分　类　号：　TP31　　单位代码：　10183
研究生学号：　2017532026　　密　　级：　公　　开

吉 林 大 学

硕士学位论文

(学术学位)

基于目标检测算法的智慧车库系统

A Smart Garage System Based on Object Detection Algorithm

作者姓名：高放

专　　业：计算机软件与理论

研究方向：数据挖掘应用

指导教师：王利民　教授

培养单位：计算机科学与技术学院

2020 年 5 月

基于目标检测算法的智慧车库系统

A Smart Garage System Based on Object Detection Algorithm

作者姓名：高放

专业名称：计算机软件与理论

指导教师：王利民 教授

学位类别：工学硕士

答辩日期：2020 年 5 月 30 日

吉林大学博士(或硕士)学位论文原创性声明

本人郑重声明：所呈交学位论文，是本人在指导教师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：高放

日期：2020 年 5 月 30 日

关于学位论文使用授权的声明

本人完全了解吉林大学有关保留、使用学位论文的规定，同意吉林大学保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权吉林大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

（保密论文在解密后应遵守此规定）

论文级别：■硕士 □博士

学科专业： 计算机软件与理论

论文题目：基于目标检测算法的智慧车库系统

作者签名： 指导教师签名：

高放

2020 年 5 月 30 日

作者联系地址（邮编）： 吉林省长春市前进大街 2699 号吉林大学计算机科学与技术学院（130021）

作者联系电话：17386861217

摘 要

基于目标检测算法的智慧车库系统

人们开车进出自家车库时需要停车拿出遥控钥匙，这一过程会分散驾驶员的注意力，有一定的安全风险，同时也不方便。虽然国内大型停车场和办公楼等区域已经普及了根据车牌自动开关闸门的设备，但这些设备一般都需要按照场地定制而且价格较贵，不适合人们在自家车库中安装。随着计算机硬件计算能力的提高和计算机视觉技术的发展，家用低成本的稳定智慧车库系统已经可以实现。

本文描述了一个家用智慧车库系统的算法和软硬件设计及其实现。包括车牌号识别的算法、车头车尾识别的算法，以及两者的实现；驾驶员驾车进入和离开车库时，自动开关车库门和车库灯的方法及软件实现；用户在不泄露个人隐私的前提下远程遥控车库门等电器的方法和实现。同时本文还为上述功能进行了用户图形界面设计，并对这一设计进行了基于浏览器-服务器模式的软件实现。

首先，本文在第一部分介绍了利用目标检测算法进行车头尾和车牌识别的方法，包括算法的设计和改进过程，并详述了此深度学习算法所需的数据收集、数据清洗、数据标记、模型训练的过程和结果，结果证明经过改进的基于目标检测算法的车辆识别系统可以快速识别出摄像头视频流中的车牌号和车头车尾信息。

接着，本文在第二部分介绍并实现了通过计算机与嵌入式设备的通信，以蓝牙或射频信号对非智能车库门、灯、报警器等设备进行无线控制的方法。实验结果显示经过硬件连接后，所编写的程序可以将非联网家用电器通过计算机网络进行远程无线控制。

然后，本文对比了不同的用户界面实现方式，讲述了基于浏览器与服务器模式的用户图形界面设计，以及用户控制中心的服务器端实现和用户使用界面实现。本文所编写的服务端软件使得用户可以利用手机、电脑、平板电脑等计算设备对家庭车库中的电器进行配置和主动控制。

最后，本文在第三部分介绍了如何将前两部分所述算法、外部设备、用户界面在 Linux 系统中整合，成为可供家庭用户使用的一套智慧车库系统。这套系统已经实际运行了 18 个月，并可以在日间稳定识别家庭车辆并开关车库门、灯等，极大方便了人们的生活。

关键词：

目标检测，深度学习，车牌识别，树莓派，Ruby on Rails，YOLO。

Abstract

A Smart Garage System Based on Object Detection Algorithm

When people drive back home, fetching remote control to open garage gate is a distraction, it's neither convenient nor safe. Even though automatic parking lot gate is widely used in most commercial centers and companies, which can recognize the license plate number and switch the gate accordingly, they're not applicable for civil home garage uses or installations. These gates are expensive and require customization for different use cases. With the improvement in speed and cost of computer hardware, and the breakthrough of computer vision technology, we can build a low cost and stable smart garage system for home use now.

This paper discusses the algorithm and implementation of a home smart garage system, including its hardware and software designs and their implementations. The paper includes the design and tradeoff of the algorithms for license plate recognition, car front or car rear recognition. The methods and software for automatically controlling garage gate and garage light while user enters or leaves garage are discussed. This paper also addresses the methods for remote control of the garage without leaking user personal information and presents the design and implementation of a browser-server based graphical user interface for the functionalities above.

In the first part, this paper introduces an object detection algorithm, and how to detect and recognize license plate number as well as car front / car rear with this algorithm, including the algorithm design and improvements. The data gathering, data cleaning, data labelling, model training procedures of this deep learning algorithm are described with details. The result shows that the presented detection algorithm could successfully identify the car plate numbers as well as car front or rear.

In the second part, this paper presents the methods of hardware communication. This includes computer and embedded device communication; how to improve and control non-smart garage gate, light or alarm with Bluetooth or radio frequency signals, as well as the implementations. The experiments show that, after hardware installations, the program this paper presents could control non-smart devices with ethernet remotely and wirelessly.

Following the hardware part is the graphical user interface part. This paper compares

different approaches to build a user interface for a control center. It presents a browser-server based graphical user interface design, the implementation of a server-side program, and of a user interface. The server-side program this paper writes enables users to control their garage devices with smart phones, tablets or PCs.

In the last part, this paper describes how to compose the algorithm, devices and user interface in a Linux operating system, to build a complete smart garage system that can be useful for home garage users. The presented system has been running for 18 months. It works stably at daytime. It has given people a lot of convenience.

Keywords:

Object Detection, Deep Learning, License Plate Recognition, Raspberry Pi, Ruby on Rails, YOLO.

目　　录

第 1 章　绪　论

1.1　研究背景和意义

随着经济的发展和生产力的发展，很多家庭都拥有了自己的小汽车和单独的家庭车库，驾驶员在驶入和驶出车库时需要用车库遥控钥匙开启或关闭车库门，这需要驾驶员将驾驶注意力从路面转移到存放遥控钥匙的地方，比如车的抽屉或衣服兜。注意力的转移对驾驶是一个安全隐患，同时，取遥控钥匙的过程也不方便。

尽管大型停车场和商业中心及部分单位、公司已经普及了通过识别车牌自动开关门的智能停车场系统，但这一类闸门设施既价格昂贵又不便于安装，不适合家庭车库的改装升级。因此，一套可以利用家庭用户现有设备进行低成本改进，使之成为能够识别家庭小汽车的智慧车库系统更适合于家用。

计算机硬件水平的提高为基于深度学习的目标检测算法提供了计算能力，在 Nvidia GeForce 10 系列 GPU 的帮助下，低成本实时高帧率的目标检测已经成为可能。而对于小型网站应用（Web App），近几年发售的类似树莓派三代的具备 ARM Cortex-A53[1]处理器的小型嵌入式设备可以提供足够的计算能力，其电力功耗可以忽略不计，常见供电电流为 0.8 安培，电压为 5 伏特。对于其他如蓝牙、无线遥控模块的控制，树莓派等小型嵌入式设备也有足够的 GPIO（General Purpose Input Output）接口和蓝牙接口用于连接外部设备，进行硬件之间的通信。

在计算设备和控制设备价格都不昂贵的今天，构建一套可以通用于不同人家庭的智慧车库系统可以极大的方便人们的生活，提高驾驶员的安全感和幸福感。作者认为这样的提高人们生活水平的软硬件系统对整个社会具有一定的价值。

1.2　国内外研究现状

本研究的核心问题在于实时的稍复杂背景环境下的目标检测算法，其中，车头、车尾、车牌以及车牌的数字均可视为不同的目标，完成了这些目标的识别，即可以做到识别家庭小汽车。

车头和车尾的识别对于家庭车库的意义在于，控制系统需要此信息来确定驾驶员的行驶状态为正在离开车库或是正在进入车库。在公共交通中，识别车头和车

尾的意义不大，因为在公共交通路线中，行车方向基本由所在的道路确定，逆行为违反交通规则的行为且极少发生；而在进入和驶出某公共停车区域时，停车场大部分为双闸门双车道，小汽车在正常情况下不允许逆行。因此，识别及分辨车头和车尾较少有人研究，但依然有使用 SVM[2]（Support Vector Machine）、kNN[3]（k Nearest Neighbor）等方法进行车头车尾识别的方法[4]。

对于车牌和车牌号识别的方法，国内外已经有很多[5][6][7]。大多数方法基于车牌区别于周围车身的色彩信息来进行[8][9]，比如色彩梯度，而后将分割出的图像再利用 SVM 等算法进行分类，此类方法可以稳定地识别车牌号。大多数的商用软件和开源软件也可以优秀地完成车牌识别的任务，但作者在查询文献时没有找到可以用于生产环境的开源代码：尽管这些程序可以识别一个车牌的号码，但对于图片的视角有一定要求。比如常见的商用停车场门闸系统，摄像头需要直接对着车牌，对驾驶员的行进路线和摄像头的摆设位置都有要求，而这在家用车库环境下无法实施。另外，部分车牌识别软件也无法进行实时的计算，因为图像识别的计算量对 CPU 资源消耗较大，尤其是在家庭车库外的较复杂背景环境下，有额外的计算开销，因此这些软件达不到实时识别对处理速度的要求，这些已有的软件也无法采用。

1.3　本文工作

本文原创了一种新的车牌实时检测方法，将车牌、车牌号和车身识别的问题转换为了三个不同的目标检测的问题，并使用深度学习算法一次解决三个问题，降低了复杂背景下识别个人小汽车的难度，并使识别的准确率达到可以稳定家用的水平，这里的稳定家用指的是在摄像头及计算机软件系统调试完毕后，在人类在摄像头所传图像中找到目标车辆后两秒内，自动识别系统也能令车库门进行相应动作。

为了解决上述三个目标检测的问题，本文要进行三个深度学习模型的训练和测试：

目标检测模型一：模型一为车头和车尾识别的模型。车头和车尾识别是本文进行的第一个工作，本文对从网络上各高校和研究机构公开的数据集中收集的高速路上行车的图片，以及本文作者在市内街路社区拍摄的机动车图像进行了图像的标注，将车头和车尾以方形框出，作为训练数据。而后进行训练，获得深度学习目标检测模型一。

目标检测模型二：模型二为车牌识别的模型（不含车牌上车牌号识别）。本文使用模型一所使用的训练集，同时作为模型二训练集，而后对该数据集进行标注，训练和验证。

目标检测模型三：模型三为字符识别模型（车牌号识别）。在模型二成功识别出车牌后，模型三负责模型二所切割下来的字符的检测。本文所采用的模型三对 0-9、A-Z 这 36 个字符进行识别。对于车牌号中的汉字，因为一家车库和邻居的车牌数字字母完全一致而仅省区号不一致的概率几乎为 0，故而本文没有识别代表省区的汉字。将切割下的车牌图片上的字符进行标注而不用原始大图片进行标注的意义在于：第一，车牌图片占原始图片的比例较小，在电脑屏幕全屏展开后鼠标很难点选；第二，整个系统在三个模型串联运行过程中，模型三也总是会获得模型二提供的车牌图片而不会获得整幅大图片，此举可以提高模型三的敏感性。本文会在第二章中详述此模型多次实验尝试中成功和失败的经验。

除目标检测算法外，本文的工作还包括了处理计算机与外部设备交互的问题。本文使用树莓派和弹簧天线对射频信号进行了发射接收和编解码，模拟了遥控钥匙的工作过程，将非智能的遥控门、遥控报警器和蓝牙遥控灯等设备改进成可以由计算机控制的智能设备。

本文最后完成的工作是一个使用 Ruby on Rails[10]自主编写的基于浏览器-服务器的 Web 客户端，它可以方便用户使用上述功能。

1.4　本文结构安排

下面介绍本文每个章节的主要内容：

第 2 章简述了所使用的目标检测算法，提出了家庭车辆识别算法；详述了构建车辆识别模型的过程和成果。

第 3 章首先处理了外设硬件的控制和通信问题，包括普通非智能车库门、车库灯、报警器等。接着讨论了基于浏览器-服务器的用户图形界面，及其设计和实现成果。

第 4 章描述了将前两章的算法、硬件、软件成果进行组合，成为一个可用的智慧车库系统的过程，同时介绍了系统的实验结果和配置细节，以及在不同场景中可能存在的问题及问题解决思路。

第 5 章对全文内容做出总结，并讨论下一步可能研究的主要工作。

第 2 章　基于目标检测算法的家庭车辆识别

2.1　目标检测算法实验环境的计算机硬件

识别车牌号的算法有很多种，其中既有利用色彩空间变换和色彩梯度的经典方法，又有基于 SVM 或其他数据挖掘算法的方法，本文采用的是基于深度学习的方法。

采用基于深度学习的方法，需要特别关注的一点就是计算机的计算能力。因为深度学习算法的主要计算是对图像像素点进行的加减乘除、取最值、判断大小等，对每个像素点每次运算的计算量都比较小，但因为像素点很多，以及模型的层数很多，因此总的计算量大而且对并行计算的要求很高。故而，大多数深度学习程序[11]都使用了 GPU 进行计算，而不采用 CPU，因为 GPU 一般核心数（CUDA 单元）多达 1280 个以上，家用 CPU 常为 8 核心以内。

既然本文的目标为家用的识别系统，所采用的 GPU 和计算机也应当设计得符合普通家庭能够具有的条件，因此本文最后采用的 GPU 为 Nvidia 公司的 GTX1060 6G GPU，这一 GPU 在实验完成时证明，它可以出色得完成计算任务，作者使用的 GPU 算力统计工具显示识别系统在正常工作时仅仅用了计算能力和显存的 30%左右，即 35%设施使用率（utility）和 2.2GB 左右的 GPU 记忆存储（2.2 / 6 GPU Memory）。摩尔定律预测每 18 个月计算能力就可以翻一倍，未来也有希望可以在嵌入式设备中使用本算法，但在 2018 年本文进行调研时，作者并未找到可以对 1080P 视频以 25 帧每秒（FPS）的速度进行实时深度学习目标检测的小型嵌入式设备。而对于 CPU 和内存容量，尽管本文的实验环境下使用的是 AMD 公司生产的 Ryzen 7 1700 8 核心 CPU 和 2133MHz 16GB DDR4 的内存记忆体（Memory），但实际使用时，考虑到 CPU 在本算法中主要计算任务仅仅为将摄像头传送的 1080P 25FPS 视频流搬运到 GPU 上，按照作者的估计，此配置已经远超算法所需，理论上不低于此配置 50%的 CPU（以核心数计）和内存（以容量计）都应可以出色的完成任务。

2.2　YOLO 目标检测算法简介

YOLO（You Only Look Once: Unified, Real-Time Object Detection）算法是华盛顿大学（University of Washington）的两位学者 Joseph Redmon 与 Ali Farhadi 于 2016 年在 Computer Vision and Pattern Recognition 会议上发表的一篇论文[12]，其后作者又于同年发表了改进版本 YOLO9000[13]，在 2018 年，作者又按照不同的新技巧将其方法改进[14]，撰写了 YOLO-v3 版本的目标检测算法[15]，本文所采用的算法基于 YOLO-v3，以下称为 YOLO 算法。

YOLO 算法的主要目的是检测一幅图中是否有某些目标，以及这些目标各属于哪种分类、在图像中的位置。举例来说，人们想利用计算机推测狗、自行车、人是否在一幅图中出现，以及若这些目标如果出现了，在图中的哪里。即假若将狗所在位置用方框框出，这个方框的中心距图片左上角顶点的水平和垂直相对位置、这个方框的宽和高占图像总宽度、高度的百分比，并依据这些信息在图像中定位该目标。

假定算法为函数f，输入图像为p，目标所属类别为c，目标所在位置的横坐标、纵坐标、宽度、高度分别为x, y, w, h，那么在预测过程中，算法的基本功能可以描述为：

$$f(p) = \{(c_i, x_i, y_i, w_i, h_i), \dots, (c_j, x_j, y_j, w_j, h_j)\} \quad \text{.....(2.1)}$$

其中i、j为不同的目标。

而在训练过程中，输入为若干幅训练集图像及其标注好的目标信息，输出为函数f中包括的深度学习模型特征提取器（feature extractor）各 filter 的参数。

2.3　YOLO 算法与其他目标检测算法的比较

在本文作者于 2018 年调研目标检测算法时，比较流行的算法是 Faster R-CNN 改进后的 Mask R-CNN[16][17][18]，SSD（Single Shot Multibox Detector）[19]和 YOLO。

Mask R-CNN 与 SDD、YOLO 的主要区别在于，Mask R-CNN 是像素级别的两步检测算法（Region based two-stage detector）。像素级别检测是指其可以精确的推断出每一个像素点属于哪个目标，继而将这个目标的轮廓精确的从图像中勾勒出来；而另外两种算法都是得到一个方框，其中框住想要的目标。两步检测算法是指，

第一：算法要进行目标所在位置的检测，第二：算法要进行该位置的目标是哪一类别目标的检测，即其使用了两个串联的神经网络分别发挥目标拣选和目标分类的作用。在实际算法表现的比较中，R-CNN 类算法的计算复杂度较高，它每秒可检查的帧数（FPS）在三个算法中是最少的，如果不需要像素级别的分析，这一算法实际上不比另外两种算法效果更好。

SSD 算法和 YOLO 算法类似，是一步（single-stage）目标检测算法，YOLO 的网络结构又称为端到端的网络（End-to-End），也就是一次推测将目标的位置和类别都推测出来。二者的不同点在于候选框的生成过程，SSD 预先定义好 8732 个候选框，而后在其中选择目标；YOLO 在第一版本中则只有 98 个候选框。而后均使用 NMS（Non-Max-Suppression）[20]来进行候选框的整合。

无论 YOLO、SSD 或是 Mask R-CNN，他们的准确率相差都不多，他们的 COCO mAP（COCO 训练集，mean Average Precision）都在 30mAP 附近[21][22]。但 YOLO 远优于其他两个算法的地方在于，YOLO 的速度非常快[23]，几乎是 SSD321 算法的三分之一（YOLO / SSD321 = 22ms / 61ms）。尽管 mAP 不如两步算法，但在速度上比采用两步算法的 FPN（Feature pyramid network）的 172 毫秒快近 85%。

基于此，本文认为在家用车牌识别的任务中[24][25][26][27]，权衡速度与准确率，YOLO 算法要优于其他的目标检测算法。原因是，其一：考虑到家庭计算机更新换代的周期为 3 年以上，家庭计算机的配置至少落后最新的家用计算机产品 2-3 年，如果采用了高精确度的算法而牺牲速度，将使部分计算机没有足够的计算能力来运行这一任务，因此放弃了 Mask R-CNN 类的两步检测算法；其二：YOLO 与 SSD 的准确率并无太大差距，没有必要为极小的（3%左右）mAP 差距而牺牲 3 倍的速度，并且本任务实际并没有 COCO 数据集的目标类别多，理论上无论使用哪个算法都可以完成检测任务，故而优选 YOLO 算法，次选 SSD 算法；其三：YOLO 算法已被作者开源[28]，并且有了很多程序员和研究人员的试错和调整，使得程序更加可靠，也有社区贡献者贡献了开源的文档和说明，使作者在调试和修改代码时可以节约很多时间。对比了一些其他目标的目标检测算法应用后[29][30][31]，本文选用了 YOLO 算法进行智慧车库系统的目标检测任务。

2.4　未能成功识别家庭车辆的方法及改进过程

本文将首先描述实验过程中不能完成车辆识别任务的三种失败方法，并以此指出 YOLO 算法的特点和局限性，作者认为失败的经验与成功的经验同样重要，因此将首先用较短的篇幅进行介绍，而后再在此基础上讲述如何改进此失败的方法，以得到可以成功识别家庭车辆的方法。在未能成功的方法中，涉及到的训练、标注的过程，将在后文成功识别家庭车辆的方法中详尽介绍，在此仅用训练、标注两词进行描述，二词指的是深度学习模型所涉及到的训练和标注过程。

未能成功的方法一:

第一个未能成功的方法的思想是很朴素的，每台轿车都不完全一样，比如车牌上的图形就是完全不同的，而车身颜色，车前玻璃内饰，车型更是可以分辨出不同的汽车，人类在日常生活中就是这样找到自己的汽车的。既然算法可以检测目标（object），不妨直接训练识别用户个人车辆的模型。

基于这一想法，本文拟将想要识别的家庭车辆作为唯一的目标（target），训练一个模型，不识别具体的车牌号，只将目标车辆的图片作为模型的训练集，只要能够识别出这一车辆，即可完成任务，即 YOLO 算法需要识别两类目标：目标车辆的车头、目标车辆的车尾。

作者首先收集了 252 张目标车辆的各个角度的照片，这些照片都为同一台小轿车，但拍摄的远近、角度、背景不同。而后进行标注，模型训练。

经过实验，算法完全可以检测到这一台车出现在视频流中。

但此方法的问题在于，这一模型的伪正例（False Positive）比率非常高，大部分小轿车，都会被认为是目标车辆，这是不可接受的。因此作者对训练集进行了改进，做了第二次模型的训练。

未能成功的方法二:

第二个未能成功的方法思想延续了第一个未能成功的方法的优点，并减少了伪正例。思想是：为了较少的计算量，仍然仅训练一个模型，但为了减少伪正例，需要将非目标车辆的图片也加入训练集，使模型知道什么样的车型不是目标，将非目标的特征提取出来加入模型当中，即 YOLO 算法需要识别三类目标：目标车头（target）、目标车尾、非目标车辆（non-target）。

根据方法二的思路，作者再次收集了 350 张非目标车辆的照片，进行了标注，

此时标注的内容为，在这些单纯非目标车辆的图片中，标注为类别 3 非目标车辆，即图片中没有目标。而后训练模型，进行测试。

方法二测试的结果为，伪正例的比率大大减少了。作者在几次测试中，发现和目标车辆不同颜色的车几乎不会被分为正例，但作者同时也注意到，由于目标车辆是一台黑色大众品牌小轿车，而测试中有一台同品牌不同型号的黑色大众小轿车，这台非目标的黑色小轿车经常引起模型的误判，将其识别为目标车辆。所以方法二的模型也是不能接受的。

经过上述两个未能成功的方法的实验，作者发现 YOLO 算法在权衡性能的时候，为了将正例尽可能识别出来，而使部分负例也分为了正例。因此本文认为使用 YOLO 算法仅建一个模型直接按照车辆来识别目标这一思路不可取。

放弃了按照车辆直接识别目标的思路后，本文开始尝试未能成功的方法三。方法三同时利用车辆本身的信息和车牌的图形信息来识别目标。

未能成功的方法三:

方法三的思路也较为朴素：因为不同车辆的车牌图形是完全不一致的，那么目标检测算法应当可以将不同的车牌视作不同的目标，而要达成本文的任务，只需要将车牌分为目标车辆车牌和非目标车辆车牌这两类，辅以方法二中的车辆信息，即仅当目标车辆的车头或车尾和目标车牌同时出现，才视为检测到了目标。也即要使用 YOLO 算法进行四种目标的分类：目标车头、目标车尾、非目标、目标车牌。

按照方法三的思路，本文重新标记了训练集，将目标车辆的车牌在图片中进行标注，在其他图片中，非目标车辆的车牌不进行标注，总图片数约为 1000 张，其中 252 张为目标车辆，其余为非目标车辆。而后训练新的深度学习模型。

方法三的实验结果为：无法在高置信度下检测到车牌。

摄像头拍摄的视频为 1080P 分辨率，因为摄像头摆放需要离行车道较远（约 10-15 米），视距较远，所以车牌在图像中所占的比例很小，宽度为视频水平宽度的 1 / 15 左右。这一现象主要会导致两个问题：第一，无法在一幅图中同时稳定的识别到车头尾和车牌这两个类别。算法经过改进，设计为在 5 秒内同时识别到车头尾和车牌即视为识别到目标车辆，改进后的算法可以解决绝大多数视频流中的此类问题，但降低了识别的敏感度。第二，有时摄像头视野内会出现其他车的车头尾和目标车的车牌，且与目标车辆的行进方向相反，算法会同时检测到目标车辆的车牌，以及反向了的车头尾，导致了离开车库的车辆被视为即将进入车库的车辆，而

对车库门进行反向误操作。

以上两点问题都令方法三稳定性下降，使得此方法无法可靠地分辨目标车辆。

在三次尝试后，本文放弃了使用一个模型直接识别目标车辆的方法，改用多阶段的识别方法。

2.5　成功识别家庭车辆的方法：三阶车牌识别

从本节开始至本章结束，详述本文所提出的基于目标检测算法的家庭车辆识别的方法。

总结上文的实验条件和结果，可以得出以下结论：

1、 仅使用车的图形作为模型训练集不能有效地将目标车辆和其他车辆区别开。

2、 车牌由于尺寸较小，占整幅图的比例小，不易于从图片中识别出来。

3、 为了识别车牌而增加神经网络的层数将使计算量提高，会降低计算速度。

4、 仅识别车牌图形而不识别出车牌具体的数字，将有伪正例的风险，使车库门误动作。

5、 不能单纯地根据图像上是否同时出现了目标车头尾以及目标车牌来判断目标车辆，而是需要目标车牌必须包含于目标车头尾所在的图像区域中。

基于上述结论，本文最终设计了如下的检测思路：

将车辆识别的过程分为三步，每步单独训练一个目标检测模型，单独负责一项目标检测任务，并将前一个模型的检测结果（输出）作为后一个模型的输入图像，继续进行下一步检测。同时保留中间结果，提取出车辆的车头车尾、车牌号信息，作为判断是否是目标车辆的依据。

第一个模型是车头车尾检测的模型，此模型负责从摄像头传来的视频流中检测出每一帧是否包含汽车的车头或车尾图像，对于包含车头或车尾的图像，将它具体的位置推断出来，并按照 1.15 的比例扩大候选框后把这一部分图像从输入的视频流图像中剪切下来，作为第二个模型的输入图像传给第二个目标检测模型。第二个模型的功能是，检测模型一传来的车头车尾图像中是否有车牌，如果有则从车头车尾图像中找到具体的位置，并按照 1.15 倍的比例扩大候选框后把车牌的图像剪切出来，并将切下来的应当包含车牌的图像传给模型三作为模型三的输入，如果没

有则返回。模型三是字符识别的模型，它将字符作为目标，对模型二传来的车牌图像进行检测，识别车牌图像上的字符，以达到识别车牌号的目的。

而后根据模型一和模型三所提取出的车头车尾与车牌号的信息，判断视频流中的车辆是不是目标车辆，如果是，迎着摄像头的是车头还是车尾，并依据这两点信息控制车库门的开关信号。

2.5.1　目标检测模型一：车头车尾识别

本节讲述三阶串联目标检测模型中的第一个模型，用于车头车尾识别的模型。

车头车尾识别模型作为视频流最先经过的模型，其功能是识别视频流中是否存在车辆，如果有，还需分辨出面向摄像头的是车头还是车尾。此模型为后续车牌和车牌号的检测提供了基础，将会降低后续模型的计算量和分类难度，同时也提供车库门应当开启还是关闭的信息。

识别的方法为 YOLO 目标检测算法，本节按照数据收集、标注、训练、测试的过程进行讲述。

数据的收集：

本文模型一的数据集（车辆图像）主要来源于作者自行拍摄的照片和雅典国家技术大学（National Technical University of Athens）多媒体技术实验室公开的科学数据集[32]。数据包括 212 张自行收集的属于目标车辆的照片，333 张自行收集的各种家用轿车的照片，以及 547 张 NTUA 公开的外国车辆的照片，共计 1092 张国内外车辆照片，这些照片均可由人眼分辨出其中的汽车。如此选择数据集的好处是，第一，本文的目的是为家庭车辆做检测模型，多选用目标车辆的照片能够提高识别的准确率，第二，选用国内外不同车型的照片可以提高模型的广泛适应性，让模型能够识别各种车型。

数据的标注：

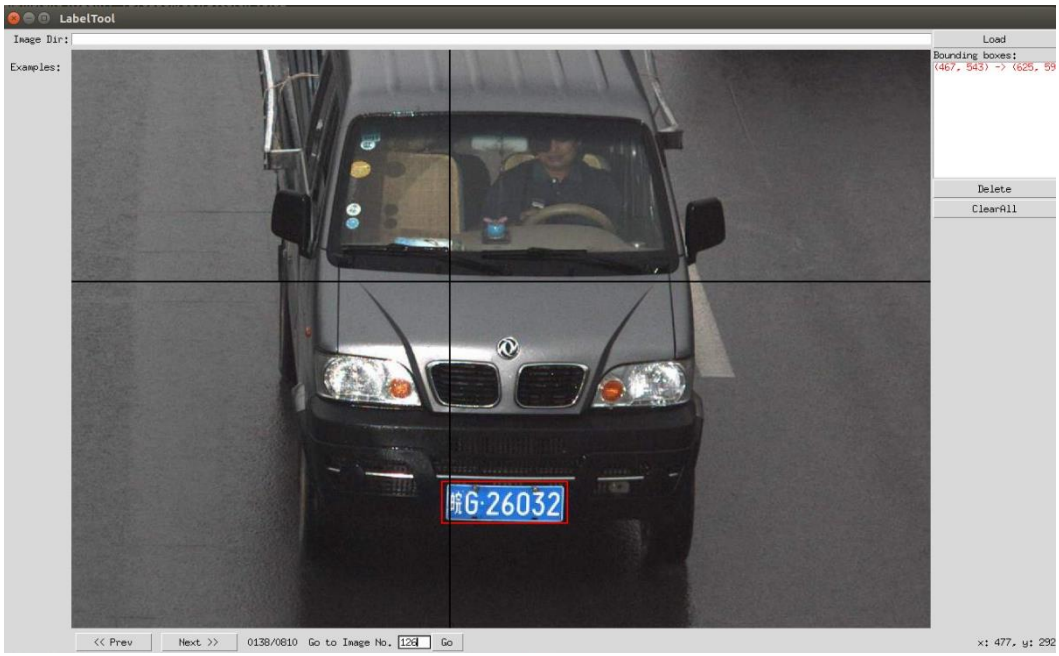

图 2.1　本文使用的图片标注工具 Yolo-Mark

YOLO 算法训练模型时，需要对于每个目标需要标注 5 项数据，分别是：类别（以整型数字代替），目标中心距图片左上角顶点的水平距离、垂直距离以及目标宽度、高度占整幅图片水平、垂直长度的百分比，距离数据均为 0～1 之间的小数。在模型 1 中，将车头标注为类别 0，车尾标注为类别 1。为了标注方便，作者使用了开源工具 Yolo-Mark[33]，如图 2.1 所示。此工具可以以图形化界面使用鼠标来选中目标所在的位置，而后由工具自行生成数据标注文件，若图像中有多台车辆，则可依次标注，生成的标注文件中以另起一行视为另一个目标；如果没有车头或车尾出现在图像中，则生成对应图片文件名的空文档。作者共标注了 1092 张图片作为模型一的训练集，既有车头和车尾单独的图片，又有车头和车尾同在一张出现的图片，作者并未统计共有多少张车头的图片与多少张车尾的图片。

模型的训练：

深度学习模型的训练使用了 YOLO 作者 Redmon 开发的使用 C 语言编写的 Darknet 深度学习框架和开源的 YOLO 算法代码，在 GTX-1060 6G GPU 下训练了约 30 小时，模型以 tiny-yolo-v3 作为基础。作者在训练时并未使用验证集来关注提前终止点（Early Stopping Point），仅手动调整学习率（Learning Rate），观察训练集的损失函数值小于 0.1 且 IOU（Interception over Union）值大于 0.8，或二者趋于不变时停止训练。

每次停止训练后，使用 19 幅全新测试图片手工测试模型的效果，没有进行批

量的测试或计算 mAP。当 19 幅图中有模型不能识别的车辆但人眼可以识别的，根据损失函数值调整学习率继续训练模型，视情况增加或降低学习率，一般情况下，首先将学习率提高 5 倍，最大为 0.01，观察损失函数，若损失函数抖动较大，说明此时学习率过大，再降低学习率至调整前的数值，而当损失函数已经较小时，将学习率降低到 0.001，观察损失函数的抖动，最低降低至 0.0001，不可再降低以防止过拟合。

模型一的结果：

模型一训练完成后，利用训练集图片进行了测试，结果是模型可以识别出每张图中所有车头和车尾，而后单独取 19 幅全新测试图片进行测试，结果是模型可以识别出全部的车头和车尾；对于没有车头或车尾的图片，模型也没有发生误分。

虽然模型在训练集表现得比较好，但不能依此认为模型可以准确识别出各种车型的车头和车尾，然而本文并未就此进行深入探讨，因为作为家用的识别车辆的方法，只要训练集中有足够的目标车辆（target vehicle），实际得到的模型可以准确识别目标车辆，就可以满足需求。再次提高泛用性需要做更多的工作，收集更多的训练数据集，做更长时间的模型训练，理论上可以得到更好更具泛用性的模型，但本文认为所得模型已经足够，不再继续改进此模型。

至此，本文的算法已经可以从图片中检测到车头和车尾的存在，并将车头和车尾的图片单独剪切下来，送到模型二中作为模型二的输入图像，尽管在后期实验中发现并不是每一帧的图像都可以检测到车头和车尾的存在，但对于每秒 25 幅图像的视频流（25FPS），算法可以保证每秒 25 幅图像的视频流中至少有 10 幅图像可以检测到车头和车尾，这为实时识别车牌提供了足够的信息。结果见图 2.2。

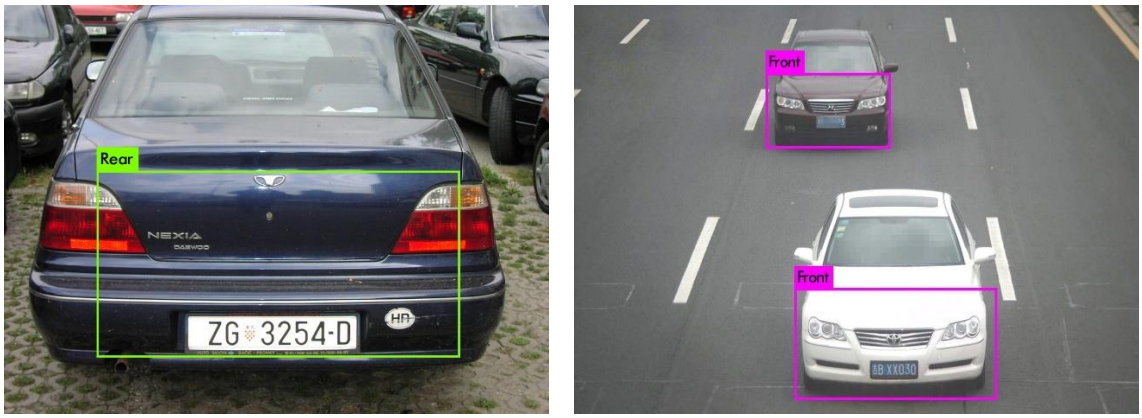

图 2.2　车头车尾（模型一）识别结果

2.5.2　目标检测模型二：车牌识别

模型二是一个独立的车牌识别模型，它的作用是识别车牌及其位置（但不识别车牌号）。在本文提出的车辆识别算法中，模型二将作为模型一的下级，将模型一的输出作为自己的输入，识别输入图片中是否有车牌存在，以及车牌在图像中的具体位置。尽管模型二仅作为模型一的下级使用，但实际上它可以用于任何场景中，但为了保证速度和减少计算量，需要整幅图中车牌图形占据的尺寸比例不能小于10%（以水平宽度计），才能有较好的识别效果，模型二难以识别远处较小的车牌目标，只能识别离摄像头较近（10 米内）、占图像百分比尺寸较大的目标。

数据的收集：

模型二的数据采用了中国车牌和外国车牌两类车牌。中国车牌数据集中有 940 张为互联网上找到的高速路摄像头拍摄到的机动车图片，有 33 张为作者采集的目标车辆图片，有 130 张为中山大学智能交通研究中心共享的科研数据库机动车图片。外国车牌数据集为 200 张来自克罗地亚萨格勒布大学 2003 年公开的车牌识别计划数据集[34]。共计 1303 张图片，如图 2.3 所示。

图 2.3　模型一的识别结果将作为模型二的输入图像

数据的标注：

和模型一标注过程类似，作者依然使用了 Yolo-Mark 工具进行数据标注。本数据集需要标注的只有车牌这一类数据，因此，按照模型一的格式，车牌标注为数值 0，其余四项依旧为水平、垂直距离占比，宽度、高度占比。一幅图有多个车牌的，

在文本文档中按行分别记录，其他目标不标注，没有车牌的图像不标注。

模型的训练：

同模型一的训练过程，仍旧以 tiny-yolo-v3 作为模型基础，将基础模型的网络参数进行迭代计算，训练为可识别车牌的模型；同时关注训练集的准确率和损失函数值以及 IoU，视损失函数值的变化情况，每隔数小时暂停训练，转为识别车牌的实际测试，而后按照测试结果决定应继续训练、调整学习率继续训练或终止训练。

模型二的结果：模型二在训练集的准确率达到了 100%。10 张测试图片中能全部识别，但因为可标注的数据集有限，没有预留更多测试集。模型二在框选车牌位置时，经常出现整体向右侧偏移车牌宽度 10%左右的距离的问题，即车牌的左侧未在方框当中，而方框的右边界又超出了车牌右边界，这是本模型的不足之处，作者认为需要重新计算 YOLO 算法中划分网格的参数，在训练时调整该参数，才能解决此问题。但是由于车牌号识别算法本身并不需要严格界定此边界，只要全部车牌号的字符都在方框内即可，因此本文对剪切方框的边界人为扩大了 15%，以保证所需字符必定会出现在即将被剪切的车牌图像之中，以便传入下一级车牌号识别模型进行检测。结果见图 2.4。

图 2.4　车牌图形识别结果（模型二）

2.5.3　目标检测模型三：车牌字符识别

模型三是车牌字符识别模型，是车牌识别三阶串联目标检测模型中最后的一

个，同时也是复杂度最高的一个。模型三将模型二输出的车牌图片作为自己的输入，经过模型的识别以后，将识别出的字符及各字符在车牌图片中的位置方框、类别作为输出。

相比于标准的深度学习建模过程，本模型在建模时对两处进行了调整改进：第一，模型使用了分阶段训练方法，牺牲了训练的时间，但降低了手工标注数据的工作量；第二，模型在目标检测过程中，使用分类数据集作为检测数据集，极大降低了手工标注数据的工作量，同时提高了模型分类字符的准确率。另外，模型对字符检测结果使用了经过设计和改进的纵坐标保序排序算法，降低了车牌图像偏斜对识别造成的影响。

数据的收集：

本模型要分的类别极多，为 0-9，A-Z 这 36 个字符中除去英文字母 I 和 O 所余下的 34 个字符，因此如何减少误分类是一个较大的挑战。为了能够达到位置检测和字符分类的目标，需要大量的数据进行模型训练，因此标注数据的工作量也很大。另外，如果直接使用高速路上的整幅图片作为字符识别的训练集，就需要在整幅图片中标注出字符的位置方框，但这很有难度，因为每个车牌字符只占车牌尺寸的不到 15%，而车牌只占图像上汽车尺寸的 10%左右，车辆占高速路摄像头所拍摄的整张照片宽度不足 30%，所以要在这么小比例的图片中标注出字符目标的方框，很容易操作失误，也很难操作。

本文为解决上述问题，考虑了本文车牌识别算法的特性，即模型三实际上总是在识别模型二所传入的单独的车牌的图像。利用这一特性，模型三的目标检测环境要宽松许多，因此作者决定仅标注单独车牌的图片上的字符，而不考虑其他背景环境的图片数据。

因为互联网找不到公开的纯车牌图像的数据集，作者直接使用本文所建的模型一和模型二生成了 1385 张车牌图像数据供研究使用。生成车牌图像的方法是，将模型一所使用的训练集作为数据集，在此数据集上先使用模型一检测出车辆的位置，而后将车辆所在位置的局部图片剪切下来送入模型二中，由模型二识别出车牌所在位置，并将车牌图像从汽车图像中剪切下来，就得到了单纯只有一张车牌的图像的数据集。全体车牌数据集共包括 1385 张车牌图片，和 20 张整幅的汽车图片，共 1405 张。

解决了目标位置检测数据集的问题后，还需收集多目标分类问题的数据集，因

为作者经过计算，1405 张图片，按照中国车牌平均每张车牌 6 个字符，总计约 8400 个字符的图片数据，按照 34 个字符在车牌中平均出现的假设，每个字符约有 247.05 项训练数据。

按照作者的经验，要用 247 个样本识别一类图形是很困难的，每个字符 247 个训练图片所得到的深度学习模型准确率将不够高，虽然“字符在车牌中平均出现”这一假设与实际情况是有出入的，但无论哪种字符的出现频率提高，都会使另一种字符减少，因此实际上分类准确率最差的字符将总是差于平均分类准确率，从而不影响“在这些数据下训练出的模型分类准确率不足”这一结论。

根据以上分析，作者针对提高字符分类的准确率进行了思考，首先，训练集必须是大量的数据，每种字符必须都出现很多次；其次，大量的数据将需要进行大量的标注工作，尽管有 Yolo-Mark 这一辅助标注工具，工作量依然巨大，按照仅考虑 24 个英文字符，每个字符 800 张图片，标注一个字符 5 秒钟计算，约需要 26.67 小时注意力集中的工作。

因此，本文在继续收集数据时，考虑并结合了 Andrew Ng 在机器学习课程[35]和 Redmon 在 YOLO9000 论文中分别提出的观点，以求减少训练模型的工作量、提高检测准确率和分类准确率：

Andrew 提出，人们想进行目标检测，而目标检测数据集标注又很困难，为了解决目标检测数据集量小、难以标注的问题，可以对分类数据集加上一些填充（这种填充可以是随机的）边框（Padding），因为已经知道是如何填充的了，也知道原目标的位置，就相当于生成了目标检测数据集。由于公开的目标检测数据集和分类数据集的数据数量相差一个量级（分类数据远多于目标检测数据集），那么就可以按此方法大量的制造标注好的目标检测数据集了。

Redmon 提出，可以使用分类数据集作为目标检测模型的训练，由标注好位置的数据提供定位信息，再由分类数据提供分类信息。Redmon 以此方法训练了一个识别 9000 种不同物体的目标检测模型，发表了 YOLO9000 论文。

本文认为尽管 YOLO9000 的效果并不尽如人意（22%左右的 mAP），但是可以结合其思想和 Andrew 提出的填充（Padding）的思想，创造新的目标字符检测数据集，来做字符的识别。

根据以上思路，作者使用了英国萨里大学（University of Surrey）公开的英文字符数据集 Chars74k[36]，这个字符数据集包含英文字母（大写和小写）和阿拉伯数

字，作者取了其中 24 类大写英文字母，共 19407 张图片，每个英文字母约 600-800 张图片。

对数据集进行扩充之后，总计收集的数据扩展到了 20812 张图像，其中手工标注车牌图像 1405 张，如图 2.5，其余 19407 张为单字符图片数据集，如图 2.6。

图 2.5　车牌图形识别结果作为模型三的训练集

数据的清洗：

不同于模型一和模型二，作者在模型三建模前额外对数据集进行了清洗。因为英文字母写法很多，有些写法不太常见，以人眼观察无法做出它是哪个英文字母的判断，同时作者认为他们和车牌号上的字母图形也相去甚远，因此需要对这些数据进行清洗，将它们从数据集中去除。

数据清洗的过程中，作者完全通过人眼观察来判断图片是否合格，作者认为没有更好的办法可以剔除不合格的数据，因为删除图片工作量不大，只需看见图片形状和常见的不符即可删除，但用异常检测等算法来检查，工作量和准确率都无法估量，因此人工挑选是最适合的方法。作者剔除的图像已经不包含在上文提到的 19407 张图片中。

数据的标注：

根据前文的分析，本模型共应对 1405 张车牌、19407 张字符总计 20812 张图片的 34 类目标进行定位和分类标注。

这个庞大数据集的标注过程有三个问题：

1、所使用的数据标注软件 Yolo-Mark 在操作过程中需要选择所标目标的类别，对于数字，这一问题较好解决，可以用键盘的数字 0-9 与类别 0-9 一一对应（类别 0 是数字 0，类别 9 是数字 9），但字母 A 是第 11 类（类别 10），就需要用手在输入框中键入 10，然后再标注位置，对于 A，只需要计算 9+1，即前十个数字后的第一个英文字母，但对于 K，O，Q 等字母，每次都要计

算他们属于第几类非常的麻烦，比如字母 K 是第 11 个英文字母，加上数字 10 个（类别 0 至类别 9），应当是类别 20；字母 O 是第 15 个英文字母，不考虑删除 I、O 字母的情况下，应是类别 24；但如若考虑不算字母 O（因为与数字 0 相似而排除出了数据集）与 I，那么在计算字母 Q 的时候，要考虑 Q 是第 17 个字母，但删除了前面的 O 和 I 这 2 个字母，加上 10 个数字，17-2+9=24，字母 Q 应为类别 24。这个计算非常麻烦，假设要标注 1000 张图片，每个图片有 3 个字母，即使使用查表法，每次查表并在标注程序中键入对应类别编号也要 5 秒，这将会是约 4 小时的额外工作量。

2、标注 1405 张图片，每张 6 个字符，将是 8430 处记录，假设每处标记（选类、标位置）耗时 8 秒，将是约 18 小时的工作量，这也是很大的。

3、余下 19407 张字母图片的训练集，都是一张方正的图片中存在一个字母，按照每张图片选类别、标注加切换共计 10 秒计算，将是近 50 小时的工作量。

针对如何解决数据标注工作量大的问题，本文提出了三点想法，分别解决上述三个问题：

对于问题一，本文首先对算法和算法的分类效果进行了权衡，本文认为如果一个目标在训练集中从未出现过，那么它在 Yolo 算法运行时也不会出现，因此，尽管模型三需要对 34 类目标进行分类，但加上字母 I、O 两类，也不会影响模型对其他字母的分类效果。基于此，本文实际的模型中要检测的有 36 个类别，即数字 0-9 和字母 A-Z，但其中字母 I 和 O 没有训练集存在目标。变化后的模型中，字母 A、K、O、Q 将分别是类别 10、20、24、26，即 9 加字母在字母表中的位次。

而后，作者对 Yolo-Mark 开源标注工具进行了功能改进，使其在标注过程选择目标分类时，不仅可以直接按键盘 0-9 数字键标注类别 0-9，还可以按键盘的字母键 A-Z 直接标注类别 10-35。这样每个类别选择分类的手工操作时间直接减少到了 1 秒钟，比之前的 5 秒减少了 80%，极大提高了标注的速度。

对于问题二，作者实验室的学术讨论班的讨论中，导师曾提出可以考虑让模型自己进行标注，以生成更多的训练数据集，讨论的原始问题虽然不是字符模型标注，但却可以用在本文的数据标注场景下。本文处理问题二的思路是：考虑到本模型的目的是标注字符，如果使用部分训练集将模型训练好了以后，可以用训练好的模型标注剩余的训练集的字符，那么就可以减少一些额外的标注工作。

例如，使用 600 张图片仅进行车牌图片中的数字的标注，训练好模型后，用生成的模型对余下 805 张图片的数字进行识别，对于这 805 张图片经过自动标注的结果，再修正其中存在标注错误的图片。这样，全部 1405 张图片的数字就都得到了标注。

最好的情况下，其中 805 张可以完全没有人工的参与，只是牺牲了计算机训练的时间。而后仅需标注英文字母即可，标注英文的过程也可以如法炮制、分批操作。作者将这种标注、训练迭代进行的方式称为分阶段训练方法，分阶段训练可以以牺牲计算机计算时间的方式减少人工的工作量。

图 2.6　英文字符分类数据集作为模型三的辅助训练集

对于问题三，作者分析了数据集的特点，即一张方形背景图片中心有一英文字符，又考虑到这些分类数据集的作用主要是提高分类准确率，而卷积神经网络提取的定位信息主要由另外 1405 张图片提供，本文决定基本上按照图片的边框向内缩进几个像素点形成的边框作为目标的定位框。

作者编写了脚本对全部字符图片都标注为（C，0.5，0.5，0.985，0.985）。这个向量表示：图片的分类为各字符对应的类别 C、目标的中心就是图片的中心（0.5，0.5）、目标的宽度和高度略小于图片本身的宽度和高度（满值为 1），为图片宽度和高度的 0.985 倍（0.985，0.985）。处理后，50 小时的工作量降低到 30 分钟，尽管标注的方框可能会稍微大一点，但作者认为这个数据集的字符几乎是填满了整张图片的，所以牺牲掉的信息量可以忽略不计。

模型的训练：

模型训练过程采用了上述问题二解法中所提到的分阶段训练方法进行迭代训练。

首先作者标注了 589 张从模型二识别并剪切下的车牌图片的数字，含英文字符的车牌仅标注其中数字的部分，不标注英文字符的部分。训练好 YOLO 算法模型后，使用此模型标注余下 816 张车牌图片中的数字，其中不能自动识别出的数

字和识别错误的数字进行手动更正。这样全部 1405 张车牌图像的数字标注完成，结果如图 2.7。

接着作者手工标注了 1405 张图片中的英文字符，其中 816 张图片是从模型一在中山大学数据集识别出的车辆图片中，使用模型二检测并剪切下的车牌图片。余下 589 张为上步标注好数字的 2003 年的高速路数据集。标注好的 1405 张图片和英文字符分类数据集的 19407 张图片一同进行 YOLO 模型的训练，考虑到车牌是宽度比高度大，而且字符不会进行镜像翻转或上下翻转，因此在训练时将 YOLO 算法的一些训练参数进行了更改，关闭了随机翻转图像的功能，也把网格宽度设为了高度的 2 倍（480 × 240分辨率）。

图 2.7　模型三第一阶段训练结果（数字识别）

模型三的结果：

模型三作为车牌字符识别和检测的模型，经过测试可以很好的识别国内车牌中的数字和英文字符，作者测试了 20 张较标准的（车牌水平，图像较清晰）车牌图片，全部字符都可以识别出来，但少部分字符位置框与实际位置有所偏差。对于国外车牌，因为有些字符和国内字体不同，不能全部识别出，本文认为这是数据集的问题，如果增加对应国家车牌的数据集，可以解决这个问题。尽管在外国车牌上的效果（图 2.8）不如中国车牌，本文认为这一模型的效果已经可以用于国内家用车牌识别，见图 2.9。

图 2.8　外国车牌字符识别模型结果

图 2.9　中国车牌字符识别模型结果

2.6　完整车牌识别算法

车牌字符检测出后，需从中提取出车牌号的信息。YOLO 是目标检测算法，但并不会按照水平或垂直坐标的顺序输出其所检测到的目标，也即模型三输出的字符仍需进行排序才能整合为车牌号。

如果车牌是一行字符，可以只对输出各目标的水平中心坐标 X 进行排序，顺序为从小到大，即可得到具体车牌号，因为车牌字符很少，所以对排序算法的特性并无要求，无需考虑复杂度和保序性，本文采用了快速排序。

如果车牌是两行字符，则需要按识别出的各字符（各目标）的水平中心坐标 X 进行排序，而后对排好序的各目标再按垂直中心坐标 Y 进行排序，顺序均为升序。这样做的结果是，较上方的一行字符会排在下方一行字符的前面，同一行的字符按从左至右的顺序排列。需要注意的是，两行字符的排序在进行 Y 坐标排序时，需要使用具有保序性的排序算法，以保证排序后水平坐标依然是有序的，因字符数较少，仍然无需考虑算法复杂度，本文在此采用了归并排序算法。

但此方法存在一个问题，因为车牌图片受路面和摄像头摆放角度的影响，可能在显示时处于倾斜状态，倾斜状态的车牌的纵坐标排序会出现问题。这一问题主要体现在，如果以车牌中心为原点，车牌图像是顺时针旋转的偏斜，那么排序时无影响，如果车牌图像是逆时针旋转的偏斜，那么排序时会产生错误。此错误是：由于

车牌图像逆时针旋转偏斜，即最左侧的字符低，最右侧的字符高，高的字符 Y 坐标小，在按 Y 坐标排列后会出现在字符串的首位，但最右侧的字符应当是最后一个字符。要解决这一问题，本文提出，为确定哪一个字符在上一行、哪一个字符在下一行，不对 Y 坐标进行严格比较，而是比较两个 Y 坐标的差值是否小于某一范围，即：

$$y_1 < y_2 + \frac{1}{4}h \quad \text{.....(2.2)}$$

其中h是识别出的字符的高度。

满足这一条件，才视为某一个字符的纵坐标小于另一个字符的纵坐标，即改变了归并排序算法的比较条件，再进行归并排序。

修正后的车牌字符排序算法可以识别出不同行数的车牌号，而且对以一行字符为主的家用小汽车仍然适用，在不同的应用环境下，使用者也可以调整这一差值以适应所在环境的车牌偏斜角度，这一调整无需重新训练模型。

完成车牌号序列重组后，算法就全部完成了。模型一可以提取车头或车尾的信息，模型二识别模型一给出的车头或车尾的车牌图像，模型三从模型二识别出的车牌中提取车牌号序列，即得到了完整的车牌识别算法。

实际使用时，程序应当以视频流作为模型一的输入，仅当模型一识别到车头或车尾时，将剪切下的图片输出，同时保存车头或车尾的判断结果到内存中作为可能用到的输出；模型二和模型三的输入将是图片，模型三的输出仅有车牌字符串。整个程序的输出是模型一的判断结果和模型三的车牌号字符串，仅当模型三识别到了 6 位字符的字符串时，才认同其为有效输出。

算法结果：

在本文设置的硬件环境中，算法可以对视频流进行实时的检测，无延迟或卡顿。在车速平均 5 米/秒，由远及近再由近及远行进 40 米的一段测试录像中，总计可以识别出车牌号 60 余次，其中车头识别出 40 余次，车尾识别出 20 余次。可以满足实时家用车辆识别的要求。

作者对另一端随机摄录的视频进行了车辆识别和车牌号识别，结果如图 2.10-2.14 所示。

图 2.10　家庭车辆识别模型结果（一）

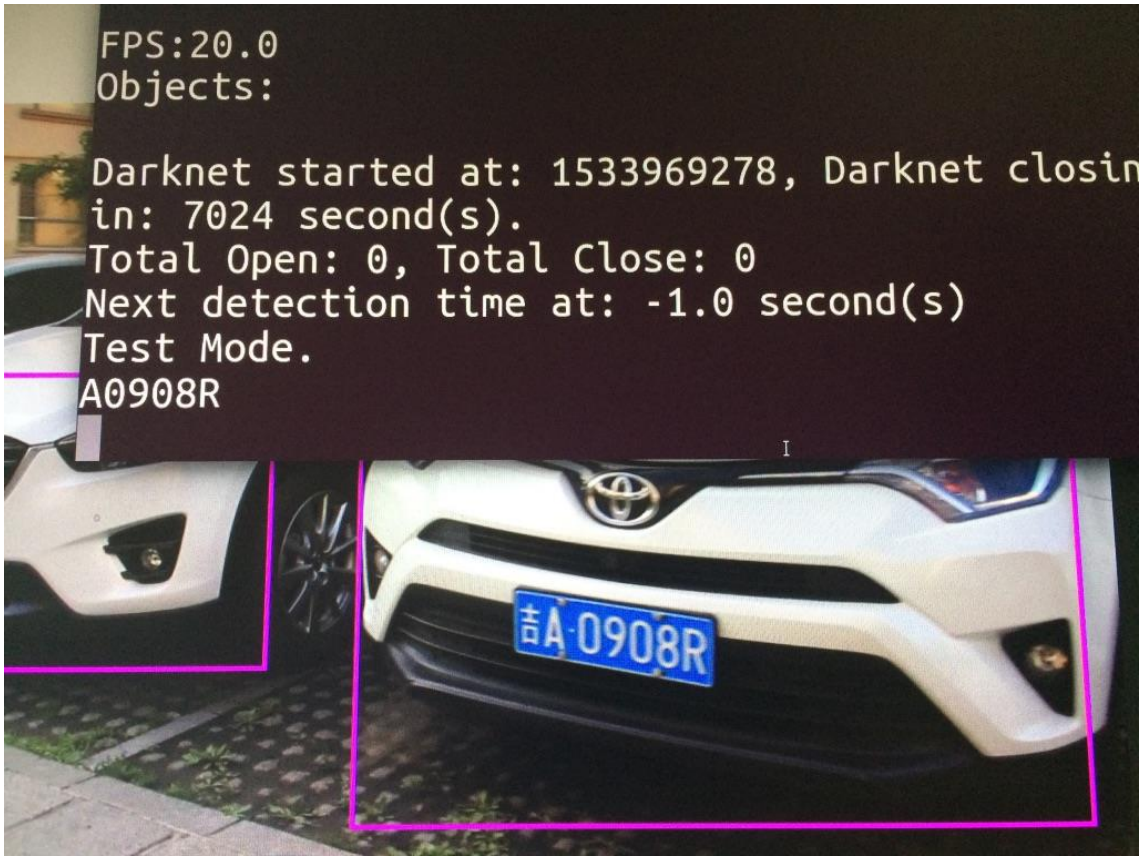

图 2.11　家庭车辆识别模型结果（二）

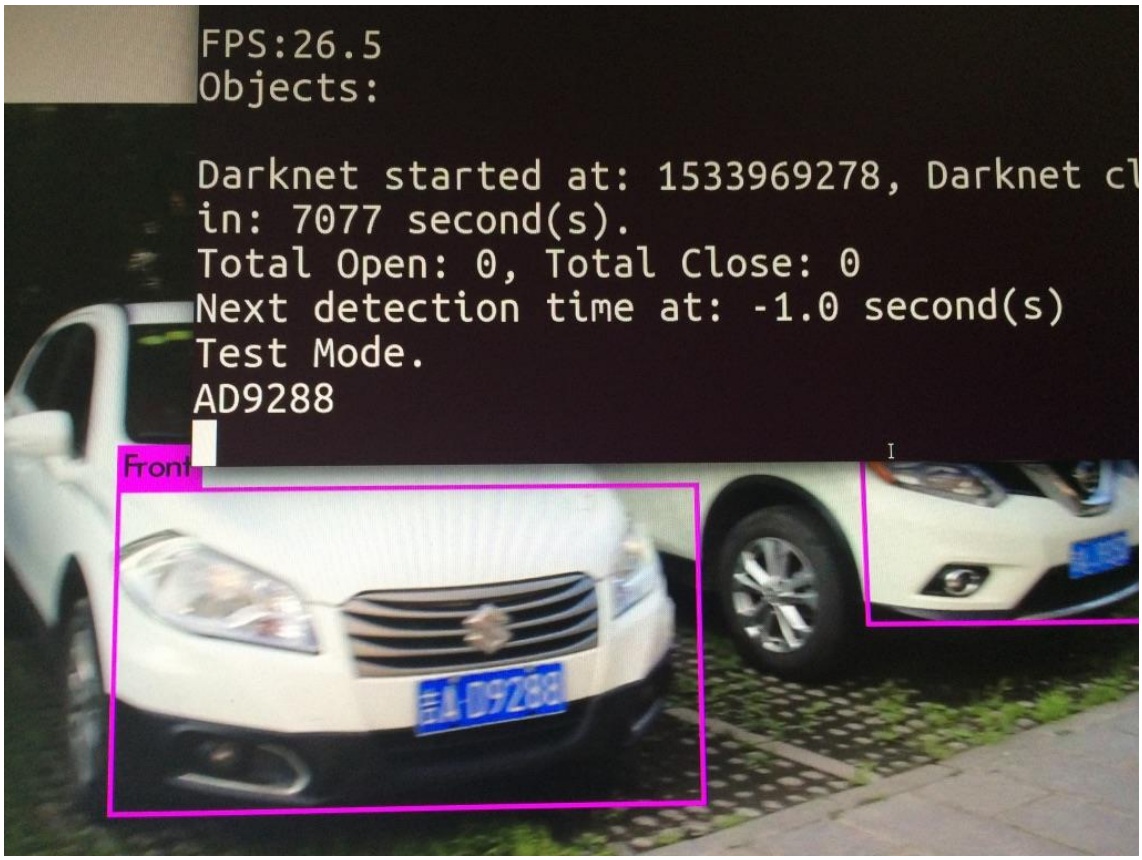

图 2.12　家庭车辆识别模型结果（三）

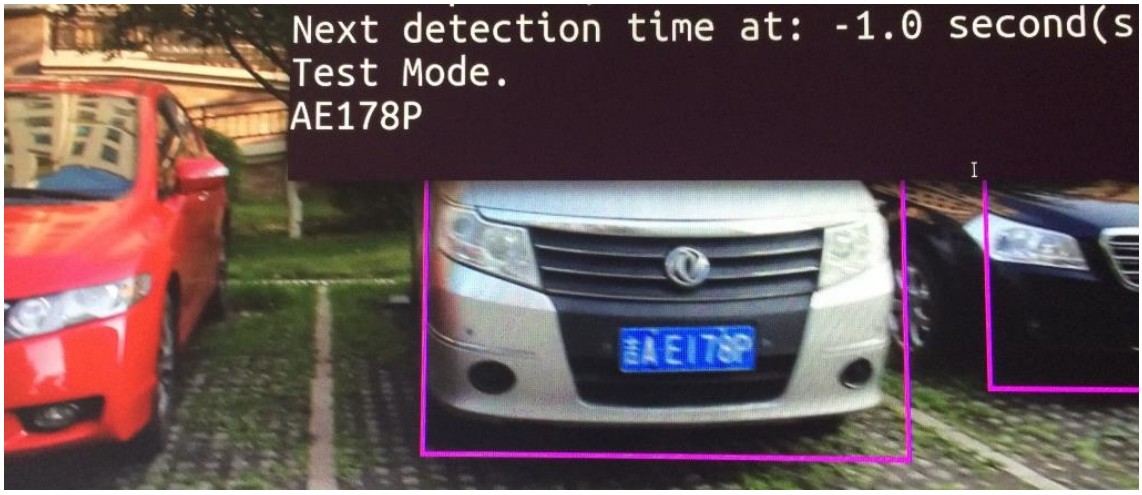

图 2.13　家庭车辆识别模型结果（四）

图 2.14　家庭车辆识别模型结果（五）

2.7　本章小结

本章主要讨论了不同的车牌识别算法、目标检测算法，提出了新的基于目标检测算法的实时车头车尾识别算法、车牌识别的算法、车牌字符识别算法，基于上述三个算法，本章组合它们形成了家庭车辆识别的算法。同时本章对数据标注的方法进行了改进，减少了数据标注的工作量。实验结果显示本算法可以完成家用车辆识别的任务。

第 3 章　硬件设备与用户界面

在掌握了家庭车辆识别的方法后，要根据算法的识别结果控制家庭电器设备，还需要进行下一步的工作。

本章前半部分研究了将遥控车库门、车库灯等家庭常见车库硬件设备改造成可由计算机控制的设备的方法，以便由检测系统识别家庭车辆后发出指令对其进行控制。其中讨论了对硬件设备进行控制需要的嵌入式设备、外设模块，以及控制方法和实验结果等。

本章后半部分主要关注系统的易用性，主要涉及基于浏览器-服务器的用户界面设计、实现与成果。

智慧车库系统是否廉价、稳定、易用是本章在进行设计和权衡时的主要原则。

3.1　硬件设备

本节讨论与智慧车库所用硬件相关问题。

3.1.1　基本目标和思路

作者认为，车库的改造不应脱离人们的现实生活，不能以高价格、高技术要求、高耗能作为改造的必要条件，而是应尽可能的廉价、稳定同时又便于施工、易于使用。因此，要尽可能的在家庭已有的设备上改进，比如不重新安装车库门、不重新布置电路，不做不必要的工作。

在这一目标下，本文对硬件设备的改造思路如下：

可以利用单片机或其他嵌入式设备来发出控制信号对各种设备进行控制。比如：模拟车库门的遥控器工作的物理过程，使嵌入式设备连接射频信号模块发射与遥控器同波形的信号，以控制车库门等受射频无线控制器控制的设备；再比如可以将车库灯的墙壁开关改造成单火线结构的蓝牙开关，用嵌入式设备的蓝牙模块根据墙壁开关的协议控制车库灯和其他蓝牙协议的设备。

本文不考虑如小程序、基于某一智能设备平台的解决方案等，因为它们都有机会将用户的个人信息泄露给第三方机构，可能对用户造成的损失。

而为了达到易用性，本文考虑了终端命令行、手机原生 APP、Web APP 三类

用户界面。终端命令行的优点是方便编写，不需要设计和实现图形界面，缺点是对于非专业人士较难使用；手机 APP 的优点是，可以充分发挥手机的性能，作出令用户喜欢的特效，开关速度快，用户使用体验最好，缺点是要处理安卓和 iOS 两个操作系统，还要适配不同的屏幕尺寸和机型，工作量比较大，需要分发应用程序、需要用户使用前下载；Web APP 是基于浏览器和服务器的用户界面架构，通过 HTTP 协议进行通信，显示界面接口已由浏览器定好，优点是无论何种手机、电脑、平板电脑等个人设备都可以使用，只要有浏览器并连入了网络就可以做出想做的控制，不盗取用户信息，也不需要下载可执行文件，安全性有保证，缺点是没有原生手机 APP 流畅，用户体验差于手机 APP。

本文决定结合终端命令行和 Web APP 两种用户操作界面。放弃手机 APP 的原因是开发手机 APP 的工作量太大，但如果不考虑工作量，手机 APP 是很好的选择。对于常用的功能且实现不复杂的，本文使用 Web APP 完成，对于不常用的功能或者开发复杂的，使用终端命令行以开源软件完成。

3.1.2　嵌入式硬件设备

为了控制各种外设，本文采用了树莓派作为控制信号的收发设备。

树莓派，如图 3.1 所示，是一种小型的嵌入式设备，价格低廉（约 30 美元 / 片），可以运行 Linux 操作系统，可以以有线网络或无线网络与家庭网络进行连接，配有蓝牙、相机的接口，以及 20 余个通用输入输出接口（GPIO），非常适合作为各种模拟、数字信号的控制和通信中心节点；又因为树莓派耗电量极小，且可以通过 SSH 和 PC 机通信，无需配置显示器，所以可以全天 24 小时开启，作为服务器使用。

本文认为树莓派的价格已经足够低廉，而且稳定耐用，无特殊情况如外力破坏和供电异常等，至少可以使用五年。树莓派并不是唯一可用的设备，大多数单片机、微控制器都可以完成本章所进行的工作，但是需要编写额外的驱动程序和底层接口，在考虑大规模制造的经济因素时可以选择自制设备，本文主要利用了树莓派中联网、蓝牙、相机、GPIO 的功能，自制设备时满足上述功能即可。

本文使用的树莓派运行了 Raspbian 操作系统，Raspbian 是开源的基于 Debian 系统进行改造的操作系统，实际上是一个 Linux 的运行环境。

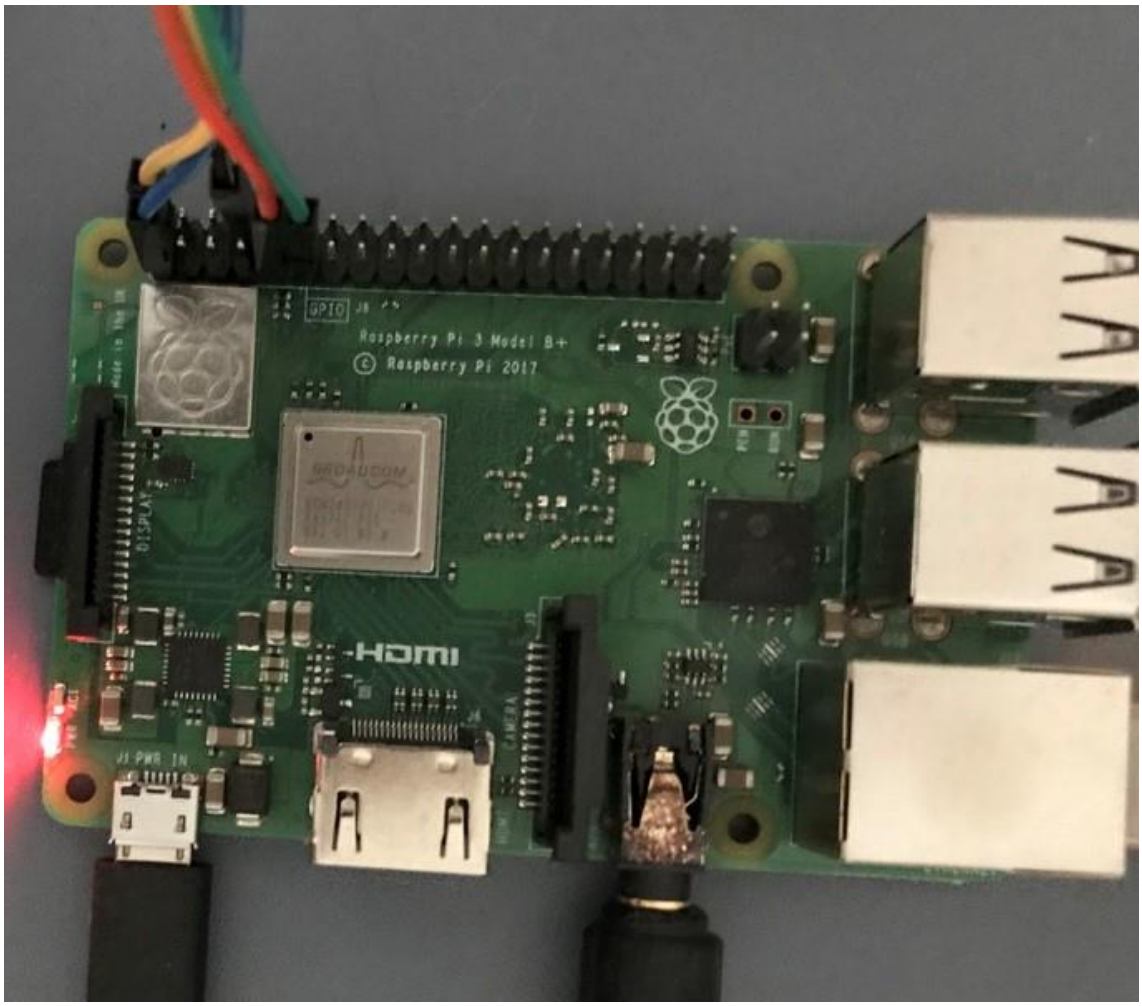

图 3.1　树莓派可经杜邦线连接发射天线等模块

3.1.3　315MHz / 433MHz 射频无线遥控模块的控制

家用车库门一般都有遥控器，可以远程遥控车库门开启或关闭。本文以 315MHz / 433MHz 遥控接收器（图 3.2）为例，说明使用计算机控制车库门的方法。

遥控车库门一般由三部分组成：卷帘门、管状电机、电机控制器。其中电机控制器是一个小型的电路板，带接收天线，是遥控器信号的接收装置，它在收到遥控器发出的信号后，解码遥控器信号，并对管状电机进行相应操作。如果可以使用计算机发出与遥控器同样波形的电磁波，就可以模拟遥控器的工作过程，使电机控制器做出同样的响应。

为了实现信号重放，首先需要录制遥控器所发出的波形，要用计算机接收此波形，需要有能接收同频率电磁波的天线，并读入计算机之中。作者采用了常见的 ASK / OOK 模块监听遥控器发出的波形，并将模块连接到树莓派上，使用 loblab

开源的 rfask 程序[37]读取接收到的信号。

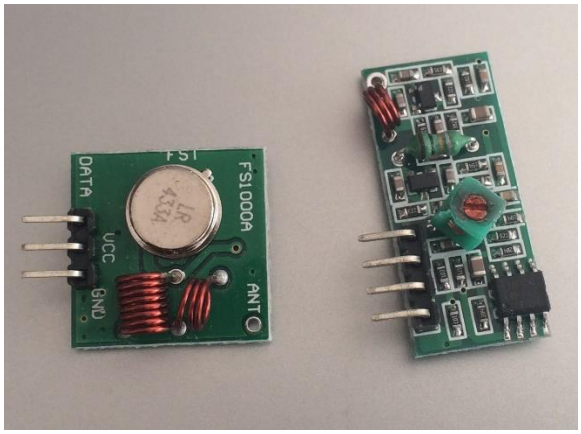

图 3.2　射频无线信号发射和接收模块（一组约人民币 5 元）

读取、解码信号时可能由于天线的质量和信号强度不同，每次读取的信号不完全一致。出现此种情况时，应当多次使遥控器发出信号，取其中出现频次最高的一两组信号作为监听到的控制信号。解码后即不再需要连接接收模块，可以使接收模块和树莓派断开连接。

发射模块连接到树莓派上，根据解码得到的遥控器信号进行信号重放。通过使用 Python 更改 rfask 程序中的代码，可以重放记录下的控制信号。有的发射模块自带了发射天线，对于没有自带发射天线的，可以按照四分之一波长制作弹簧天线，焊接到发射模块的天线引脚上。

对于 433MHz 的信号发射模块，天线长度（缠绕后）应为约 17 厘米：

$$\lambda = \frac{1}{4} \times \frac{c}{f} = \frac{1}{4} \times \frac{3\times10^8}{433\times10^6} = 0.173(m) \quad \text{.....(3.1)}$$

其中λ为四分之一波长，f为天线信号频率，c为光速。

两步完成后，即可使用计算机控制车库门开关。

除车库门以外，使用同频率电磁波控制的车库报警器等设备也可以用此方式进行信号重放，以便计算机进行控制。

3.1.4　蓝牙无线模块通信和控制

因为家庭电路布线时如果没有特殊设计，一般都是单线型，所以为了控制车库灯等其他电器，可以使用单火线方式的蓝牙墙壁开关改造传统机械式的船型墙壁开关。这种改造无需重新布设墙壁内的 220V 电线，而零火线方式的蓝牙墙壁开关则需要重新布设墙壁内的电线。

蓝牙墙壁开关属于蓝牙低功耗设备，厂家出厂时附带控制开关的手机 APP 供用户使用，但需要自行分析墙壁开关蓝牙协议，才能模拟这些控制信号。尽管各种蓝牙开关的程序设计不完全一致，但因为它们都使用蓝牙低功耗协议，且本文的主要目的是控制开关通断，按照电子设计的经验，这个控制量应是一位布尔值或用一个字节来表示，只需要找到这个值，并且用树莓派的蓝牙模块对这个值进行置位，理论上就可以控制开关的通断。

本文以某公司生产的单火线低功耗蓝牙开关作为例子，讲述在没有技术手册（Datasheet）的条件下如何找到这一开关控制位。其他品牌的开关也可以按照此种方式进行探索。

为了探索蓝牙墙壁开关的控制方法，本文使用了三类探索软件，三类需要共同使用互相配合。第一类是开关生产厂家配备的蓝牙开关 APP，在没有任何信息的情况下，需要用这一软件切换开关的状态；第二类是蓝牙状态读取软件，本文使用了 iOS 平台的 LightBlue 软件，这一 APP 可以读取蓝牙开关内的状态量，通过观察在灯通断变化时哪个量发生了改变来推测哪个量是开关通断的控制量；第三类软件是树莓派的蓝牙操作软件，通过对树莓派的蓝牙进行编程控制，模拟手机 APP 对蓝牙的控制过程，本文使用了开源软件 bluepy。

整个探索过程是一个迭代试错的过程，并无技巧可言。操作顺序如下：首先使用厂家蓝牙开关 APP，对软件内的功能进行探索，观察每个蓝牙开关都可以进行哪些操作，可能需要记录哪些状态量，比如开关通断、开关声音、是否有背景灯、背景灯的亮度。大致了解了功能后，分析需要几个状态量记录这些功能，比如上述四个功能或许是一个四位二进制，或许是一位十六进制。而后使用 LightBlue 搜索到此蓝牙开关，进行握手连接，读取其中的信息，查找哪些状态量是四位二进制，对可以写入的位随机写入 0 或 1，观察开关的变化，直到可以用 LightBlue 控制蓝牙开关各种状态，记录并分析哪些位是对哪些功能进行控制的。最后使用 bluepy 在树莓派上编程，模仿 LightBlue 的操作，达到使用树莓派控制蓝牙开关的目的。

本文对所用蓝牙开关探索后发现，写入 0x01 为开关此墙壁开关，写入 0x07 为开关此墙壁开关的强背景光，每次写入都会切换一次状态。这样即可使用树莓派控制开关通断，并且在等候用户关灯读秒时，可以交替变换背景灯的通断，做出呼吸灯的效果，给用户以提示。

以上为用树莓派控制蓝牙墙壁开关的方式。通过控制开关，可以控制车库灯，

继而可以在用户开车驶入车库时自动打开车库灯，也可以通过监测蓝牙墙壁开关的状态量，在用户关闭车库灯时自动关闭车库门，对于不同的生活场景可以进行不同的设置。

3.2　用户界面

从本节起至本章结束，本文讨论智慧车库系统的基于浏览器-服务器的用户界面。

3.2.1　用户界面设计思路

作者认为一个好用的算法不应当只存在于大学实验室和研究所里，而是应当易于学习、易于被非专业人士使用，尽可能的应用于有需求的人们的生产生活之中，为社会生产力提高做出贡献。

因此本文设计并编写了基于浏览器-服务器的用户操作界面，可以方便地控制家庭车辆识别算法的启动和停止，独立地进行车库门、车库灯的开关，以及使用车库内监控拍照等。作者没有采用世界上已有的成熟的大公司或小公司的开源、闭源物联网平台，因为无论哪种第三方平台，如 Apple 公司的 Home App、开源的 HomeBridge 或者国内的云监控平台，都有信息泄露的风险，归根结底，服务器设备只有运行在内网中，并做好家庭路由器防火墙隔离，非必须数据不上网，才能保证家庭控制设备和照片等信息的安全。

本部分的主要目标是编写一个方便用户控制各种车库设备的网站，行业内有多种开源的网站开发框架，如 Python 语言编写的 Django、Flask 框架，Ruby 语言编写的 Ruby on Rails 框架，JavaScript 语言编写的 NodeJS 框架等。这些框架使用的基本思路都比较相似，可以实现的效果也没有太大区别，主要差异在于不同框架处理问题的角度和细节以及易用性；而从软件执行的效率上来看，对于本文想实现的开关功能，他们都可以达到同样的效果，因此无论选择哪个框架都可以完成用户界面的实现任务。

本文最终采用的是 Ruby on Rails（RoR）网站开发框架，这一框架使用 MVC 架构，有数据库集成接口 Active Record，浏览器页面使用 Embedded Ruby 由服务器端编译生成，前端页面样式使用 CSS（SCSS）和 HTML 格式编写。RoR 已经经

过 16 年的开发进入了 6.0 版本，可以自动处理类似 CSRF（Cross Site Request Forgery）类的安全问题，集成度高，开发速度快，帮助程序员节省了很多精力。

3.2.2　用户界面功能与后台数据库表设计

用户界面应当有的基本功能：

1. 用户应当可以注册和登录智慧车库控制中心网站，以防止非家庭人员控制家庭设备。
2. 用户应当可以在登录网站后，点击按钮控制家庭设备（如车库门、车库灯）。
3. 用户应当可以在网站增加、删除、查找、更改自己家里的设备的条目。

为了实现上述三个功能，需要编写三个页面：登录页面、设备控制页面、设备管理页面。

根据以上需求，进行数据库表的设计。数据库表设定为两张，第一张表为用户表（users），第二张表为设备（devices）。

用户注册、登录和认证、保存登录信息功能由开源库 devise gem 处理，使用 users 表存储。Devise 是业内常用的 RoR 网站管理用户及权限的开源库，因为在用户信息安全方面需要处理的问题很多，因此直接采用了开源的软件，降低系统风险，不再在此赘述。

设备表主要需要以下 6 个键：设备名称、设备类型、设备所在位置、设备开启命令、设备关闭命令、设备显示顺序。

设备名称：字符串型数据。

设备类型：字符串型数据，为门、灯、报警器、照相机等。

设备所在位置：字符串型数据，可为空。

设备开启命令：文本型数据，为存储设备开启命令的脚本文件。

设备关闭命令：文本型数据，为存储设备关闭命令的脚本文件。

设备显示顺序：整形数据，为设备条目在页面上的优先级，在有多条设备信息的网站中控制设备出现的先后顺序。

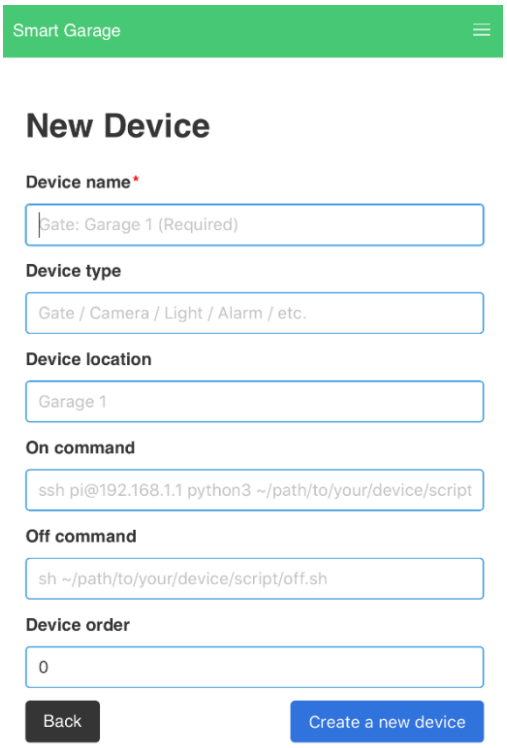

图 3.3　新设备的数据条目

用户进入网站后首先进行注册和登录，而后在设备管理页面添加新的设备（图 3.3），如车库门，以及控制开关车库门的脚本，这些脚本为前文中信号重放小节编写的。添加所有新设备后，在设备控制页面点击对应按钮对设备进行控制。

3.2.3　用户界面外观与交互设计

用户界面的外观应当能够自动适应不同的屏幕尺寸，即响应式设计（Responsive Design）。本文使用了 Bulma 开源样式库[38]，以 SCSS 控制样式。对不能满足需求的样式，本文自行设计所需的样式表。Bulma 开源样式库能够自动适配各种尺寸的屏幕，基于 Flexbox 编写，不含任何 JavaScript，便于使用。

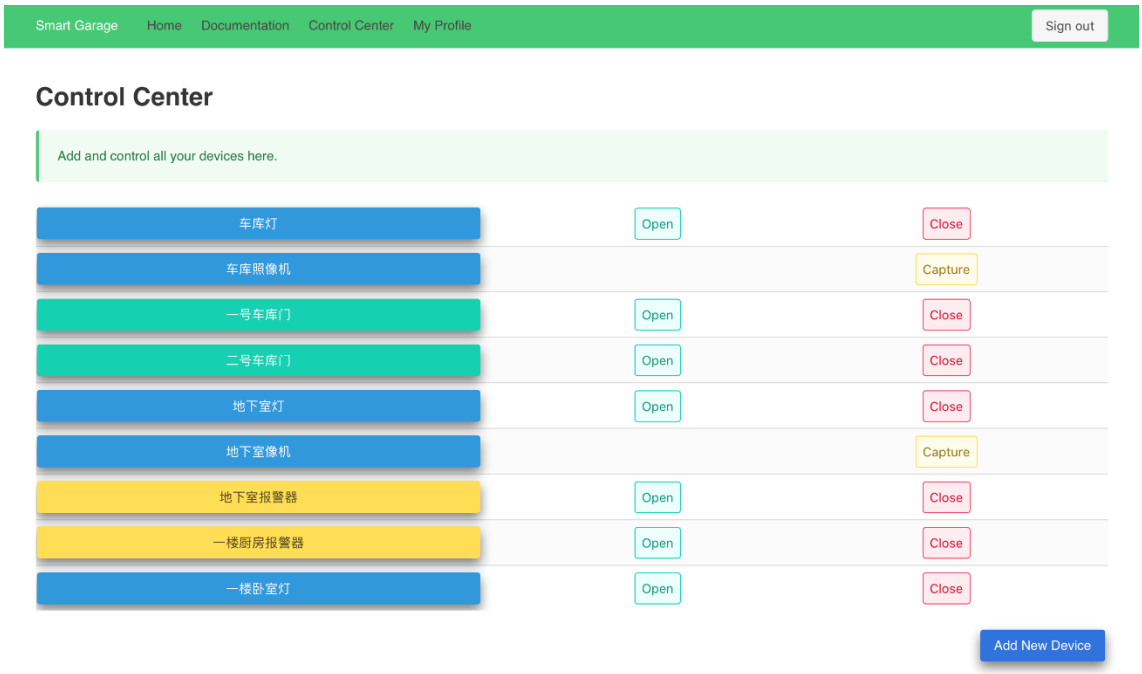

图 3.4　响应式设计的网页（PC 版设备控制页面）

图 3.5　响应式设计的网页（手机版设备控制页面）

在用户交互过程中，首要考虑的是易用性。如图 3.4，图 3.5 所示，用户应当在登录后无需再进入二级页面就可以发出指令，减少用户点击的次数，因此设备控制页面将作为登录成功后的第一个页面。

在设备控制页面中，用户可以单击开启、关闭按钮控制对应设备。对于不需要返回执行结果的脚本，将他们独立开启一个线程运行，同时显示一条横幅告知用户操作正在进行中，以达到用户无需等待服务器返回执行结果的效果；对于需要等待返回结果的脚本，如拍摄监控照片，以单线程执行，而后将拍摄的照片刷新在页面上，再次刷新页面或做出其他控制时，隐藏掉监控照片。

在设备管理页面中，以表单的形式允许用户增、删、查、改各数据（图 3.6），在用户确认修改或放弃修改时，弹出浏览器警告窗口使用户二次确认以防止误操作。

图 3.6　设备管理页面

在用户界面程序编写完成后，可以在树莓派上运行，为了方便用户访问，应当对树莓派所分配的 IP 地址进行固定，这一点可以在路由器中设置。

如果需要在家庭外访问家庭内网运行的树莓派，可以在路由器中进行端口转发，转发地址是树莓派被分配的内网 IP 地址以及 Rails 程序开放的端口号。若要在外网方便的访问，可以考虑设置路由器的 DDNS（动态 DNS），并用域名访问。

本文编写的用户界面已经在测试环境中稳定运行了 18 个月。

3.3　本章小结

第一，本章对智慧车库系统的硬件控制部分进行了讨论，讲述了使用树莓派控制遥控车库门、蓝牙墙壁开关等设备的方法和实现过程，实验结果表明这种方法是可行而且简便的。

第二，本章针对算法的可用性和易用性问题，设计了基于浏览器-服务器的用户交互界面。使用户可以在各种电子设备的网络浏览器中对控制中心进行操作，如启动和停止车辆识别功能，控制车库门、灯、报警器开关等，为算法的实际应用提供了便利。

第 4 章　智慧车库系统的整合

本章主要讨论智慧车库系统工程应用方面的问题。

尽管识别算法、外设控制硬件、用户界面软件都可以独立工作，但将这些模块整合成为一套系统，还需要进一步的工作。

本章将首先介绍整体系统的工作流程，而后讲述模块之间的信息交换方式，接着对每处接口的设置进行分析和处理，最后对算法的应用细节和可能存在的问题以及应对方式进行探讨。

4.1　智慧车库整体结构

最基本的智慧车库系统应当有一台路由器、一台运行检测算法的计算机、一台 24 小时工作的树莓派、一台用于采集视频的相机或摄影机、以及一些遥控外设。

计算机、树莓派、网络摄影机连接到同一台路由器上，或以交换机连接在同一网段内，以便流量互通，建立一个内部网络（如图 4.1）。

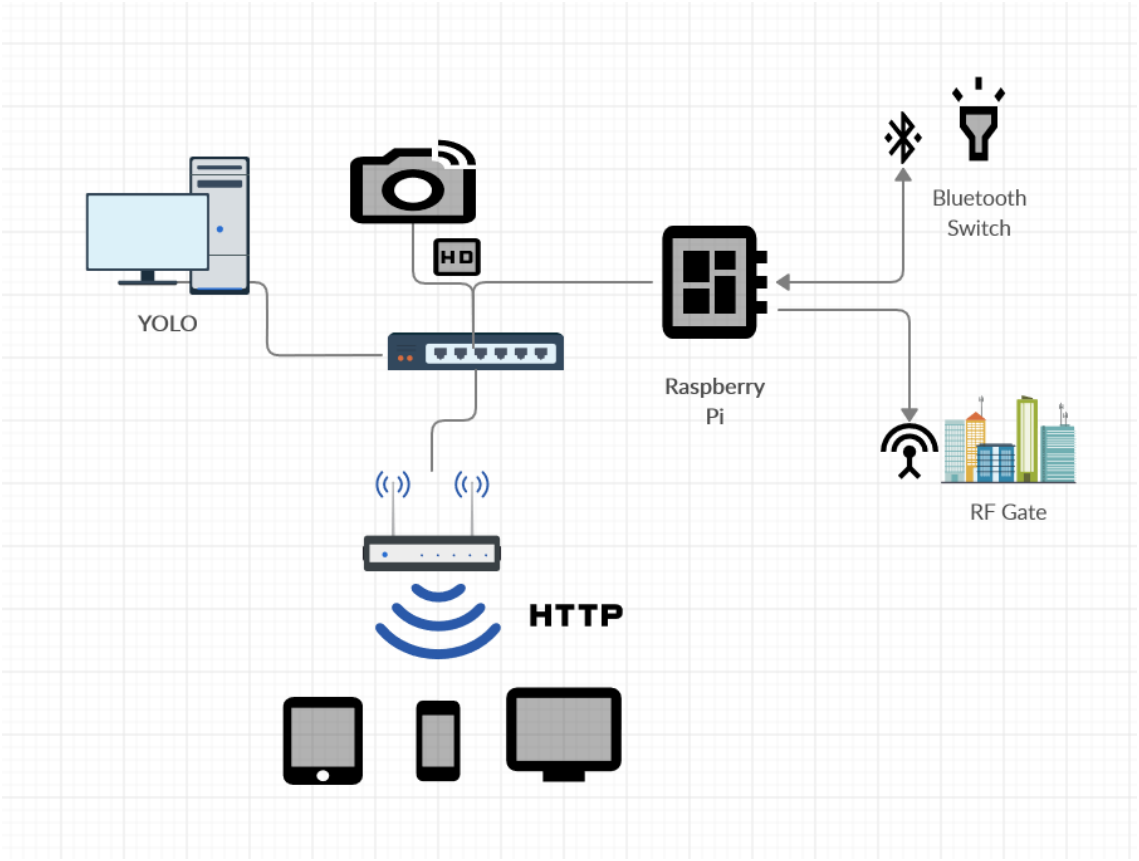

图 4.1　硬件关系图

树莓派作为控制中枢，全天工作。树莓派负责向其他设备发出控制信号，这些信号包括：

1、 通过网络向计算机发送开机、关机信号，发送启动识别算法、停止识别算法信号。

2、 通过 ASK / OOK 射频无线发射模块向车库门控制器发射开关车库门的控制信号。

3、 通过蓝牙与蓝牙开关通信，读取开关目前的通断状态、发出开关信号。

4、 作为用户界面网站的服务器，树莓派运行的 Ruby on Rails 服务端响应用户客户端发来的 HTTP 请求。

网络摄影机：全天工作，以 RTSP 协议（Real Time Streaming Protocol）将视频流发送到计算机上，供计算机作目标检测使用。

计算机：在收到树莓派的开机指令和启动检测指令后，接收摄影机发送来的视频流，进行目标检测，将要做出的控制指令以 SSH（Secure Shell）协议命令的方式传递给控制中枢树莓派，而后由树莓派执行指令，如打开车库门、车库灯等。

4.2　视频流采集

本文使用了某公司生产的 1080P 枪机网络摄像头，因为其具有网络连接功能，可以提供 RTSP 协议的视频流，运行在计算机检测算法程序中的 OpenCV 接口则可以取用这种协议的视频流。对于其他品牌的摄像头，只要其具备网络功能，可以支持 RTSP 协议，均可使用。选择摄像头时同时需要注意使用场景，如是否具备电力供应，实际的大概使用距离，同时可以依照不同的现场需求选择由网络供电的摄像头及支持 PoE（Power over Ethernet）以太网供电的路由器，以及合适焦距的摄像头或者变焦摄像头。

摄像头安装时应当尽量正对汽车前进方向，使车牌能较清晰的出现在屏幕上，否则可能降低识别的速度和准确率。

摄像头的网络连接应当尽可能的稳定无延迟，否则会影响实时识别的效果，尽管本算法在软件实现中已经处理了网络故障类的视频流中断问题，但是由视频流中断产生的信息缺失问题算法无法进行弥补，如果恰逢网络故障时车辆经过，则将无法识别到车辆。

摄像头采集视频的锐度、白平衡、曝光补偿、色彩明亮度等参数，没有唯一的设置标准，本文建议按照具体使用场景进行调整、现场进行测试。调整的底线是人眼可以从视频中分辨出车牌号，在此基础上按照算法检测结果进行选择性调整，找到最佳的设置参数。但对于帧率，本文推荐采用摄像头可以采集到的最高帧率，如本文使用的摄像头为 25 帧，提高帧率可以降低检测的延迟，尽早发出控制信号，也可以给检测算法提供更多的图像，提高检测灵敏度；但这样做同时会增加计算机的计算量，调整时需注意 GPU 的运行负荷。调优的目标则是可以稳定检测到车辆通过。

在实验中，作者发现摄像头会自动进行曝光调整，但有一定的延时，比如较黑的环境下车灯突然直射到摄像头，所采集的视频将是过曝光的视频，无法从图像中看到任何物体，只有一片白色。这种情况下摄像头无法识别出车辆，因此建议停用自动曝光调整，同时对夜晚和白天设置相同的参数，以防自动切换昼夜摄像头模式时引起图像曝光抖动。

4.3　能源节约

作者使用英伟达公司提供的 GPU 运行检测工具 nvidia-smi 软件观察了 GPU 的功耗，结果为在 Arch Linux 系统中，未运行目标检测算法的功耗为 10 瓦特左右，运行车牌目标检测算法时 GPU 的功耗为 45 瓦特左右。

本文认为除非有必要，否则不应全天开启计算机运行车辆检测程序，因为它会消耗可观的能源。因此本文将计算机不运行检测软件时设置为休眠状态，以有线网络连接，采用 WakeOnLan 进行计算机唤醒，仅在需要时才启动软件的运行，这样可以节约大部分的能源。

不设置为关机状态而是设置为休眠状态的原因是，休眠状态网卡总是有供电的，但关机状态主板电源管理策略可能停止网卡供电，这将使计算机不会响应 WakeOnLan 的指令。

WakeOnLan 是大部分计算机主板都支持的通过有线以太网网卡进行计算机唤醒的方法，在 Windows、Linux 系统都有相应的支持，此功能默认是关闭的，需要手动进行开启。因为不同主板和操作系统开启此功能的方式不同，因此需要按照主板的用户手册和操作系统的官方文档各自设置。需要进行的设置大致相同，首先应

在主板上启用通过网络唤醒功能，这一步的目的是保持网卡在关机后仍然供电，以监测其他设备通过路由器转发来的魔术包（Magic Packet）；接着应当在操作系统中配置支持 WakeOnLan 唤醒，这一步一般在网卡的设置中进行；最后要注意路由器的设置，因为发送唤醒魔术包一般是以广播的方式进行的，因此需要路由器记录计算机网卡的物理地址（Mac Address），一般的路由器会在计算机开启后自动记录，而后在一段时间后自动清除，此时应当为计算机设置固定 IP 地址，而后运行脚本定时添加计算机网卡地址的记录。

计算机设置好 WakeOnLan 功能后，发送唤醒魔术包的工作由 24 小时开启的树莓派完成，魔术包的格式唯一，按魔术包文档配置即可。

日常家庭使用时，需要关注家庭成员上下班的出门和回家时间，在这个时间段内开启车辆检测算法。统计好可能的时间段后，使用树莓派 Linux 系统自带的定时执行任务软件 crontab 进行定时触发任务，crontab 的配置按照官方手册进行即可。这一定时任务的主要目的是唤醒计算机（如处于休眠状态），而后启动车辆检测算法。如果计算机已经打开，则直接启动车辆检测算法，同时注意检测结束后不能自动休眠，以免影响正在使用计算机的用户的工作。测试计算机是否开机可以通过建立一个测试管道（socket）来完成，如果无法建立 socket 或建立超时则视为计算机未启动，此时通过 WakeOnLan 唤醒计算机，应等待 15 秒左右（也可按照计算机实际开机时间设置此延迟），再启动车辆检测算法，车辆检测结束后，自动休眠，以继续节约电能。

当手动开启车辆检测算法时，应当再启动前首先测试车辆检测算法是否已经开启，这一过程可以使用 Linux 自带的进程统计工具 ps 进行。如果车辆检测算法已经开启，应当使用 pkill 命令杀掉进程，而后再开启新的进程，以免造成资源泄漏。

4.4　目标检测算法配置

本节介绍目标检测算法针对不同现场场景时需要进行的调整及其他设置。

显存分配问题：

车辆检测软件使用时，输入图像需要串行经过三次目标检测模型，每个模型在运行时需要分配 GPU 显存，第一个模型的显存是在软件刚刚启动时分配的，但第

二、第三个模型，仅当它们的前置模型有输出时，才会进行显存分配。这种显存分配方法可能造成问题，因为显存分配需要时间，本模型所需时间一般在 1 秒左右，两次分配则为两秒。在分配的时间段内，第一个模型无法继续进行检测，这会降低成功识别到目标车辆的几率，因为 2 秒的时间车辆可能已经驶过视频采集区域，而导致失去了 50 帧左右的图像。考虑到这个问题，本文在软件开始运行时就将三个模型的显存全部分配完毕，而不等待模型一识别到了车头车尾才分配模型二的显存，这可以降低第一个经过摄像头视野的车辆的漏检率。

摄像头视野内机动车停泊问题：

摄像头可视角度大时，视野内可能出现道路两侧停泊的机动车。车辆检测系统会持续的检测这些道路两侧停泊的机动车，将它们的图像输入模型二、模型三，消耗大量的计算资源，不但没有价值，还可能使检测系统卡顿，造成计算资源的浪费。为了解决这个问题，本文采用的方法是，对摄像头输入计算机的视频流进行裁剪，仅保留行车通道，即没有车辆长时间停泊的通道。这一裁剪过程可以使用摄像头自带的功能，以减少计算机的工作量，也可以用本算法实现中提供的裁剪函数，按比例裁剪输入视频流。裁剪的尺度应按现场实际情况确定，尽可能保证裁剪后的视野内没有其他车辆停泊的位置，但也要注意不能过度裁剪，以免无法拍摄到目标车辆或因拍摄的时间太短而降低识别率。

目标检测模型阈值问题：

YOLO 算法对每个目标都计算一个概率值，如果其超过设定的阈值（threshold）则视为检测到一个目标，否则放弃这个目标。对于本文的三个模型，作者经过实验测试设定了三个阈值：模型一是车头车尾检测的模型，这一阈值设定的比较高，因为视频流有很多帧，有很多机会可以得到一个可信的被判断为是车辆的图像，设为 0.86；模型二是车牌图形识别的模型，车牌是较小的一块图像，占车的比例小，不易被检测到，因为由模型一传入模型二的图像已经是车辆，而且模型二只有一类目标，只要检测出目标，有极大的可能性检测到的目标就是车牌，因此根据实验分析效果，将阈值设定得较低，为 0.50，尽可能保证不浪费模型一的输出图像；模型三是字符识别的模型，经过了模型一和模型二两次认定，已经可以确定传入模型三的是车牌图像，模型三主要可能出现的问题是误分类，即将 3 分为 8，将 0 分为 D 或 Q 这类错误，故而不能像模型二一样将阈值设定得太低，但又因为字符很小，也不容易被识别到，因此也不能设定太高以免漏检，根据实验结果，此阈值设定为 0.65。

这些阈值并不唯一，不同的应用场景可以按需进行更改、权衡。

4.5　本章小结

本章主要讨论了智慧车库系统的工程细节问题，如硬件网络连接、信息的流动方向、进行视频采集的摄像头的配置，还介绍了可能的减少能源消耗的方法，以及算法的一些配置参数等等。对于不同的使用场景，这些设置和参数均可按需调整或扩展，以便达到最好的检测效果。

这些步骤虽然不包含复杂的学术问题，但在构建基于目标检测算法的家庭智慧车库系统时非常重要。经过了整合的算法、软件、硬件、用户界面，构成了廉价、稳定、易用的一套车辆识别系统，系统在利用本章的方案针对性的进行配置后，可供不同需求的家庭用户使用。

第 5 章 总结与展望

5.1　工作总结

本文完整地介绍了基于目标检测算法的智慧车库系统的算法思想、硬件要求、软件设计和本系统的整合方法。

本文所做的主要工作如下：

1、提出了基于目标检测算法的车头、车尾识别和分类方法，并进行了模型建立和实验，结果证明此方法可以检测并分类图像或视频中汽车的车头、车尾。

2、提出了基于目标检测算法的车牌号识别方法，建模并进行实验，实验结果证明模型可以识别车牌号。

3、提出了在深度学习字符识别建模过程中，降低数据标注工作量的方法，并结合分类数据集和目标检测数据集，提高了车牌字符识别的分类准确率。

4、成功进行了用树莓派和计算机控制车库门、车库灯等非智能设备开关的实验。

5、编写了基于浏览器-服务器架构的用户界面，使用户可以在手机、电脑或平板电脑上控制车库门、车库灯等设备的开关和识别算法的启停。

6、对算法在不同应用场景下需要调整的配置进行了讨论，以便检测算法在不同环境下能得到最好的使用效果和用户体验。

5.2　工作展望

尽管本系统可以满足家庭用户的使用需求，但它仍然存在部分缺陷：

1、日落后、夜晚无法识别车辆。尽管在雪天、阴雨天本算法都可以识别到车辆，但在日落后人眼仍可识别车牌时，如吉林地区的十一月至次年二月，下午四点三十分后，算法通常无法识别到车辆。作者初步认为这一问题可能在于训练数据集和光照，训练集大多是在光线较好的条件下或有高速路闪光灯辅助照明的条件下拍摄的，因此模型在天色较暗时无法识别到车辆。而在夜晚，没有辅助照明的条件下，人眼无法识别车辆，本算法也无法识别车辆，这一问题很难解决，可能要考虑不可见光波段的信号。

2、算法的计算量大。一块 GTX1060 6G 的 GPU 最多同时运行二到三个检测实例，通常只运行一个检测实例。尽管这对于家庭用户没有影响，但要进行扩展或在未来使用云计算来进行家庭车辆检测，这一算法的计算量仍然过高，如果可以考虑用 MapReduce 算法[39]或 Raft 算法[40]的思想对其进行辅助的分布式计算，用小型的不昂贵的硬件设备集群进行大规模的计算，可能可以降低检测的成本，但是这两种算法是否可以应用在这一识别问题上尚需讨论。

3、能源节约设置影响软件易用性。为了节约能源，算法仅在家庭成员可能出行或回家时运行，这需要考虑家庭成员的通常作息时间，对时间不规律的用户则无法采用这种定时运行的方式。作者曾考虑对检测算法的启停使用基于手机位置的触发方式，如用户（按照手机基站定位或 GPS 定位）靠近家庭地址 1 公里到 1.5 公里范围时启动识别，但这一方法不可靠，问题在于定位信号发送通常有延时，经常发生用户已经离开某个基站，才收到用户在那个基站的消息，而使用树莓派主动向手机请求位置又将会减少手机电池的寿命，因此放弃了这种基于位置的触发方法。如果可以找到基于位置启动检测算法启停的方法，使用位置作为粗略的数据、使用车辆信息作为精确的数据，控制车库门的开关，将对整个系统的易用性有很大的提升。

智慧车库系统可做的调整还有很多，在未来技术发展后可以针对系统不易使用的地方进行改进，使其更适合人们使用，为工作提高效率、为驾驶员提高安全水平、为人们提升幸福感。

参考文献

[1] ARM Holdings. ARM Cortex-A53 MPCore Processor Technical Reference Manual [EB/OL]. ARM, (2016-02-08)[2020-03-04]. https://developer.arm.com/docs/ddi0500/g/preface.

[2] Cortes, Corinna, and Vladimir Vapnik. Support-vector networks[J]. Machine learning, 1995, 20.3: 273-297.

[3] Altman N S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression[J]. The American Statistician, 1992, 46(3):175-185.

[4] Sara Baghdadi, Noureddine Aboutabit. Front and Rear Vehicle Classification[J]. Advanced Intelligent Systems for Sustainable Development, 2019(4): 264-272.

[5] Li H, Shen C. Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs[J/OL]. arXiv preprint arXiv: 1601.05610, 2016[2016-01-21]. https://arxiv.org/abs/1601.05610.

[6] Du S, Ibrahim M, Shehata M, et al. Automatic License Plate Recognition (ALPR): A State-of-the-Art Review[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2013, 23(2):311-325.

[7] Yuan Y, Zou W, Zhao Y, et al. A Robust and Efficient Approach to License Plate Detection[J]. IEEE Transactions on Image Processing, 2017, 26(3):1102-1114.

[8] 肖驰. 一种高效的车辆颜色识别方法[J]. 现代计算机(专业版), 2017(17):73-77.

[9] 陈宏彩. 一种基于深度卷积神经网络的车辆颜色识别方法[J]. 河北省科学院学报, 2017, 034(002):1-6.

[10] David Heinemeier Hansson. Ruby on Rails[CP/OL]. Github Repository, (2005-12-13)[2020-03-04]. https://github.com/rails/rails.

[11] Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung. Convolutional Neural Networks for Visual Recognition[Z/OL]. 2017-7[2020-03-04]. https://cs231n.github.io.

[12] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: unified, real-time object detection[C]. Conference on Computer Vision and Pattern Recognition (CVPR). San Francisco: IEEE, 2016:779-788.

[13] Redmon J, Farhadi A. Yolo9000: Better, faster, stronger[C]. Computer Vision and Pattern Recognition (CVPR). San Francisco: IEEE, 2017: 6517-6525.

[14] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J/OL]. arXiv preprint arXiv:1502.03167, 2015[2015-02-11]. https://arxiv.org/abs/1502.03167.

[15] Redmon, Joseph, and Ali Farhadi. Yolov3: An incremental improvement[J/OL]. arXiv preprint arXiv:1804.02767, 2018[2018-04-08]. https://arxiv.org/abs/1804.02767.

[16] Girshick R. Fast R-CNN[C]. International Conference on Computer Vision (ICCV). Santiago, 2015:1440-1448.

[17] Ren, Shaoqing, He, Kaiming, Girshick, Ross, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 39(6):1137-1149.

[18] 王林，张鹤鹤. Faster R-CNN 模型在车辆检测中的应用[J]. 计算机应用, 2018, 38(3):666-670.

[19] Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector[C]. European conference on computer vision. Springer, Cham, 2916:21-37.

[20] Bodla N, Singh B, Chellappa R, et al. Improving Object Detection with One Line of Code[J]. The IEEE International Conference on Computer Vision (ICCV), 2017: 5561-5569.

[21] Lin, M., Chen, Q., Yan, S. Network in network[J/OL]. arXiv preprint arXiv:1312.4400, 2013[2013-12-16]. https://arxiv.org/abs/1312.4400.

[22] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]. Inter. Conf. on Neural Information Proc.Sys.,2012(1):1097-1105.

[23] Jørgensen, Hogne. Automatic License Plate Recognition using Deep Learning Techniques[D]. Norway: Norwegian University of Science and Technology, 2017.

[24] 陈寅鹏，丁晓青. 复杂车辆图像中的车牌定位与字符分割方法[J]. 红外与激光工程, 2004, 33(1).

[25] 马俊莉，莫玉龙，王明祥. 一种基于改进模板匹配的车牌字符识别方法[J]. 小型微型计算机系统, 2003, 24(9).

[26] 赵伟，张南楠. 深度学习在复杂环境下车牌定位算法中的应用[J]. 现代电子技术, 2019(17).

[27] 刘泽康，孙华志，马春梅. 基于卷积神经网络多特征联合的车辆识别模型[J]. 计算机科学, 2019(B06):254-258.

[28] Redmon J. Darknet: Open source neural networks in c[CP/OL]. (2016-05-06)[2020-03-04]. http://pjreddie.com/darknet.

[29] 蒋芸，彭婷婷，谭宁. 基于 YOLO 算法的眼底图像视盘定位方法[J]. 计算机工程与科学, 2019(9).

[30] 付振华，纪祥，赵坤旭. 基于 YOLO 算法的智能交通灯控制系统模型[J]. 单片机与嵌入式系统应用, 2019(9).

[31] 高宗，李少波，陈济楠. 基于 YOLO 网络的行人检测方法[J]. 计算机工程, 044(005):215-219,226.

[32] National Technical University of Athens. Medialab LPR database[DB/OL]. (2000-4-27)[2020-3-3]. http://www.medialab.ntua.gr/research/LPRdatabase.html.

[33] Alexey. Yolo_mark[CP/OL]. Github Repository, (2019-4-29)[2020-3-3]. https://github.com/AlexeyAB/Yolo_mark.

[34] Zoran Kalafatić, Slobodan Ribarić, Tomislav Hrkać. License Plate Detection, Recognition and Automated Storage[DB/OL]. (2003-10-11)[2020-3-4].

http://www.zemris.fer.hr/projects/LicensePlates/english/results.shtml.

[35] Andrew Ng. Machine Learning[Z/OL]. (2017-07)[2020-03-04]. https://www.coursera.org/learn/machine-learning.

[36] De Campos, T., Bodla, R. B., Varma, M. The chars74k dataset[DB/OL]. (2012-10-15)[2020-3-4]. http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k.

[37] loblab. rfask[CP/OL]. Github Repository, (2018-05-26)[2020-03-04]. https://github.com/loblab/rfask.

[38] Jeremy Thomas. Bulma[CP/OL]. Github Repository, (2020-01-26)[2020-03-04]. https://github.com/jgthms/bulma.

[39] Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters[C]. Proceedings of Sixth Symposium on Operating System Design and Implementation (OSD2004). 2004.

[40] Ongaro D, Ousterhout J. In search of an understandable consensus algorithm[C]. 2014 USENIX Annual Technical Conference. 2014: 305-319.

作者简介及在学期间所取得的科研成果

作者简介：

姓名：高放

出生年月：1991 年 7 月

民族：汉族

籍贯：吉林省长春市

吉林大学计算机科学与技术学院，工学硕士

研究方向：数据挖掘应用

联系方式：gaofang17@mails.jlu.edu.cn

科研成果：

1. Cui Mengzhao, ***Gao Fang***, Gang Xiaokun, Wang Guixia. Diagnosis of sarcopenia in T2DM--using data mining methods, 中华医学会第十一次骨质疏松和骨矿盐疾病中青年学术会议, 2019/04.
2. Mengzhao Cui, Xiaokun Gang (co-first author), ***Fang Gao (co-first author)***, Gang Wang, Xianchao Xiao, Zhuo Li, Xiongfei Li, Guang Ning, Guixia Wang. Risk assessment of sarcopenia in patients with type 2 diabetes mellitus using data mining methods, Frontiers in Endocrinology, 2020/03/07 (SCI, 中科院 2 区 (2020)，IF (2020) = 3.634).

致　谢

首先我想感谢我的家人，在艰难的时期，我们互相支持，在幸福的日子，我们分享欢乐。在他们的支持下，我顺利的完成学业，在他们的陪伴下，我走完这奇特的旅程，没有他们的支持和帮助，我的生活就不会如此美好和充满回忆。

接着，我想感谢一直指导我学习和科研工作的王利民老师和李雄飞老师，他们一直支持我的想法、宽容我的错误，帮助我在学业和生活上成长，一直以乐观的心态和独特的人格影响我，让我在迟疑时不畏惧、困惑时不迷茫。

同时，我还想感谢我本科时的指导老师许军老师，尽管我已毕业多年，他一直支持我在电子电气领域的探索，为我提供了很多帮助，没有他的帮助，我就不会在交叉学科中发现自己的兴趣爱好，施展自己的才能。

最后，我想感谢所有帮助过我的同学、朋友、同事，你们的善意和付出令我的世界更加美好，希望在未来我也可以用我的知识和技能令你们的世界一样美好！