马鞍山学院

Ma′anshan University

2026 届本科毕业设计（论文）

题　　目	基于视觉的循迹机器人控制系统设计
学 生 姓 名	李想
学　　号	**22402137**
专 业 班 级	机器人工程器 **R222**
学　　院	智造工程学院
指 导 教 师	张丰丰
完 成 日 期	**2026** 年 **5** 月 **20** 日

教务处 制

马鞍山学院学位论文原创性声明

本人声明：所呈交的学位论文是本人在导师的指导下，独立进行研究取得的成果。除文中已经注明引用的内容外，本论文不包含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后果，并承诺因本声明而产生的法律结果由本人承担。

学位论文作者： 

日期： 2026 年 5 月 20 日

马鞍山学院学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并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。本人授权马鞍山学院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、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保　密☐，在__年解密后适用本授权书。

不保密☑。

学位论文作者签名：李想　　指导教师签名：张丰丰

日期：2026 年 5 月 20 日　　日期：2026 年 5 月 20 日

马鞍山学院

毕业设计(论文)任务书

课题名称	基于视觉的循迹机器人控制系统设计
院　　别	智造工程学院
专业班级	机器人工程器 R222
姓　　名	李想
学　　号	22402137

毕业设计（论文）的主要内容及要求：

1.查阅与课题相关文献资料 15 篇以上，其中英文资料不少于 5 篇；
2.基于视觉的循迹机器人控制系统设计方案研究，并撰写课题开题报告；
3.通过对基于视觉的循迹机器人控制系统的课题调研，收集相关资料，开展以下研究工作：
（1）调研视觉循迹机器人或相关移动机器人的研究现状与发展趋势；
（2）制定基于视觉的循迹机器人控制系统总体设计方案（包括硬件平台选型、视觉传感器选型、图像处理算法设计等）；
（3）基于视觉的循迹机器人控制系统开发（完成图像采集与预处理、路径识别算法等核心功能模块的软硬件实现与调试）。
4.设计说明书一份（1.5 万字左右、页数为 40 页左右，要求电脑打印，说明书组成包括毕业设计任务书、中外文摘要及关键词、目录、正文、主要参考文献、致谢和附录等）。

指导教师签字： 张丰丰　　**日期：2026 年 1 月 15 日**

摘　要

人工智能和智能制造技术在快速发展，移动机器人在物流，巡检这类领域里的应用正变得越来越广泛，路径识别和跟踪是它自主导航的重要环节。传统的红外传感器循迹方式容易受到光照，地面材质等环境因素的干扰，整体的稳定性也不够，为此，本文基于视觉设计了一套关于循迹的机器人控制系统，能通过机器视觉的方法来提高路径识别在环境适应性和抗干扰性方面的表现。这一整体用智能车作为平台，先使用摄像头采集赛道上的图像，再通过灰度化、高斯滤波以及基于 HSV 颜色空间阈值分割的方式来做图像预处理，本文还运用到小连通域过滤的方法，减少预处理阶段可能产生的额外噪点。到了路径识别阶段，系统整体会结合扫描线检测与边缘检测这两种方式构成的组合视觉算法，提取赛道的左右边界，接着算出路径的中心线以及横向偏差。运动控制方面使用的是 PID 算法，它能根据像素点的偏差实时调节电机的转速，进而让机器人达到稳定的循迹。实验结果表现出，系统可以在嵌入式平台上有效处理复杂光照条件下的图像，准确识别出路径的边界，通过闭环控制使机器人沿着赛道的中心线平稳行驶。本文为移动机器人的视觉巡线提供了一条可行的技术思路，在机器人环境感知与自主运动控制方面同样有一定的参考意义。

关键词：视觉循迹；路径识别；PID 控制；OpenCV；摄像头巡线

Abstract

With the rapid development of artificial intelligence and intelligent manufacturing technology, the application of mobile robots in the fields of logistics and inspection is becoming more and more extensive. Path recognition and tracking is an important part of its autonomous navigation. The traditional infrared sensor tracking method is easily disturbed by environmental factors such as illumination and ground material, and the overall stability is not enough. Therefore, this paper designs a set of robot control system based on vision, which can improve the performance of path recognition in environmental adaptability and anti-interference through machine vision. This whole uses the smart car as a platform, first uses the camera to collect the image on the track, and then performs image preprocessing through grayscale, Gaussian filtering, and threshold segmentation based on HSV color space. This paper also applies the method of small connected domain filtering to reduce the additional noise that may be generated during the preprocessing stage. In the path recognition stage, the whole system will combine the combined visual algorithm composed of scanning line detection and edge detection to extract the left and right boundaries of the track, and then calculate the center line and lateral deviation of the path. The PID algorithm is used in motion control, which can adjust the speed of the motor in real time according to the deviation of the pixel points, so that the robot can achieve stable tracking. The experimental results show that the system can effectively process images under complex illumination conditions on the embedded platform, accurately identify the boundary of the path, and make the robot run smoothly along the center line of the track through closed-loop control. This paper provides a feasible technical idea for the visual inspection of mobile robots, and also has certain reference significance in robot environment perception and autonomous motion control.

Keywords: Visual Line Tracking; Path Recognition; PID Control; OpenCV; Camera Line Following

目　录

1 绪论

1.1 研究背景与意义

循迹机器人是一种能依靠自身感知能力、沿着预设路线自主行驶的智能小型移动设备，在仓库货物搬运、校园物资配送、展馆引导以及智能竞赛等多种日常场景中都得到了广泛应用。循迹机器人有好几种主流的循迹方式，不同方式所适配的场景以及各自的优劣特点差异都比较明显，传统的仓储物流中心仅仅依赖人进行生产的工作模式很难适应当前的仓储物流需求实际应用中[1]，因此智能化转型升级已成为仓储物流中心发展的主流趋势[2]，而其中的具体对比就像下面表 1.1 所示，能更清楚地看出各类循迹方式分别适用于哪些情况。

表 1.1 主流循迹的优缺点对比

巡迹方式	主要传感信息	典型应用场景	优势特点	主要局限性
红外循迹	地面反射强度	简单封闭产线	成本低、实现简单	易受光照和材质影响
视觉循迹	形状、纹理、颜色、边界等	智能制造物流、巡检机器人	信息量大，可识别多类型标记与障碍物	算法与算力要求较高

对比表格里的内容可以发现，不一样的循迹方式对应着不同的应用场景。视觉循迹有信息量大，适配场景更广泛这两点优势，是目前这类机器人最常用，实用性也最强的一种导航方式。和贴磁条、金属感应这些老旧的循迹模式不一样，视觉循迹就像给机器人装上了一双电子眼睛，能让设备像人一样看清路面上的路线，自主判断前进方向，灵活完成直行、转弯这些动作。现在视觉循迹机器人在各类实际场景中的使用变得越来越频繁，对它的整体控制系统进行设计和优化也成了当下机器人领域的一个研究热点。

在特种作业里，输电线路巡检机器人这类平台，能靠稳定性强的姿态和轨迹跟踪控制，在狭窄又危险的环境里沿着设定好的路径平稳移动，这能表现出精准路径控制对保证作业安全和提高维护效率的重要作用[3]。接下来结合当前行业的发展情况，将详细介绍视觉循迹机器人在国内外的发展现状。

1.2 国外研究现状

国外在循迹机器人方面的研究很早就开始了，相关技术也相对成熟。如 2021 年 Carlos Campos 等人发布了 ORB-SLAM3，ORB-SLAM3 是一种基于视觉的实时 SLAM 系统，广泛应用于机器人和自动驾驶等领域[4]。2021 年，波士顿动力

推出专为仓储和物流设计的仓储移动机器人 Stretch。Stretch 用于自动化搬运箱装货物，配备了移动底盘、可伸缩的机械臂及智能视觉系统，可以自主导航并抓取不同尺寸的货箱[5]。

国外在工业和物流机器人这方面，对于导航和路径识别技术已经在商业化产品中得到了广泛应用。图 1.1 展示的是以 Amazon Robotics 为代表的仓储机器人整体，这类机器人早期主要靠地面上的二维码来达到路径引导的目的。随着柔性物流需求的提高，它的新一代移动机器人慢慢用上了多摄像头的视觉感知整体。机器人可以通过识别货架结构和地面标识来完成环境理解与路径规划，还能识别通道边界。在实际运转当中，机器人会把视觉信息和里程计数据结合起来做自主导航，还能在动态环境中实时调整路径，减少对固定标识物的依赖。

图 1.1 Amazon Robotics 仓储机器人系统

在移动机器人的细分领域里，清洁机器人和配送机器人是视觉循迹技术大规模商业应用的两个重要场景。美国的 iRobot 在它的 Roomba 高端系列里越来越多地使用视觉传感器，不再单纯依靠碰撞和红外传感。它会用顶部的摄像头和 SLAM 算法形成家庭环境的地图，识别房间边界，家具轮廓和地面纹理，接着沿着规划好的覆盖路径进行循迹清扫。在较新的产品中，Roomba 的视觉系统可以支持全局地图的形成，还能识别地面上的常见障碍物，比如电线和宠物排泄物。它能在路径跟踪的过程中动态调整局部的轨迹，提高了清扫效率也增强了安全性。Roomba 的任务不完全等同于传统意义上“沿着指定线路巡迹”，不过它那套基于视觉的路径识别与规划模式，和本课题在技术上存在比较高的相似性。

在服务机器人和自动驾驶相关方面，视觉导航技术也成了重要的发展方向。美国 iRobot 公司在它的扫地机器人产品里引入了基于摄像头的视觉 SLAM 技术如图 1.2 所示，它会识别房间边界和家具轮廓，还会识别地面纹理来形成环境地图，然后在规划好的路径上达到高效的覆盖式运动。在自动驾驶领域，Tesla 和 Waymo 等公司形成了以多摄像头为核心的视觉感知整体，用深度神经网络去识

别车道线和可行驶区域，还会识别动态障碍物，再结合路径规划和控制算法来达到车辆的自主循迹与导航。视觉不只是进行简单的路径跟踪，还需要能在复杂环境中完成环境理解和决策控制，这代表了移动机器人导航技术的一个重要发展趋势。

图 1.2 iRobot 扫地机器人 Roomba960

在进入深度学习时代之后，国际上的主流研究路线逐渐转向了基于大规模数据驱动的视觉路径识别框架。而典型的工作有像 NVIDIA 提出的端到端自动驾驶网络，它把前向相机采集到的原始图像通过卷积神经网络直接映射成方向盘转角的输出，在大量真实道路数据的训练之下，这个网络会自动学习到跟车道线、路面边界相关的特征，从而对传统算法中那些复杂的特征工程形成了一种替代。

一些欧洲企业在室内服务机器人和巡检机器人方面更重视视觉和语义理解的结合。比如瑞士的 ANYbotics 公司推出的四足机器人 ANYmal 如图 1.3 所示，它配置有多目摄像头、深度相机等视觉传感器，用来识别楼梯、走廊、管道和设备等，还能在工业场景中进行巡检和路线跟随。这类四足平台主要面向巡检，这意味着视觉路径识别技术对轮式巡线机器人的路径识别和控制也有较强的启发意义。

图 1.3 ANYbotics 研发的四足自主机器人

Google Research 等机构还在室内和半结构化环境里研究依赖视觉完成自主导航和巡迹控制的方式。图 1.4 展示的是基于 Google 的 Cartographer 等方案的整体 SLAM 结构，它主要依赖激光雷达，不过在扩展版本里也加入了摄像头，达到了视觉和激光融合的建图与定位效果。在那些没法安装激光雷达的轻量化机器人平台上，研究者也尝试国使用纯视觉交会定位以及稀疏特征点地图，在地图里标注出重要路径和目标点，机器人再通过实时定位和局部路径规划来完成对指定路线的跟踪与巡线，这为低成本的视觉循迹机器人提供了一种可以参考的思路。

图 1.4 Google Cartographer 的路径图

从国际领域的情况看，基于视觉的移动机器人路径识别和控制技术表现出逐步变化的趋势。先是用传统图像处理方法，之后慢慢发展到深度学习和多传感器融合，现在又到了端到端的决策优化。商用机器人公司和自动驾驶企业会在官网和公开技术文档里公布整体方案，视觉已经成为路径识别和导航方面一种核心的传感方式。它会和激光雷达、雷达、超声波这些传感器一起形成多模态的感知模式。这一发展现状为基于视觉的循迹机器人控制系统设计提供了比较成熟的理论基础和工程经验方面的借鉴。把传统图像处理方法和深度学习结合起来，再加上决策优化方法，是未来视觉循迹机器人研究的一个重要方向。

1.3 国内研究现状

我国在循迹机器人领域的研究起步相对晚了一些，但最近几年，随着人工智能和电子信息技术这些领域的快速发展，再加上国家对智能制造、机器人产业在政策上给了不少支持，视觉循迹机器人这个方向已经实现了比较快的突破。无论是在技术研发还是产业化应用方面，都取得了相当不错的成绩，跟国外先进水平之间的差距也在逐步缩小。杜毅和付太猛等人提出了基于空间推理的实时视觉语言导航系统，该系统在低功耗机器人上集成了高效的空间推理，与传统依赖单一图像级特征相似性来引导机器人的方法不同，改方法将像素级视觉

语言特征与好奇心驱动的探索相结合，使得机器人能够在多样化的环境中依据人类指令进行导航[6]。

有一种常见的实验工作会用 OpenCV 来形成智能车巡线的整体，工作人员会对图像做去噪，二值化，透视变换和线特征提取。这些操作可以让赛道的中心线或是边界线被快速定位，还能在常见教学赛道上达到稳定的路径识别和转弯控制，验证了传统视觉算法在低成本平台上的工程可行性[7]。

从工程落地层面看，AGV 机器人主要采用有轨或无轨的方式搬运货物，其智能化水平相对比较低，码垛机器人可针对不同的货物进行不同程度的码垛，尤其是针对较重的货物，其能够有效搬运货物、实现货物码放功能[8]。国内在工业 AGV 如图 1.5 所示和仓储物流机器人上的视觉巡线研究，表现出一种视觉加多个传感器融合的发展趋势。在这类工程项目里，视觉系统除了进行路径识别，还可以用于货位识别、托盘检测以及作业区域的安全监测，它是移动机器人综合感知整体里一个比较重要的组成部分。

图 1.5 航发机器人 AGV 平台

国内在自动驾驶和视觉感知方面，具有代表性的企业就是百度的 Apollo 自动驾驶平台如图 1.6 所示，还有像京东这样已经在多个城市投放的自动配送车，可以在比较复杂的环境中完成自主导航和路径决策。以及卓驭科技也就是原来的大疆车载，也在车载视觉感知这一块持续推进技术的落地。

图 1.6 Apollo 自动驾驶平台

在工业和物流移动机器人领域，国内企业发展速度很快。未来机器人就主攻带视觉导航功能的叉车机器人，它的产品用 3D 视觉来达到环境识别和路径规划，已经在物流和制造业中得到广泛应用。励微机器人会提供智能物流的整体解决办法，在智能仓储和工厂自动化里做到了自主导航和调度。传统制造企业华晓精密，也在 AGV 整体方面拥有较大的市场规模，它的产品被广泛使用在汽车制造等工业场景当中，AGV 如图 1.7 所示。国内的 AGV 行业正在朝着视觉加上激光融合导航的方向快速发展。

图 1.7 华晓精密复合型双叉举 amr 产品

像国内工业视觉和移动机器人融合发展这方面，海康威视及其子公司海康机器人有较强的代表性。这家公司靠着自己的技术在图像采集和计算机视觉领域积累的技术，把视觉感知能力拓展到工业移动机器人整体当中，进而形成了集环境感知和目标识别于一体的智能物流解决方案。他们的自主移动机器人在具体应用里，一般会使用摄像头和激光传感器，识别地面上的标识和货架的位置，再识别环境的结构，以此来达到对路径的自主规划和动态调整，完成仓储搬运和物料配送这些任务如图 1.8 所示。

图 1.8 海康机器人叉取机器人

和传统上那些依靠固定轨道或者单一传感器的导航方式比起来，海康机器人的设计更注重视觉信息在环境理解里的作用。通过工业相机获取环境图像紧

接着做特征提取，不仅可以帮助机器人完成定位校正，还能协助机器人的识别路径，减少在复杂工厂环境里导航出现的误差。它的架构还能对多台机器人进行协同调度，能在大规模仓储场景里达到路径优化和任务分配的目标。这种机器视觉加移动机器人的融合，表现出国内企业在视觉技术工程化应用方面的优势，也为视觉循迹与自主导航技术在实际工业场景中的落地提供了一份重要的参考。

电商和快递行业发展速度越来越快，京东、阿里巴巴这类电商企业也开始着手智能仓储的布局。从 2014 年开始，京东就启动了智能物流模式的建设工作，还接连在全国多个地方建起了“亚洲一号”智能物流园区。他们在仓储整体里大量使用 AGV（自动导引车）和 AMR（自主移动机器人）进行拣选，搬运和分拣作业，订单处理流程已经达到了高度自动化水平。如图 1.9（c）为京东物流机器人。阿里巴巴旗下的菜鸟网络在全国建起了一系列智能仓储园区，比如“菜鸟未来园区”，这里部署了很多 AMR 与 AGV，用来完成拣选，搬运和分拣作业。如图 1.9（d）为菜鸟推出的智能仓储机器人[9]。

图 1.9 国内仓储机器人应用现状

我国在视觉循迹机器人领域的发展和国外先进水平相比是比较缓慢的，并且在一些方面还是有一定的不足。首先是高端传感器，对于核心芯片这类重要硬件，有一部分还得靠进口，自己研发的高端产品在性能稳定性上还有提高的空间。其次就是软件算法在仔细程度和覆盖范围上都不够，面对复杂环境时的自适应能力，多机器人之间的协同控制，和国外先进技术比仍有差距。最后就是产业化水平还有待提升，像是部分产品停留在实验室阶段，在规模化生产和市场化应用方面的能力较弱。随着国内科技水平的不断提高，再加上相关企业和科研机构持续地投入，这些问题正在逐步得到解决，视觉循迹机器人在国产化和智能化方面的水平也会不断提高，未来可以在更多的领域进行广泛的应用。

1.4 本文主要研究内容

在前面这些研究的基础之上，整体上采用 IFLYTEK 单目摄像头作为主要的传感器，搭建一个包含了视觉采集、图像预处理、路径特征提取以及控制决策在内的循迹机器人平台。在算法层面，构建一套以 OpenCV 为核心的图像处理流程，重点去实现灰度化、滤波去噪、HSV 颜色分割、自适应阈值分割还有形态学处理等预处理操作，并且在预处理变换之后对赛道 ROI 进行裁剪以减少噪点的干扰，从而提升对多种赛道以及光照变化场景的适应能力。在路径识别这个层面，设计一种基于八领域扫描识别边界线与 Canny 边缘检测相融合的双边界提取算法，用来稳健地检测赛道的左右边界，再采用二次多项式拟合的方式去获取中心线和横向偏差，这样一来就能为 PID 控制器提供可信度比较高的输入信息。论文研究中的关键内容如图 1.10 所示。

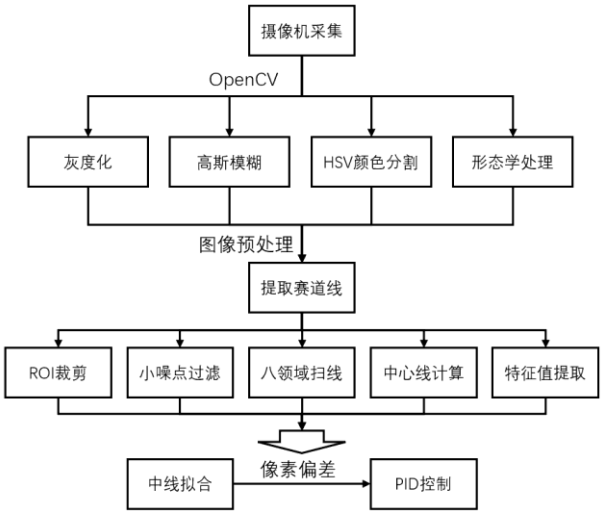


图 1.10 论文主体研究内容

对于传统视觉循迹在复杂场景中的不足，本文进一步引入了轻量化目标检测思想，在保证实时性的前提下借鉴车道线检测中对细长目标进行特征增强与鲁棒识别的方法，为复杂标线与局部缺失工况提供算法储备[10]；同时参考边缘计算设备在保证检测精度与实时性之间权衡的设计思路，将路径识别计算量与嵌入式平台算力进行匹配，以兼顾循迹控制闭环中的延时与稳定性[11]。在控制与场景处理方面，研究了基于视觉偏差的 PID 循迹控制器建模与参数整定方法，建立起偏移量、小车角速度、线速度之间的联合控制关系，并把角速度发布到 ROS Master 的话题当中；确保小车在复合赛道环境下能够保持连续且平滑进行路径跟踪。整个研究通过系统搭建、图像预处理以及实车实验这几方面相结合的方式，对所提出来的算法与控制策略在抗干扰性和可行性上进行综合性的验证。

1.5 论文章节安排

本文结构安排如下：第一章为绪论，阐述课题的研究背景与实际意义，梳理国内外视觉循迹与移动机器人控制相关研究现状，明确本文的研究目标与技术路线。第二章给出基于视觉的循迹机器人系统总体设计，从功能需求出发提出系统架构，构建软硬件总体架构与关键技术路线。第三章围绕 OpenCV 图像处理流程，详细设计图像预处理、赛道左右边界识别与中心线拟合方法，并结合 HSV 和 ROI 裁剪讨论路径识别精度与鲁棒性，其中在路径特征提取环节借鉴电力巡检中面向复杂背景提升细小目标检测能力的图增强 YOLO 思想，以增强细窄赛道线在复杂纹理地面的可分辨性。

第四章重点研究智能车运动控制与特殊场景处理策略，在系统分析整体运动学特性和执行机构动态特性的基础上，构建基于视觉偏差的位置 PID 循迹控制器，给出偏差量提取、高斯滤波与归一化等预处理流程，系统论证偏差量到 ROS Master 话题控制及其对循迹精度、响应速度和稳定性的影响。围绕直线、缓弯、急弯等不同路段的轨迹跟踪需求，比较多种线速度角速度分配与转向控制策略的优缺点，并结合实际硬件约束对控制律进行积分限幅与死区处理。结合单目相机 PID 巡线项目在实践中将比例、积分、微分增益参数通过 Ziegler-Nichols 经典方法调代码参数。

第五章介绍在调试过程中，所使用的调试方法和调整方案以及展示的部分成果，还有 Debian10 系统的部署与调试。还对视觉采集，图像处理，决策控制和底盘执行这些功能模块做了联调验证。在路径识别和运动控制的性能方面，设计了光照变化，路面差异这些特殊情况。开展了路径识别精度，处理时延和系统实时性的测试。之后用闭环循迹实验，测试控制系统在直线，弯道等情况下的跟踪误差，速度稳定性。对比不同 PID 参数，不同速度条件下的实验结果，用数据和实验分析控制策略对循迹效果的影响，为后续的算法优化和工程应用提供依据。

第六章总结了全文做的工作，梳理了在视觉信息方面的处理，运动控制策略和软硬件整体集成等方面的主要研究成果。

2 基于视觉的循迹机器人系统总体设计

2.1 系统需求分析与设计目标

本系统的需求与设计目标主要围绕路径识别、PID 控制以及特殊情况处理这三个方面来展开。在路径识别方面，需要在摄像头视场之内对赛道进行稳定且连续的检测，要求算法即便在光照不均、地面纹理出现变化以及局部存在遮挡的情况下，仍然能够准确地提取出左右边界并计算出中心线的偏差，同时还要支持直线、弯道、圆环等多种不同的赛道形态。在 PID 控制方面，系统应当在嵌入式平台上完成图像的采集、预处理、阈值分割、赛道线提取以及中点像素偏差的计算，并在控制周期之内把控制信息以 ROS 话题的形式输出，整个控制回路需要具备较小的延时与抖动，这样才能保证智能车在中高速运行条件下依然保持平稳的循迹状态。

在功能需求方面，整体要围绕图像采集，图像处理，路径识别和运动控制协同运转具体内容如表 2.1 所示。其中路径识别模块要达到灰度化，高斯滤波，HSV 颜色阈值分割，Canny 边缘化和形态学处理的要求，在这个基础上使用扫描线结合边缘检测提取左右边界，计算中心线与横向偏差，再计算出中心线与横向偏差，还要保证在直线，急弯这些区段内都能保持识别结果的连续性。运动控制模块要把中线像素偏差转化为期望转向量与速度指令，通过 PID 控制达到调速与转向的目的。基于视觉处理的智能车在提高车辆自主感知与控制一体化方面有明显优势[12]。基于 STM32 的巡检小车在巡线，数据采集与异常报警等方面表现出传感融合和远程监控的实用价值[13]，这两项研究为整体在功能集成与任务扩展上提供了可以参考的方向。

表 2.1 功能模块设计

功能模块	具体子功能	性能需求描述
图像处理	预处理、阈值分割、边界提取	在光照变化与局部遮挡条件下保持识别稳定，减少噪点
路径识别	中线拟合、特征值识别	多项式拟合中线以及实现对特殊赛道线的特征值提取
运动控制	PID 调节、控制策略	角速度响应平滑、超调小、满足中低速循迹需求
系统交互与扩展	参数配置、辅助功能、可视化	完成对功能调试的各种辅助

2.2 系统总体架构设计

结合上面提到的功能需求，本课题提出一个由感知层，决策层和控制层组成的三层整体模式。感知层主要由前视摄像头，超声传感器和与之配套的图像采集与数据采集电路构成，负责进行赛道图像采集，距离信息测量及基础传感数据预处理，还能通过高速串口或 USB 等接口把结构化数据发送到上位计算单元。决策层设置在嵌入式计算平台上，运转使用 OpenCV 的图像处理与路径识别算法，一种是灰度化，滤波，自适应阈值分割，二是逆透视变换，边界提取与中心线拟合等功能，同时融合障碍物检测信息，计算横向偏差，航向角及障碍物相对位置，生成期望速度，转向控制量及避障决策。控制层由微控制器，电机驱动模块和执行机构组成，主要接收决策层下发的控制指令，进行 PID 运算及差速分配，把目标速度与转向量映射为左右电机的转速与转向调节量，并实时回传编码器等状态信息，和决策层形成闭环控制链路。这个三层模式在逻辑上达到感知，计算与执行的解耦，可以进行算法升级与硬件扩展，提高整体的可维护性与工程应用价值。

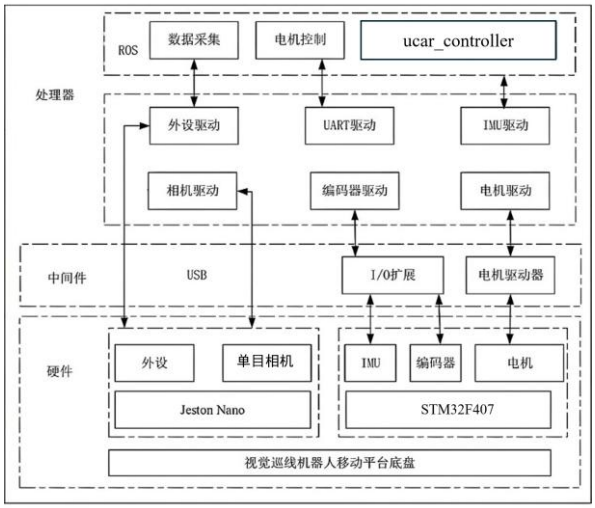


图 2.1 视觉巡线机器人整体框架设计

2.2.1 硬件平台选型与搭建

本系统以科大讯飞二代智能车为硬件载体，如图 2.2 所示，在原厂底盘结构基础完成视觉感知、嵌入式计算与运动执行机构的一体化集成与参数优化。计算单元采用科大讯飞二代车双层异构嵌入式架构上层 NVIDIA Jetson Nano 4GB 主控加下层底层 STM32F402 运动控制单片机，串口通信联动，系统环境是 Debian 10、ROS 20.04。上位机负责图像采集、OpenCV 视觉处理、路径识别与运动决策。下位机采用 STM32F407 这种高可靠单片机，专门承担电机闭环调速、编码器信号采集、舵机角度闭环控制及底层安全保护逻辑，实现感知到决

策再到执行的分层架构。主控板与下位机之间采用异步串行 UART 串口通信通道，视觉传感器选用二代车单目超广角 RGB 相机分辨率与帧率为 1920×1080 @ 30FPS，固定安装于车头中轴线支架，在蓝色赛道加白色引导线场景下可覆盖前方大约 0.3–1.5m 赛道区域，保证充足横向视野与前向预瞄距离。车轮结构采用麦克纳姆轮可以灵活转向，四轮底盘电机配置则是 4 路带 AB 相编码器直流电机。IO 扩展则包含板载预留的 GPIO、UART、I2C、SPI、ADC 扩展排针，可外接传感器、按键、OLED 等。标配外设含有支持外接激光雷达、高清摄像头、语音模块、串口屏、避障传感器。最后小车的供电是整车专用车载供电系统，适配双主控稳定供电，各个模块设计如表 2.2 所示。

表 2.2 功能模块设计

设备选型	设备性能
AI 传感器 1：环形 6MIC 阵列	6 个麦克风、360° 声源定位、5 米收音范围、USB 通信、支持自定义唤醒词（无限次数）、板载声源定位及回声消除算法、支持实现全双工语音交互。
AI 传感器 2：IMU 姿态检测模块	俯仰 / 横滚（静态）：0.05°；俯仰 / 横滚（动态）：0.1° 绝对航向（磁辅助）：0.5° RMS；速度精度：±0.05m/s
AI 传感器 3：RGB 相机	像素：1920×1080@30FPS； 视场角：水平 FOV 124.8°、垂直 FOV 67°、对角 FOV 160°
AI 传感器 4：激光雷达	三角测距，频率：4000–9000Hz 扫描距离：0.12m–16m（扫描频率 4000Hz），0.28m–16m（扫描频率 9000Hz），相对误差为 2%； 扫描角度：0°–360°，角度分辨率为 0.28°

图 2.2 科大讯飞 U-CAR-02

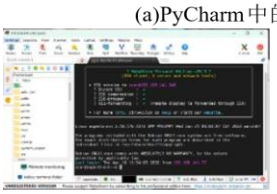
2.2.2 软件框架与开发环境

本课题的软件部分采用了一种两级架构，也就是基于 Python 的上位机视觉处理框架与下位机实时控制固件协同工作。上位机运行在嵌入式 Linux 平台的 Debian10 系统上面，其核心由 OpenCV、ROS 以及自定义的算法库构成，负责

完成图像的采集、预处理、路径识别以及控制量的计算。采集图像后，先经过预处理生成稳定的二值图，然后再利用八领域的思想去扫描并搜索左右边界，接着进行多项式拟合，实时输出中心轨迹的点列以及横向偏差。为了保证在连续弯道和岔路这类场景下能够维持较高的计算帧率，图像处理线程采用了固定ROI与分辨率下采样的策略，平均处理频率可以稳定在20fps左右。上位机通过UART串口这种通信方式，把目标车速、转向补偿值等控制指令封装成定长的数据帧，再通过 ucar_controller 功能包向下发送出去。下位机这边则基于 STM32F407 单片机来运行控制固件，负责完成指令的解析、编码器的测速以及 PID 闭环调速等实时控制任务。Xiaowei Bi 和 Ye Liu 在论文中提出无人机输电线路巡检系统同样采用上位机深度学习感知与机载控制器解耦的架构以兼顾复杂算法与飞行实时性，对本系统软件分层设计具有可推广性参考[14]。

对于嵌入式的开发，往往需要复杂的编译烧录过程，因此本文采取了远程开发的开发环境以便后续的各类调试和代码调试。为了实现 Windows11 系统作为代码环境远程开发 Debian10 系统，本文采取了三种远程连接分别配合完成所有的调试工作，其中 PyCharm 的 ssh 远程连接如图 2.3（a）所示，这是代码调试的主要工具。realVNC 则是部署 Debian10 系统的关键工具如图 2.3（b），他可以直接远程控制小车，也是 PID 控制环节主要调试工具之一。最后则是画面延时较高的 MobaXterm 如图 2.3（c）所示，他的作用是是前两者的结合，PyCharm 无法远程显示摄像头画面，realVNC 调试代码较为麻烦，而 MobaXterm 则都可以满足，但鉴于采取网络传输延迟较高无法实时调试。后续本文将根据不同情况进行不同的远程控制工具的选择。

(a)PyCharm中的ssh远程连接



(b) MobaXterm

(c) realVNC Viewer

图 2.3 三种远程控制软件

2.3 关键技术路线与方案

本课题的关键技术路线通过感知、理解、决策到执行的完整闭环展开，构建从图像实时采集到运动闭环控制的一套视觉循迹方案。系统以单目广角摄像头为核心感知单元，以固定安装姿态与俯角持续采集前方赛道区域图像，通过

USB2.0 高速接口将视频流实时传输至嵌入式 Linux 主控平台，通过 OpenCV 进行视觉处理流水线，保证图像数据与控制指令可以联合控制。

在图像预处理阶段，本文尝试了多种视觉处理方案的组合，并选用组合效果最好的一套预处理方案，其中涉及到的有灰度化、高斯滤波去噪、HSV 颜色空间分割、固定阈值分割、形态学上运算以及 Canny 算子等，通过尝试不同的预处理组合来有效抑制环境光照不均等干扰因素，从而显示出赛道与背景之间的对比度差异。本文还采用了 ROI 对区域进行裁剪的策略，剔除图像上方那些无效的天空与背景区域，这样在保证路径识别精度的前提之下能够大幅降低计算开销，让嵌入式平台可以满足实时性的要求。

在路径识别与计算阶段，系统基于图像预处理阶段的二值图，进行其他视觉处理操作，例如采用扫描线搜索与边缘检测相结合的组合算法。在二值图上按行去搜索赛道的左右边界点，同时利用 Canny 边缘检测来剔除噪声干扰，再通过多项式拟合的方式生成连续且平滑的赛道左右边界曲线，最后计算得到赛道的中心线和横向偏差的控制参数，为后续的决策环节提供可靠的输入信息。

在运动决策与控制阶段，系统采用位置式 PID 作为核心的控制算法，把视觉计算得到的中线像素偏差当作反馈量，再结合转向角速度进行控制，为了防止震荡，同时引入积分限幅以及死区处理的策略，这样可以避免车辆在弯道、转弯角等复杂场景当中出现超调、震荡或者失稳的现象。

本章主要介在后续实验过程中的三个阶段如图 2.4 所示，而解决巡线的任务关键就是围绕这三个阶段展开，将图像预处理、路径识别与计算和运动决策与控制三者融合为一个整体，完成对小车的巡线控制。

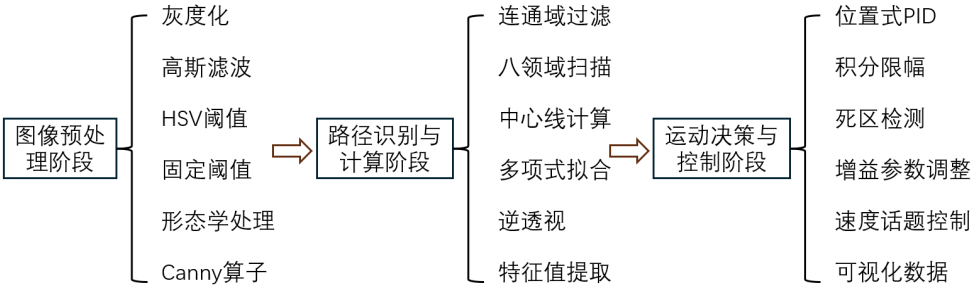


图 2.4 三个阶段的关键技术

3 基于 OpenCV 的路径识别算法设计与实现

3.1 图像预处理模块设计

图像预处理模块的目标是抑制噪声，强化赛道的边缘，再生成稳定的二值图像，为后续的车道线提取提供一个可靠的基础。从设备采集的地图颜色偏暗和地图本身固有的颜色特征来看，本文使用了一套轻量化的预处理流程，这套流程包含灰度化、高斯滤波、HSV 颜色空间阈值分割、Canny 边缘融合和形态学运算等步骤，它在保证实时性的同时可以有效提高赛道特征的清晰度。该流程基于 OpenCV 常用图像处理算子达到，结构简洁、运算量小，适合嵌入式视觉巡线系统[15]如图 3.1 所示。在阈值分割阶段，为进一步提高白色赛道在强光照条件下的抗干扰能力，本文使用了 HSV 颜色空间，这样可以更好地完成白色赛道的提取，把颜色域的特征和亮度域的特征结合起来，最后再通过形态学上的闭运算修复轮廓上的断点。针对赛道光照不均的情况，该方法可以动态地调整阈值，保证赛道线在不同环境下都有较高的连通性。它还能应对局部出现阴影和高亮反光的情况，保证赛道线有不错的对比度效果。

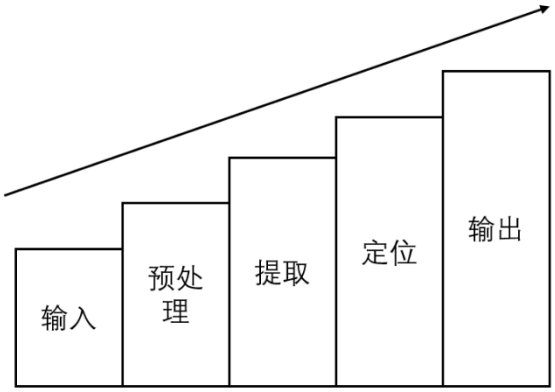


图 3.1 图像预处理各阶段流程

3.1.1 图像灰度化与滤波去噪

在本系统中，为兼顾实时性与识别精度，采集图像先统一缩放至320×240像素，减少后续卷积与扫描计算量；将采集到的彩色赛道图像首先进行加权平均法灰度化，将 RGB 三通道压缩为单通道亮度信息，以减少颜色冗余和后续运算量。灰度值$\mathrm{I(x,y)}$按$\mathrm{I} = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}$计算，当赛道像素为$(\mathrm{R},\mathrm{G},\mathrm{B}) = (210,210,210)$时，就可得$\mathrm{I} \approx 210$，而背景像素$(40,40,40)$对应$\mathrm{I} \approx 40$，形成明显的亮度对比，为阈值分割也提供良好的灰度分布基础。

为了防止车体振动和图像传感器造成的频繁噪声，本文可以使用模板的高斯滤波，对每帧图像进行卷积平滑。在基于摄像头循迹的麦克纳姆轮小车实验

整体中，也把图像灰度化和滤波作为前置步骤，保证 HSV 二值化和赛道特征点提取的稳定性，支持多类型赛道识别和轨迹规划，在多帧实验中，做完上述处理后，预处理阶段平均耗时控制在约 25ms 以内。为后续自适应阈值分割与路径特征提取提供了有效输入。用 OpenCV 形成的巡线系统表现出，灰度化和高斯滤波的组合可以在保证赛道边缘清晰度的前提下，把图像噪声水平减少到能支撑实时边界提取与中心线拟合的范围而处理时间与噪点数如图 3.2 所示。

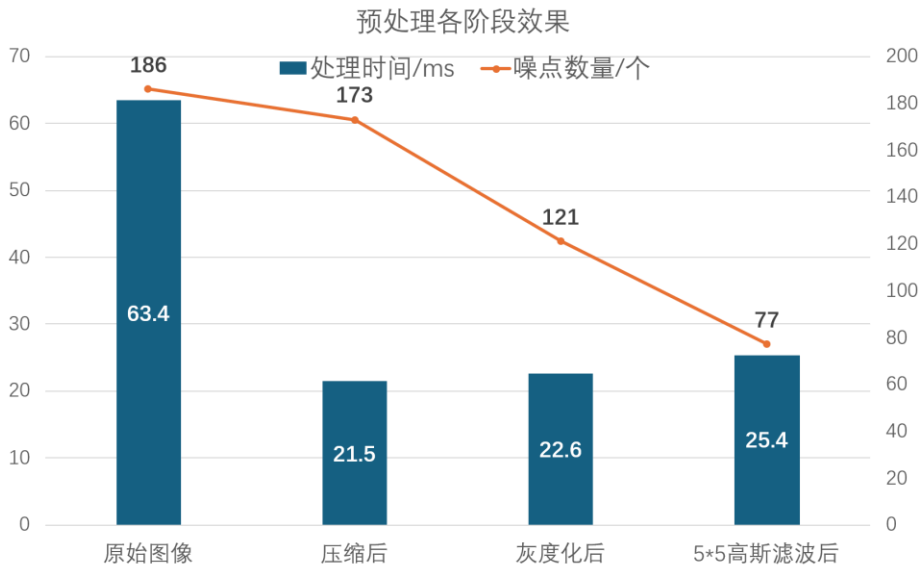


图 3.2 各步骤噪声抑制效果

公式如下：

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

$$I_g(x,y)=0.299R(x,y)+0.587G(x,y)+0.114B(x,y) \tag{3.2}$$

当取$\sigma=1.2$、$(x,y)=(1,1)$时代入式（3.1），高斯权值为

$$G(1,1)=\frac{1}{2\pi\times1.2^2}e^{-\frac{1^2+1^2}{2\times1.2^2}}\approx0.0552 \tag{3.3}$$

在5×5滤波核归一化后，中心像素的输出灰度将接近周围像素的加权平均值，例如邻域灰度为$\{200,205,210,\ldots\}$时，滤波后中心值约为 205，有效削弱尖锐噪声峰，同时保持赛道边缘梯度变化。

灰度化和高斯模糊是传统巡线意义上的基础预处理步骤，其中对于本文来说也是预处理阶段的必要步骤，这可以有效的减少图像的颜色干扰，而有利于

后续的颜色提取基础图像。灰度化和高斯滤波的使用在 OpenCV 中体现如下，例如高斯滤波的参数采用了$\sigma \approx 1.1$，$(x,y) = (5,5)$，灰度化采用 OpenCV 自带的算法，式（3.2）中的公式就是 OpenCV 自带算法的计算原理，在多数情况下使用灰度化和高斯滤波是多数图像处理的基础步骤，本文采取（5,5）是因为多数情况下摄像头采取的图像噪点偏多，则需要模糊更强的图像，巡线所需的细节并不多且可以通过形态学进行弥补，所以在通常情况下，需要强降噪少细节来增强图像的抗干能力。预处理的参数配置如表 3.1 所示，代码体现如图 3.3 所示。

表 3.1 预处理步骤的参数设置

处理环节	参数设置	作用
图像缩放	320×240	降低计算量，提高帧处理速度
灰度化	0.299R+0.587G+0.114B	保留亮度信息，削弱颜色干扰
高斯滤波	5×5 卷积核	抑制高频噪声并保留主要边缘
输出特性	边界更连续	提升后续二值化稳定性

```
GAUSSIAN_KERNEL = (5, 5)
gray = cv2.cvtColor(warped, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, GAUSSIAN_KERNEL, sigmaX: 0)
```

图 3.3 灰度化和高斯滤波在代码中的体现

3.1.2 阈值分割与二值化

对于二值化效果，是为了是图像中的边缘显示更加明显，而在本文中这一步就是决定了是否能提取赛道线的关键所在。阈值分割又分为固定阈值分割和自适应阈值分割。固定阈值二值化直接依据赛道与背景亮度差完成前景分离，计算开销小，适合嵌入式实时处理，但受阴影、反光与局部亮度漂移影响明显，易出现边界断裂或白区粘连。自适应阈值无需反复调阈值，光照均匀变化，如整体变亮或则变暗时，阈值自动跟随，线的完整性更稳定，但是他对噪点的敏感度很高，如果出现一个干扰项会直接打乱原本的轮廓，原因是在二值均方图中某一值突然变大导致。

因此，把固定阈值分割的结果与 Canny 边缘检测图进行融合，一方面依靠前者来保证赛道主体区域的完整性，另一方面借助后者来补偿边缘定位方面的精度。刘潇威、朱泓宇、胡艺、杨喆辰和赵春锋在目标灯塔追踪的研究当中，通过二值图像处理的方式确定了中心位置并实现了 AGV 的稳定控制，这说明简洁的阈值分割方法在实时视觉闭环场景里是具备工程可行性的[16]。本系统选取的

阈值 T 等于 177，凡是高于这个阈值的像素就判定为赛道像素，在测试当中这一设定的表现比较好。为了进一步增强抗干扰能力，本文先将固定阈值与 Canny 算子做了融合处理，由此得到第一版的效果图，具体的公式如下所示。

$$B(x,y)=\begin{cases}1, & I_f(x,y)\geq T\\ 0, & I_f(x,y)<T\end{cases}, E=\text{Canny}(I_f;50,150), F=B\vee E \qquad (3.4)$$

融合后有效前景占比为 F=12.5%，较单纯边缘图更连续，较单纯二值图更利于后续边界搜索，而固定阈值和 Canny 算子融合后的边界连续性如图 3.4 和图 3.5 所示。

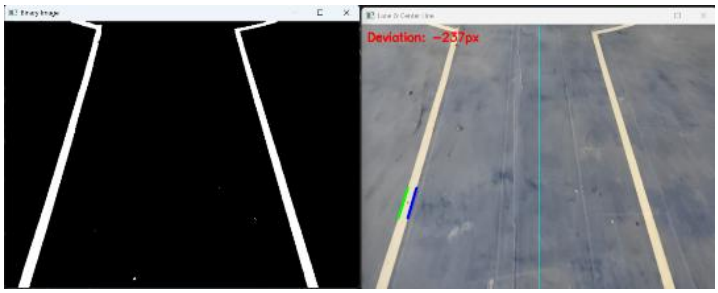


图 3.4 灰度化和高斯滤波在代码中的体现

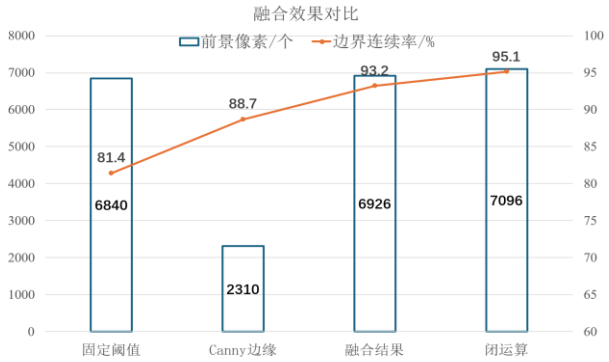


图 3.5 固定阈值与 canny 算子融合对比

3.1.3 基于 HSV 颜色空间的白色赛道阈值提取

HSV 是一种颜色模型，H 是色调，S 是饱和度，V 是明度，调节对应参数即可筛选出所需区域，因此为减弱亮度波动对白色赛道分割的影响，图像由 RGB 转换到 HSV 空间，利用白色在色调上不敏感、饱和度低、明度高的特征进行阈值提取。设像素为$\mathrm{p}=(\mathrm{H},\mathrm{S},\mathrm{V})$，则白色赛道判定为$\mathrm{S}\leq \mathrm{S_t}$且$\mathrm{V}\geq \mathrm{V_t}$。结合实验取$\mathrm{S_t}=43$、$\mathrm{V_t}=200$，可得：

$$M(x,y)=\begin{cases}1, & S(x,y)\leq 43\wedge V(x,y)\geq 200\\ 0, & \text{otherwise}\end{cases} \qquad (3.5)$$

在320×240图像中，检测总像素$N = 76800$，满足条件的白色候选像素$N_M = 8602$，占比

$$\eta = \frac{N_M}{N} = \frac{8602}{76800} = 0.1120 \tag{3.6}$$

计算可知占比为 11.2%。OpenCV 户外寻迹提出以视觉图像处理完成路径区域识别与车辆自主行驶，说明颜色空间分割对复杂环境寻迹具有直接工程价值[17]。与灰度阈值方法相比较，HSV 方法在存在阴影的区域当中仍然能够保持比较稳定的赛道连通性，具体实现的效果如图 3.6 所示。

(a) 未调整过的图像　　(b)修改参数后的图像

图 3.6 调整 HSV 参数的效果图对比

3.1.4 特征融合与形态学闭运算优化

将固定阈值得到的二值图与 HSV 白色掩膜按逻辑或的方式进行融合，既可以兼顾边缘突变的那些信息，也能够利用颜色方面已有的先验知识，从而获得更加完整的赛道候选区域。在复杂背景下的小目标检测研究当中，通过图增强的方式强化了结构之间的关联，这实质上说明保留邻域上的拓扑连续性有助于稳定地表达目标边界，跟赛道线连通性优化的思路是一致的[18]。除此之外，还对处理完成之后的二值图进行了连通域判断，假如某个连通域里面的像素点个数少于 15 个，就把这个区域判定为噪点并忽略不计，这样就能够消除融合之后产生的多余像素点区域，具体效果如图 3.7 所示。

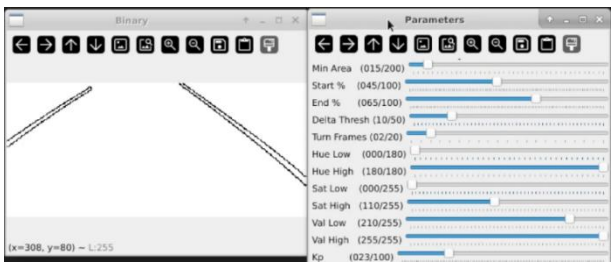


图 3.7 固定阈值与 canny 算子融合对比

对于原始图像，本文先做了颜色反转和左右镜像这两项操作，目的是让图像显示成肉眼所习惯的视角，这样便于后续对关键帧进行判断以及提取特征值。在使用逆透视的过程中发现存在边界丢失的情况如图 3.8 所示，为了让边界线的判断更准确一些，逆透视这个功能并不适合本课题的巡线使用。

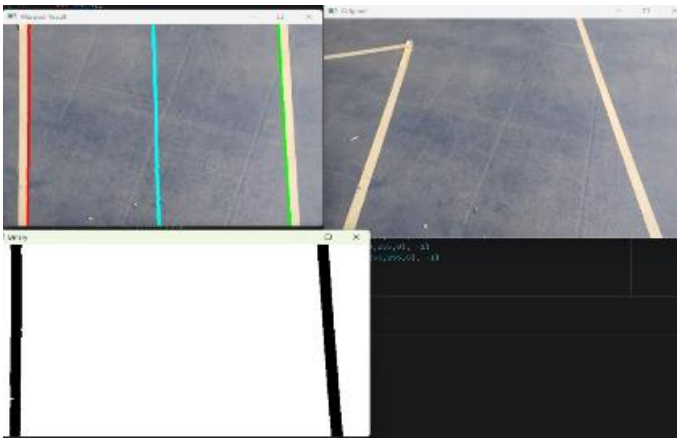


图 3.8 逆透视边界线丢失

最后，本文采用了灰度化、高斯滤波、HSV 颜色阈值分割、Canny 边缘检测、形态学处理、颜色反转、左右镜像以及小连通域判断等一系列操作来完成图像处理阶段的工作，融合之后得到的效果如图 3.9 所示，基本能够满足巡线方面的要求。

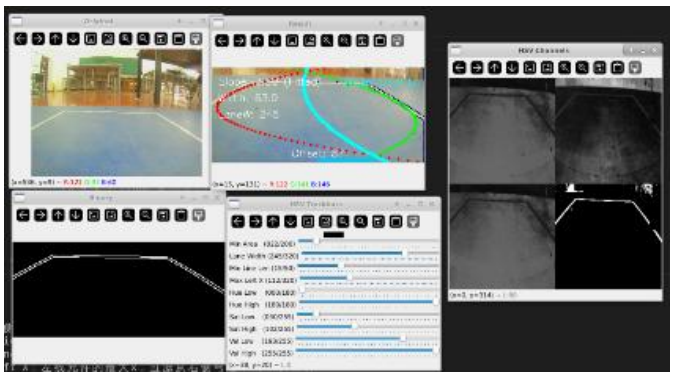


图 3.9 图像识别效果图

3.2 左右边界识别与中心线拟合

在预处理优化后的二值图上面，采用自下而上的多条水平扫描线去搜索左右边界。对于每一条扫描线，从图像的中心位置开始分别向左右两侧寻找第一个满足前景到背景梯度发生突变的像素点，同时结合 Canny 边缘的响应来做一致性校验，这样可以剔除掉噪声干扰以及局部的伪边缘。假设第 i 条扫描线提取

出来的左右边界点分别是 $L_i(x_{Li},y_i)$ 和 $R_i(x_{Ri},y_i)$，那么中心点就是 $C_i((x_{Li}+x_{Ri})/2,y_i)$。

把这些扫描线上得到的中心点集合输入到最小二乘法当中进行中心线拟合，就可以得到赛道在当前视野范围内的整体走向，并据此计算出图像中心与拟合中心线之间的横向偏差。如果某一条扫描线只检测到了单侧的边界，那就利用相邻有效行的边界斜率去做插值补偿，从而保持中心线的连续性。这种方法既兼顾了扫描线搜索在计算效率方面的优势，也保留了边缘检测在定位精度上的长处，比较适合在 ROS 控制周期下进行实时的路径感知。

3.2.1 基于八领域扫描线与边缘检测的组合边界提取

为了提高弯道以及断裂区域当中边界定位的稳定性，本文在中线的底部设置了一个种子点 $S(x_c, H-1)$，然后按照八邻域的方向分别向左和向右进行扩展搜索，其中八邻域的方向集合为{(-1,-1),(0,-1),(1,-1),(-1,0),(1,0),(-1,1),(0,1),(1,1)}。左侧的优先搜索序列依次取(-1,0)、(-1,-1)、(0,-1)、(-1,1)，右侧则依次取(1,0)、(1,-1)、(0,-1)、(1,1)，相应的原理示意图如图 3.10 所示，大致可以理解为顺时针方向对应左侧的搜索，逆时针方向对应右侧的搜索。当某个候选点满足二值前景连续、Canny 响应为 1 并且与当前行进方向之间的夹角变化小于设定的阈值时，就把它判定为有效的边界点如图 3.11 所示；假如连续 k 次匹配失败，那就回退到上一个有效的点，并在其邻域之内重新启动搜索过程。

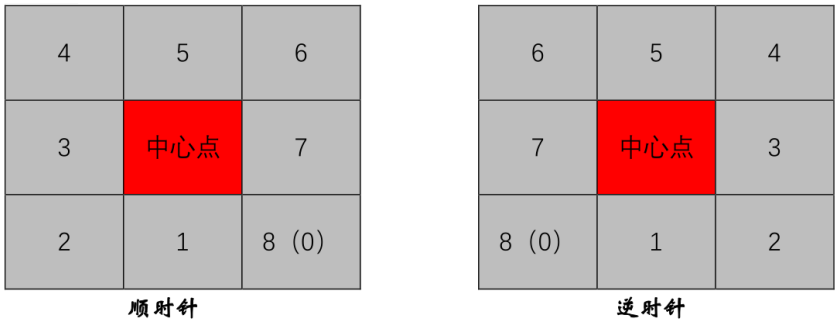


图 3.10 八领域两种实现原理

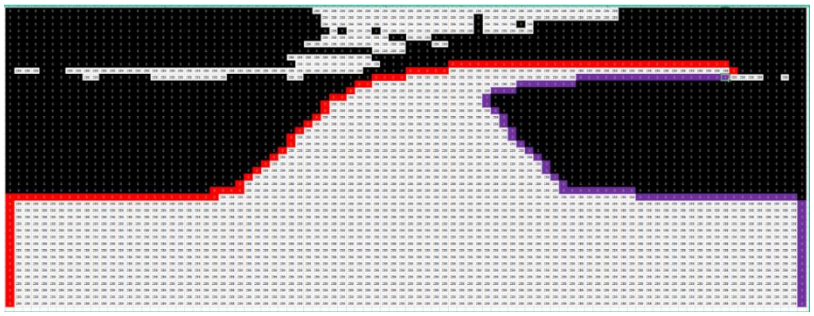


图 3.11 八领域两边扫描结果转化为 Excel 表格

3.2.2 中心线拟合与横向偏差计算

在提取了左右车道线的特征点之后，按照同一扫描行建立对应的中点集，记左侧边界的横坐标为 $x_L(y)$，右侧边界的横坐标为 $x_R(y)$，那么中心线的位置就等于左右两边横坐标之和再除以二，也就是满足两边之和除于二的条件。为了抑制由局部缺失所引发的抖动，采用最小二乘法去拟合一条二阶的中心线，并以图像的主点 x_0 为基准来计算横向偏差 e，其计算公式为 $e=x_c(y_r)-x_0$，其中 y_r 代表所选取的参考行，相关内容如表 3.2 所示，具体的代码实现如图 3.12 所示。Bi 与 Liu 在面向电力巡检的视觉感知研究中，将点云与深度学习结合起来完成故障定位，其核心思想是通过稳定特征之间的聚合来提升在复杂场景下的定位鲁棒性，这与本文利用多行特征点来拟合路径中心线的思路是一致的。

表 3.2 左右边界以及中线拟合像素点

参考行 y_r	左边界 x_L	右边界 x_R	中心点 x_c
20	52	106	79
40	55	107	81
60	58	110	84
80	60	112	86

```
if len(region_pts) < 5:
    return 0.0

xs = np.array([p[0] for p in region_pts])
ys = np.array([p[1] for p in region_pts])

try:
    coeffs = np.polyfit(ys, xs, deg: 2)
    fitted_xs = np.polyval(coeffs, ys)
    center_x = (img_w / 2) - 18
    deviations_from_center = fitted_xs - center_x
    avg_deviation = np.mean(deviations_from_center)
    return avg_deviation
except:
    return 0.0
```

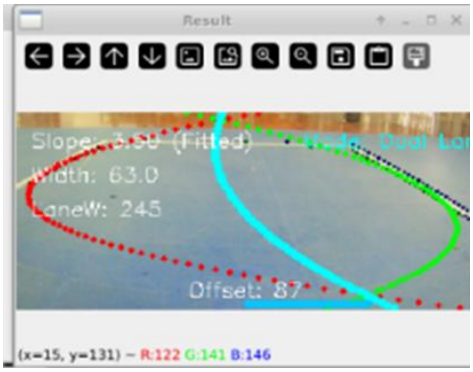


图 3.12 中线拟合代码及部分效果图

3.2.3 右转检测算法设计

右转检测在中心线偏差之外，引入车道线宽度变化、左右边界斜率及连续帧三类特征如表 3.3 所示。设参考区域内宽度为$w(y) = x_R(y) - x_L(y)$，当平均宽度明显增大、右边界斜率绝对值减小时，判为候选右转。该思路与张斐、张泽建、薛明、党小红、余建群将 K-means++、遗传算法和改进 TEB 结合，以提升动态场景实时性与稳定性的处理逻辑一致，即通过多约束联合抑制单一特征误判[19]。为避免噪声触发，设置连续N帧投票机制，仅当候选帧数达到阈值时输出右转标志。

表 3.3 右标志位确认情况

判据项	计算方式	右转触发条件
车道线宽度	$w(y) = x_R(y) - x_L(y)$	下部均值小于上部均值
右边界斜率	$k_R = \Delta x_R / \Delta y$	k_R ∨ 接近 0
连续帧确认	候选帧累计数	$n \geq N_{th}$

右转检测在的关键就是引入了像素宽度计算、斜率计以及右外侧车道线，外侧车道线的作用就是计算宽度，当然为了防止误差，本文选取了图的 75%-55%的区域作为检测区域，这样是为了有效减少断线或则特殊情况时的误差，导致斜率和宽度判断失效。多个判断是为了增加识别的准确率保证在受到外界干扰的情况下仍能正常工作。有了斜率和宽度的双重判断就可以过滤大部分的误判情况，为了防止极少数情况的误判另外加入了连续帧判断，主要就是过滤这些特殊情况如图 3.13 所示。

图 3.13 断点误判

在修改完判断条件后，如果符合所有的条件，对转角进行画圈这样就可以数据可视化判断，宽度条件以及斜率条件设置是否合理，对于可视化内容本文加入了斜率和宽度的实时显示，这样方便观察特征值。具体效果图如图 3.14 所示。

图 3.14 右转角准确判断

3.2.4 终点检测算法设计

终转检测的方式相较于转角更加简单直接，首先在实际运行过程中，小车的摄像头是固定的，因此可以发现当小车到达终点的时候画面底部会出现一条白线，故可以选取该条白线作为判断条件。选取底部 10%-15%的区域作为判断条件，遍历该区间所有的像素值，如果当像素点的占比达到一定条件就可以判定为终点。这样简单的判断必须必须保证判断区域足够准确以及不能选取过大的临界值，不然也会使小车停不下来或则提前停止。

本文选取的判断条件是超过阈值的 6%，是因为在预处理阶段对图像进行了颜色反转，后续判断都是基于识别出来的赛道线进行判断，这样可以有效减少计算量，而使用前置图像会增大很多不必要的计算，因此画面中的黑色像素偏少但是会有一个明显的增加。具体实现效果如图 3.15 所示。

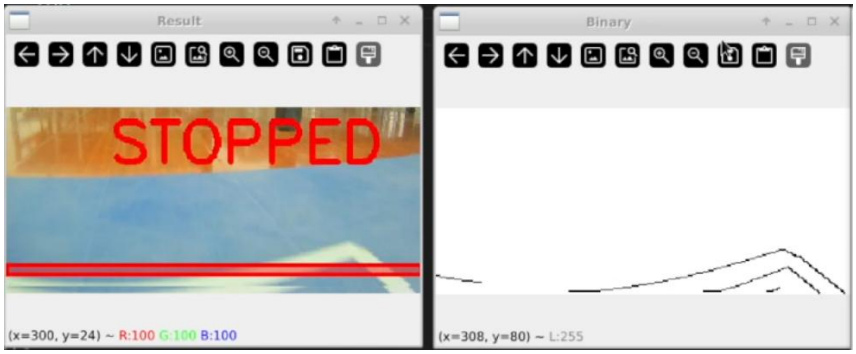


图 3.15 终点位置的准确判断

除此之外本文设置了，停止恢复功能和停止时的阈值显示，具体的日志信息如图 3.16 所示。经过测试已经具备完整的巡线判断条件包括直线、弯道、转弯以及终点，后续配合 PID 的控制就可以完成小车的自主巡线功能。

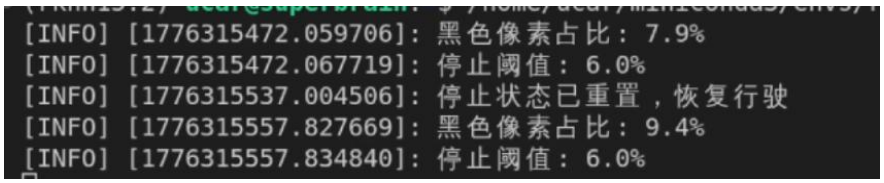


图 3.16 Python 的端口输出信息

4 基于 ROS 与 PID 的循迹运动控制算法设计

4.1 循迹运动控制系统总体架构

循迹运动控制系统以视觉模块所输出的横向偏差作为核心输入量，在 ROS 的不同节点之间完成偏差感知到控制计算再到速度发布这样一个闭环流程，控制层采用的是离散化的位置式 PID 如图 4.1 所示。该系统会把当前的偏差、历史上累积的误差以及相邻时刻之间误差的变化量共同映射成转向的修正量，再以 ROS 话题发布的形式作用到驱动电机的 PWM 信号上，从而让智能车在直线路段以及一般的弯道当中都能够保持稳定的路径跟踪。

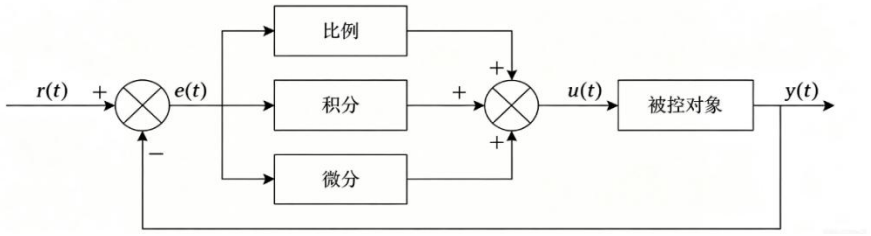


图 4.1 位置式单闭环 PID 模型

在工程实现的过程当中，连续的控制规律会被转换成一组可以在线整定的 Python 文件参数，这样便于底层的程序进行部署和调试。系统在获得中心线的偏差之后，会结合底盘的运动关系对 PID 的输出量进行分配，目的是增强在连续弯道场景下的动态响应能力，减小超调与振荡现象，从而提高循迹过程当中的平稳性以及抗干扰能力。

$$u(t)=K_p\left[e(t)+\frac{1}{T_i}\int_0^t e(t)dt+T_d\frac{de(t)}{dt}\right] \tag{4.1}$$

将其离散化，以 T 作为采样周期，表达式 4.1 表达式 4.2：

$$u_k=K_p\left[e_k+\frac{T}{T_i}\sum_{j=0}^{k}e_j+T_d\frac{e_k-e_{k-1}}{T}\right] \tag{4.2}$$

把表达式 4.2 中的参数整定后得到下面的表达式 4.3：

$$u_k=K_pe_k+K_i\sum_{j=0}^{k}e_j+K_d(e_k-e_{k-1}) \tag{4.3}$$

上面的表达式中：

k——采样序号，k＝0，1，2，……；

uk——第 k 次采样时刻的计算机输出值；

e(k)——第 k 次采样时刻输入的偏差值；

e(k－1)——第 k－1 次采样时刻输入的偏差值；

Ki——积分系数，Ki＝Kp*T/Ti；

Kd——微分系数，Kd＝Kp*Td/T；

将位置式 PID 控制应用到本篇论文的关键就是将输入和运动控制进行整合，本文使用的方式是通过官方给出的 ROS 底盘控制功能包 uca_controller 进行控制，对 ROS 发布对应的速度话题即可实现运动控制，具体架构模型如图 4.2 所示。

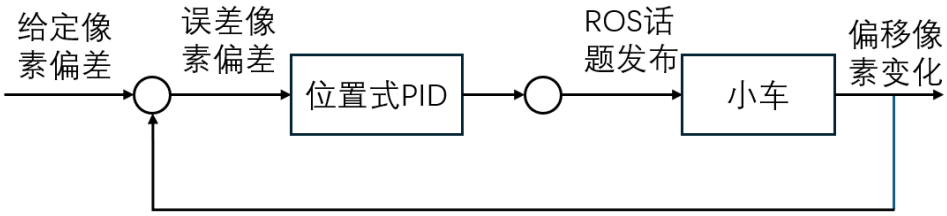


图 4.2 ROS 和位置式 PID 的结合使用架构

4.1.1 位置式 PID 控制框架实现

本系统采用位置式 PID 作为核心的控制算法，它直接输出绝对的控制量，跟底盘角速度指令之间的匹配度比较高，因此比较适合用在视觉循迹的实时闭环场景当中。离散位置式 PID 的表达式见式（4.3）。在这个表达式里头，u(k)代表第 k 个时刻输出的转向角速度，e(k)是中心线在横向上的像素偏差，而 Kp、Ki、Kd 则分别是比例、积分和微分系数，这些系数在代码中的具体体现如图 4.3 所示。崔春蕊把比例、积分、微分这些控制原理转化成可调试的代码参数，并用 C 语言完成了 PID 的编写与整定工作，这说明离散控制算法具有良好的工程实现性[20]。

```
# 比例项
p_term = self.kp * error

# 积分项
if abs(error) < 50:
    self.integral += error * dt
    self.integral = max(-self.integral_limit, min(self.integral_limit, self.integral))
i_term = self.ki * self.integral

# 微分项
derivative = (error - self.previous_error) / dt if dt > 0 else 0
d_term = self.kd * derivative
```

图 4.3 K_p、K_i、K_d 三个增益的使用

4.1.2 ROS 话题控制框架实现

系统基于 ROS 的话题通信机制来实现视觉节点与控制节点之间的解耦，采用标准的消息格式来保证通用性与实时性。视觉模块会把中心线的横向偏差封装成 std_msgs/Float64 类型的数据，然后发布到/lane_deviation 这个话题上面；PID 控制节点订阅了该话题，在计算出相应的运动指令之后，再以 geometry_msgs/Twist 类型的消息发布到/cmd_vel 这个话题，其中 linear.x 表示前进方向的线速度，angular.z 则表示转向的角速度。本文所使用的这套框架不需要自定义消息，兼容性比较好，调试起来也方便，可以直接对接智能车的底层驱动，满足嵌入式平台对于轻量化运行的要求。在虚拟机当中，添加一个小海龟之后就可以查看话题，从而看到完整的话题内容如图 4.4 所示，再使用 rqt 工具就能够看到完整的 tf 树如图 4.5 所示，这就是 ROS 提供的完整运动控制框架。

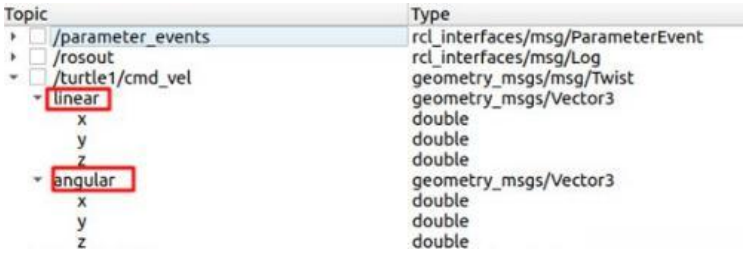


图 4.4 topic 话题列举

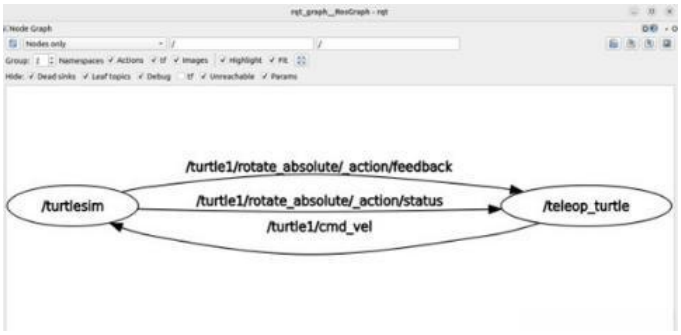


图 4.5 rqt 显示完整的 tf 树

4.1.3 ucar_controller 功能包的介绍

ucar_controller 是科大讯飞智能车官方提供的底层运动控制功能包，就是用来驱动小车底盘的包。它达到了对四个电机，彩色 LED 和 IMU 的控制。这个功能包负责把 ROS 发来的指令转换成电机实际的执行信号，是上层算法与硬件底盘之间一个重要的桥梁。具体来看，该功能包主要有四个方面的功能。一种是订阅/cmd_vel 这个话题并解析其中的线速度和角速度，再转换成左右轮的差速控制量。二，集成了编码器的反馈信号，达到电机转速的闭环调速，从而提高运动的平稳性。第三，通过串口与底层的 STM32 进行通信，完成指令的下发和状态的回传，把硬件层面的细节都屏蔽掉了。在上层的视觉模块和 PID 模块中，

只需要发布标准的/cmd_vel 指令，剩下的工作就可以交给 ucar_controller 去稳定执行。这样一来就大幅减少了开发的复杂度，同时也能保证整体的实时性和可靠性

虽然这个功能包降低了开发的难度，但也在一定程度上增加了通信的延时，使得完整的控制周期变长了，而控制周期的长短又直接受到路径识别耗时的约束。综合考虑到图像的采集、预处理、边界提取、中心线拟合以及 ROS 消息传输等各个环节带来的延迟如表 4.1 所示；面向仓储巡线移动平台的研究中，王钢围绕地面引导线的识别与机器人运动控制之间的协同实现，强调了感知周期与执行周期的匹配对于运行效率和稳定性的基础性作用。

表 4.1 不同环节的关键策略

环节	主要内容	时间约束/策略
图像处理	预处理、识别、拟合	单帧尽量小于 25 ms
控制计算	偏差滤波、PID 更新	周期设为 33 ms
通信发布	ROS 话题发送	控制延迟小于 1 个周期
异常工况	丢线、突变、断点	保持上一帧的控制量

在使用功能包的时候需要运行 launch 文件，launch 文件会调用底层的串口配置参数，这样就不需要单独控制每个轮子的电机转速来控制运动而是转换成 ROS 的话题来控制运动如图 4.6 所示。

```
port: /dev/base_serial_port  # 串口设备名，在startup_scripts的脚本中已经将底里的串口连接为/dev/base_serial
baud: 921600                 # 串口波持率，晓mini的版本因为加了imu数据所以用了更高的921600
bese_frame: bese_link        # 底盘坐标系名
odom_frame: odom             # 里提计坐标系名
odom_topic: /odom            # 里提计话题名
wheel_avg_distance: 0.2169   # 在有轮理与前后轮理之和的均位，用于解算旋转时的角速度
wheel_radius: 0.04657        # 抢半径
vel_topic: /cmd_vel          # 订同的速度话题，向这个话题发布20Hz的速度指令即可以控制小车移动
# publish_odom_tf: true      # 是否提供 base_frame <- odom_frame 的坐标变拽
cnd_timeout: 0.2             # cmd指令超时时间（单位：s），即距离上一个 vel_topic 的多久以后自动作止运动，电
sertal_timeout: 50           # 串口招时（单位：ms，不宜肢动）
linear_speed_max:  3.0       # 底虽最大线速度，用于限制底里的最大平移速度（单位：m/s，有效指围:(0,1.9)），电
angular_speed_max: 3.14      # 底虽最大线速度，用于限制底里的最大旅转速度（单位：rad/s），由电机可达到的实际.
Mileage_file_mame:  car_Mileage_info.txt  # 智能驾驶小车记录行放里程的文件名，该文件持线记录小车电机的行放生
debug_log:  false            # 是否打印ROS的log。
<launch>
  <node pkg="ucar_controller" name="base_driver" type="base_driver" output="screen" >
    <rosparam command="load" file="$(find ucar_controller)/config/driver_params_mini.yaml" />
    <!-- <rosparam command="load" file="$(find ucar_controller)/config/driver_params_xiao.yaml" /> -->
  </node>
  <!-- <node pkg="joy" name="joy_node" type="joy_node"/> -->
</launch>
```

图 4.6 driver_params_mini.yaml 参数和 launch 文件配置

4.2 位置式 PID 设计内容

为了进一步优化循迹精度与行驶平稳性，本节根据视觉反馈的偏差特性对位置式 PID 控制器做了系统性的改进，重点围绕中心线偏差的预处理、死区设

置、积分分离以及积分限幅这几个方面展开，以解决小偏差下的振荡问题、大偏差时的超调现象以及积分饱和等典型控制难题。任斌、冯继和在视觉引导AGV的控制研究中，将坐标偏差、航向角偏差与路径信息结合起来作为巡线控制的依据，说明仅仅依赖单一的横向偏差很难满足复杂路径下稳定跟踪的实际需求[21]。

在第三章处理得到的中心线信息经过坐标系统的统一之后，转换为横向偏差 e(k)，并以图像的中心点作为零点完成归一化处理，再进一步映射成线速度 v 与角速度 ω 两个控制量。其中，e(k)的符号代表了车身相对于赛道中心线所处的偏移方向，而它的幅值则反映了需要施加的修正强度；控制节点会根据偏差的数值大小来确定转向增益，同时叠加输出限幅与死区抑制机制，避免小幅偏差引起频繁的舵量变化，从而形成一条从视觉感知、偏差计算、指令生成再到电机执行的完整闭环链路。位置式 PID 的具体设计内容如图 4.7 所示。

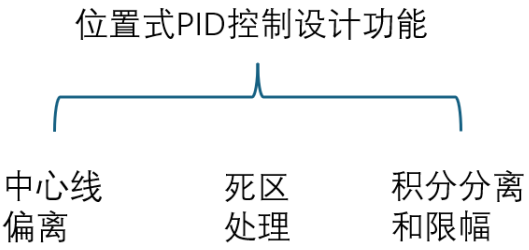


图 4.7 位置式 PID 的设计内容

4.2.1 中心线偏差数据处理

PID 控制器的重要反馈量是中心线横向像素偏差，不过这个数值会因为图像噪声，车体震动和局部标线破损等因素出现毛刺与突变，所以必须对它做预处理。一种处理方式是对原始偏差进行滑动窗口均值滤波。二，要进行异常值的剔除，当相邻两帧之间的偏差变化超过设定的阈值时，就把当前帧判定为无效点并保持上一帧的有效输出。第三，完成坐标与符号的统一处理，把图像坐标系下的偏差映射到车体运动坐标系当中，保证偏差的符号与转向方向之间形成严格对应的关系。经过这几步处理之后的偏差变得平滑且稳定，明显提高了 PID 输入信号的质量。

考虑摄像头抖动、边缘误检及通信瞬时波动会引起偏差，系统采用限幅剔除机制约束输出范围，也就是当像素偏差变化只在 5 以内时不做反应如图 4.8 所示，ERROR_FILTER_SIZE 的值放在全局变量中手动设定。这样输入 PID 的偏差序列就能保持连续稳定。该处理思路与徐大勇提出的规划—控制分层中由内层控制器抑制非线性扰动的思想一致，有助于减弱感知噪声向执行层放大的现象。张古海在电气设备巡检机器人路径跟踪系统中，将路径偏差计算与 PID 控

制器、自适应调整及多传感融合协同设计，以提升定位精度和跟踪性能，这与本文对偏差标准化处理后再输入控制层的思路一致[22]。

```
# 误差滤波
error_buffer.append(raw_deviation)
```

图 4.8 中心线误差滤波的实现

4.2.2 PID 死区处理

在偏差较小时，像素噪声易导致 PID 输出频繁小幅调整，使车体出现蛇形抖动。为此引入偏差死区机制，设定死区阈值 e_{dead} 本系统取 5 像素，控制逻辑为：

$$e(k)=\begin{cases}0, & |e(k)|<e_{dead}\\ e(k), & |e(k)|\geq e_{dead}\end{cases} \tag{4.4}$$

死区处理采用相邻帧突变量与统计阈值联合判定，设当前偏差为$e_n(k)$，则有

$$\Delta e(k)=e_n(k)-e_n(k-1)\vee,\Delta e(k)>\theta\Rightarrow e_n(k) \tag{4.5}$$

式中取 θ 等于 0.35，假如 e_n(k-1)为 0.12 而 e_n(k)为 0.61，那么 Δe 就等于 0.49，这个数值大于 θ，因此当前这一帧就会被剔除，转而用前一个有效数值或者滑动窗口的均值来替代它。这种机制能够抑制由赛道反光、边缘断裂以及误检等情况所造成的突变输入，从而提高 ROS 控制节点输出指令的连续性与可靠性。

4.2.3 积分分离和积分限幅

在视觉循迹整体里，赛道上会同时出现直线和弯道等不同路况，传统位置式 PID 控制器的积分环节在大偏差或者长时间持续偏差的场景下会不断积累，很容易引发积分饱和的问题。一种表现是，智能车进入弯道时偏差会持续增大，积分项快速累加，造成转向输出超调，车体可能直接冲出赛道。二，等到车驶出弯道偏差归零后，积分项没办法快速释放，又会造成转向上的滞后，回正动作也变得缓慢。第三，在路径识别短暂丢失、光照发生突变的异常场景下，积分项的累积会让控制指令持续偏离正常值，严重影响到循迹过程的稳定性和安全性[23]。

针对上述问题，本系统结合视觉中心线偏差的动态变化特性，将积分分离算法与积分限幅算法融合应用，从积分启停控制和积分幅值约束两个方向实现双重抗积分饱和优化，代码实现如图 4.9 所示，这样的优化逻辑与偏差预处理、死区处理形成完整的控制链，适配智能车嵌入式运行环境。如图 4.10 所示，即

使出现小幅度像素偏离超过 5，但是当有积分分离和限幅存在时，PID 控制也不会再次发布速度话题来控制速度，以此达到稳定的控制效果。

```
# 积分项
if abs(error) < 50:
    self.integral += error * dt
    self.integral = max(-self.integral_limit, min(self.integral_limit, self.integral))
i_term = self.ki * self.integral
```

图 4.9 积分分离和限幅

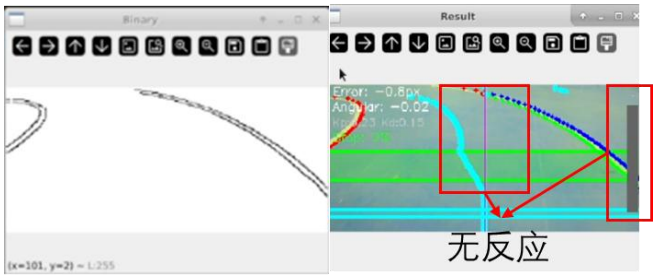


图 4.10 积分分离和限幅效果图

4.3 ROS 话题控制部分

系统所有运动指令均通过 ROS 话题实现发布与调度，本节重点说明速度指令话题发布与定时运动控制实现方式。在基于视觉的智能车循迹系统中，图像处理节点、PID 控制节点、底盘驱动节点分属不同功能模块，若采用直接函数调用或全局变量共享，会导致模块调试困难、扩展性差。ROS 话题通信作为一种异步、点对点、分布式的消息传递机制，可完美解决上述问题，使各模块独立运行、按需通信，同时支持实时数据观测、在线参数调优与多节点协同。话题之间的转换结构如图 4.11 所示[24]。

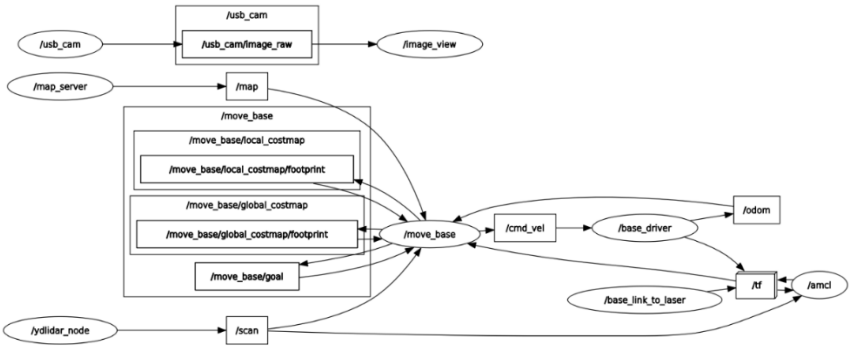


图 4.11 智能车端 rqt 显示完整的 tf 树

本文涉及的整体使用的 ROS 话题通信架构如图 4.12 所示。这套标准流程的运转方式是这样的。视觉路径识别节点要进行实时计算，算出赛道的中心线偏差后把它发布成对应的消息。PID 控制节点是核心的调度单元，它会订阅这条偏差消息，然后进行控制量的解算。底盘驱动功能包会订阅运动控制方面的消息，最后驱动电机去执行具体的循迹动作。AGV 智能巡迹避障物流车在巡线方式上，强调的是自动导引和自动行驶这两项功能的协同达到，节点之间规范地传输状态信息和控制量，是保证整体能够连续稳定运转的一个重要基础。

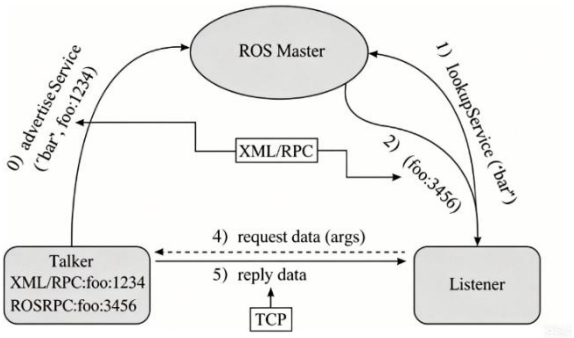


图 4.12 Topic 话题的基础架构

4.3.1 /cml_vel 话题的速度发布

/cmd_vel 是 ROS 移动机器人领域中一个通用的标准运动控制话题，也是本系统连接上层控制算法与底层硬件执行的关键环节，其消息类型为 geometry_msgs/Twist，其中包含了三维线速度与三维角速度共六个自由度的控制量。针对麦克纳姆轮智能车底盘的实际需求，本系统只使用了纵向前进的线速度 linear.x 以及垂直方向的转向角速度 angular.z 这两个有效分量，其余分量均固定为零，这样既能保持消息格式的轻量化，也能让指令解析更加高效，具体如图 4.13 所示。

```
wheeltec-client@ubuntu:~/mycore_ws/src$ rosmsg info
o geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

图 4.13 查看/cmd_vel 速度话题

在循迹控制的流程当中，/cmd_vel 话题的发布完全由 PID 控制节点负责，它的发布频率和图像处理的帧率保持严格同步，固定在 30Hz，保证控制指令没

有延迟也不会产生堆积。linear.x 是直行时的基准速度，会根据底盘的性能和赛道的复杂程度设定为一个恒定的安全数值，例如本整体取 0.2 米每秒，这样既能保证智能车平稳行驶，也留有足够的纠偏响应时间。angular.z 是动态的转向控制量，直接由位置式 PID 控制器输出，负责根据赛道中心线的偏差实时修正行驶方向，达到左偏时向右调整、右偏时向左调整的闭环循迹效果。发布过程中要使用 ucar_controller 功能包，如图 4.14 所示。

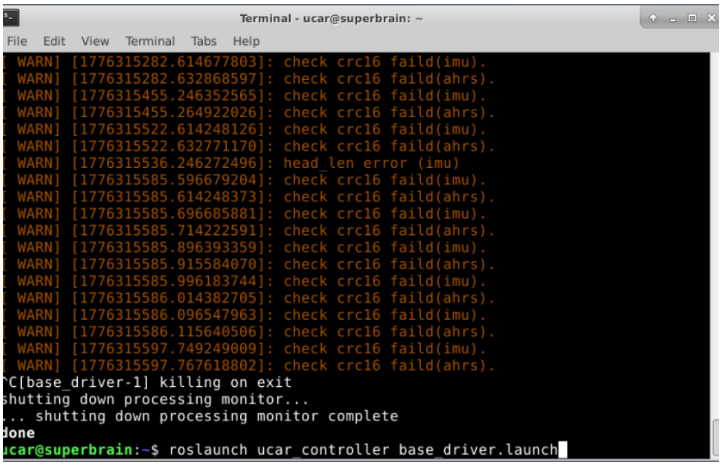


图 4.14 ucar_controller 话题调用

这个话题有很强的通用性和兼容性，不用自定义消息格式，也不用修改底层的驱动代码，能直接对接 ucar_controller 这类官方的底盘控制功能包，这样就能大幅减少整体的部署难度。还可以用 rostopic echo、rqt_plot 等 ROS 工具实时监测速度指令的波形，方便做 PID 参数的整定和控制效果的分析。系统还会对输出做严格的限幅处理，保证速度和转角都控制在底盘的安全范围之内，防止出现侧滑的情况。

4.3.2 ROS 的定时运动实现方式

为了保证图像处理、PID 解算以及指令发布这几个环节能够同步且稳定地运行，系统采用了 ROS 的定时循环配合时间戳校准的方式来实现精准的定时运动控制。通过使用 rospy.Rate(30)把控制频率设定为 30Hz，对应的周期大约是 33 毫秒，这个频率与视觉处理的帧率是相匹配的。在每一个控制循环当中，系统会依次完成图像的读取、路径的识别、偏差的计算、PID 的更新以及指令的发布，最后通过 rate.sleep()这个函数来实现精准的周期阻塞。

系统核心的定时工具就是 rospy.Rate 这个类，设定的控制频率为 30Hz，对应的单周期时长大约是 33 毫秒，这个频率既兼顾了嵌入式平台在算力方面的限制，也满足了视觉循迹对实时性的要求。在程序的主循环里面，每完成一次完整的控制流程，就会调用一次 rate.sleep()函数进行周期的阻塞，从而强制把循环频率稳定在预先设定的数值上。

定时的作用就是可以控制在识别到特征值的时候进行规定的运动，这样就可以实现类似转弯和停止等各类动作，但是同时他对特征值的识别要求极高，如果出现了误判就会影响后续控制。在代码中 30fps 下循环 15 次约 0.5 秒，因此可以实现特定的转弯动作，如图 4.15 所示。代码设定为 range（6）大约会运动 0.2s，当线速度为 0.2m/s 的基础上，运动 0.2×0.2×1.5=0.06m，根据测试实际实际时长可达 0.4-0.5s，因此会运动大约 12-15cm，也就是出现转弯角的时候可以快速通过。

```
# 发布控制指令
cmd_msg = Twist()

if turn_flag:
    cmd_msg.linear.x = LINEAR_SPEED * 1.5
    cmd_msg.angular.z = 0.0
    for i in range(6):  # 30fps下15次约0.5秒
        cmd_pub.publish(cmd_msg)
        rate.sleep()
else:
    cmd_msg.linear.x = LINEAR_SPEED
    cmd_msg.angular.z = angular_speed
    cmd_pub.publish(cmd_msg)

deviation_pub.publish(filtered_deviation)
```

图 4.15 代码中定时运动的使用

5 系统实验平台搭建与测试分析

5.1 实验平台搭建与参数配置

实验平台用科大讯飞二代车模型作为形式，完成了软硬件的一体化搭建。计算端装的是 Debian 整体和 ROS 运转环境。开发端则使用 Windows 系统结合 PyCharm 的 SSH 远程调试功能来开发，如图 5.1 所示。视觉传感器装在车头的前上方位置，俯角控制在 20°到 25°之间，这样既能兼顾近场赛道的边界信息，也能保证一定距离的前视视野。图像处理节点基于 OpenCV 来达到，输出中心线的偏差和边界的状态，还有转向标志等信息，并通过 ROS 话题把这些数据发送给控制节点。底层的执行部分由双轮差速驱动和电机编码器的反馈共同构成，再加上 PWM 调速链路，形成一条从感知到决策再到执行的完整闭环。

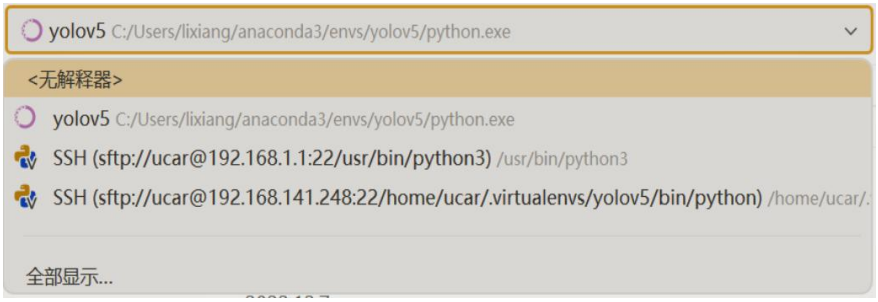


图 5.1 PyCharm SSH 进行远程调试

在重要参数的设置方面，本文使用的摄像头采集分辨率定为 320×240 像素。处理帧率可以稳定在 25 到 30 赫兹之间，控制周期则设为 33 毫秒。一种是二值化的阈值，二是 HSV 白色提取的阈值，第三是形态学运算所用的核大小，这些都是根据赛道上实际的光照条件来标定的。位置式 PID 的初始参数设定为 Kp=0.23、Ki=0.1、Kd=0.15，还把线速度设置在 0.2m/s，这样可以保证直道上的响应速度，也能维持弯道运转时的稳定性

5.1.1 硬件平台与执行机构

系统平台选用了搭载 Debian10 系统的智能小车作为移动载体，车体上集成了 RGB 相机、IMU、激光雷达以及双轮差速驱动机构，由此构成了一套感知与执行一体的硬件系统。其中的 RGB 相机配置为 1920×1080 的分辨率、每秒 30 帧的采集速率；IMU 在俯仰和横滚方向上的静态精度达到 0.05°，动态精度为 0.1°，主要用于辅助小车姿态的补偿计算；激光雷达的扫描角度覆盖 0°到 360°，角分辨率为 0.28°，测距范围在 0.12 米到 16 米之间。麦克纳姆轮通过左右两侧电机的转速差来实现转向，这种结构相对简单，也便于与 PID 控制量直接进行映射。模块化的执行机构设计有利于传感器快速响应以及系统稳定的协同工作，

这与朱玉提出的攀爬机器人与抓举机器人采用模块化结构以便快速部署的机械系统思路具有一致性[25]。

5.1.2 跨平台远程调试环境搭建

为了提高嵌入式端程序联调的效率，整体搭建了一套 Windows 开发主机加 Debian 车载平台这种跨平台的远程调试环境。在 PyCharm 里面用 SSH 完成代码的编辑和工程管理，可以直接对车载端的 ROS 节点、OpenCV 算法进行同步的修改。终端的连接、文件的传输由 MobaXterm 负责如图 5.2 所示，这样便于快速执行编译、启动程序以及排查进程方面的问题。图形界面的监控使用的是 VNC 远程桌面，能够达到对算法窗口、赛道画面的一体化观察。这种调试方式可以在 1920×1080 分辨率、每秒 30 帧的视频流条件下持续查看白色赛道的识别结果，并且结合左右边界、中心线的信息及时去修正相关的参数。

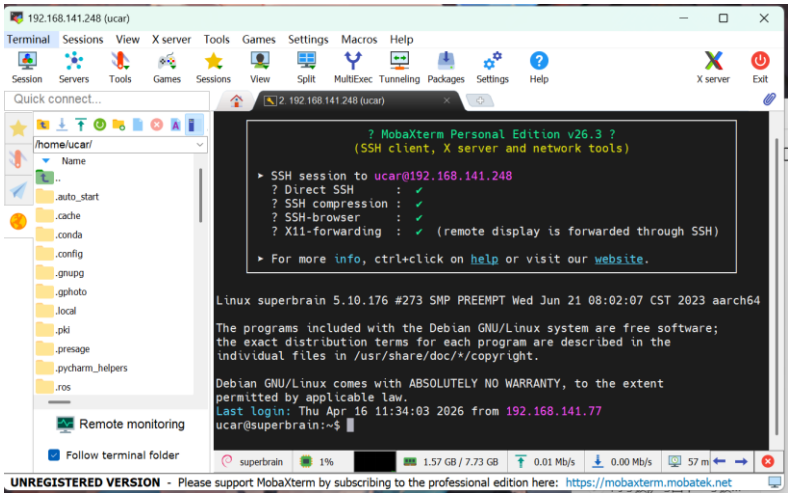


图 5.2 MobaXterm 远程代码调试

在实际调试中，远程环境可实时观察二值化结果、边界拟合曲线和蓝底白线赛道实景画面，减少频繁切换带来的麻烦。MobaXterm 还可用于 ROS 话题显示和部署脚本执行，使代码修改、结果验证与画面监控更加方便。

5.1.3 系统关键参数配置

系统关键参数的配置主要围绕视觉采集、ROS 通信以及控制策略这三个部分来展开。在 ROS 层，采用了固定的话题命名方式、固定的消息发布频率以及确定的节点启动顺序，视觉节点会输出偏差量和转向标志，控制节点在订阅到这些信息之后会按照设定的周期执行滤波、限幅以及差速映射等操作。控制策略当中的基础线速度、弯道减速的阈值、位置式 PID 的 Kp、Ki、Kd 参数以及输出上限等，都统一放在配置文件里面进行管理，再配合异常值剔除和通信超时保护机制，这样就能保证系统在重复部署的条件下始终保持一致的循迹响应。各个模块的参数总览如图 5.3 所示，图中包含了所有模块的参数调整情况

```python
GAUSSIAN_KERNEL = (5, 5)
THRESHOLD_VAL = 200
DENOISE_KERNEL = np.ones( shape: (3, 3), np.uint8)
MIN_CONTOUR_AREA = 15

RESIZE_WIDTH = 320
RESIZE_HEIGHT = 240

# 默认参数
START_RATIO = 0.45
END_RATIO = 0.65
DELTA_THRESHOLD = 10
TURN_FRAMES = 2
MIN_SLOPE = -1.0
MAX_SLOPE = 0.4

# HSV默认值
HSV_VALUES = {
    'hue_low': 0,
    'hue_high': 180,
    'sat_low': 0,
    'sat_high': 110,
    'val_low': 210,
    'val_high': 255
}
```

```python
# PID参数（保守初始值）
PID_KP = 0.23
PID_KI = 0.1
PID_KD = 0.15
MAX_ANGULAR_SPEED = 0.4
MIN_ANGULAR_SPEED = -0.4
LINEAR_SPEED = 0.2

# 误差滤波参数
ERROR_FILTER_SIZE = 5
DEADZONE = 5.0

# 停止检测参数
STOP_THRESHOLD = 0.06
```

图 5.3 各参数配置合集

5.1.4 关键参数可视化窗口调节

在参数调节过程中，可以发现众多模块掺杂在一起，导致无法有效调节参数或则及时看见参数变化导致的效果变化，这使调节难度大大增加。因此本文加入了滑块窗口，这样便实现了实时调节，而且可以对模块进行有效区分。

参数可视化的加入会使整体的调整更加复杂，为此本文加入参数文档功能，保存参数和载入参数的实现便会让参数调试更加方便可观如图 5.4 所示，可以有效减少后续的 PID 参数调整的时间和难度。

```python
def create_trackbars():  1 个用法
    """创建调试滑动条"""
    cv2.namedWindow( winname: "Parameters", cv2.WINDOW_NORMAL)

    cv2.createTrackbar( trackbarName: 'Min Area',  windowName: 'Parameters', MIN_CONTOUR_AREA,  count: 200, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Start %',  windowName: 'Parameters', int(START_RATIO * 100),  count: 100, lambda x: None)
    cv2.createTrackbar( trackbarName: 'End %',  windowName: 'Parameters', int(END_RATIO * 100),  count: 100, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Delta Thresh',  windowName: 'Parameters', DELTA_THRESHOLD,  count: 50, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Turn Frames',  windowName: 'Parameters', TURN_FRAMES,  count: 20, lambda x: None)

    cv2.createTrackbar( trackbarName: 'Hue Low',  windowName: 'Parameters', HSV_VALUES['hue_low'],  count: 180, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Hue High',  windowName: 'Parameters', HSV_VALUES['hue_high'],  count: 180, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Sat Low',  windowName: 'Parameters', HSV_VALUES['sat_low'],  count: 255, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Sat High',  windowName: 'Parameters', HSV_VALUES['sat_high'],  count: 255, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Val Low',  windowName: 'Parameters', HSV_VALUES['val_low'],  count: 255, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Val High',  windowName: 'Parameters', HSV_VALUES['val_high'],  count: 255, lambda x: None)

    cv2.createTrackbar( trackbarName: 'Kp',  windowName: 'Parameters', int(PID_KP * 100),  count: 100, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Ki',  windowName: 'Parameters', int(PID_KI * 100),  count: 50, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Kd',  windowName: 'Parameters', int(PID_KD * 100),  count: 100, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Max Angular',  windowName: 'Parameters', int(MAX_ANGULAR_SPEED * 10),  count: 20, lambda x: None)
    cv2.createTrackbar( trackbarName: 'Deadzone',  windowName: 'Parameters', int(DEADZONE),  count: 30, lambda x: None)

    cv2.createTrackbar( trackbarName: 'Stop Thresh%',  windowName: 'Parameters', int(STOP_THRESHOLD * 100),  count: 100, lambda x: None)
```

图 5.4 各参数配置合集和代码实现

5.2 路径识别算法性能测试与分析

在前面所说的基础配置条件之下，进一步对路径识别算法在强光、弱光、室内均匀光照以及蓝色赛道背景干扰这几种情况下的性能做了测试。本实验基于 1920×1080 分辨率、每秒 30 帧的图像输入，算法在经过灰度化、高斯滤波、二值分割、HSV 白色阈值提取以及闭运算等一系列处理之后，能够稳定地保留住

白色边界的主要特征。在直线路段以及缓弯区域当中，左右边界的连续性比较好，中心线的拟合也比较平滑，识别出来的结果跟实际赛道的走向基本上保持一致。

为了验证本文所设计的路径识别算法在实际场景中的有效性与稳定性，本节从图像预处理效果、边界与中心线的识别精度、实时处理性能以及特殊情况下的参数调整这三个方面展开了测试。实验是在实验室标准的蓝底白线赛道上进行的如图 5.5 所示，环境光照采用室内的常规照明，路面上会有部分的破损与遮挡情况，一定程度上能够确保测试结果客观地反映出算法本身的性能。

图 5.5 实验所使用的蓝底白线赛道

5.2.1 图像预处理模块效果验证

图像预处理是提升视觉识别抗干扰能力的前提，它的作用在于抑制噪声、强化赛道特征并消除环境中的各种干扰。本文采用的处理流程，包含了灰度化、高斯滤波、HSV 白色分割、自适应阈值分割、Canny 边缘融合、形态学闭运算以及小连通域滤除等步骤。

在对比实验当中，没有经过预处理的原始图像存在光照不均匀、路面纹理杂讯以及车体振动带来的噪声等干扰因素，直接对它进行边界提取会出现大量的误检点；而经过完整的预处理流程之后，图像能够有效地滤除掉蓝色背景的干扰，清晰地保留住白色赛道的轮廓。具体来看，高斯滤波能够平滑掉高频噪声；HSV 阈值分割可以在阴影或者弱光的条件下稳定地提取出白色赛道的区域；自适应阈值与 Canny 边缘相互融合之后提升了赛道边缘的连续性；形态学闭运算修复了细小的断裂部位；而最小连通域的过滤则去除了孤立的噪点以及光斑的干扰。

实验结果表明，经过预处理之后得到的二值图像具有完整赛道线、背景干净以及噪声微弱这几个特点，能够明显地提升后续在边界识别与中心线拟合方面的可靠性，从而为 PID 控制提供高质量的视觉输入。

在边界提取这个环节当中，通过逐行扫描的方式成功地锁定了赛道的边缘，准确标记出了红色与蓝色所代表的左右边界轮廓，并且平滑地拟合出了表示导航路径的蓝色中心线如图 5.6 所示。即便是面对弯道以及交汇路口这类比较复杂的赛道结构，中心线的提取偏差也都被控制在较小的像素范围之内，并且本系统在静态场景下具有较高的准确度以及较强的抗干扰能力。

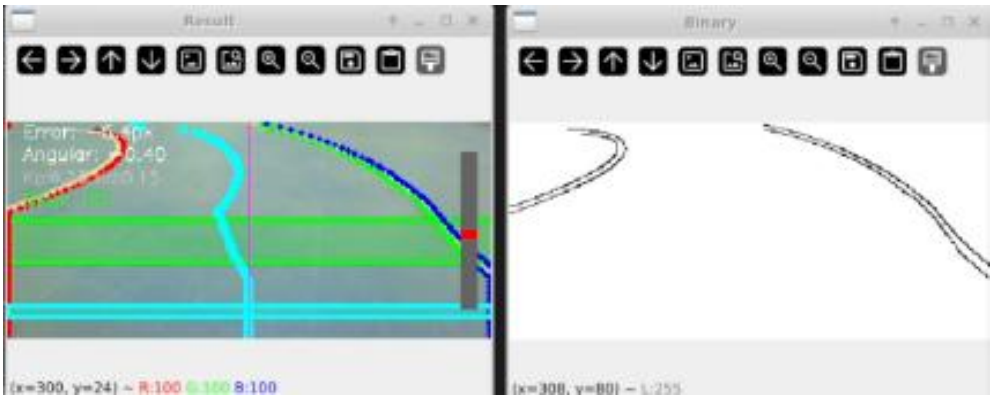


图 5.6 图像预处理模块效果

5.2.2 边界识别与中心线拟合测试

本文采用自下而上扫描线搜索法提取赛道左边界、右内边界与右外边界，通过左右边界中点计算中心线，并使用二次多项式拟合实现轨迹平滑。在测试过程中不断加入了众多的算法改变中线拟合效果如图 5.7 所示。

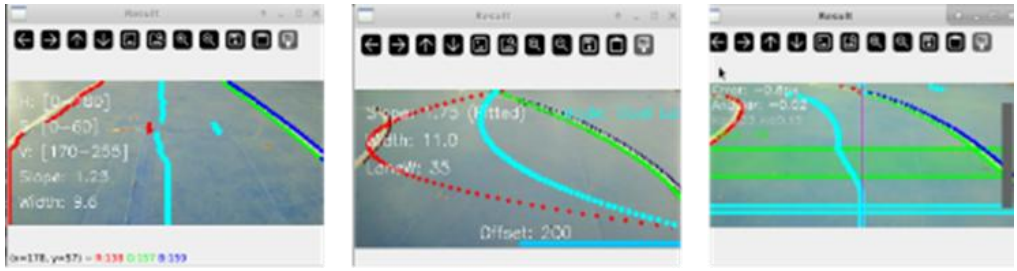


图 5.7 中心线拟合的效果

测试结果显示，直线赛道中边界、缓弯赛道边界偏离中心情况极小。在右转情况的检测中，本文使用宽度变化和斜率判断这两项约束，再加上连续帧确认，让识别基本无误，有效防止噪声误触发。终点停止线用底部区域黑色像素占比判断准确率极高。

5.2.3 视频流处理及特殊帧参数整定

为了验证系统在动态视频流方面的处理能力，对连续的视频画面进行了测试，并统计了图像采集、预处理、边界提取、中心线拟合以及偏差输出等各个环

节所耗费的时间。结果表明，系统单帧图像的平均处理时间为 21.8ms，最大处理时间为 28.6ms。由于控制周期设定为 33ms，视觉处理链路能够在一个控制周期之内完成有效偏差的更新，没有出现明显的帧累积或者控制滞后的现象，能够满足循迹运动控制对于实时性的要求。引入了遗传算法和粒子群算法来优化 PID 参数，无论在实车测试还是仿真当中都表现出了更好的误差抑制能力、响应速度与鲁棒性，这也说明离散控制结构需要优先考虑动态稳定性方面的问题[26]。

针对出现类似转弯角、转弯角、终点等特殊帧如图 5.8 所示，系统通过观察关键帧的数据来修改参数。将视频帧处理找到关键位置再进行参数调整就可以有效找到征值和是否能检测出转弯角这类特殊情况，这样可以使系统具备良好的环境适应性。

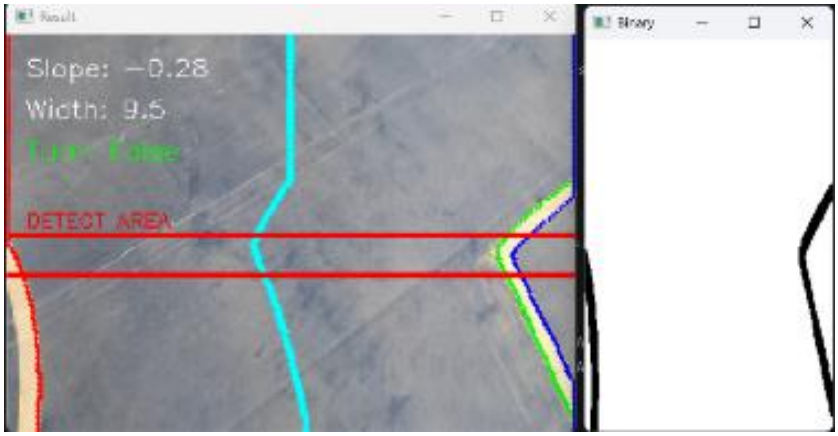


图 5.8 视频帧处理关键位置

5.3 循迹运动控制系统测试与分析

在实际的赛道闭环测试里，整体使用 1920×1080@30FPS 的视觉输入，搭配中心线偏差计算，增量式 PID 调节和差速驱动来完成整车循迹验证。测试赛道包含直线和缓弯。大曲率转弯路段的白色边界识别连续稳定，车辆可以沿赛道中心平稳运转。实验中控制周期保持 33 ms，视觉链路输出和底盘响应匹配良好。连续运转过程中没出现明显丢线、抖振现象，也没有过度修正的情况。进入弯道后，机器人可以依据偏差变化及时减速修正，车身姿态恢复平顺，本文有较好的跟踪精度和闭环稳定性。

5.3.1 Ziegler-Nichols 法调试增益

这个方法要先通过实验获取整体的临界增益 Kc 和周期 Tc，再用这些参数来计算比例积分和微分参数。一种是先把整体的环路关闭，慢慢增大比例参数 Kp，直到整体开始出现振荡，这时就把增益 Kc 和周期 Tc 记录下来。二是根据 Ziegler-Nichols 法则计算比例、积分和微分参数。

表 5.1 参数调整的策略

路况状态	参数调整策略	控制目标
直道小偏差	适当增大K_p，减小K_d	快速回中、保持速度
弯道中等偏差	降低K_p，增大K_d	抑制超调、平稳转向
急弯大偏差	暂时冻结K_i，限幅输出	防冲线、防振荡
噪声抖动场景	保持K_p，提高滤波权重	降低误调、增强鲁棒性

Ziegler-Nichols 法它的基本思想是通过试验获取系统的临界增益 Kc 和周期 Tc，然后根据这些参数计算出合适的 PID 参数。

5.3.2 转弯角调速与安全策略测试

为了保证硬件运转的安全，本文通过/cmd_vel 话题发布的像素偏差加入了双重限幅保护。线速度固定为每秒 0.3 米，减少通讯延时带来的震荡。转向角速度则限制在负 0.4 到正 0.4 弧度每秒这个范围之内，限制最高角速度可以有效减少震荡。话题发布分为三种模式一种是在直线弯道循迹，输出的是连续且平滑的速度指令。二是当检测到右转路口时，会自动将转向角速度清零并短暂提高直线速度，达到直行通过路口的动作。第三是一旦识别到终点停止线，就会立即发布全零的速度指令，使智能车能够安全地停下来。

结　论

本文完整设计了一套基于机器视觉的循迹控制系统。在路径识别算法方面，用 OpenCV 形成了一套轻量化的图像处理流程，使用灰度化、高斯滤波、HSV 白色赛道提取、自适应阈值分割和形态学闭运算等预处理操作的组合，有效抑制了环境噪声并强化了赛道特征。同时提出一种八领域扫描线和 Canny 边缘检测相融合的边界提取方式，能够精准定位赛道的左右边界，再用二次多项式拟合得到中心线，从而稳定地计算出横向偏差。本文还设计了右转检测算法，能识别复杂赛道的相关特征，满足直线、弯道等场景下的识别需求。

在运动控制算法方面，本文使用了 ROS 分布式模式来控制智能车的整体运动，将位置式 PID 作为核心控制算法，加入了偏差滤波，死区处理，积分分离和积分限幅等内容，有效解决了智能车在小偏差下的抖动，大偏差时的超调以及积分饱和的问题。本文通过/cmd_vel 这一标准话题实现了视觉节点和底盘驱动之间的通信联动，结合角速度和限幅的策略，保证了循迹过程的平稳性和硬件的安全性。本文借鉴了 Ziegler-Nichols 法完成了 PID 参数的调试，提高了控制的精度和响应速度，还提高了机器人自主导航稳定性方面的表现。

在系统搭建和测试方面，完成 Jetson Nano 加 STM32F407 双层硬件搭建与 Windows 远程调试环境部署。系统对单帧图像处理平均耗时大约在 20ms，可以满足 30Hz 控制周期实时性要求。系统对直线赛道边界识别、弯道识别、右转检测和终点识别准确率高。机器人可以沿赛道中心线平稳行驶，无明显抖振冲线与失稳现象，达到了预期设计目标。

本文采用的硬件平台为实验室基础配置，而后续可针对实际应用场景优化硬件结构，减少抖动对视觉采集和运动控制的影响。后续可简化远程调试流程，搭建可视化上位机调试界面，实现图像实时预览和 PID 参数在线调节，提升系统调试效率。长远来看，可基于本系统架构进行仓储巡检、室内搬运、园区巡逻等实际应用场景，推动基于机器视觉的智能移动机器人从实验室场景走向工程化、实用化应用。未来的机器人控制系统还可进一步拓展多模态人机交互功能。通过集成环形麦克风阵列实现声源定位与全双工语音交互，结合视觉感知与多传感器测距信息，使循迹机器人能够脱离单一的赛道环境，适应更为复杂的非结构化自主导航与调度场景。

参考文献

[1] 徐翔斌, 徐翔斌,马中强. 基于移动机器人的拣货系统研究进展[J]. 自动化学报,2022,48(1):1-20.

[2] 昌校宇. 智于科技慧于生态——中国外运发布《新时代中国智慧物流发展报告》[J]. 物流时代周刊, 2019(1): 6.

[3] Xiang Kai He,Zhi Huan Chen,Xue Gang Dai,Yang Chen,Tyrone Fernando,Herbert Ho Ching Iu.Design of Quantitative Dynamic Surface Sliding Mode Fault-Tolerant Controller for Power Line Inspection Robots[J].International Journal of Control.Automation, and Systems,2026,24(1):

[4] Campos C, Elvira R, Rodríguez J J G, et al. Orb-slam3: An accurate open-sourcelibrary for visual, visual–inertial, and multimap slam[J]. IEEE transactions onrobotics, 2021, 37(6): 1874-1890.

[5] Ackerman E. A Robot for the Worst Job in the Warehouse: Boston Dynamics'Stretch can move 800 heavy boxes per hour[J]. Ieee Spectrum, 2022, 59(1): 50-51.

[6] Du Y, Fu T, Chen Z, et al. VL-Nav: Real-time Vision-Language Navigation withSpatial Reasoning[J]. arXiv preprint arXiv:2502.00931, 2025.

[7] 梁晓妮,楚朋志,肖雄子彦.基于 OpenCV 图像处理的智能车巡线系统设计[J].传动技术.2022, (01): 139-149

[8] 黄捷,詹维捷,潘聪捷,等.面向智慧工厂的多仓储机器人路径规划仿真教学研究[J].实验技术与管理,2024,41(11):100-108.

[9] 王钢.基于视觉巡线的机器人移动平台控制系统研究[D].中国优秀硕士学位论文全文数据库.2025

[10] 卢嫚,朱世博.基于 OpenCV 和 YOLOv5 的车道线检测与识别[J].国外电子测量技术.2024, (06): 8-12

[11] Dong Wei,Pang Chen,Yang Han,Liqun Zhang,Jie Xue,Qianfeng Liu.Research on Transmission Line Personal Protective Equipment Detection Algorithm Based on Improved YOLOv11.[J].Journal of visualized experiments : JoVE.2025,(226):

[12] 吴恒,李彦超,窦浩,肖金甫,徐春钰.一种摄像头智能车的系统设计方案[J].科技创新与应用.2025, (17): 115-120

[13] 魏芳波,冯吴穹,杨佳敏,黄玥桐,樊希.基于 STM32 的智能巡检小车的设计[J].电子制作.2025,33(16):83-86

[14] Xiaowei Bi,Ye Liu.Autonomous UAV-Based Power Line Inspection Using Lidar Point Clouds and Deep Learning for Fault Localization[J].Journal of Circuits.Systems and Computers,2025,(prepublish):

[15] 胖墩会武术. OpenCV 图像处理基础教程：阈值、滤波、形态学与边缘检测 [EB/OL]. CSDN 博客，2022. https://blog.csdn.net/shinuone/article/details/126022763.

[16] 刘潇威,朱泓宇,胡艺,杨喆辰,赵春锋.基于目标搜索与识别的智能车追踪系统[J].现代计算机.2020,(21):18-21

[17] 张伊.基于 OpenCV 图像处理的智能小车户外寻迹算法的设计[J].电子技术与软件工程.2018, (18): 85-90
[18] Jiangtao Shi,Wenqi Fan,Zhi Yang,Junhui Li,Mengxuan Li.Improving small object detection in power line inspection with graph-enhanced YOLO[J].Microsystem Technologies.2026,32(2):
[19] 朱玉.电力巡线清障机器人登塔挂线机械系统设计及仿真[J].机械工程与自动化.2026,55(01):66-68
[20] 崔春蕊.中学机器人单光感巡线 PID 控制的项目式教学设计研究[J].山东教育.2025,(35):61-64
[21] 杨宗信.复杂环境下基于机器视觉的巡线 AGV 自主导航算法研究[D].导师：杨旭.　浙江理工大学.2023
[22] 徐桂敏,张艺潇.基于摄像头循迹的麦克纳姆轮智能小车实验系统设计[J].湖北第二师范学院学报.2025,42(08):20-26
[23] 韩彩霞,黄艺,罗子波.智能循迹小车设计[J].电子设计工程.2023,31(23):58-62
[24] 冯海喜.基于永磁同步电机无位置传感器巡线机器人控制系统研究[D].导师：郭家虎.　安徽理工大学.2025
[25] 朱玉.电力巡线清障机器人登塔挂线机械系统设计及仿真[J].机械工程与自动化.2026,55(01):66-68
[26] Dong Wei,Pang Chen,Yang Han,Liqun Zhang,Jie Xue,Qianfeng Liu.Research on Transmission Line Personal Protective Equipment Detection Algorithm Based on Improved YOLOv11.[J].Journal of visualized experiments : JoVE.2025,(226):

致　谢

总以为来日方长，毕业遥遥无期，转眼便要执笔作别。突然就想起了那句欲买桂花同载酒，终不似，少年游。二十载寒窗行至尾声，我的大学时光，也悄然落下帷幕。我曾经看到过很多热泪盈眶的致谢，也曾在心底默默预演了数遍，而今真的落笔时，只觉思绪翻涌，百感交集。回首这段岁月，仍有万般不舍，四年的时光也似白驹过隙般悄然而逝了。第一次进入马院，觉得马院很大，走了很久才到寝室，到了大二觉得马院很小，随便逛逛就能转完一圈，而让我最后望着来时的路又发现其实学校的每一处都有了我的身影。常常说思念不是真的在想一个人，而是触景生情想起一件事，而此刻我就是这般感慨。完成论文的最后一部分，也是完成了大学生涯的最后一部分。在这我由衷感谢每一位给予我帮助的老师和朋友。

一朝沐杏雨，一生念师恩。致谢我的毕业设计指导老师丰丰老师，而真正的感谢或许要从大一开始，第一次比赛就觉得丰丰老师特别平易近人，都说良师益友，我也认为良师亦友。到论文的选题开题、初稿打磨到反复修改，您在每一个关键节点都为我指引方向、悉心点拨。亦感谢四载春秋里所有传道授业的师长，最大的收获还有每位实验室老师传授的知识，叶增林老师和王鹏老师教我的 PLC，孙洒老师教的机器人视觉都让我无论是在比赛还是论文中都受益良多。你们的教诲与关怀，如灯如炬，照亮我求学之路。惟愿诸位恩师，万事顺遂，桃李芬芳。

树高千尺不忘根深沃土。感谢我的父母二十余载对我的栽培以及生活中的爱与支持。每当我遇到困难，父母总是第一个给我鼓励的人。感谢这么多年来你们无怨无悔的付出，永不停歇的照顾。你们是我求学路上最大的底气，你们的充分信任，让我勇敢大胆的做出生命中的每一个选择。唯愿父母家人平安康乐，岁月无恙。

愿岁并谢，与友长兮，感谢我的挚友，总在我失意时给我安慰和包容，让我拥有了前进的勇气和力量。遇到善良又有趣的你们是我最宝贵的财富，我们一起分享喜怒哀乐，一起探讨人生未来，一起聚餐逛街，一起旅行，这些相互鼓励和陪伴的日子，让我的青春更加绚烂。花开终有落，唯有友情藏心中，愿我们永远自由，平安喜乐，前程似锦！

以梦为马，不负韶华。最后感谢这一路走来的自己，四年大学生涯，有收获，有失去，有成长，有笑语，有泪水，有疯狂，也有遗憾。所有的经历都有意义，感恩所有的相遇。关关难过关关过，前路漫漫亦灿灿。谨以此篇，送给我的二十一岁，我最热烈而真挚的青春。