车牌识别主要需要是实现定位车牌以及识别字符两个步骤，并且对于定位车牌的过程进行优化以及错误处理。对于车牌识别的干扰项有一定的处理能力和修复能力。

项目的系统架构有两层，第一层是界面UI层包含TK，TTK，图片，第二层是算法层包含图像边缘算法，车牌颜色定位，SVM，EasyPR训练等等，车辆牌照识别整个系统主要是由车牌定位和车牌字符识别两部分组成，识别的步骤是先对图像进行缩放处理进行颜色定位，然后对图像进行矫正，字符分割，最后对字符进行识别。

项目成果方面：

本项目成功实现了客户端车牌识别，支持上传本地或远端汽车图片，并能快速给出蓝、绿、黄三种颜色车牌的车牌识别结果。

特色以及创新：

本项目的特色以及创新是定位车牌以及识别字符两个方面，并且对于定位车牌的过程进行优化以及错误处理，对于车牌识别的干扰项有一定的处理能力和修复能力。

后续计划以及工作安排：

本项目后续计划以及工作安排是把项目进一步延展，实现更清晰的识别车牌，解决外部环境条件的变化而产生了明显的差异，因为太阳光的照射，改变了那些字的明亮度，所以从早上到夜晚，这些字符的灰度会随着时间的推移产生很明显的变化。同时，由于周围的各种灯光和阴影，以及其他一些周边的物体，都会对这些字符产生变化，还有实现在不同的天气环境同样识别出车牌信息，比如下雨，下雪，雾等。

## 5.2 团队组织管理小结

心得与经验：

由于我们这次项目关于图像处理方面的内容，我们之前没有遇到过，也不知道从哪下手，所以团队之间的互相帮助和一去讨论和学习的气氛起到了很大的作用，有了有效的团队合作才能保证每个人在开发过程中更加得心应手，所以我们在分工编写前先在一起讨论，确定了系统的主要模块及其功能，统一它们的名称，然后各自编写各自所分工的部分，发现应该添加新功能或原先的设计有问题时再一起讨论。

项目管理中必须要考虑风险问题，那么，风险在哪里，如何进行有效的跟踪呢？一个典型的问题是，很多团队意识到了风险，却没有将风险记录下来，导致风险只停留在线下，没有传播到团队各个成员和领导层中，更没有人对风险进行把控。合理的把控风险能降低开发中遇到的问题。

如何保障团队持续过程改进，也是管理者需要关注的问题，其前提是团队需要有可量化的效能指标。传统的研发数据统计大多是通过人工或半人工的手段完成，不仅费时费力，还可能产生数据精准度问题。另外，选用的数据指标是否客观、有说服力，不同成员间的数据分散在不同的系统中难以打通，这些都是导致团队效能难以度量的重要原因。

对于自己所做功能中有需要用到其他组员模块中的数据时，积极和其他团队中的成员进行沟通，保证数据在传送的过程中的正确性，由于功能会相互影响，所以大家进行的多次讨论可以发现很多问题和之前没有想到的细节，不断发现问题，不断改正错误，通过本次的项目学会了最新图像处理技术。

反思与教训：

经过这一学期的课程设计中我们学到了很多，我们明白了团队意识的重要性，在交流和学习中，我们都在不断的成长。在设计过程中虽然遇到了一些问题，暴露了我们的知识欠缺和经验不足，实践出真知，通过亲自动手制作，是我们掌握的知识不再是纸上谈兵。在今后的学习与实践过程中，一定要不懈努力，遇到问题不能退缩而要迎难而上，这个课程设计带给我们一些做项目的经验。

规划阶段，团队战略⽬标到执⾏层层关联并对⻬，公开透明，并通过充分的沟通保证成员理解目标。在怎么做上，总是以优先级排序所有工作，保证团队目标聚焦。

执行阶段，需要建⽴标准⼯作⽅法与流程，利⽤软件⼯具数字化、自动化⼯作流程，提升效率。同时，每日检查项目进展情况，提前发现和应对风险，保证项目按时、高质量交付。

分析阶段，通过⼯作数据的结构化存储，聚合量化。利用数据进行团队考核时，应选取业务结果数据，如按时交付率、缺陷密度。而在做业务优化时，则选择过程数据，如技术评审通过率、故事点完成速率。

首先得明白一个项目不仅仅是一个人就可以完成的。所以这期间的分工合作就非常重要，一个合理的分工方式往往会对整体效率造成不小的影响。另外就是对项目的各模块之间的名称命名标准得做统一性要求，不然最后合并时会产生很多不必要的麻烦。

其次就是每天的任务进度报告，这看起来毫无影响的东西其实非常重要。它会向总的管理者提供你每天的任务状态，以方便对一些问题及时处理，也会为接下来的工作安排提供指导性。

## 6. 团队成员总结

## 6.1 团队人员白子煜个人总结

开发任务：

开发界面代码（Surface.py）以及进行OpenCV SVM训练、编写文档算法设计部分以及文档最终审核。

界面代码利用Python GUI库TK实现，并且加入对于结果的颜色展示和裁剪的车牌区域展示。其中利用gird布局确保展示结果无遮挡且不受界面缩放的影响依旧可以正常使用。

算法部分对于黄绿色的车牌在进行测试时发现背景色偏暗导致车牌无法识别，通过查阅资料了解到借助颜色反转便可以实现对应的车牌识别，而且避免了偏暗环境下直接无法获取到车牌的情况。

在进行编写过程中发现车牌的边缘和车牌的铆钉会对车牌的识别结果产生影响甚至导致车牌识别错误。最终通过去除铆钉和车牌边缘的宽度判断的方式降低了这两种因素对于车牌识别最终结果的影响。

在测试过程中对于出现的识别数据错误和失败，再次进行SVM训练和EasyPR训练结果导入，增加了识别的成功率。

对于最终的项目启动加入了bash脚本进行项目的依赖安装以及项目的启动，避免了项目在不同设备下出现项目代码无法启动或者启动命令不统一的情况。

开发成果：

界面正常展示分析结果、界面功能无异常、训练结果计算成功率超过80%。

团队协作：

推荐使用github进行代码管理，并且对于开发进度进行调整以及安排。利用github提供的代码私有化仓库和git的分布式团队开发管理开发流程和开发进度，极大的提高了代码的可维护性和追根溯源的效率。

进行Python技术分享和技术答疑，对于不同阶段的技术问题及时处理。在本次开发过程中，在python的学习过程中主要掌握了python的class以及函数的使用，并且了解了python GUI编程的相关技巧以及信息。

对于团队成员的算法存疑进行各个不同平台的信息查询整合，在建立的群组里进行转发和讨论，提高了团队成员的积极性并且避免了其他团队成员在相同或类似的问题上再次踩坑。

收获：

采用Python开发车牌识别理解了Python的代码特点以及结合Python使用数据处理。利用OpenCV进行SVM训练和车牌识别的过程模拟，并且了解了回归算法的特点以及应用。

使用SVM训练了解不同车牌识别方式的特点以及准确性。训练的内容其实更多的是对训练数据的依赖，当电脑性能达到一定高度并且有非常多的训练数据时，可以大幅提升训练速度和识别的准确率。

了解了合作开发的重要性。团队开发不同于个人，作为团队成员的管理者更是需要对于团队的整体情况有清晰的把控和对于团队问题的快速定位，并且能快速针对问题提出合理的解决方案或者解决思路，利用编程的通用性特点快速帮助团队成员进行开发和问题的处理以及修复。对于团队每个人开发的任务进行合理的把控以及进度的跟进，以此来实现对于整体的把控。

反思：

刚开始全员投入对于python的学习和openCV的使用，花费了大量的时间去让每个人做同样的事情，对于团队整体来说浪费了宝贵的时间。团队开发中需要合理安排每个人的工作以及任务，并且需要经常进行进度同步以及遗留问题处理。学习新技术的时候应该是以快速上手为准，从基础开始学习不能快速满足开发需求以及应用场景。

团队初期在对于进度的把控上没有去考量每个成员的个人开发水平以及算法基础，导致前期项目推动困难并且项目的整体流程过慢。对于需要开发的内容需要提前考量技术方案以及技术难以程度，结合自身以及团队成员进行合理的安排和计划。

## 6.2 团队人员阿卜莱提个人总结

开发任务：

Python语言实现GUI（用户界面）的实现，ANN算法设计，数据整合，EasyPR去做训练，提高识别结果的准确性。

开发成果：

本项目中我实现了通过Python语言设计GUI，设计ANN算法，数据整合和EasyPR去做训练，提高识别结果的准确性，GUI的实现，采用Tkinter模块来实现，因为Tkinter模块是[Python](https://baike.baidu.com/item/Python/407313)的标准Tk GUI工具包的接口，tkinter提供了更丰富的窗口组件，实现GUI更加方便。

算法设计采用ANN算法，ANN是指由大量的处理单元(神经元) 互相连接而形成的复杂网络结构，是对人脑组织结构和运行机制的某种抽象、简化和模拟。 人工神经网络（Artificial Neural Network，简称ANN ），以数学模型模拟神经元活动，是基于模仿大脑神经网络结构和功能而建立的一种信息处理系统，它是物流合作伙伴选择方法中合作伙伴选择的[神经网络算法](https://baike.baidu.com/item/%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E7%AE%97%E6%B3%95)的另一种名称。它是20世界80年代后迅速发展的一门新兴学科，ANN可以模拟[人脑](https://baike.baidu.com/item/%E4%BA%BA%E8%84%91/1432766)的某些智能行为，如知觉，灵感和[形象思维](https://baike.baidu.com/item/%E5%BD%A2%E8%B1%A1%E6%80%9D%E7%BB%B4/7417887)等，具有自学性，[自适应](https://baike.baidu.com/item/%E8%87%AA%E9%80%82%E5%BA%94/4862182)和非线性动态处理等特征。

 团队协作：

作为一个团队交流很重要，我们还建立了小组讨论群，有什么问题大家随时能进行交流，监督彼此的任务进度，通过这样的方法，大家都会变得积极起来了。因为通过团队合作，有利于激发团队成员的学习动力。

大部分人的心里都有期望他人尊敬自己的欲望，都有不服输的心理，都有精益求精的欲望。这些心理因素都不知不觉地增强了成员的上进心，使成员都不自觉的要求自己要进步，力争在团队中做到最好团队项目最忌讳“闭关锁国”，自己一个人埋头苦干是不会有什么大的进展的，而且根本做不好一个项目，我们小组积极交流，齐心协力才能完成了这次的课程设计任务。

 收获：

 这个课程设计带给我们一些做项目的经验。首先得明白一个项目不仅仅是一个人就可以完成的。所以这期间的分工合作就非常重要，一个不合理的分工方式往往会对整体效率造成不小的影响，另外就是对项目的各模块之间的名称命名标准得做统一性要求，不然最后合并时会产生很多不必要的麻烦。其次就是每天的任务进度报告，这看起来毫无影响的东西其实非常重要，它会向总的管理者提供你每天的任务状态，以方便对一些问题及时处理，也会为接下来的工作安排提供指导性。最后就是对代码的要求了，代码显然是整个项目的核心，它的简洁与否直接影响最后对项目程序的调试的难易程度，因此代码必须简洁，规范。当然我们也从这其中认识到团队的重要性，也使我们更多的了解到团队意味着什么，通过本次课程设计，深刻体会到软件开发文档编写的重要性，软件开发文档是指导我们进行项目开发的指挥棒。

编写好一份高效可行的软件开发文档对项目开发非常重要。还有对系统的需求分析非常重要，需求的变成直接影响项目的开发进度。这次课程设计首先对系统的需求理解不透彻，走了一点弯路，浪费了一些时间。在以后的软件开发过程中尽量吸取本次课程设计的经验教训，提高项目开发效率。

反思：

研发系统中对遇到的问题和解决方案进行记录是十分重要，这样下次再遇到类似的情况，我们很快就找到合理的解决方案，也不用花费大量时间查询解决办法，这也是积累经验的一种方式，还有研发系统之前的合理的分工任务也很重要，因为小组内的每个组员的能力，强项和弱项不一样，没有合理的分工组员任务，会导致组员没办法发挥自己最佳能力或者有些组员可能觉得自己的任务太难没办法实现导致影响项目的进度，然后彼此之间的任务进度监督也很重要，有了彼此之间的监督也不会出现因为懒惰影响项目研发进度的情况，也会帮助我们养成一个拖延事情的好习惯。

项目研发之前组内必须得讨论和统一好项目的主要功能，项目需求，项目风格，模块等等，这样能避免项目研发过程中的一些不必要的修改过程，也会对最后的整合整个项目有所帮助。

最后就是需要认真撰写项目研发报告，因为我们写报告过程中会对整个研发过程进行回顾，这会帮助我们跟进一步了解研发技术，同时我们能发现在自己身上存在的好多问题。

## 6.2 团队人员高浩个人总结

开发任务：

 C/S架构设计与部署、 SVM部分算法设计、程序测试、资料查找和文档撰写。

开发成果：

 对与架构设计来说，首先考虑到C/S和B/S架构的优缺点，C/S结构在技术上很成熟，它的主要特点是交互性强、具有安全的存取模式、网络通信量低、响应速度快、利于处理大量数据。因为客户端要负责绝大多数的业务逻辑和UI展示，又称为胖客户端。它充分利用两端硬件，将任务分配到Client 和Server两端，降低了系统的通讯开销。然而B/S架构个性化特点明显降低无法实现具有个性化的功能要求，在跨浏览器上，B/S架构不尽如人意，客户端服务器端的交互是请求-响应模式，通常动态刷新页面，响应速度明显降低（Ajax可以一定程度上解决这个问题）。无法实现分页显示，给数据库访问造成较大的压力，所以综上来说选择C/S架构。

对于算法设计，选择SVM算法，Svm（support Vector Mac）又称为支持向量机，是一种二分类的模型。当然如果进行修改之后也是可以用于多类别问题的分类。支持向量机可以分为线性核非线性两大类。其主要思想为找到空间中的一个更够将所有数据样本划开的超平面，并且使得本集中所有数据到这个超平面的距离最短。点到超平面的距离公式和最大间隔的优化模型。

要识别车牌字符，前提是先进行车牌字符的正确分割与提取。字符分割的任务是把多列或多行字符图像中的每个字符从整个图像中切割出来成为单个字符。车牌字符的正确分割对字符的识别是很关键的。

测试采用鲁棒性测试，上传同种图片进行车牌识别，测试结果：识别出相同车牌号；上传不同省的车牌号照片进行识别，测试结果：能识别出相同的车牌号；上传不同颜色的车牌号进行识别：测试结果：能识别出相同的车牌号

团队协作：

使用git进行多人开发。团队的运作会遇到非常多复杂的场景，要提高团队成员间的合作效率，就要在复杂当中识别规律，建立各种有效的机制，来提高团队协作的效能。

团队成员个人，大多都具有自身的特长，背景，经验。互补的技能组合、多元化的观点，让团队有更大的可能面对挑战，取得成果。

收获：

本次项目让我对车牌识别系统的实现和研究方法进行了深入了解。分别从图像预处理、车牌定位、字符分割以及字符识别等方面进行了学习和分析。

在车牌定位前，先对图像进行预处理，再进行边缘检测操作等方法，进而为定位打下基础。通过实验运行的结果可以看出，图像预处理、sobel边缘检测、灰度跳变等对车牌的定位都是十分有效的。

反思：

对多组图片进行实验能很好的定位车牌，准确的进行车牌字符的分割，得到了很好的实验效果。本实验程序车牌的识别部分并不是十分准确，但是通过对代码的学习和网上资料查询，深切体会到方法的多样性和复杂性，不断改进的方法促使人不停的进步与学习。