

# 第十八届中国研究生电子设计竞赛

## 技术论文

智能停车场导航系统

Intelligent parking navigation system

---

## 摘要

随着国民经济的提高，汽车保有量呈现快速增长的趋势，智能停车等静态交通发展成为民生建设的重中之重。因此开发智能停车场或者对现有停车场进行升级改造，对于解决“停车难、寻车慢”等瓶颈问题，促进静态交通快速发展具有重要的现实意义。本课题充分结合计算机视觉和传感器等相关技术，采用 C/S 架构，实现对进入停车场的车辆进行车位预定和车位导航。运用 YOLOv5 和 CRNN 技术，实现车牌识别，结合 A\*算法来搜索最近的空闲车位。另外在每个路口安装了方向指示器（由开发板、摄像头、显示屏组成），用于采集车牌信息，并根据车牌信息去服务器查找并预分配的车位，根据车位和当前坐标对车辆进行导航。为了管理车位信息，使用了 Redis 作为存储系统。当车辆驶入停车场时，根据车牌预定车位，并记录停车时间，当车辆离开车位时，通过出口识别判定车辆离开，并及时将车位标记为空闲状态。

在车牌识别的检测与识别中，运用两阶段的迁移学习策略训练了 YOLOv5 模型和 CRNN 模型，且精度较高满足识别的需求。针对实时的检测效果，采用了 OnnxRuntime 对训练好的模型进行加速，从而让模型的效果可以达到实时性要求。

通过这一解决方案，能够实时监测停车场的状况，提高停车位利用率，有效缓解停车场拥堵问题，为车主提供更便捷的停车体验。

**关键词：**智能停车导航系统；YOLOv5；CRNN；Redis；迁移学习

---

## Abstract

With the improvement of the national economy, the trend of rapid growth in car ownership, intelligent parking and other static transportation development has become a top priority in the construction of people's livelihood. Therefore, the development of smart parking or the upgrading of existing car parks is of great practical significance to solve the bottleneck problems of "difficult parking and slow car search" and promote the rapid development of static transportation. This project fully combines computer vision and sensors and other related technologies, and adopts C/S architecture to realise parking space reservation and parking space navigation for vehicles entering the car park. Using Yolov5 and CRNN technology, license plate recognition is implemented, combined with the A\* algorithm to search for the nearest available parking space. In addition a direction indicator (consisting of a development board, camera and display) was installed at each junction to capture the license plate information and to go to the server to find and pre-assign spaces based on the license plate information and to navigate the vehicle based on the space and current coordinates. To manage the parking space information, Redis is used as a storage system. When the vehicle enters the car park, the space is reserved according to the license plate and the parking time is recorded. When the vehicle leaves the space, the vehicle is judged to have left by exit recognition and the space is promptly marked as vacant.

In the detection and recognition of license plate recognition, the YOLOv5 model and CRNN model are trained using a two-stage migration learning strategy, and the accuracy is high to meet the needs of recognition. For real-time detection, OnnxRuntime is used to accelerate the trained models so that they can meet the real-time requirements.

This solution enables real-time monitoring of parking conditions, improves parking space utilisation, effectively relieves parking congestion and provides a more convenient parking experience for car owners.

**Keywords:** intelligent parking navigation system; YOLOv5; CRNN; Redis; migration learning

---

## 目录

<b>第一章 绪论</b>	<b>1</b>
1.1 研究背景和意义	1
1.2 停车场智能化导航与管理研究现状	1
1.3 作品难点和创新点	2
<b>第2章 系统设计与硬件架构</b>	<b>3</b>
2.1 系统需求分析	3
2.2 基于深度学习的目标检测	4
2.2.1 基于 YOLOv5 的目标检测算法	4
2.2.2 基于 CRNN 的车牌识别算法	4
2.2.3 模型训练	5
2.3 硬件系统	6
2.3.1 硬件平台介绍	6
2.3.2 硬件及传感器介绍	7
2.3.3 硬件系统架构	8
2.4 软件系统	9
2.4.1 后端系统	9
2.4.2 前端系统	10
<b>第三章 系统设计与实现</b>	<b>12</b>
3.1 系统整体设计概述	12
3.2 车位分配实现	17
3.3 数据存储实现	18
3.3 停车场车位无感导航实现	19
3.4 各终端通信实现	21
<b>第四章 后期工作</b>	<b>23</b>
4.1 停车场导航改进	23
4.2 反向寻车	23
<b>第五章 总结</b>	<b>24</b>
<b>参考文献</b>	<b>25</b>

## 第一章 绪论

### 1.1 研究背景和意义

随着中国城市现代化的发展，城市汽车保有量逐年增加。城市中心停车设施不足的问题越发明显，挤占正常机动车道停车的行为随处可见，严重的影响了城市交通。为了解决停车问题，势必会扩大停车场规模和增加停车位，传统停车场的弊端也越加明显，概括而言，传统停车场存在以下不足之处<sup>[1]</sup>：

- 1.管理者对于各个区域剩余停车位无法准确掌握，只能人工检查。
- 2.传统停车场出入口还只是单纯设置道闸，停车取卡；离开停车场时，人工收费离场，在停车场高峰时间会造成出入口拥堵，降低停车场利用率，浪费用户时间。
- 4.停车场内引导信息不够详细，只有针对部分区域的车位使用情况引导，无法详细发布每个车位使用情况。
- 3.反向寻车难。在陌生的大型停车场中，容易在停车场内迷失，不容易快速准确找到车辆。

随着计算机技术的不断进步和硬件设备的提升，如 Jetson Nano 等嵌入式设备的广泛应用，智能停车场导航系统的实施成为可能。因此，引入智能停车场导航系统成为解决停车场拥堵问题的一种切实可行的方案。

### 1.2 停车场智能化导航与管理研究现状

国内的智能停车场还是只是处于起步阶段，最近几年，国内的一些一线城市如北京、上海、广州等地才刚刚建立了一些智能停车场。二三线城市大多数的智能停车场管理系统还只是对出入口处建立道闸，取卡机和收费亭。缺乏对停车场内部信息的管理，包括对停车场内停车位的管理，停车场内车辆位置管理，更不用说是信息的共享，管理者对停车场内车位使用情况一无所知，并且场内缺少车辆引导，用户需要花费时间来寻找空位，并且在返回场内寻找车辆时，由于忘记自己车辆位置而造成寻找车辆困难。总的来说，智能停车场现阶段发展还是不够完善<sup>[2]</sup>。许多院校和研究所也开始投入人力物力资源对停车场智能化测控管理技术开始研究，研究范围涉及到车位引导、车辆识别、控制平台、反向寻车、以及组网方式等方面。

目前，国外停车场管理系统利用互联网信息共享技术已经实现了在区域内多个停车场内随意停车。在管理系统中，车位信息和计费结算等功能都是由区域内系统统一进行处理的。可以在未出发之前，用户通过手机终端等方式查询目的地

停车场车位信息，并直接预订车位，可以实现自动扣费。这种管理方式不仅提高了停车场的利用率，并且给用户带来了便利，节约了时间，使停车场的功能得到了极大扩展和延伸。

2010 年美国大学的 Jae Kyu Suhr 等人利用运动立体声技术进行三维重建实现停车位检测，这是车位检查的一项新技术的应用[3]。

2012 年意大利帕玛尔大学的 Alessandro Grazioli,Marco Picone 等人设计了一个面向服务的模块化智能泊车系统包括停车场运营商和最终的 Web 应用程序，系统已在当地校园开始使用[4]。

综上所述，目前停车场管理手段趋向于智能化、信息化、人性化。所以相应的停车场导航，视频引导等功能的不断完善研究开发，符合停车场智能化管理技术发展的新方向，本文将围绕着基于停车场导航展开研究，符合停车场智能化的发展趋势。

### 1.3 作品难点和创新点

基于图像识别的智能停车场导航系统尤其是针对于那些人流量大经常造成拥堵的停车场存在强烈的社会需求和发展前景[5]。为了解决这一个问题我们开展了一个创新项目。

我们的项目集成了多项技术，以提升停车场的管理和停车体验。首先，我们采用了 YOLOv5 和 CRNN 两个模型来进行车牌的检测和识别。这些深度学习算法能够准确地提取车牌信息，帮助我们实时获取车辆的标识。同时，我们利用 A\* 算法进行车位最短路径搜索，通过分析车牌信息和已分配的车位信息，快速找到最近的空闲车位，并指示车辆如何前往目标车位。

为了实现车位状态的实时更新和管理，我们采用了 Redis 作为车位导航系统的存储工具。Redis 数据库可以高效地存储和更新车位的占用状态，确保信息的准确性和及时性。当车辆离开车库时，我们使用图片检测车子是否离开，通过检测车辆的进入和离开，及时更新车位的状态，以提供准确的空闲车位信息给车辆导航系统使用。

此外，我们还注重项目的部署和适配性，将整个系统部署到 Jetson Nano 等嵌入式平台上。通过优化算法和适配嵌入式设备的计算资源和功耗限制，我们实现了高效的车牌识别和车位管理功能。

我们的项目旨在提升停车场管理的效率和用户的停车体验，减少停车拥堵现象，节约时间和资源。通过运用先进的技术和创新的方法，我们相信这个项目将为城市交通带来积极的影响，为人们的出行提供更便捷和高效的停车解决方案。

## 第 2 章 系统设计与软硬件架构

### 2.1 系统需求分析

本作品实现了一个大型停车场车位管理和导航系统，解决在高峰期空车位寻找困难的问题。保证用户根据指示能用最短的时间寻找到空闲车位，且做到无感式服务，即不需要用户下载额外的导航软件。本系统由一台服务器和若干方向指示器（由摄像头、超声波传感器、开发板、显示屏）组成。系统需求如下：

- 1) 当车辆进入停车场时识别车牌号，并根据当前车位空余情况分配最近的车位。
- 2) 路口的方向指示器能够感应到车辆，并且抓拍图片上传服务器，服务器返回识别到的车牌号以及预定的车位信息，然后根据车位坐标计算出当前车辆应该往哪个方向行驶。
- 3) 车辆可以根据方向指示器的箭头以最短的路径到达预定的车位。
- 4) 要求系统有容错能力，当车辆没有按某个方向指示器的箭头行驶，或者当前路口的方向指示器没有正常工作，仍然可以从当前位置以最短路径到达预定车位。
- 5) 作为实时系统，要求可以实时处理车辆信息，保证系统的可靠性。需要对图像识别模型进行加速。

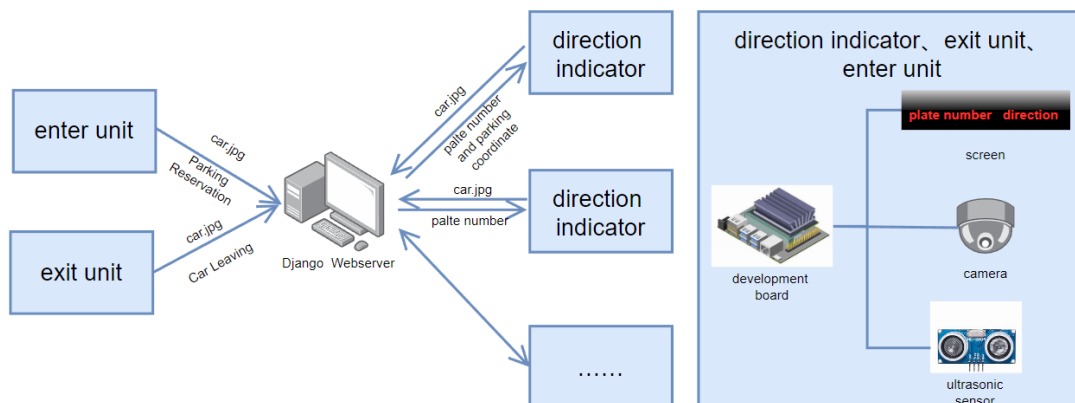


图 2.1 系统架构图

## 2.2 基于深度学习的目标检测

### 2.2.1 基于 YOLOv5 的目标检测算法

由于复杂的背景环境会对车牌检测造成干扰，而车辆停放在拥挤的街道或有大量其他物体的区域时，车牌可能被遮挡或与背景混合，导致检测困难增加。基于 YOLOv5 的车牌检测相比图像检测具有显著的优势。它利用先进的目标检测架构和特征提取技术，能够高精度地检测和定位车牌。同时，YOLOv5 还具备较快的推理速度，适用于对实时性要求较高的场景。这意味着它可以在短时间内快速准确地识别车牌，提供高效的车牌检测解决方案。

鉴于车牌识别的特点和需求，训练出来的 YOLOv5 模型无法达到实时性要求，因此我们需要对训练出来的模型进行压缩和加速，这里使用 OnnxRuntime 对模型进行加速，可以将训练好的车牌识别模型转化为高效且快速的推理模型，从而在车牌检测应用中实现更高的性能和响应速度。这种加速技术可以提升系统的实时性能，使车牌识别能够更快速、准确地完成，提供更好的用户体验<sup>[6]</sup>。

使用 Netron 可视化工具描述的车牌识别算法的 YOLOv5 网络结构图如图 2.2 所示：

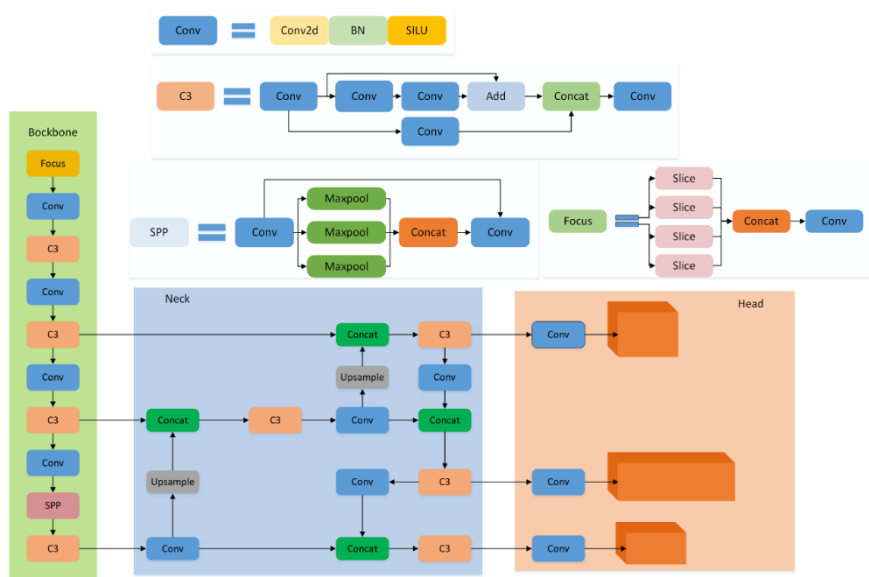


图 2.2 网络结构可视化

### 2.2.2 基于 CRNN 的车牌识别算法

在前面使用 YOLOv5 模型提取出车牌的位置，这时候我们需要去提取车牌里面的车牌号，传统 OCR 技术具有一定的局限性，对复杂场景实用性很差，对光照条件比较敏感，对尺度和角度变化的鲁棒性较低。因此采用基于 CRNN 算法的车牌识别。



CRNN (Convolutional Recurrent Neural Network) 是一种结合了卷积神经网络 (CNN) 和循环神经网络 (RNN) 的混合模型, 主要用于文本识别和语音识别。CRNN 通过 CNN 提取图像特征, 并利用 RNN 进行上下文建模和序列识别。它在文本识别领域表现出色, 能够处理各种文本样式和长度, 并具备良好的特征提取和上下文建模能力<sup>[7]</sup>。CRNN 网络结构图 2.3 如下所示:

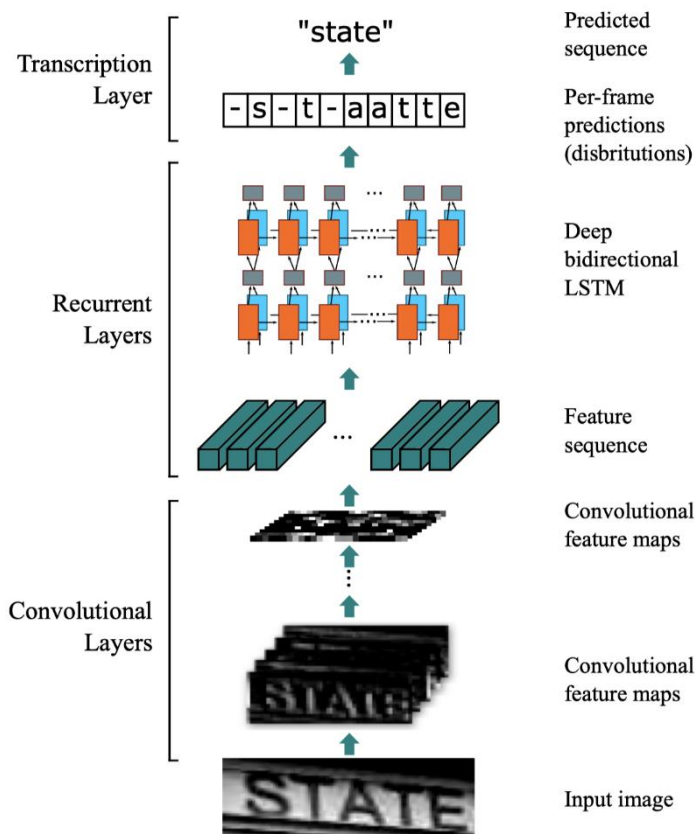


图 2.3 CRNN 网络结构图

CRNN 算法相比于其他算法在文字识别结合了 CNN 和 RNN, 可以处理不同大小和字体的文本, 对长文本的处理能力很强且具有较高的准确性和鲁棒性。

### 2.2.3 模型训练

本项目采用迁移学习的策略去训练深度卷积神经网络。即在训练好的 YOLOv5 模型上用自已的数据进一步训练, 本次训练采用 CCPD 和 CRPD 数据集进行训练, 以提高在车牌检测中的泛化能力。CRNN 我们采用 GitHub 上已经给出的预训练模型, 然后利用 YOLOv5 截取的数据集对 CRNN 模型继续训练。

本次训练分为两个阶段首先训练 YOLOv5 的车牌检测模型, 之后再通过 YOLOv5 模型检测出车牌位置并进行截取, 截取出的图片放入 CRNN 上继续训练, 在第一阶段 YOLOv5 的训练中, 我们对载入的预训练模型权重冻结在卷积层, 仅仅训练全连接层, 训练 40 个 epoch, 之后我们将卷积层解冻, 继续训练 40

个 epoch，这样我们可以在短时间内获取一个检测器，之后为了使得模型在对车牌位置的特征提取精度更高，我们解冻了所有的卷积层，进一步训练网络，训练过程 loss 曲线的变化如下图所示：

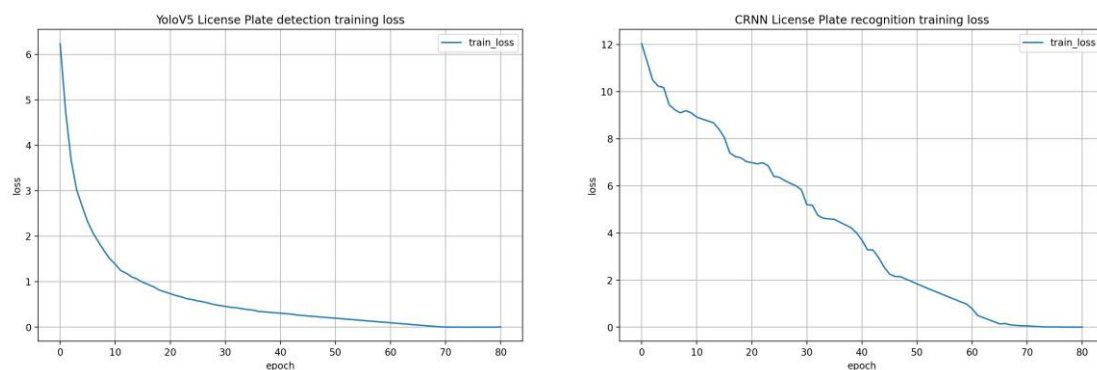


图 2.4 loss 变化

## 2.3 硬件系统

### 2.3.1 硬件平台介绍

硬件平台采用 Jetson Nano 和树莓派 3B+等嵌入式开发板，由于硬件资源有限，如果条件允许只采用一种开发板也是可以的。

Jetson Nano 是 NVIDIA 公司研发的基于 GPU 处理器的嵌入式开发板，其 ARM 端移植装载 Ubuntu18.04LTS 系统，支持目前流行的 TensorFlow, Caffe/Caffe2, PyTorch, keras 等 AI 框架和算法，同时包含 OpenGL、Opencv 等 GPU 的运行环境以及相应的调用接口<sup>[8]</sup>。Jetson Nano 支持两路 CSI 视频输入接口和 USB 摄像头的输入，能够同时捕获两路 CSI 视频以及接入的 USB 摄像头数据并实时显示处理。Jetson Nano 也能够将图像数据信息通过系统视频编码器转化为 H.265 视频流格式，实现在远端控制台实时显示。

树莓派 (Raspberry Pi) 是一款卡片式计算机，自带蓝牙与 WIFI 模块，它具有 4 个 USB 主控制接口 (其中 2 个是 USB 3.0)、1 个 RJ45 有线网接口、耳机音频输出插口、2 个微型高清视频 HDMI 输出接口和一组 20×2 的 GPIO 引脚。板上的 CSI (Camera Serial Interface, 摄像头串行接口) 用于连接摄像头。树莓派 GPU 支持 OpenGL ES 3.x、硬件加速的 OpenVG，和高至 4Kp60 HEVC 视频硬件解码。本项选取树莓派 3B+开发板，处理器是 64 位的核 心架构 (Cortex-A53 和 Cortex-A72)，运行基于 Debian 的 Raspberry Pi OS。

树莓派这类计算机结构简单、体积小、耗电低，却拥有与普通计算机几乎相同的功能和性能，可以很方便地植入各种应用系统中。

### 2.3.2 硬件及传感器介绍

本项目采用了 HC-SR04 超声波传感器及 0.96 寸 OLED 显示屏、IMX-219 摄像头模块以及 USB 摄像头连接开发板构成方向指示器。

超声波传感器一般用于机器人，小车的避障，物体的测距，液位检测，停车检测等领域。本项目采用 HC-SR04 超声波传感器模块，超声波传感器模块上面通常有两个超声波元器件，一个用于发射，一个用于接收。电路板上四个引脚：VCC、GND、Trig（触发）、Echo（回应），工作电压与电流为 5V，15mA 感应距离为 2~400cm，感测角度不小于 15 度。超声波测距原理为：声波速度为 340m/s（空气中的音速在 1 个标准大气压和 15°C 的条件下约为 340m/s），通过记录发射时间和接收时间，计算出间隔，根据（声波速度 \* 时间间隔）/ 2 就可以得到距离。

传感器工作原理为超声波工作时超声波模块的触发角（Trig）输入至少 10us 以上的高电位，即可发送超声波，在触发角（Trig）发送超声波“响应”角（Echo）接收到传回的超声波这段时间内，“响应”角（Echo）位始终呈现出一个高电平的状态<sup>[9]</sup>，可以从“响应”角（Echo）持续高电平脉冲的时间来换算出被测物的距离，如下图 2.5 所示。

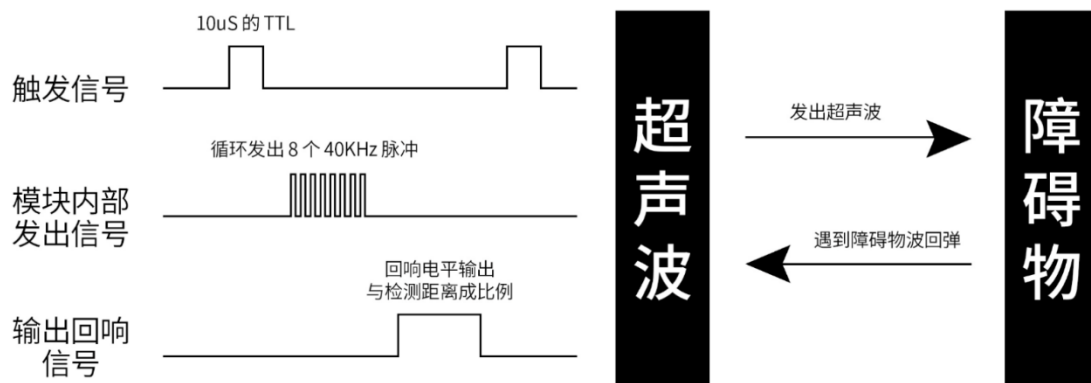


图 2.5 超声波测距原理图

OLED 屏幕即有机发光二极管（Organic Light Emitting Diode）。OLED 由于同时具备自发光，不需背光源、对比度高、厚度薄、视角广、反应速度快等被认为是下一代的平面显示器新兴应用技术。本项目使用的是 0.96 寸 OLED 显示屏，分辨率为 128\*643，通过 IIC 接口控制。屏幕采用 SSD1306 进行驱动，SSD1306 是一个单片 CMOS OLED/PLED 驱动芯片，可以驱动有机/聚合发光二极管点阵图形显示系统。由 128 segments 和 64 Commons 组成。该芯片专为共阴极

OLED 面板设计，SSD1306 在发送或接受任何信息之前必须识别从机地址。设备将会响应从机地址，后面跟随着从机地址位（SA0 位）和读写选择位（R/W# 位）。

摄像头采用 Logitech C920 USB 高清摄像头，静态分辨率为 1280×960，支持最大帧率为 30FPS，支持自动对焦。

### 2.3.3 硬件系统架构

硬件连接如图 2.6、2.7 所示。超声波传感器 trigger 引脚连接 jetson 开发板 13 引脚，设为输出模式；echo 引脚连接 18 引脚，设为输入模式(树莓派 18 脚连 ECHO，7 脚连 TRIGGER)，传感器检测到高电平脉冲的时间换算成距离，设定判断阈值为 5cm，当汽车驶过，超声波传感器检测到有车遮挡，距离小于 5cm 时，返回一个 True 值，摄像头开始抓拍。传感器检测到距离大于阈值（此处为 30）即表明汽车驶离，摄像头停止抓拍。

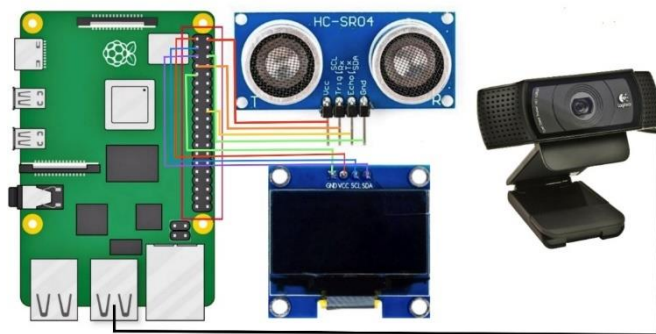


图 2.6 硬件连接图

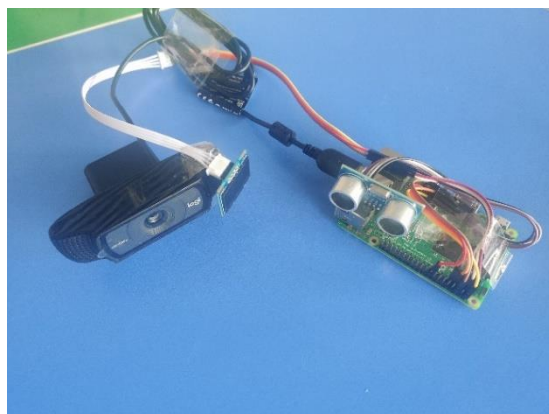


图 2.7 硬件实拍图

屏幕连接 jetson nano 的 3, 5 引脚，使用 I2C1 接口（树莓派 3B+同样使用 I2C1，连接 3,5 引脚），使用 Adafruit\_SSD1306 库，设置字体为宋体，当车辆行驶过方向指示器，超声波传感器检测到车辆，开启摄像头进行抓拍，传输照片给服务器进行识别，服务器返回车牌号与车位指示方向，在 OLED 屏幕上进行显

示，第一行显示车牌信息，第二行显示“请根据指示寻找车位”，第三行显示车位指示箭头方向。显示效果图如图 2.8 所示。



图 2.8 0.96 寸 OLED 显示效果图

## 2.4 软件系统

### 2.4.1 后端系统

后端部署采用 Django 去进行程序开发，对于每个与开发板的通信都会利用 socket 去进行通信，在后端创建线程池去开启每个 socket 服务。当 Django 启动的时候，主线程会对所有服务进行开启，在后端会一直监听 socket 是否发送消息。如果在入口检测到车子进入会对汽车进行抓拍，发送到服务器端，服务器会分配最近的空位，然后写入数据库。如果在路口检测到车子那么也会进行抓拍发送到服务器，服务器会对车牌识别去查预分配的停车位然后对这辆车子进行导航。

如果在出口检测到汽车，对车子进行抓拍发送到服务器端，服务器会根据汽车离开的时间计算停车费用。智能停车场导航系统的 Django 后端架构图如下图 2.9 所示：

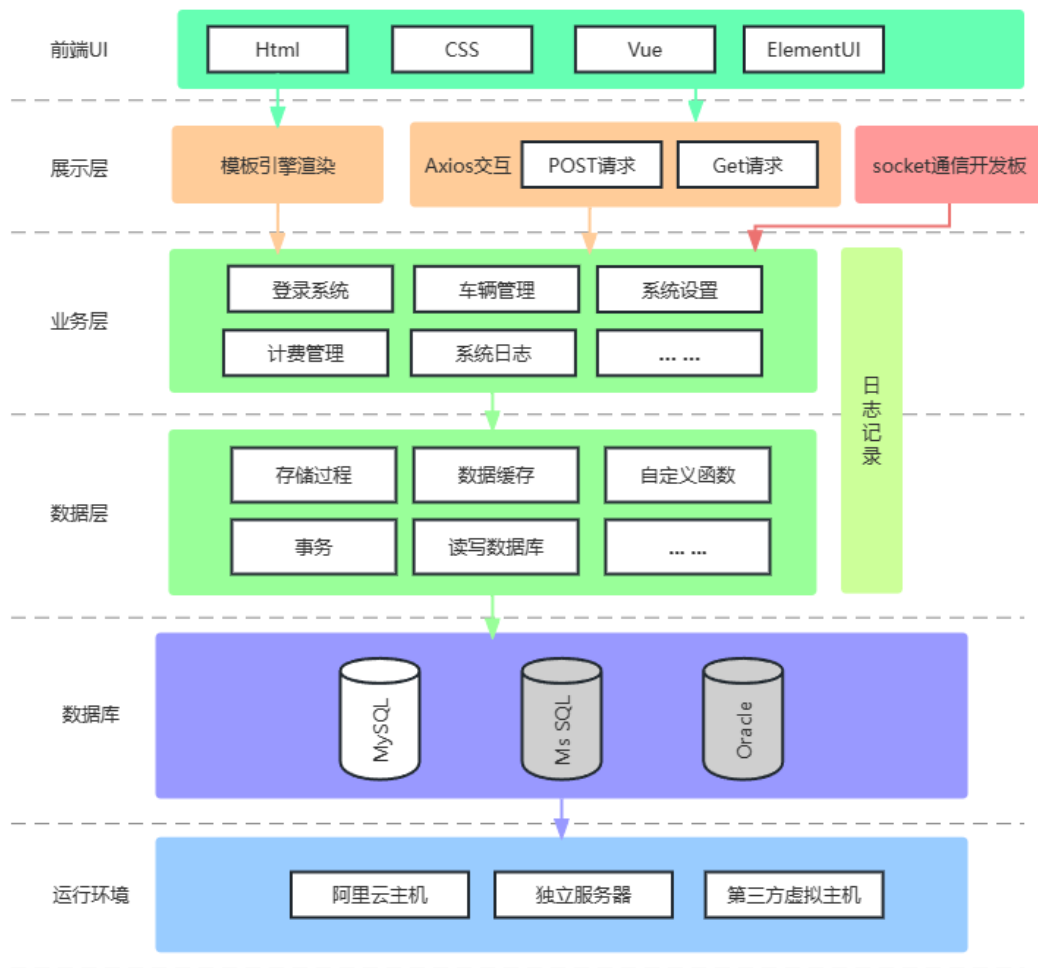
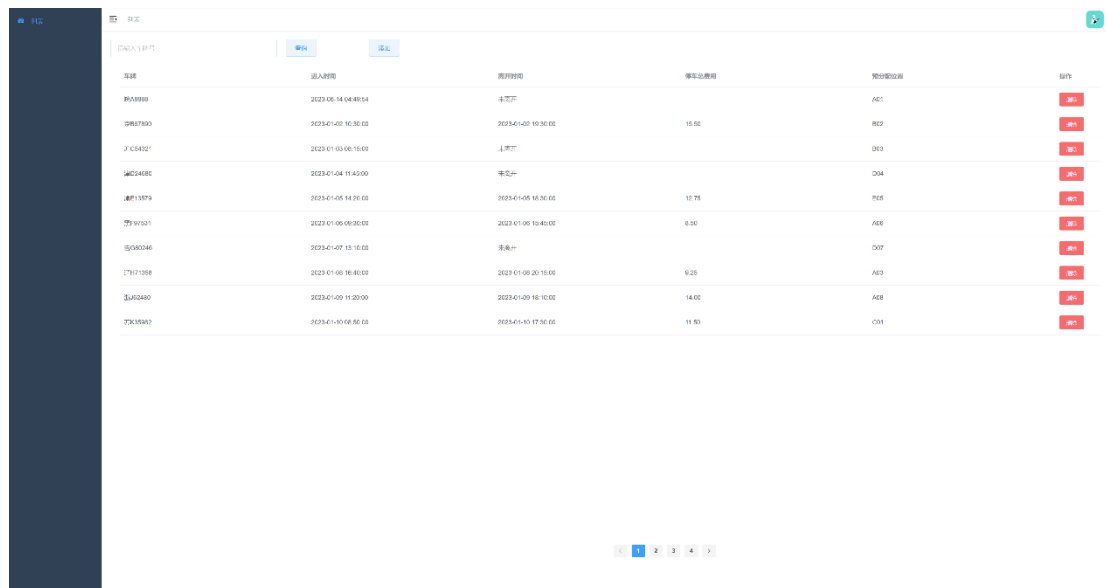


图 2.9 后端架构图

#### 2.4.2 前端系统

为了提供停车场管理员用户与交互的体验，我们采用了前端页面展示的方式，直观地展示车辆的停车时间、预分配车位和收费情况。整个前端采用了Vue.js 和 ElementUI 来进行实现。在车辆进入停车场时，我们会预先分配车

位，而车辆离开时，会自动计算停车费用。以下是页面效果图：



The screenshot displays a web application interface for a smart parking navigation system. On the left is a dark blue sidebar with a '统计' (Statistics) menu item. The main content area has a light gray header with a search bar and two buttons: '查询' (Query) and '导出' (Export). Below the header is a table with six columns: '车牌' (License Plate), '进入时间' (Entry Time), '离开时间' (Exit Time), '停车总费用' (Total Parking Fee), '分配车位号' (Assigned Parking Space Number), and '操作' (Action). The table contains ten rows of data, each with a red '删除' (Delete) button in the '操作' column. At the bottom of the table is a pagination bar showing '1', '2', '3', '4', and '5'.

车牌	进入时间	离开时间	停车总费用	分配车位号	操作
粤G88888	2023-01-14 14:48:54	未离开		A01	删除
京A87890	2023-01-02 10:30:00	2023-01-02 10:30:00	15.00	B02	删除
J C54321	2023-01-03 00:15:00	未离开		D03	删除
冀B54321	2023-01-04 11:45:00	未离开		D04	删除
粤H11579	2023-01-05 14:30:00	2023-01-05 18:30:00	12.75	E05	删除
鄂F 91231	2023-01-06 09:30:00	2023-01-06 10:40:00	8.50	A06	删除
苏B80345	2023-01-07 13:10:00	未离开		D07	删除
川F12356	2023-01-08 16:40:00	2023-01-08 20:10:00	9.25	A08	删除
辽A02450	2023-01-09 11:20:00	2023-01-09 16:10:00	14.00	A09	删除
苏K35642	2023-01-10 08:50:00	2023-01-10 17:30:00	19.50	C01	删除

图 2.10 页面效果图

通过这个用户界面，停车场管理员可以方便地查看每辆车的停车时间和费用信息，以及已经分配的车位情况。这种直观的展示方式可以帮助管理员更好地管理停车场，并提供快速而准确的服务。同时，采用 Vue.js 和 ElementUI 的前端技术，确保了页面的高效性和良好的用户体验。



## 第三章 系统设计与实现

### 3.1 系统整体设计概述

系统由一个服务器和若干方向指示器组成，方向指示器分布在每个路口，如图 3.1 所示。在功能上可以分为车位管理系统和车辆导航系统。车位导航系统主要负责车位的分配和回收，车辆导航系统主要负责对车辆进行无感式导航到最近车位。

当在入口处检测到有车辆时，假设会首先确认是否还有空车位，如果有则识别其车牌号（假设是皖 A123456）并为其分配最近车位，如下图 3.1 红框所示，并且记录车牌-车位数据到数据库。

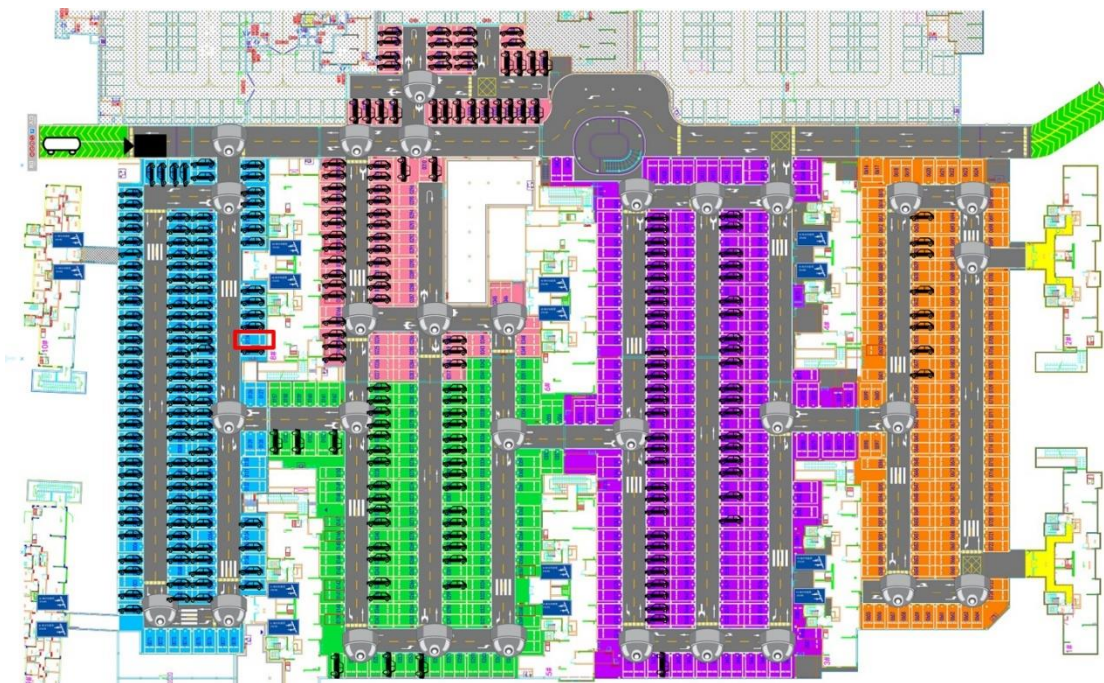


图 3.1 停车场中间时刻状态图

之后当车行驶路口时，如图 3.2 所示，传感器会感应到前方有车，然后开启摄像头连续抓拍图片，并把图片传到服务器，服务器会识别图片中的车牌号，并根据车牌号查找该车需要去的车位，然后把车牌号和车位数据打包发给方向指示器。方向指示器根据算法（具体导航时如何实现的在后续 3.3 节会说明）可以计算出该车应该往哪个方向走可以到达预定的车位，然后在显示屏上显示信息，比如：皖 A123456 ↓，后续每个方向指示器都会做同样的事，最终将车导航到车位。如果车辆行驶过快错过某个方向指示器也无关紧要，依然可以保证它从当前位置可以以最短路径到达预定车位，具体请看 3.3 节。



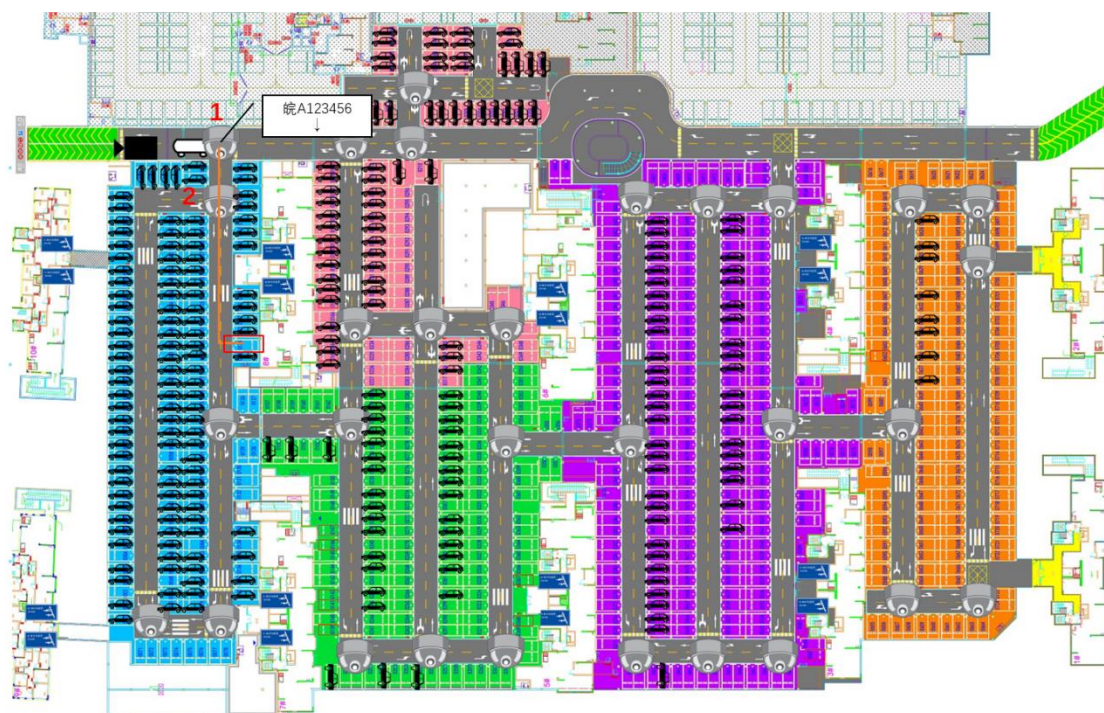


图 3.2 路口导航说明

上面的例子比较简单，请看如下示例，假设当前停车中里入口最近的车位是图 3.3 蓝框所示位置，请看导航系统是如何完成导航的。



图 3.3 单车复杂情况过程 1



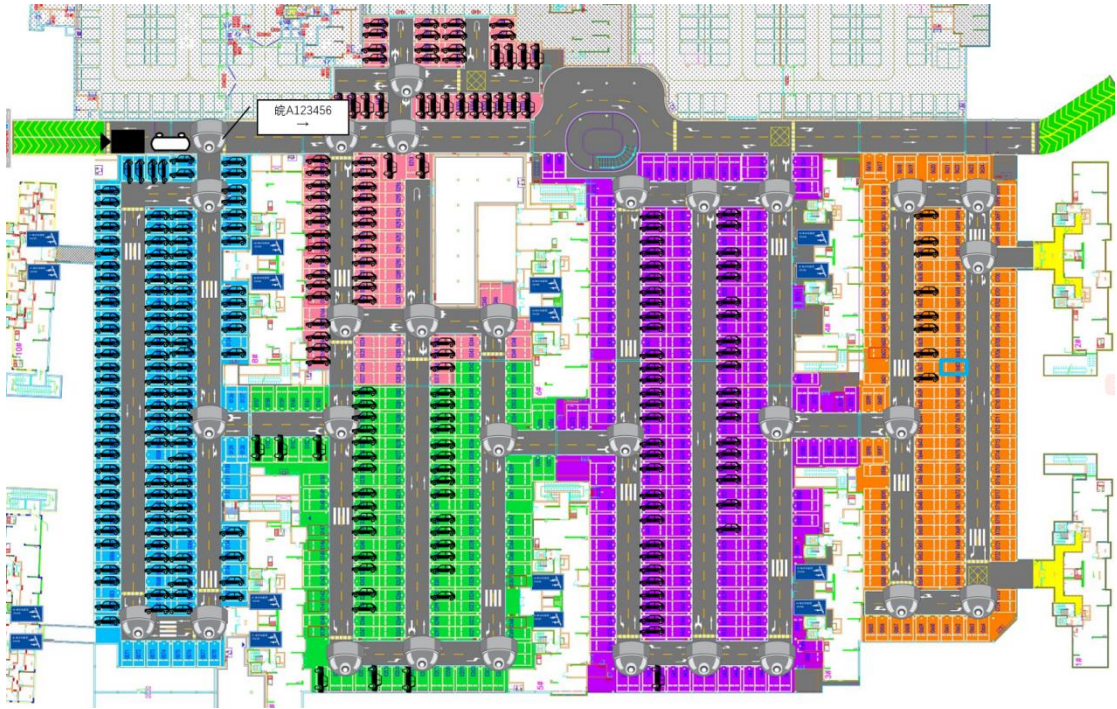


图 3.4 单车复杂情况过程 2

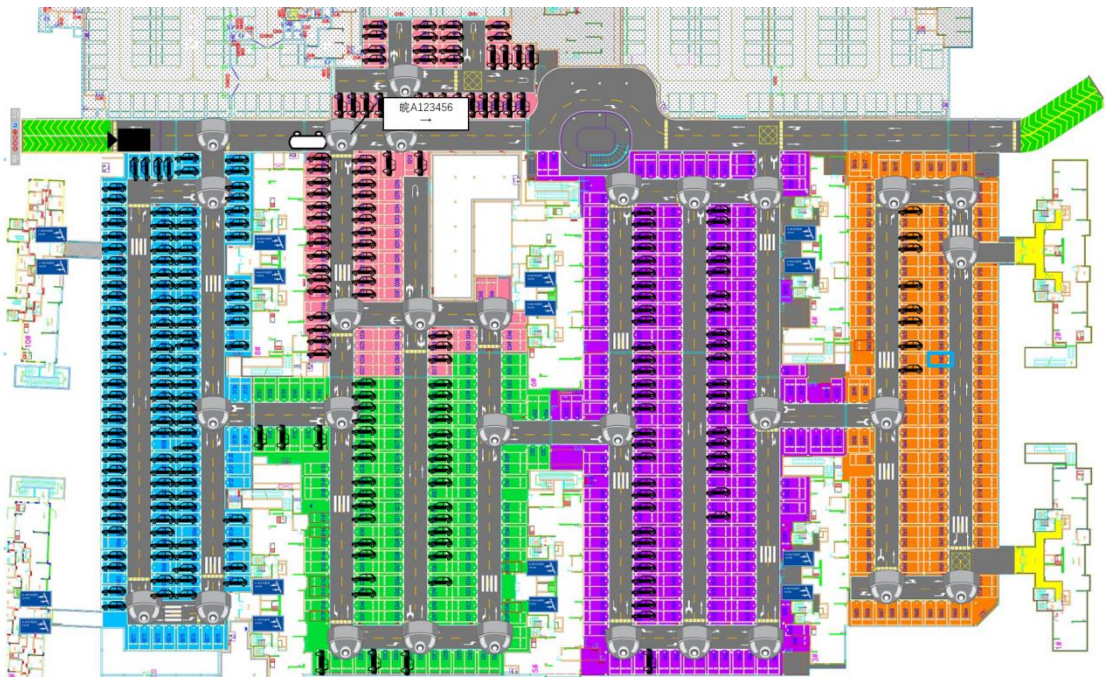


图 3.5 单车复杂情况过程 3



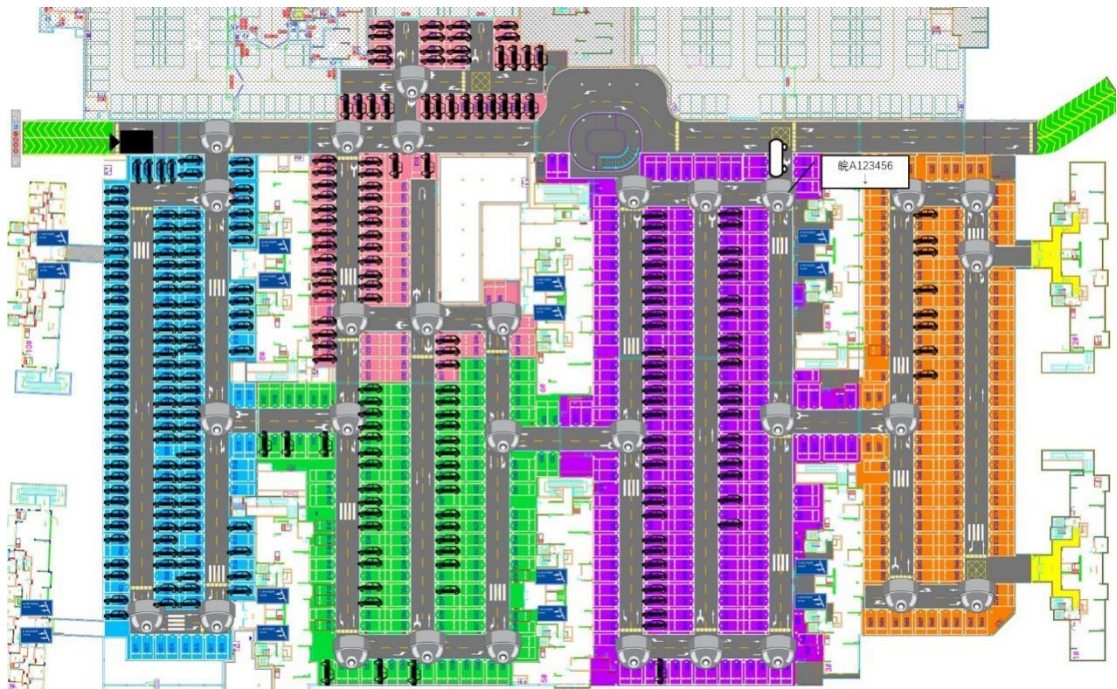


图 3.6 单车复杂情况过程 4

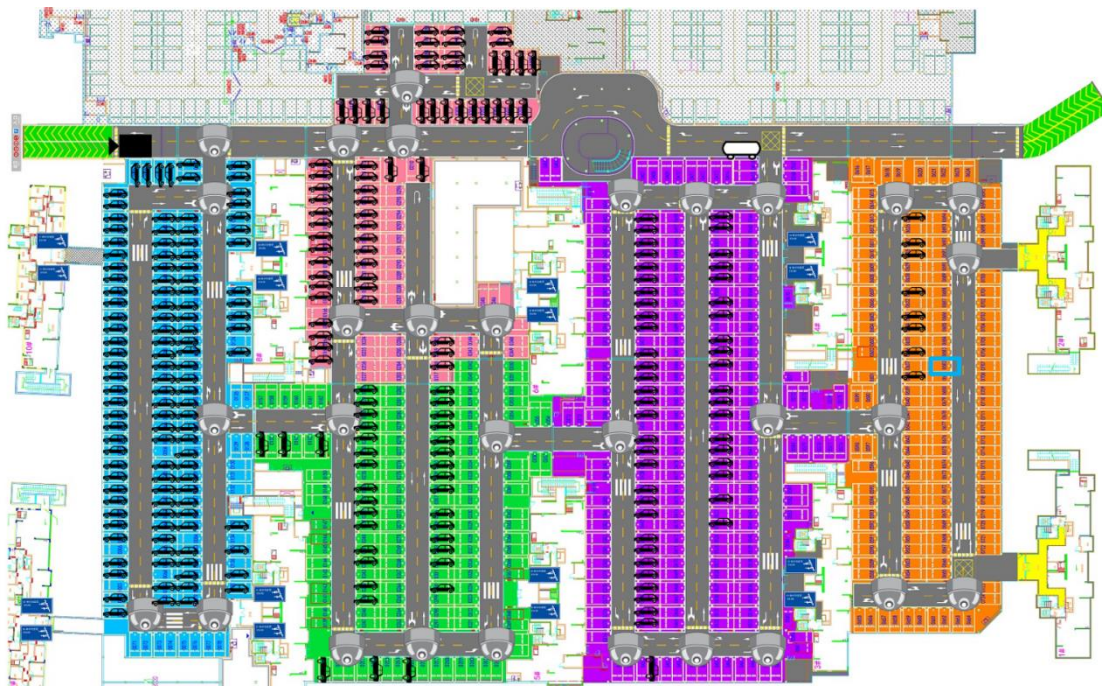


图 3.7 单车复杂情况过程 5



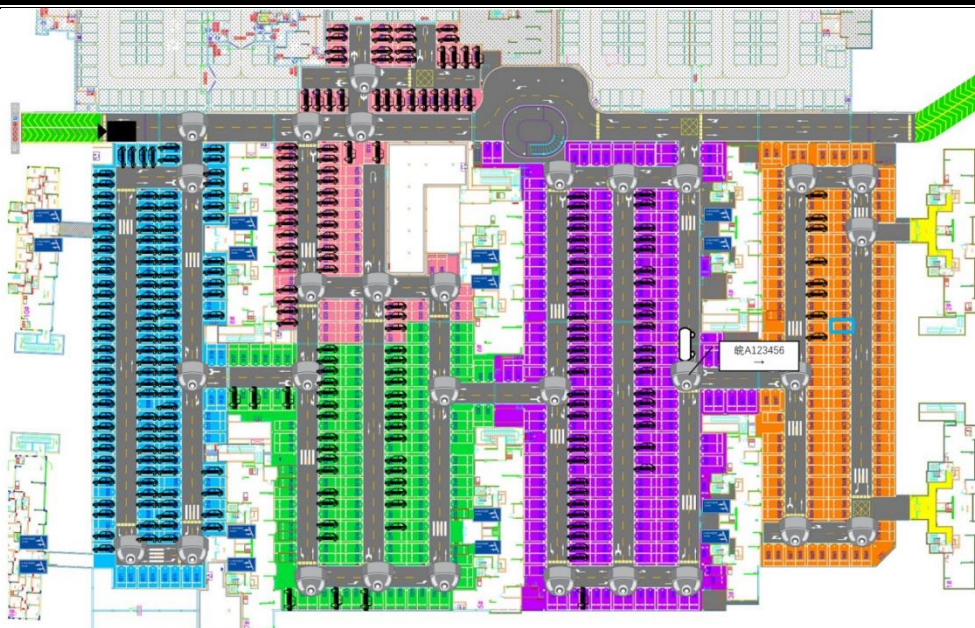


图 3.8 单车复杂情况 6

后续过程就不再展示了，只要车辆跟着方向指示器的箭头走，就一定可以保证它可以到达车位。

特殊情况：如图 3.9，车辆本来按照当前路口方向指示器所显示地箭头直行并按红色路线可到达红框所示位置。

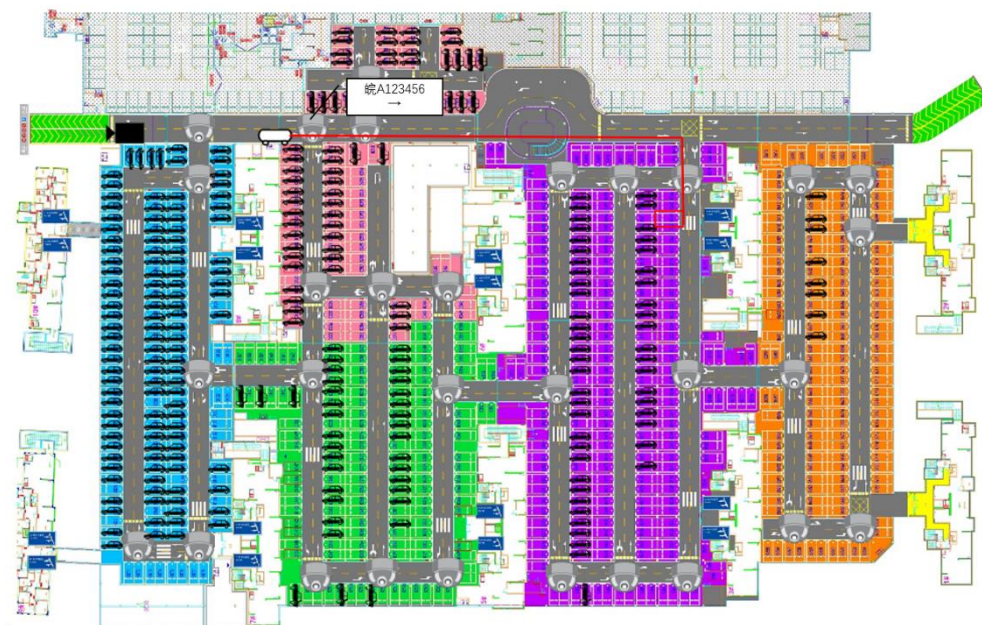


图 3.9 特殊情况说明 1

当用户忽略某个方向指示器地箭头或者该路口地方向指示器故障没有正常工作。如图 3.10，用户选择右拐，当到达下个路口时，方向指示器将会为他重新规划路线。

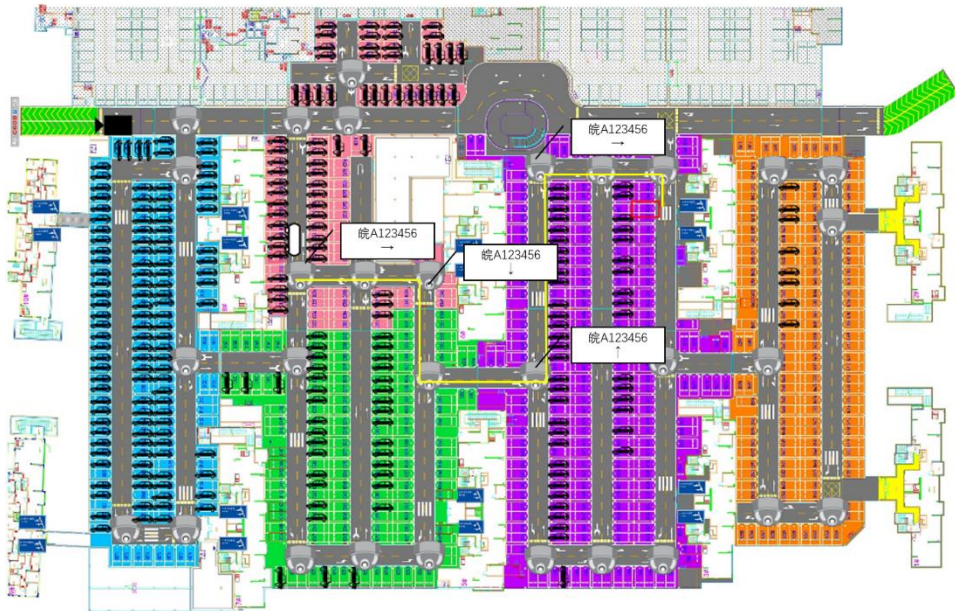


图 3.10 特殊情况说明 2

## 3.2 车位分配实现

车辆驶入停车场入口时会触发感应器，然后摄像头开始连续抓拍图片，并把图片上传服务器，识别识别其车牌号后为其分配最近的车位，并把车牌-车位数据写入数据库。

由于是车辆是运动状态，拍摄的图片不免出现模糊等情况。为了最大限度的保证车牌号的正确性，我们使用以下两个方法。



图 3.11 车牌示例

(1) 正则表达式匹配车牌。根据《中华人民共和国机动车号牌》规则，92 式机动车号牌（现行）由一个代表一级行政区的简体汉字配以一个拉丁字母和一组 5 位数字或拉丁字母组成，通常简体汉字之后的拉丁字母就是决定了该市对该省份的重要性，例如“A”是用于省会（自治区首府）或直辖市的市中心区所注册的车辆，省会以外的城市就会按照其在 1994 年的经济发展水平和影响力

而从“B”开始排列。

(2) 计算前后车牌相似度。由于车牌拍摄是连续过程，即会对一辆车连续拍摄多张图片，但是系统只会对一个车牌号分配一个车位，在次识别出同一辆车不做什么事情。假设车牌通过正则表达式匹配规则，那是不是就可以直接为其分配车位呢？考虑车牌号“皖 AC0001”，首次识别时，系统为其分配车位，在运动过程中或者遮挡，系统可能误识别成“皖 AO0001”，但是其实是同一张车牌，如果再次分配，系统中将浪费一个车位。为了解决这个问题，可以计算字符串的相似度。字符串相似度计算可以由很多方法，我们采用了编辑距离方法。

编辑距离是指将一个字符串转为另一个字符串需要用到的最少操作数。操作包括：插入一个字符、删除一个字符、替换一个字符。例：word1 = "horse", word2 = "ros"。编辑距离是 3，即用 3 次操作就可以将 word1 转为 word2。说明：第一步、horse -> rorse (将 'h' 替换为 'r')。第二步、rorse -> rose (删除 'r')。第三步、rose -> ros (删除 'e')。

当前后两个车牌的编辑距离过小时，说明车牌号高度相似，系统不在为其分配车位。项目中我们设置的阈值为 2，即如果前后车牌编辑距离小于 2，判定为同一张车牌。

### 3.3 数据存储实现

目前常见的数据库分类可以分为关系型数据库如 Oracle、MySQL、SQL Server 等等，以及非关系型数据库，又称 NoSQL，使用键值对、文档、列式、图形等方式存储非结构化或半结构化数据，如 Redis、MongoDB、HBase 等。关系型数据库数据库的访问速度相比非关系型数据库会慢很多，原因在于关系型数据库需要把数据存到磁盘中，磁盘需要 IO 读写，而非关系型数据库比如 Redis 是直接将数据存到内存中查找速度很快，并且关系型数据库通常要求强一致性，保证事务的 ACID 特性，而非关系型数据库通常可以牺牲一致性来提高可用性和性能，采用最终一致性或 BASE 模型。



考虑到数据规模、数据访问效率和响应速度，最终选择 Redis 作为数据存储。在系统初始化时会将车位信息存储到 redis 中，单条数据如图 3.12 所示：

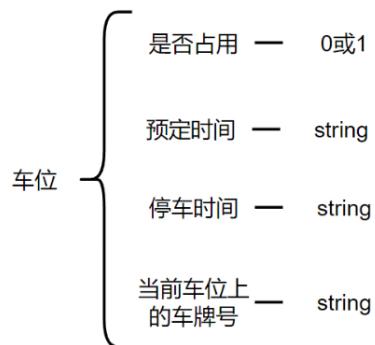


图 3.12 单条数据展示

除了车位信息，还会记录车牌号与车位的键值对数据，当车辆驶入停车场预定车位时记录，当车辆驶离停车场时删除。

### 3.3 停车场车位无感导航实现

目前大多数大型停车场都建在地下，受限于停车场规模和室内导航精度，只能靠视觉寻找路径。这里我们提出一种基于 A\*算法的车位路径导航算法<sup>[10]</sup>。

首先要想计算最短路径先要将停车场抽象成矩阵图。以图 3.13 所示停车场平面为例。与之相对应的矩阵如图 3.14 所示。

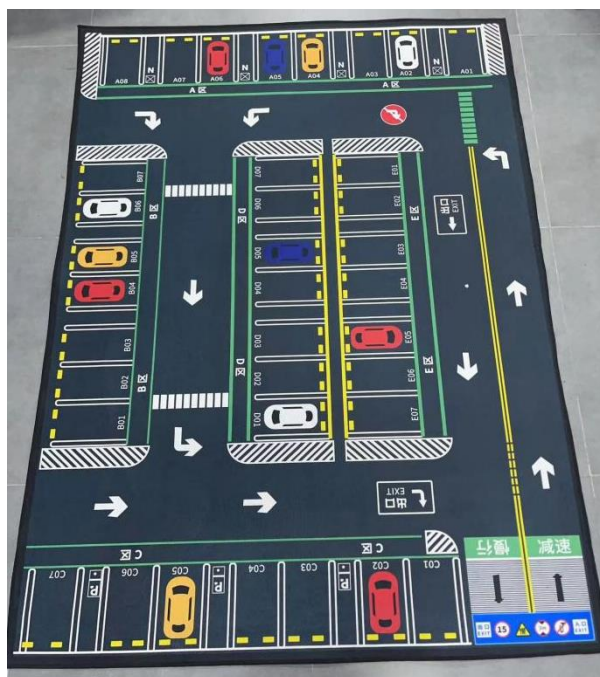


图 3.13 停车场示例图

```

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 0
[2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2], # 1
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 2
[2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2], # 3
[1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 1], # 4
[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2], # 5
[1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 1], # 6
[2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2], # 7
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 8
[2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2], # 9
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 10
[2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 11
[1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 12
[2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2], # 13
[1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 1], # 14
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2], # 15
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], # 16
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2], # 17
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], # 18

```

图 3.14 对应矩阵图

其中 0 表示路径，1 表示墙壁或者不可达，2 表示车位。即每个车位都有独一无二坐标，然后可以根据以某点为起点求到达某个车位的 shortest 路径。方向指示器位于每个路口，自然也可以有其对应坐标，比如图 3.11 中红框所示位置。系统初始化时会把停车场对应的矩阵图发送给每个方向指示器，然后每个方向指示器根据矩阵图和自身的坐标计算到每个车位的 shortest 路径，然后就可以得出从当前位置应该往哪个方向走可以最快到达预定车位。然后会到达下一个路口，这个路口的方向指示器也会做同样的事情，最终将车辆导航到车位。

例：假设矩阵如下表 3.1 所示，第一行和第一列为坐标序号，初始点在坐标 (5, 0) 位置，需要前往 (5, 5) 位置，假设车辆按照 (5, 1)、(5, 2)、(4, 2) 行驶，在 (3, 2) 点可以有两个方向可以走，向左或者向右。(3, 2) 点的方向指示器就会以当前的坐标为起点以该车辆需要前往的坐标为终点计算 shortest 路径，计算可得 shortest 路径为：(3, 2)、(3, 3)、(3, 4)、(4, 4)、(5, 4)、(5, 5)，取路径上的第二个坐标 (3, 3) 与自身坐标向减得 (0, 1)，表示“—>”方向。在实际场景中要想对车辆实现导航作用，还需要结合摄像头的朝向来显示箭头。

表 3.1 示例矩阵

Y \ X	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	1	1	0	1	0
3	0	0	0	0	0	0
4	0	1	0	1	0	0
5	0	0	0	1	0	2



所以这也就解决了不论当前车辆行驶到了停车场哪个位置，只要它再次跟着方向指示器的指引方向行驶，它就会以从当前位置开始最短的路径到达预定车位，因为每个方向指示器都是计算的最短路径，所以它就会以最短路径（此最短路径相较于一开始就跟着方向指示器走不是最短路径，只是当前位置的最短路径）到达预定车位。

### 3.4 各终端通信实现

本系统架构为经典的 C/S 架构，即一台服务器需要为多个客户端服务。当方向指示器的传感器感应到车辆时开始连续抓拍并把照片发送给到服务器，服务器识别图片中的车牌号，并根据车牌号查找该车预定的车位，然后把信息打包发送给方向指示器，注意服务器需要具备同时为多个方向指示器服务的能力。

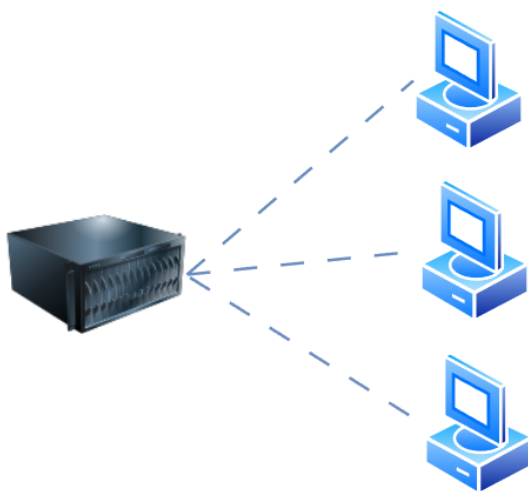


图 3.15C/S 架构图

当客户端数量较少时可以为每个连接的客户端创建单独的线程为其服务，且这些线程会一直存在，不论客户端有没有发来消息。这样的好处是能够对客户端的响应及时，但是当客户端数量很多时，系统就要创建大量的线程，而创建一个线程，操作系统需要为其分配一个栈空间，如果线程数量越多，所需的栈空间就要越大，那么虚拟内存就会占用的越多。系统创建的线程：1、PID 号数值限制。系统会为每个线程分配一个独一无二的 PID 号，而 PID 号是有上限的，linux 中最大的线程号存放在 `/proc/sys/kernel/pid_max` 中。2、内存限制。具体来说，在 32 位 Linux 系统里，一个进程的虚拟空间是 4G，内核分走了 1G，留给用户用的只有 3G。如果创建一个线程需要占用 10M 虚拟内存，那么一个进程最多只能创建 300 个左右的线程。

---

在处理大量客户端的情况中，多路 IO 复用是很常见的一种手段，它可以让一个线程或者进程同时监听多个文件描述符（通常是 `socket`），从而提高系统的并发性能和资源利用效率。多路复用常用的实现方式有三种：`select`、`poll`、`epoll`。其中性能和扩展性最好的是 `epoll`，它是 `linux` 特有的一种多路 IO 复用机制，它使用了一个内核维护的事件表来存储文件描述符和事件，只需要在注册或取消文件描述符时进行拷贝，而且返回的是就绪的文件描述符列表，不需要遍历。在系统开始时每个方向指示器将于服务器建立 `TCP` 连接，服务器将这些 `socket` 加入到 `epoll` 对象中进行监听，当有客户端发送来消息时，服务器会创建工作线程处理信息，工作线程的生命周期只到处理完信息，这与创建单独线程去监听 `socket` 不同。

---

## 第四章 后期工作

### 4.1 停车场导航改进

后续可以通过增加车位传感器来实现实时识别车位的占用状态，以便动态调整路径规划，解决用户不按预先分配车位停车导致的问题。车位传感器可以安装在每个停车位上，通过检测车辆的存在与否来判断车位的占用状态。

一旦车位传感器检测到车辆占用了预先分配给其他用户的车位，系统可以立即进行反应。通过实时更新车位占用状态，路径规划算法可以在用户到达停车场后重新计算最佳的可用车位，并为用户提供新的导航指引，以避免出现停车位被占用的问题。

为了解决车位被抢占等问题，还可以设计增加智能地锁系统。该系统可以采用舵机模块控制车位的锁定和解锁。当车位上的车辆靠近时，智能地锁可以自动解锁，允许用户停车。一旦车辆离开车位，智能地锁可以自动锁定，确保车位不会被其他车辆占用。

通过结合车位传感器和智能地锁系统，可以实现一个智能的停车场管理系统。该系统能够实时监测和管理车位的占用状态，并通过动态调整路径规划和智能地锁控制，解决用户不按预先分配车位停车和车位被抢占的问题。这样可以提高停车场的利用率，减少用户的停车困扰，并提供更好的停车体验。

### 4.2 反向寻车

当面对停车场空间巨大、视觉特征重复以及反向寻车难的问题时，我们可以考虑引入更智能、高效的反向寻车系统，以提供更好的用户体验和解决痛点。一种可能的解决方案是利用先进的无线定位技术，如全球定位系统（GPS）、蓝牙、无线传感器网络等，结合智能手机应用程序来实现反向寻车功能。这些智能反向寻车系统可以通过智能手机应用程序提供方便的界面，让用户轻松输入查询信息，并获得准确的车辆位置和导航指引。相较于传统的基础设施投资大、维护困难和用户操作繁琐的方法，这些技术方案更加智能化、高效，并且能够显著改善用户的反向寻车体验。

除此之外，随着技术的不断发展和创新，还可能会出现更多创新的解决方案，如车辆图像识别、车辆传感器技术等，以进一步提高反向寻车系统的准确性和便捷性。

---

## 第五章 总结

本项目旨在解决停车场拥堵问题，通过设计和实现智能停车场导航系统，提供车主更便捷、高效的停车体验。该系统充分利用计算机视觉、传感器技术和优化算法，实现车牌识别和车位导航功能。

在项目中，我们采用了 C/S 架构，结合 YOLOv5 和 CRNN 等技术实现了车牌识别功能。通过两阶段的迁移学习策略训练模型，并使用 OnnxRuntime 进行加速，实现了高精度和实时性要求的车牌识别。此外，我们使用 A\*算法搜索最近的空闲车位，并通过方向指示器、摄像头和显示屏等设备进行导航。

在系统实现中，我们使用 Redis 作为存储系统来管理车位信息，并记录停车时间。当车辆驶入停车场时，根据车牌预定车位，并在车辆离开时及时将车位标记为空闲状态，从而提高停车位利用率。同时，我们通过实时监测停车场状况，有效缓解停车场拥堵问题。

在项目的过程中，我们注重用户界面和交互设计，确保系统的易用性和用户体验。同时，我们也关注安全性和隐私保护，采取了加密和数据保护措施，保护车牌信息和用户隐私。

总体而言，本项目的智能停车场导航系统为停车场管理者和车主提供了许多优势。通过提高停车位利用率和缓解停车场拥堵问题，该系统为车主提供了更加便捷的停车体验。此外，系统还具备可扩展性和性能优化的特点，能够应对未来停车场规模的扩大和大规模数据处理的需求。

## 参考文献

- [1] 麻吉辉, 王丽杰, 赵原真, 孙建波. 智能停车场反向寻车系统设计[J/OL]. 哈尔滨理工大学学报:1-11[2023-06-19].
- [2] 刘军. 基于 ZigBee 的智能停车场管理系统的设计与实现[D]. 哈尔滨工程大学, 2012.
- [3] Suhr, J.K., Jung, H.G., Bae, K. et al. Automatic free parking space detection by using motion stereo-based 3D reconstruction. Machine Vision and Applications 21, 163 - 176 (2010).
- [4] A. Grazioli, M. Picone, F. Zanichelli and M. Amoretti, "Collaborative Mobile Application and Advanced Services for Smart Parking," 2013 IEEE 14th International Conference on Mobile Data Management, Milan, Italy, 2013, pp. 39-44, doi: 10.1109/MDM.2013.63.
- [5] 杨智璇, 刘辉. 基于轻量化定位的地下停车场导航路径优化研究[J]. 价值工程, 2022, 41(26):121-123.
- [6] 李世伟. 基于 YOLOv5 算法的目标检测与车牌识别系统[J]. 电子技术与软件工程, 2022(01):138-141.
- [7] 华春梦, 臧艳辉, 马伙财. 一种基于 CRNN 的车牌识别算法研究与应用[J]. 现代信息科技, 2021, 5(20):78-81+86. DOI:10.19850/j.cnki.2096-4706.2021.20.020.
- [8] 杨强强, 宋子诚, 姜凌昊等. 基于 Jetson nano 的跟随购物小车系统的设计与实现[J]. 电脑知识与技术, 2023, 19(04):7-9. DOI:10.14004/j.cnki.ckt.2023.0194.
- [9] 胡峰涛, 傅自钢, 李建民等. 基于树莓派的人物追踪系统及其小车实现[J]. 电脑知识与技术, 2017, 13(23):170-172. DOI:10.14004/j.cnki.ckt.2017.2438.
- [10] 杨城, 杨进. 基于 A\*算法的进路搜索应用研究[J]. 铁道通信信号, 2023, 59(05):20-25. DOI:10.13879/j.issn.1000-7458.2023-05.23014.