天津中德应用技术大学
Tianjin Sino-German University of Applied Sciences

本科生毕业设计

基于 ROS 系统的自主导航小车软件系统开发

Development of software system for autonomous navigation trolley based on ROS system

姓　　名　韩旭

学　　院　汽车与轨道交通学院

专　　业　车辆工程

指导教师　吕冬慧

职　　称　副教授

完成时间　2023 年 5 月 20 日

天津中德应用技术大学

本科生毕业设计（论文）选题申报表

<table>
<tr><td>学　院</td><td colspan="2">汽车与轨道交通学院</td><td rowspan="2">申报人</td><td>姓名</td><td colspan="3">吕冬慧</td></tr>
<tr><td rowspan="2">专　业</td><td colspan="2" rowspan="2">车辆工程</td><td rowspan="2">技术职务</td><td>正高</td><td>副高</td><td>中级</td></tr>
<tr><td></td><td>√</td><td></td></tr>
<tr><td>题目名称</td><td colspan="6">基于 ROS 系统的自主导航小车软件系统开发</td></tr>
<tr><td>题目类型</td><td colspan="2">自拟</td><td>题目来源</td><td colspan="3">创新竞赛</td></tr>
<tr><td>课题来源、背景及意义</td><td colspan="6">2021 年带领学生参加天津市大学生挑战杯竞赛，学生开发了基于 ROS 系统的自主导航小车，对小车软件系统进行开发，实现了小车的自主导航、自动行走、自主路径规划。</td></tr>
<tr><td>任务及要求</td><td colspan="6">学习 ROS 系统，基于 SLAM 技术进行地图算法开发，使用 A*算法对路径进行规划，完成小车自主导航、路径规划、实现主动避障等功能。</td></tr>
<tr><td>工作条件</td><td colspan="6">计算机、书籍、智能小车、rviz 软件</td></tr>
<tr><td>知识与能力要求</td><td colspan="6">汽车构造、汽车设计、汽车电子控制技术、单片机原理、Python</td></tr>
<tr><td colspan="7">系（教研室）审查意见：
无

负责人(签名)：吕冬慧　　2022 年 12 月 4 日</td></tr>
</table>

天津中德应用技术大学
Tianjin Sino-German University of Applied Sciences

毕业设计（论文）任务书

题　　目：基于 ROS 系统的自主导航小车软件系统开发

学　　院：汽车与轨道交通学院

专　　业：车辆工程

学生姓名：韩旭

学　　号：19424020128

起止日期：2022 年 12 月 1 日～2023 年 5 月 20 日

指导教师：吕冬慧

任务书下达日期:2022 年 12 月 15 日

毕 业 设 计（论文）任 务 书

1. 毕业设计（论文）课题背景及意义

随着现代科学技术的高速发展，高新技术已经广泛应用于现实生活当中，使人们的日常生活更加高效便捷。近些年来，无人配送小车、智能机器人、无人驾驶汽车等高新技术应用产品相继出现在日常生活中。特别是智能无人驾驶汽车的出现，降低了驾驶汽车的门槛，使得人们出行更加便捷。

目前，随着越来越多的人研究自动驾驶，智能车自主导航的问题成为近年来研究的热点问题。虽然在已知地图中的无人驾驶汽车的智能导航已经有许多的科研成果，但是在环境地图未知的情况下无人驾驶汽车的定位和导航仍是当前无人驾驶技术的主要研究和攻坚的方向。同时定位与地图构建（SLAM）算法可以通过智能车自身携带的传感器测量所处陌生环境信息和对自身位置的估计，能够为陌生环境中运动智能车的定位与导航提供有效的解决方案。而智能无人车的普及，必定要解决在陌生环境中智能车定位和导航的问题。因此，深入研究 SALM 算法、A*算法对提高无人驾驶智能车定位导航技术具有重要的研究意义和商业价值。

2. 毕业设计（论文）课题任务的内容和要求

内容：

（1）智能车系统搭建。

（2）地图创建与定位方法研究。

（3）智能车路径规划算法研究。

（4）智能车自主导航系统实验与测试。

要求：

学习 ROS 系统，基于 SLAM 技术进行地图算法开发，使用 A*算法对路径进行规划，完成小车自主导航、路径规划、实现主动避障等功能。

毕 业 设 计（论文）任 务 书

3．毕业设计（论文）课题成果（包括毕业设计论文、图表、实物样品等）

（1）毕业设计（论文）1 套

（2）自动驾驶智能车 1 辆

4．推荐参考资料

［1］熊安，卞春江，周海，刘成. 基于 ROS 的机器人定位与导航系统的仿真设计[J]. 电子设计工程，2018，26(24): 188-193.

［2］张文玲. 移动机器人同时定位与地图构建自适应算法研究[D]. 合肥：中国科学技术大学，2009.

［3］王珊珊. 轮式移动机器人控制系统设计[D].南京：南京理工大学，2013.

［4］李昌杰. 智能移动机器人控制系统设计研究[D].陕西：长安大学，2012.

［5］徐曙. 基于ＳＬＡＭ的移动机器人导航系统研究[D].武汉：华中科技大学，2014.

［6］孙炜， 吕云峰， 唐宏伟等. 基于一种改进 A*算法的移动机器人路径规划[J]. 湖南大学学报(自然版), 2017, 44(4): 94-101.

［7］焦富龙. 基于双目视觉的移动机器人ＳＬＡＭ系统的设计与实现[D]. 北京：北京工业大学，2015.

［8］靳士超. 基于麦克纳姆轮的全向智能移动机器人导航系统研究[D]. 苏州：苏州.

［9］曹思萌. 动态环境下移动机器人的路径规划算法研究[D]. 哈尔滨：哈尔滨工业.

［10］刘毅. 移动机器人路径规划中的仿真研究[J]. 计算机仿真，2011，28(06)：227-230.

所在专业审查意见：

无

负责人：胡恢堂

2022 年 12 月 15 日

天津中德应用技术大学
Tianjin Sino-German University of Applied Sciences

本科生毕业设计（论文）开题报告

题　　目：基于 ROS 系统的自主导航小车软件系统开发

学　　院：汽车与轨道交通学院

专　　业：车辆工程

学生姓名：韩旭

学　　号：19424020128

起止日期：2022 年 12 月 15 日～ 2023 年 5 月 20 日

指导教师：吕冬慧

开题日期:2023 年 1 月 8 日

一、 开题报告内容（课题的目的意义、与本课题有关的国内外研究（应用）情况及发展趋势、课题主要研究内容、参考文献等）

1、 课题的目的及意义

1.1 课题背景

如今，科学技术的发展日新月异，越来越多的高新技术不断地走进人们的日常生活中，给人们带来了舒适和便利。这些高新技术甚至完全改变了人们的生活方式，极大地提高了人们的生活质量。自主导航移动机器人技术的发展就是其中的一个体现，特别是融入了现代的人工智能技术以后，其智能化程度和适用性大大增强。

自主移动机器人具备环境侦测、定位导航、路径规划、自主移动的能力，能够在未知的地形中执行复杂任务。而通过对周围环境的精确感知，得到准确的定位则是实现复杂功能的前提，尤其是对于室内移动机器人，在室内环境中，由于接收到外界的 GPS 信号较弱，要想实现更加准确的定位，只能通过机器人自身的传感器感知自身状态和周围环境信息，得到位姿信息，再通过位姿信息绘制周围环境地图，来实现移动机器人在室内环境下定位与建图的功能。

相比于固定式机器人，自主导航移动机器人应用范围更广，实用性更强，占有非常重要的地位。移动机器人系统包含计算机、通信等专业领域，具有能够感知自身信息和自主移动等功能，已经在军事、餐饮和快递领域得到广泛应用。

1.2 课题目的

近年来，自主移动机器人领域出现了显着增长，其应用范围从救援和搜索到物流和制造。实现这些应用程序的关键技术之一是机器人操作系统 (ROS)，这是一个用于构建机器人软件的强大开源框架。ROS 提供了一套强大的工具来构建自主移动机器人，包括用于映射、定位和导航的库。这些工具使开发人员能够创建功能强大且高效的机器人，这些机器人可以在复杂的环境中运行，而人工干预最少。

1.3 课题意义

基于 ROS 的自主导航移动机器人的意义在于它们具有革新许多行业的潜力。 例如，在制造业中，移动机器人可用于在工厂周围移动材料和产品，从而减少对人力的需求并提高效率。在物流领域，自主机器人可用于在仓库和配送中心周围运输货物，再次降低劳动力成本并提高效率。除了这些实际应用之外，基于 ROS 的自主移动机器人具有巨大的研究潜力。通过提供高度灵活及可定制的平台，ROS 允许研究人员试验不同的算法和方法来解决机器人技术中的各种问题。总的来说，基于 ROS 的自主导航移动机器人的研究目的及意义植根于它们能够在广泛的应用中实现高级自动化，提高效率，并促进机器人领域的开创性研究。

国内外研究（应用）情况（文献综述）20%

对于自主导航移动机器人，要想在复杂的未知环境中实现自主导航，执行所要求的

任务，首先需要解决三个问题，即“我在哪里”，“我要去哪儿”，“我该如何去那里”。这三个问题分别体现了机器人的定位、路径规划和相应的控制方式三个方面。移动机器人要想实现精确的导航，其首要的条件是通过其外部传感器感知周围环境，得到精确的环境模型，建立环境地图，再利用机器人内部的传感器如里程计或加速度计、陀螺仪等惯性器件，运用航迹推算的方法得到其位姿的改变，并与机器人的外部传感器所得到的测量信息相结合，得到位姿变量的纠正值，实现机器人的准确定位。通过机器人的定位信息结合机器人当前时刻的观测信息，实现地图的增量化更新，为下一时刻机器人的位姿确定和地图更新做准备。在未知环境中，机器人周而复始地重复这一过程，在得到环境地图的同时实现了实时的定位，为机器人的路径规划与运动控制奠定了基础。因此，实现室内智能小车的自主导航功能，需要解决智能车在未知环境中的定位、建图和路径规划等关键技术。接下来对 SLAM（同时定位与建图）技术和路径规划的国内外研究现状进行介绍。

2.1 激光 SLAM 算法研究现状

SLAM 即同时定位与建图，这一概念最早于 1986 年由 Peter Cheeseman 等学者在美国加利福尼亚的 ICRA 上提出，他们首次将基于概率理论的方法引入到 SLAM 领域。Smith、Self 与 Cheeseman 在此基础上于 1987 年提出了基于卡尔曼滤波的 SLAM 算法，为 SLAM 的研究奠定了基本的理论框架。在卡尔曼滤波算法的基础上，扩展卡尔曼滤波以及无损卡尔曼滤波被分别提出并用于改进基于卡尔曼滤波的 SLAM 算法，一段时间以来成为了最为广泛的 SLAM 方法。

激光 SLAM 根据不同的数学优化框架，可分为基于滤波器的激光 SLAM 与基于图优化的激光 SLAM。激光 SLAM 算法中 EKF-SLAM（Extended Kalman Filter）是在 SLAM 概念出现后首次提出的正式 SLAM 算法，该算法为实现定位功能基于扩展卡尔曼滤波使用线性泰勒对状态变量与观测量进行近似，但就过程而言，该方法计算量大，复杂程度高、鲁棒性差，且该方法所建地图为特征地图，无法用于导航避障。针对此算法的不足，Montemerlo 等人于 2002 年提出了 FastSLAM 算法，采用粒子滤波来进行位姿估计，并通过位姿构建地图，该方法是最早实现实时建图的激光 SLAM 方案。FastSLAM 方案输出的栅格地图解决了特征地图无法导航的问题，但随之产生新的问题，即粒子滤波中随着随机重采样的次数增加会使得粒子愈发散失，从而造成建图精度严重降低。

2007 年，Grisetti 等人以 FastSLAM 算法为基础，将粒子数量限定在一个较小的范围进行预测采样，然后基于优化扫描匹配的方法优化位姿，同时减少重采样次数以缓解粒子耗散问题，此即为 Gmapping-SLAM 算法，Gmapping 算法可实时构建地图，鲁棒性强，且所建地图精度也高，是目前主流的算法之一，但算法也存在一定局限性，即算法过于依赖于里程计信息，其所构建的地图效果也取决于里程计精度，里程计精度高则建图效果好，反之亦然。

而基于图优化的激光 SLAM 方案首次提出的是 Lu 等人，通过建立非线性最小二乘来优化系统状态中累积的误差，该方法提出了一条新的优化方案，扩展了研究思路，但没有做到实时处理 SLAM 问题。为改善此缺点，Konolige 等人于 2010 年提出 Karto SLAM 算法，实现了实时 SLAM，但是 SLAM 耗费时间较长，特别是在搜索大范围环境的时候。

2016 年，谷歌公布的 Cartographer 开源算法，对 Karto SLAM 进行了优化。

2017 年，R.P.Guan 等人将 KLD-MCL 算法引入到 SLAM 提议分布中，得到了自适应粒子数的高效粒子滤波器。

2018 年，Hong He 等人在里程计信息中加入激光信息，提升了机器人定位性能。2018 年，Hyung Gi Jo 等人提出基于粒子间地图成型的改进 RBPF-SLAM 算法，在不改变地图精度的情况下减小其所占空间。

2019 年，Jia Danping 等人提出粒子权值平衡方法，用来提高粒子的特征丰富性。但是这个方法对于简单场景下的地图构建适应性较好，而在复杂场景下的地图构建情况还 没有明确的研究。

2020 年，Ragesh K 等人提出了一种新的分布式方法，利用具有全局定位能力和有限的机器人间通信的机器人群来构建一个未知环境地图，验证了分布式探测和映射策略的有效性。

国内科研机构对 SLAM 技术的研究相对而言起步较晚，但也取得了一定研究成果。

2011 年，孙玉梁提出了综合 MbICP 匹配算法和 EKF 滤波算法的优势，提高自主导航的鲁棒性和实用性。

2015 年，高帅针对复杂环境中里程计存在较大误差的问题，依靠激光雷达高频率地采集距离信息，采用 hectslam 算法，在实际环境中完成了栅格地图的构建，取得了良好的实验效果。

2017 年，雷碧波采用一种改进的特征提取方法,将改进后的算法在 ROS 上实现,采用激光匹配算法实现准确的地图构建与定位,在实验中对比分析,改进后的算法解决了构图与定位的实时性,鲁棒性也有所提高。

2018 年,齐春辉针对移动机器人自定位问题,将航迹推算定位与惯性定位进行卡尔曼融合,并与改进后的蒙特卡罗定位相结合实现移动机器人自定位。利用激光雷达数据构建栅格地图,并引入栅格占据率概念,以减少传感器噪声的影响。

2.2 路径规划研究现状

在智能机器人领域，路径规划问题占据着重要地位。路径规划问题是在有障碍物的环境中，在满足距离、计算时间、通信延迟和能量消耗等优化条件的前提下，寻找从机器人初始位置到期望位置的安全路径。在真实场景下机器人进行路径规划，首先要知道自身位置，成功规划路径后，设置机器人角速度和线速度，机器人才会顺利完成任务。全局路径规划中环境已知，局部路径规划中环境不确定，需要通过传感器了解环境信息。目前，A*

算法、Dijkstra 算法、D*算法是常用的全局路径规划算法。Dijkstra 算法中较大数量的遍历节点数会降低算法的效率。因此，P.E.Hart 等人于 1968 年提出 A*算法，与其他全局路径算法相比，其性能更优，但是该方法存在计算效率低和转折点多的问题。Holland 等人最早采用遗传算法解决机器人路径规划问题，通过定义的搜索空间获取路径。

（1） 全局路径规划算法研究现状

全局路径规划算法是在已知整体环境信息后，找到从初始点到目标点的最优轨迹，而全局路径规划算法又可分为图搜索类算法、随机采样类算法、智能算法等。

为了改进 A*算法的实时性，2014 年，Xin Y 等人提出了一种改进的基于无限邻域的 A*算法，每个方向都进行搜索，减少了路径长度和转折点的数量。

2016 年，T.Oral 等人提出了一种多目标 D* lite（multiobjective D* lite，MOD* lite）算法。为了比较 MOD* lite 算法与多目标 A*算法的求解质量和执行时间要求，设计了多目标遗传路径规划和强度 Pareto 进化算法。

2018 年，X.Zhao 等人将跳点搜索方法与 A*算法相结合，对 A*算法进行优化，得到一些关键的跳跃点，通过这些点与目标点之间的直线跳跃得到最终路径。虽然该方法避免了 A*算法中大量扩展无用节点的过程，提高了路径规划效率，但规划的路径不平滑，实现困难。

2019 年，吴鹏等人采用双向搜索的方式减少了路径规划时间。但是在生成的路径中存在较多拐点。

2020 年，R.Fareh 等人提出了一种序列线性路径方法，通过在路径中寻找线性捷径来改善路径的质量，实验结果显示出了优异的性能。

（2） 局部路径规划算法研究现状

在实际环境中难以避免出现未知障碍物，机器人要在未知环境中仍能顺利到达终点，则需要涉及局部路径规划，达到避开障碍物的目的。1997 年，D.FOX 和 W.Burgard 等人提出动态窗口法（Dynamic Windows Approach，DWA），针对该方法中速度空间不准确的问题，Gerkey 等人提出了改进的动态窗口法。该方法通过轨迹的评估函数，同时对机器人的路径空间采样，来选择最佳路径。

2015 年，J.Yoo 等人提出了一种基于注视控制的导航体系，该方法根据局部环境变 化情况来切换特定的优先级，可以安全地应用于复杂环境，但是这个方法注重局部最优， 使得机器人在终点附近位置存在障碍物的情况下会规划路径失败。

2016 年，张坤等人在机器人路径规划中引入模糊控制理论，解决了局部路径规划 中强烈依赖环境信息的问题。

2018 年，张福海等人用强化学习解决局部路径规划，通过设计合理的报酬函数， 机器人能在未知环境中不断的学习试错，最终到达目标。然而，尽管有这样的优势，强化 学习方法对最优解的收敛速度很慢。

为了解决这一局限性，2019 年，K.Jo 等人提出了一种混合局部路线生成算法，在基于感知的局部路线和基于精确地图的局部路线选项之间选择最优的局部路由，通过实验验证了该算法的优越性。

2020 年，M.N.A.Wahab 等人提出粒子群优化和条纹搜索算法（Particle Swarm Optimization and Fringe Search Algorithm，PSOFS）的混合元启发式算法，在路径平滑度 和机器人安全性两方面体现了 PSOFS 在局部路径规划方面的优良性能。

2021 年，K.Shi 等人将蚁群算法和人工势场法融合，解决了在复杂情况下移动不到目标的问题。

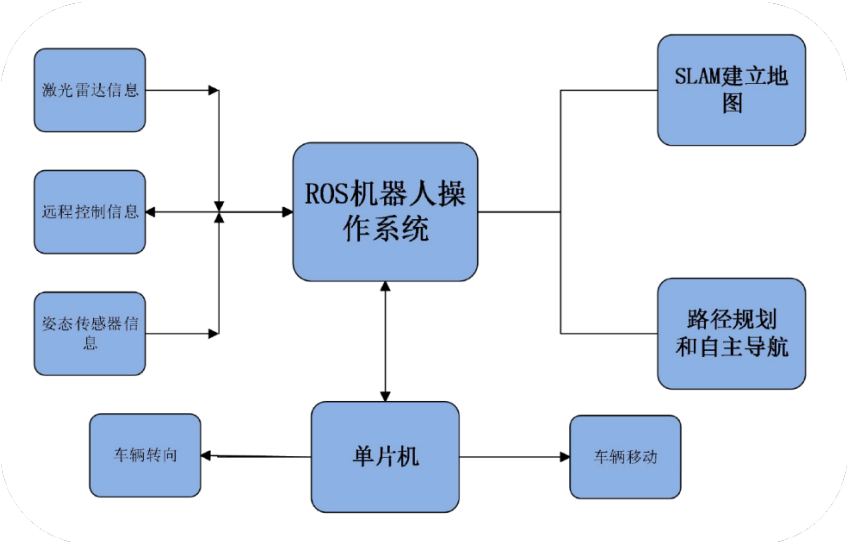
3、课题主要研究内容

3.1 主要研究内容

本课题的主要研究内容是基于 ROS 系统并结合 SLAM 算法实现一款小型的自动行驶智能小车，该智能小车具备构建室内二维地图、自动导航和躲避障碍的功能。本论文系统地阐述了智能小车的实现过程及原理。从国外的学术论文及国内的学术论文中学习算法的原理及思想，从国外的开源项目以及视频网站中学习如何仿真和具体实现一款简易的智能小车。在完成了小车的硬件设计后，通过 ROS 系统采用其成熟的经过验证的算法实现了智能小车在实验室环境下的地图构建和基于地图的导航定位等功能，完成对 SLAM 算法的学习和研究，同时在学习的基础上对构建地图和导航算法进行改进并在小车平台上实验验证和分析，最终达到理解 SLAM 算法的目的。

3.2 系统框架和章节安排

（1）系统框架



（2）章节安排

第一章 绪论

1.1 研究背景及意义

1.2 国内外研究现状

1.3 论文研究内容及章节安排

1.4 本章小结

第二章 自主导航小车系统建模及硬件架构

2.1 移动小车运动学模型

2.2 硬件架构

2.5 本章小结

第三章 整车控制系统的搭建

3.1 底层控制

3.2 顶层控制

3.3 底层与顶层的协调控制

3.4 本章小结

第四章 SLAM 与路径规划

4.1 SLAM 概述

4.2 Gmapping 算法

4.1 全局路径规划

4.2 局部路径规划

4.4 本章小结

第五章 移动小车平台搭建与自主导航实验

5.1 实验平台搭建

5.2 SLAM 建图实验

5.3 路径规划导航实验

5.4 本章小结

第六章 总结与展望

6.1 全文总结

6.2 展望

参考文献

附录

致谢

在校期间的研究成果

3.3 课题准备情况

通过文献资料查阅、理论学习，已提前完成第一章绪论、第二章的部分内容。接下来的时间将投入到第三章 SLAM 算法应用研究及第四章的路径规划导航实验上。

2、 参考文献

[1] Smith R, Self M,Cheeseman P.Estimating uncertain spatial relationships in robotics[M]//Autonomous robot vehicles. Springer, New York, NY, 1990: 167-193.

[2] 危双丰,庞帆,刘振彬,师现杰. 基于激光雷达的同时定位与地图构建方法综述[J].计算机应用研究,2020,37(02):327-332.

[3] Montemerlo M, Thrun S, Koller D, et al. Fast SLAM: a factored solution to the simultaneous localization and mapping problem[C] Proc of AAAI National Conference on Artificial Intelligence. Palo Alto, CA:AAAI press,2002:593-598.

[4] Grisetti G, Stachniss C, Burgard W. Improved techniques for grid mapping with Rao-Black wellized particle filters[J].IEEE Trans on Robotics, 2007, 23(1): 34-46.

[5] Lu F, Milios E. Globally consistent range scan alignment for environment mapping[J]. Autonomous Robots, 1997,4(4):333-349.

[6] Konolige K, Grisetti G, Kümmerle R, et al. Efficient sparse pose adjustment for 2D mapping[C].Proc of IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway, NJ: IEEE Press, 2010: 22-29.

[7] Zhang J, Jiang Y, Wang K. A modified FastSLAM for an autonomous mobile robot[C]//IEEE

[8] 胡春旭. ROS 机器人开发实践[M]. 机械工业出版社, 2018.

[9] 王珊珊.轮式移动机器人控制系统设计[D].南京：南京理工大学，2013.

[10] 李昌杰.智能移动机器人控制系统设计研究[D].陕西：长安大学，2012.

[11] 徐曙.基于ＳＬＡＭ的移动机器人导航系统研究[D].武汉：华中科技大学，2014.

[12] 雷碧波.基于 ROS 平台的室内定位算法设计与实现[D].浙江理工大学,2017.

[13] 孙炜,吕云峰, 唐宏伟等.基于一种改进 A*算法的移动机器人路径规划[J].湖南大学学报(自然版),2017,44(4):94-101.

[14] 刘毅.移动机器人路径规划中的仿真研究[J].计算机仿真，2011，28(06)：227-230.

[15] 孙玉梁.移动机器人室内即时地图构建与自主导航[D].大连理工大学,2011.

[16] 高帅.基于 ROS 的未知环境移动机器人导航研究[D].东北大学,2015.

[17] 齐春辉.基于 ROS 的室内移动机器人导航技术研究[D].河北工业大学,2017.

[18] 陈华.基于 ROS 的室内移动机器人导航技术研究[D].南昌大学,2022.

[19] Dijkstra E W. Communication with an automatic computer[D]. Excelsior, 1959.

[20] Stentz A. The D* Algorithm for Real-Time Planning of Optimal Traverses[M]. CMU-RI-TR-94-37, CMU. 1994.

[21] Kavraki L, Latombe J C. Randomized preprocessing of configuration for fast path planning[C]. Robotics and Automation, 1994. Proceedings. 1994 IEEE International Conference on. IEEE, 1994:2138-2139

[22] La Valle S M. Rapidly-exploring random trees: A new tool for path planning[R]. Ames, USA: lowa State University, 1998.

[23] 张杰.移动机器人路径规划研究[D].上海交通大学,2014.

[24] 顾幸方,陈晋音.移动机器人未知环境避障研究[J].传感器与微系统,2011,30(05):16-20.

[25] Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots[J]. International Journal of Robotics Research, 1986,5(1):90-98.

二、进度及预期结果

起止日期	主要内容	预期结果

2022.12.15-2023.1.08	理解自己选报的题目，查阅相关文献资料，分析相关技术的研究现状。攥写开题报告，完成答辩 PPT。	构建毕业设计基本框架。完成答辩 PPT，通过导师审批并完成开题答辩。
2023.1.09-2023.2.1	对 SLAM 技术及路径规划算法进行研究。	完成相关算法的研究。
2023.2.2-2023.3.15	学习 ROS 系统，搭建 ROS 软件平台并在 Gazebo 上进行仿真验证。	在 PC 端搭建好 ROS 软件平台并在 Gazebo 上完成仿真验证。
2023.3.20-2023.4.1	进行毕业设计论文中期检查。	提交中期检查表。
2023.4.2-2023.5.1	进行移动小车平台的搭建以及自主导航实验，证实自主导航技术的可行性。	搭建好小车平台，完成自主导航实车验证。
2023.5.2-2023.5.20	进行论文的修改和查重。	完成论文的修改并通过查重
2023.5.21-2023.6.1	申请毕业答辩资格，制作毕业答辩 PPT，准备毕业答辩。	通过学院审核取得答辩资格，完成 PPT 并通过毕业答辩。
2023.6.2-2023.6.30	进行毕业设计归档。	完成毕业设计归档。
完成课题的现有条件	1、 基于 ROS 的竞速车平台。 2、 围绕自主导航相关内容所学习的激光雷达原理和 ROS 知识。 3、 通过文献资料学习、网络视频教授的自主导航实现的相关理论和方法。 4、 具备的 C++、Python 编程语言基础。以便对相关算法在 ROS 上的应用和功能的实现。 5、 具备的查阅文献资料和快速学习的能力。 6、 学校的硬件资源和导师教授的知识。	
指导教师意见	无 **指导教师：** 吕多慧 2023 年 1 月 8 日	
开题答辩小组意见	无 **组　　长：** 吕多慧 2023 年 1 月 8 日	

天津中德应用技术大学

本科生毕业设计（论文）的声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导教师指导下，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，本毕业设计（论文）的研究成果不包含任何他人创作的、已公开发表或没有公开发表的作品内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本毕业设计（论文）原创性声明的法律责任由本人承担。

毕业设计（论文）作者签名：

2023 年 05 月 20 日

本人声明：该毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过设计（论文）的全部内容，并能够保证题目、关键词、摘要部分中英文内容的一致性和准确性。

毕业设计（论文）指导教师签名：

2023 年 05 月 20 日

目　录

摘　要

随着人工智能技术的进步，智能车的功能和应用领域不断扩展，从月球车、工业制造到家庭智能服务和智能物流。这些高新技术的应用不仅改变了人们的生活方式，同时也提高人们生活质量。

本课题以 ROS（机器人操作系统）为软件框架，结合室内的应用场景，基于激光雷达传感器，设计了一种基于室内未知环境下的具备自主导航和避障功能的智能车。针对室内移动机器人的同时定位与地图构建技术，本课题选择了应用广泛的Gmapping算法进行 SLAM（同时定位与建图）。针对基于已知地图的路径规划问题，在比较 Dijsktra 算法和 A*算法的优缺点后，采用了 A*算法作为全局路径规划算法，选用 DWA 算法作为局部路径规划算法。

通过综合应用 Gmapping 算法、A*算法和 DWA 算法，实现了室内移动机器人的同时定位、地图构建和路径规划，为智能车的自主导航提供有效的解决方案。

为了检验所使用算法的有效性，根据所搭建的室内环境，在 Gazebo 上进行物理仿真，验证以上算法的可行性。在仿真结果达到预期效果的情况下，将其移植到搭建的智能车平台进行真实环境下建图以及自主导航效果验证。通过仿真和实车试验表明，本文所设计的智能车具备在室内陌生环境中建立地图、进行导航和动态避障的功能，达到了预期的效果。

关键字：ROS；智能车；SLAM；自主导航

ABSTRACT

With the advancement of artificial intelligence technology, the functions and application fields of smart vehicles continue to expand, from lunar rover, industrial manufacturing to home smart services and smart logistics. The application of these high-tech technologies not only changes people's lifestyle, but also improves people's quality of life.

In this project, ROS (Robot Operating System) is used as the software framework, combined with indoor application scenarios, and based on lidar sensors, an intelligent vehicle with autonomous navigation and obstacle avoidance functions based on unknown indoor environments is designed. For the simultaneous localization and map construction technology of indoor mobile robots, this topic selects the widely used Gmapping algorithm for SLAM (simultaneous localization and mapping). Aiming at the path planning problem based on known maps, after comparing the advantages and disadvantages of Dijsktra algorithm and A* algorithm, A* algorithm is used as the global path planning algorithm, and DWA algorithm is selected as the local path planning algorithm.

Through the comprehensive application of Gmapping algorithm, A* algorithm and DWA algorithm, the simultaneous positioning, map construction and path planning of indoor mobile robots are realized, providing effective solutions for the autonomous navigation of intelligent vehicles.

In order to verify the effectiveness of the algorithm used, physical simulation was performed on Gazebo according to the built indoor environment to verify the feasibility of the above algorithm. When the simulation results achieve the expected results, they are transplanted to the built intelligent vehicle platform for real-world mapping and autonomous navigation effect verification. Simulation and real vehicle tests show that the intelligent vehicle designed in this paper has the functions of establishing maps, navigation and dynamic obstacle avoidance in an indoor unfamiliar environment, and achieves the expected results.

Key words：ROS；Intelligent vehicle；SLAM；Autonomous navigation

第一章　绪论

1.1 研究背景及意义

1.1.1 研究背景

随着科技的不断进步，越来越多的高新技术走进了人们的日常生活中。这些技术不仅为人们带来了更加舒适和便利的生活方式，同时也提高了人们的生活质量。自主导航移动机器人技术就是这种进步的典型之一，尤其是在与迅速发展的人工智能技术结合后，它的适用性和智能化程度也得到了极大的提高。

自主导航移动机器人具备定位导航、路径规划和自主移动的能力，能够在未知环境中执行各种任务。室内移动小车由于无法依赖弱或无法接收的 GPS 信号，必须依靠搭载的传感器来感知周围环境和自身状态，以保证准确的定位。与固定式机器人相比，自主导航移动机器人具有更广泛的应用范围和实用性，且具备重要地位。移动机器人系统融合了计算机和通信等专业领域，具备感知自身信息和自主移动等功能，已广泛运用于军事、餐饮和快递等领域。

1.1.2 选题的目的及意义

近年来，自主移动机器人领域出现了显着增长，其应用范围从搜索和救援到制造和物流。实现这些应用程序的关键技术之一是机器人操作系统 (ROS)，这是一个用于构建机器人软件的强大开源框架。 ROS 提供了一套强大的工具来构建自主移动机器人，包括用于映射、定位和导航的库。 这些工具使开发人员能够创建功能强大且高效的机器人，这些机器人可以在复杂的环境中运行，而人工干预最少。 基于 ROS 的自主导航移动机器人的意义在于它们具有革新许多行业的潜力。 例如，在制造业中，移动机器人可用于在工厂周围移动材料和产品，从而减少对人力的需求并提高效率。 在物流领域，自主机器人可用于在仓库和配送中心周围运输货物，再次降低劳动力成本并提高效率。 除了这些实际应用之外，基于 ROS 的自主移动机器人具有巨大的研究潜力。 通过提供高度灵活及可定制的平台，ROS 允许研究人员试验不同的算法和方法来解决机器人技术中的各种问题。 总的来说，基于 ROS 的自主导航移动机器人的研究目的及意义植根于它们能够在广泛的应用中实现高级自动化，提高效率，并促进机器人领域的开创性研究。

1.2 国内外研究现状

针对自主导航移动小车在复杂未知环境中实现自主导航和任务执行的需求，需要解

决定位、建图和路径规划三个关键问题。本课题研究的室内自主导航技术，首先需要在室内的未知环境中创建地图，然后在创建好的地图上进行定位，在地图中设置好目标点后，通过路径规划算法会在地图上生成最优的小车行进路径，小车沿着该路径移动，直至到达目标点。而在未知环境中创建地图和定位的精度将制约着小车能否精准的移动到目标未知。其中，SLAM（同时定位与建图）技术作为移动机器人在未知环境中的定位与建图的方法，一直成为国内外的研究热点。在未知环境中的定位与建图是实现小车自主导航功能基础，而在已建立的地图上进行路径规划也是实现该功能的关键技术之一。接下来对 SLAM 技术和路径规划的国内外研究现状分别进行介绍。

1.2.1 SLAM 研究现状

SLAM（同时定位与建图）问题作为机器人研究中的一个基本而重要的研究领域，一直以来都受到深入的关注和研究。最早，Randall C. Smith 和 Peter Cheeseman 提出了 SLAM 问题的概念。随后，Leonard 和 Durrant-Whyte 在处理 SLAM 问题时引入了机器人位姿的估计方法。Smith 等人采用卡尔曼滤波器来处理 SLAM 问题，并对其进行改进，这一方法得到了广泛的应用，推动了 SLAM 技术的进展。

由于卡尔曼滤波方法在 SLAM 问题中的应用，SLAM 技术得到了显著的进展。研究人员继续改进和优化 SLAM 算法，引入新的方法和技术，如粒子滤波、图优化等，以提高 SLAM 的精度和鲁棒性。SLAM 技术的发展不仅推动了机器人领域的进步，也在自动驾驶、虚拟现实等领域得到了广泛应用。

粒子滤波在解决 SLAM 问题时存在粒子退化问题。2007 年，Doucet 等人[2]改进粒子滤波算法，提出 Rao-Blackwellized 粒子滤波方法，解决 SLAM 问题，Gmapping 方法是基于此方法实现的，已经在机器人实践中得到应用。

2016 年谷歌提出了图优化 SLAM 算法 Cartographer[3]， 该算法在构建地图的过程总可以消除累计误差，并且有很好的实时性。

2018 年，Hong He 等人[4]将激光信息融合到里程计信息中，提高了机器人的定位性能。

2019 年，Jia Danping 等人[6]提出了粒子权值平衡方法，用于提高粒子的特征丰富性。然而，该方法在复杂场景下的地图构建问题还没有明确的研究结果。

相比于国外，国内对于 SLAM 研究较晚，但随着近几年的高新技术的发展，研究成果也在增加，并对经典算法进行了优化和改进。

2015 年，王立新等，提出了 LIO-SAM 算法，结合激光雷达的里程计和建图，实现了高精度的激光 SLAM，适用于复杂环境和高速移动条件。

2018 年，肖红兵等，提出了基于滤波器的激光 SLAM 算法，结合滤波器方法和传感

器数据融合，实现了高精度的室内和室外定位与建图。

2019 年，周宁等，在激光 SLAM 领域进行了深入研究，提出了多传感器融合的激光 SLAM 算法，结合惯性测量单元（IMU）和激光雷达数据，实现了更稳健和精确的定位与建图。

1.2.2 路径规划的研究现状

在智能机器人领域中，路径规划问题具有重要的地位。它涉及在包含障碍物的环境中，以满足距离、计算时间、通信延迟和能量消耗等优化条件的前提下，找到机器人从初始位置到目标位置的安全路径。路径规划的成功实现对于机器人的任务执行至关重要。路径规划可以分为全局路径规划和局部路径规划。全局路径规划是在已知环境下进行的，而局部路径规划需要通过传感器来获取环境信息，因为环境是不确定的。在全局路径规划中，常用的算法包括 A*算法、Dijkstra 算法和 D*算法。

2014 年，Xin Y 等人提出了一种改进的基于无限邻域的 A*算法，旨在改善 A*算法的实时性。该算法的改进之处在于在每个方向上进行搜索，从而减少路径长度和转折点的数量。

2018 年，X.Zhao 等人[7]提出了一种改进 A*算法的方法，结合了跳点搜索技术。该方法通过识别关键的跳跃点，并直线连接这些点与目标点，优化了 A*算法的路径规划过程。这样做可以避免扩展大量无效的节点，从而提高了路径规划的效率。 然而，这种方法生成的路径可能不够平滑，并且实现起来具有一定的难度。

2019 年，吴鹏等人[8]提出了一种使用双向搜索来减少路径规划时间的方法。通过同时从起点和目标点进行搜索，可以加快搜索过程，提高路径规划的效率。 然而，尽管这种方法在减少规划时间方面取得了成功，但在生成的路径中却存在较多的拐点。这意味着生成的路径可能不够平滑，并且在实际导航过程中可能需要频繁的转弯。

2020 年，R.Fareh 等人[9]提出了一种称为序列线性路径（Sequential Linear Path）的方法，旨在改善路径的质量。该方法通过在生成的路径中寻找线性捷径，进一步优化路径的形状和特征。

1.3 课题的研究内容

本课题选定在结构化的室内环境下设计的一种基于 ROS 系统的自主导航小车，使用单线程激光雷达和 IMU 作为主要的传感器，主要使用单线程激光雷达和 IMU 作为主要传感器，并结合搭建的阿克曼结构小车进行研究和实验。

结合自主导航中的 SLAM 和路径规划等关键技术与 ROS 系统的特点分别对智能车定位建图、路径规划进行软件设计，首先利用 ROS 系统中的 Gazebo 仿真平台，搭建了阿

克曼小车和环境模型，以便进行后续的实验研究。在该环境中，进行了针对阿克曼小车的 A*算法全局路径规划和 DWA 局部路径规划的自主导航仿真实验。通过仿真实验，验证所提出的路径规划算法在虚拟环境中的可行性和效果。然后，将实验扩展到搭建的真实环境中，对本文中所研究的阿克曼小车进行自主导航的实验和分析。验证智能车在室内的定位导航以及避障功能。

第 1 章是绪论部分，该部分首先对移动机器人的研究背景和意义进行了概述，包括移动机器人在现代社会中的重要性和应用领域。接着，对室内机器人同时定位与地图构建（SLAM）的国内外发展现状进行了简要概括和分析，包括介绍了一些主要的研究成果和方法。最后介绍了课题的研究内容及安排。

第 2 章是整车软件系统的设计，介绍了 ROS 系统的特点、通信机制和文件系统的结构，并根据智能车的各个功能模块对总体功能节点进行了规划设计。

第 3 章是 SALM 与路径规划，介绍了本课题使用的 Gmapping 算法，用于全局路径规划与局部路径规划的 A*算法和动态窗口法。

第 4 章是仿真验证和实车的自主导航实验，首先在 Gezabo 软件上创建了实车模型，在搭建的仿真环境中对所选用的 SLAM 技术和路径规划算法进行了验证。然后，使用搭建的阿克曼智能小车，在真实的环境中进行自主导航实验，验证算法可行性。

第 5 章主要对本文的工作进行了总结，并对进一步的工作进行了展望。

第二章　智能车软件系统设计

本课题采用 ROS（机器人操作系统）作为软件框架。首先，对 ROS 系统进行了深入分析，包括基本概念、文件系统、通信机制和系统特点的介绍。这为后续的软件系统设计提供了必要的理论基础。接下来，对智能车软件系统进行了整体设计。通过将智能车的各个功能部分进行划分，利用 ROS 系统的独立节点特性，进行总体功能节点的规划设计。这样的设计方式使得每个功能模块可以独立运行，并通过 ROS 的通信机制实现模块间的数据交流和协同工作。最后，对每个具体的功能节点进行了详细介绍。包括运动控制节点、定位建图节点和传感器驱动节点的软件环境搭建和设计。针对每个节点，介绍了其功能、设计思路和实现方式。通过搭建适当的软件环境，保证节点之间的协调运行和数据交换，最终完成了智能车软件系统的设计。

2.1 ROS 机器人操作系统

ROS 提供了一套灵活、可扩展的工具和库，用于构建机器人软件系统。它提供了通用的功能模块和通信机制，使开发者能够快速构建和集成各种机器人应用。通过使用 ROS 作为通用的软件框架，机器人开发者能够更加专注于机器人的特定任务和应用领域，减少重复开发工作，提高开发效率。同时，ROS 也促进了机器人技术的交流与共享，开发者可以通过 ROS 社区分享和获取各种开源的软件包和算法，加快了机器人技术的进步和创新。

2.1.1 ROS 基本概念

当从计算图的角度看待 ROS 机器人操作系统时，以下是一些基本概念的介绍：

（1）节点：在 ROS 中一个节点就是一个进程，它可以是一个可执行文件、一个脚本或者是一个库。节点可以通过发布消息和订阅话题的方式与其他节点进行通信。一个完整的机器人系统通常由多个运行中的节点组成。每个节点负责完成特定的功能或任务，通过节点之间的通信和协作，实现整个机器人系统的功能。比如本智能车中的底盘运动控制系统就是一个节点，运行中的激光雷达也是一个节点。

（2）节点管理器：节点管理器是 ROS 的一个核心组件，它负责启动、停止和监控 ROS 节点的运行状态。节点管理器还可以提供节点发现和管理服务。

（3）消息： 消息是节点之间传输的数据，可以是任何类型的结构化数据。ROS 提供了一种消息格式定义语言，可以用于定义自定义消息类型。

（4）通信机制：ROS 中的基本通信机制主要包括话题通信、服务通信和参数服务器。它们分别采用不同的实现策略，以满足不同的通信需求。

①话题通信

话题是一种发布者-订阅者模式的通信机制。发布者将消息发布到特定的话题上，而订阅者可以选择订阅感兴趣的话题，并接收相应的消息。发布者和订阅者之间是松耦合的，发布者不需要知道是否有订阅者存在，订阅者也不需要知道是否有发布者存在。通过话题通信，多个节点可以同时发布和接收消息，实现了节点间的异步通信。话题通信在实时性要求不高、数据流式传输的场景中被广泛使用，如传感器数据的发布和处理、地图数据的传输等。

在本课题中，机器人在执行导航功能时，使用的传感器是激光雷达，机器人会采集激光雷达感知到的信息并计算，然后生成运动控制信息驱动机器人底盘运动。以激光雷达为例，在 ROS 中有一个节点需要实时的发布当前雷达采集到的数据，导航模块中也有节点会订阅并解析雷达数据。再以运动消息的发布为例，导航模块会根据传感器采集的数据实时的计算出运动控制信息并发布给底盘，底盘也可以有一个节点订阅运动信息并最终转换成控制电机的脉冲信号。

②服务通信

服务通信是一种请求-响应模式的通信机制。服务提供者提供一个服务，而服务请求者可以向服务提供者发送请求，并等待响应。服务通信中存在明确的请求和响应过程，通信双方需要知道对方的存在。服务通信在需要进行请求和响应交互、需要获取特定结果的场景中常被使用，如执行特定任务、获取特定信息等。

在本课题中像 SLAM 模块就使用了服务通信，SLAM 节点需要向激光雷达节点发送接收环境信息数据，激光雷达节点处理请求，并返回处理结果。服务通信更适用于对实时性有要求、具有一定逻辑处理的应用场景。节点关系为一对多的特点。

③参数服务器

参数服务器是一种分布式的键值存储系统，用于存储和共享参数数据。节点可以从参数服务器中读取参数值，也可以将参数值写入参数服务器。参数服务器提供了一个统一的存储和访问参数的接口，使得多个节点可以共享和使用相同的参数配置。参数服务器在配置管理和参数共享方面起到了重要的作用，可以用于存储系统的配置参数、调整算法的参数等。

在本课题中，在导航实现时，会进行路径规划，像全局路径规划，设计一个出发点到目标点的大致路径。本地路径规划，会根据当前路况生成实时的前进路径。路径规划时，需要参考小车的尺寸，我们可以将这些尺寸信息存储到参数服务器，全局路径规划节点与本地路径规划节点都可以从参数服务器中调用这些参数。参数服务器，一般适用于存在数据共享的一些应用场景。

2.1.2 文件系统

在 ROS 机器人操作系统中，有一个文件系统用于组织和管理机器人软件开发所需的文件。这个文件系统称为"ROS Package"（ROS 软件包）。

ROS 软件包是 ROS 中最基本的文件和目录的组织单元，它将相关的文件按照一定规则组织在一起，以方便开发和维护，文件系统结构如图 2-1 所示。每个 ROS 软件包通常包含一个或多个节点、配置文件、启动脚本、消息类型定义、服务类型定义、库文件、测试文件等。

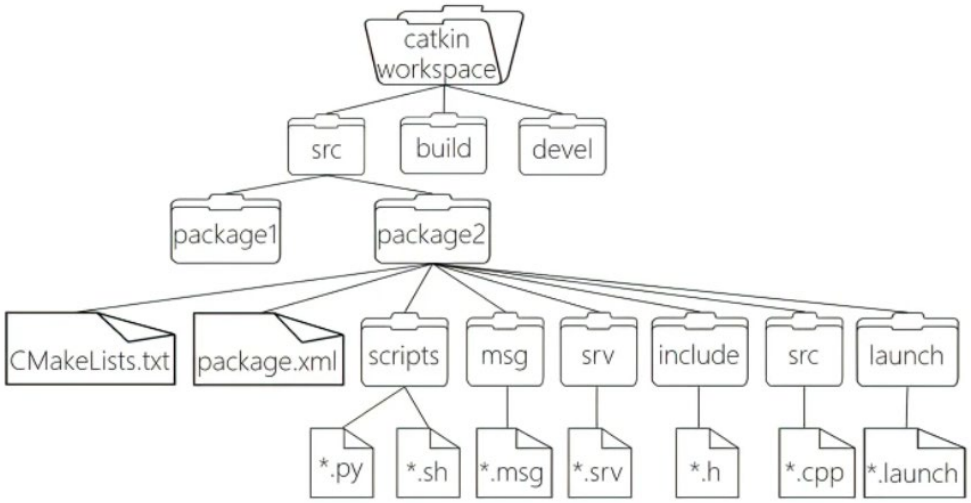


图 2-1 ROS 文件系统结构

在 Catkin workspace 文件夹下，创建一个名为“src”的文件夹是正确的第一步。“src”文件夹是用于存放开发者编写代码的地方，其中包含各种功能包（packages）。功能包是 Catkin 编译系统的基本单元，每个功能包可以包含一个或多个节点、库文件、配置文件等。接下来，可以使用“catkin_make”命令对工作空间进行编译。这将执行 CMake 和 Catkin 的构建过程，根据“src”文件夹中的功能包来生成构建目标。编译完成后，会在工作空间下生成“build”和“devel”两个文件夹。“build”文件夹用于存放 CMake 和 Catkin 的缓存信息以及编译过程中生成的中间文件。“devel”文件夹是用于存放编译后的可执行文件、库文件和其他构建输出的地方。

在 ROS 中，可执行文件一般可以分为两种来源。第一种是可执行脚本，例如 Shell 脚本和 Python 脚本，它们是直接可以运行的脚本文件。第二种是 C++文件，包括头文件（.h）和源文件（.cpp），这些文件需要经过编译后才能生成可执行文件。

2.1.3 系统特点

ROS 机器人操作系统的通信架构是其核心特点之一。ROS 采用了点对点的设计思路，其中每个功能模块或进程都被抽象为一个独立的节点。这种设计使得节点可以独立开发、测试和部署。这种设计模式不仅适用于单机器人系统，还可以在分布式环境下实现多机协同。因此，开发者可以独立地开发机器人的不同部分，并在不同的机器上进行运行，从而极大地提高了机器人系统的灵活性和扩展性。此外，ROS 还具有以下特点：

（1）开源：ROS 是一个开源软件平台，其源代码和文档都可以自由获取和使用。

（2）硬件无关性：ROS 支持多种硬件平台，可以运行在不同的嵌入式系统和普通计算机上。

（3）丰富的生态系统：ROS 拥有庞大的用户和开发者社区，可以提供丰富的功能模块和工具，帮助开发者快速构建机器人应用程序。

（4）灵活的通信机制：ROS 提供了多种灵活的通信机制，包括话题、服务和参数服务器，可以满足不同的应用场景。

（5）多语言支持：ROS 支持多种编程语言，包括 C++、Python 等，方便不同开发者使用自己熟悉的编程语言进行开发。

2.2 智能车 ROS 软件架构设计

本文在进行智能车软件设计时采用了 ROS 软件框架，并通过对功能需求的分析，将 ROS 架构划分为应用层、中间层、操作系统 OS 层和硬件层四个层级，以实现清晰的软件架构和明确的功能划分。在该架构中，ROS Master 作为节点管理器，统一管理系统中的各个节点，确保系统的稳定和有序运行。图 2-2 展示了软件框架的结构。

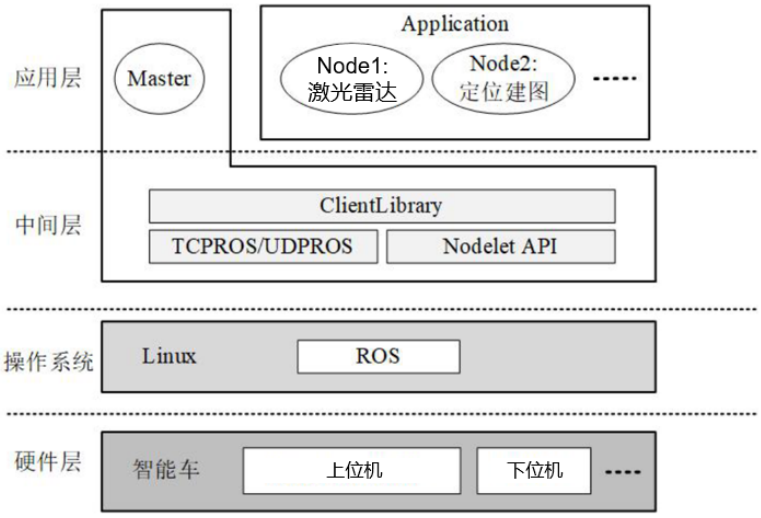


图 2-2 软件框架

对于操作系统层，ROS 是一种运行在 Linux 系统上的机器人开发软件框架，而不是传统的操作系统。每个 ROS 版本都需要与相应版本的 Linux 系统配套使用。针对本智能车项目，选择了 ROS Kinetic 版本，因此以与之相匹配的 Ubuntu 16.04 作为操作系统。

中间层在 ROS 系统中扮演着重要角色，它通过 TCP/UDP 网络协议进行通信封装，包括 TCPROS 和 UDPROS。此外，ROS 还提供了 Nodelet 用于进程间通信，并针对常用的机器人数据结构类型进行了封装，以提高开发效率。

在硬件层面，需要进行软件的设计和开发，以控制底层控制系统，并进行硬件驱动的安装和调试，以确保硬件系统按照系统设计的要求有序运行。

在应用层设计中，节点管理器负责注册和连接 ROS 系统中的各个功能模块节点，来协调整个系统的运行。针对每个功能模块，需要进行独立的设计，并将其作为一个 ROS 节点运行，以实现底盘运动控制、同步定位建图等主要功能。软件节点设计框图如图 2-3 所示。

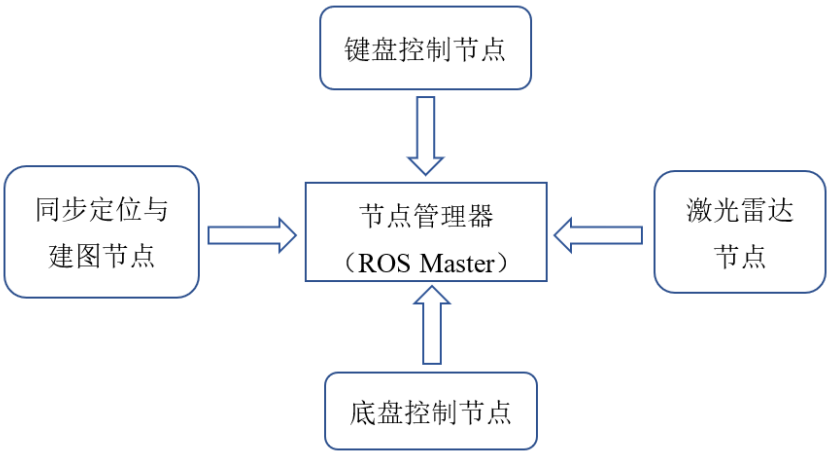


图 2-3 系统软件节点设计框图

在智能车软件系统的设计过程中，本文充分利用了 ROS 系统的特点，采用独立节点的设计方式。这种设计方法简化了复杂的软件开发过程，并根据智能车所需的具体功能进行节点的划分。

2.3 节点的软件设计

2.3.1 底盘控制节点设计

底盘运动控制节点的设计需要考虑下位机底盘控制器部分，包括制定下位机和 ROS 系统之间的串口通信协议，以确保它们能够有效地进行数据交换和指令传输。底盘控制器部分的代码一般分为系统初始化和主循环两部分。

在系统初始化部分，下位机进行一系列的准备工作，包括设置串口通信参数、初始

化底盘电机驱动器、配置传感器等。这些步骤旨在确保下位机与底盘硬件正常连接，并且各个组件都处于可工作的状态。

主循环是底盘控制器部分的核心部分，它是一个循环结构，不断地执行一系列操作来实现底盘的运动控制。在每一次循环中，下位机会读取传感器数据，例如编码器的反馈信息、惯性测量单元（IMU）的姿态数据等。然后，根据系统设计和算法逻辑，计算出底盘需要的运动指令，例如线速度和角速度。最后，将这些指令通过串口发送给 ROS 系统中的底盘运动控制节点，以实现底盘的运动控制。

底盘控制器部分的设计要考虑底盘硬件的特性和通信协议的制定，以确保下位机与 ROS 系统之间的稳定通信和正确的数据传输。通过有效的底盘控制器设计，可以实现对底盘的精确控制和运动策略的实现，从而为智能车系统提供准确的运动能力。

在下位机和搭载 ROS 系统的上位机嵌入式计算系统之间实现数据交互需要设计一个合适的串口通信协议。为了满足需求，我们约定以下规则：串口波特率设定为 38400，数据构成为 1 位停止位、8 位数据位，无奇偶校验位。帧头用于标识通信帧的开始，可以是一个固定的值或者特定的字符。功能码用于区分上位机发送的消息和下位机发送的数据，约定上位机发送的消息功能码为奇数，下位机发送的数据功能码为偶数。数据部分根据具体需求传输相应的数据内容，可以是传感器数据、控制指令等。校验码用于验证数据的完整性，可以采用简单的校验算法，如异或校验。

每个下位机具有一个唯一的 ID，用于级联设计时的标识。具体的串口数据格式可参考表 2-1。这样设计的串口通信协议能够确保下位机和上位机之间的可靠数据传输和正确解析，为智能车的控制和状态反馈提供了有效的数据交互方式。

表 2-1 串口数据格式

帧头	帧长度	ID	功能码	数据	预留位	CRC 校验
0x5a	0xXX	0x01	0xXX	0xXX…0xXX	0x00	0xXX

（1）帧头：固定值为 1 字节，使用 0x5A。

（2）帧长度：包括帧头、帧长度本身、ID、功能码、数据、预留位和 CRC 校验的全部数据，共 1 字节。

（3）ID：1 字节，用于标识下位机的编号，用于级联设计时的标识。

（4）功能码：1 字节，控制板端发送的功能码为偶数，控制板端接收的功能码为奇数。

（5）数据：根据功能码的定义，数据的长度和内容可能有所不同，最大长度为 250 字节。

（6）预留位：暂时设置为 0x00，为将来协议可能的扩展预留。

（7）CRC 校验：1 字节，采用 CRC-8/MAXIM 校验方式。

在串口通信协议中，功能码用于标识数据包的用途和类型。根据功能码的不同取值，可以定义不同的功能和操作。控制板端发送的功能码为偶数，控制板端接收的功能码为奇数，以便区分发送方和接收方。

具体的功能码定义可以根据智能车的需求进行设计和扩展，下面是一些可能的功能码如下表 2-2 所示：

表 2-2 功能码定义

功能码	功能
0x01	控制底盘运动指令
0x03	获取底盘当前状态信息
0x05	设置传感器参数
0x07	发送激光雷达数据
0x09	执行特定动作或任务
0x0B	更新固件版本
0x0D	请求系统自检

在制定好下位机的串口通信协议后，需要在 ROS 系统中设计底盘的运动控制节点/base_controller，图 2-4 是运动控制节点功能示意图。

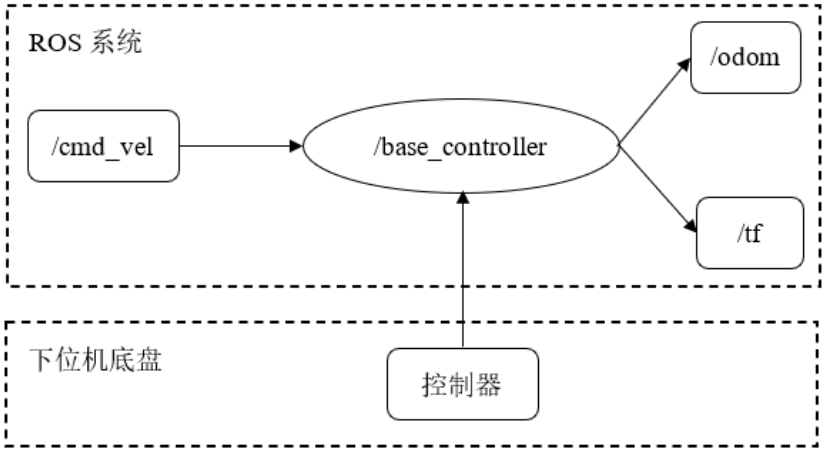


图 2-4 底盘运动控制节点功能示意图

在一般情况下，底盘的速度控制信息是通过激光雷达或其他传感器采集的数据，并经过算法处理后发送到 ROS 系统中的/cmd_vel 话题。底盘的控制节点（通常称为/base_controller）会订阅/cmd_vel 话题，以获取 ROS 系统中的速度信息。然而，由于下位机中的数据格式与 ROS 系统中的格式不一致，所以/base_controller 节点需要将接收到的速度信息转换为符合制定的串口通信协议的格式，然后将其发送给下位机。

在同步定位建图系统中，下位机中的 IMU 数据信息是必要的。为了获取这些信息，通常会使用一个名为/base_controller 的节点来接收下位机发送的数据。一旦/base_controller 节点接收到数据，它会根据预先定义的串口通信协议，将下位机的数据

转换为 ROS 系统中的数据格式。然后，转换后的数据将被发布到指定的 ROS 话题中。在这个过程中，一些关键信息的发送也是必要的。例如，坐标变换信息通常会发送到 ROS 系统中的/tf 话题中，而里程计信息则会发送到 ROS 系统中的/odom 话题中。这样，ROS 系统中的 SLAM（Simultaneous Localization and Mapping）算法就能够根据这些数据进行同步建图，将外部环境的信息转化为地图信息。

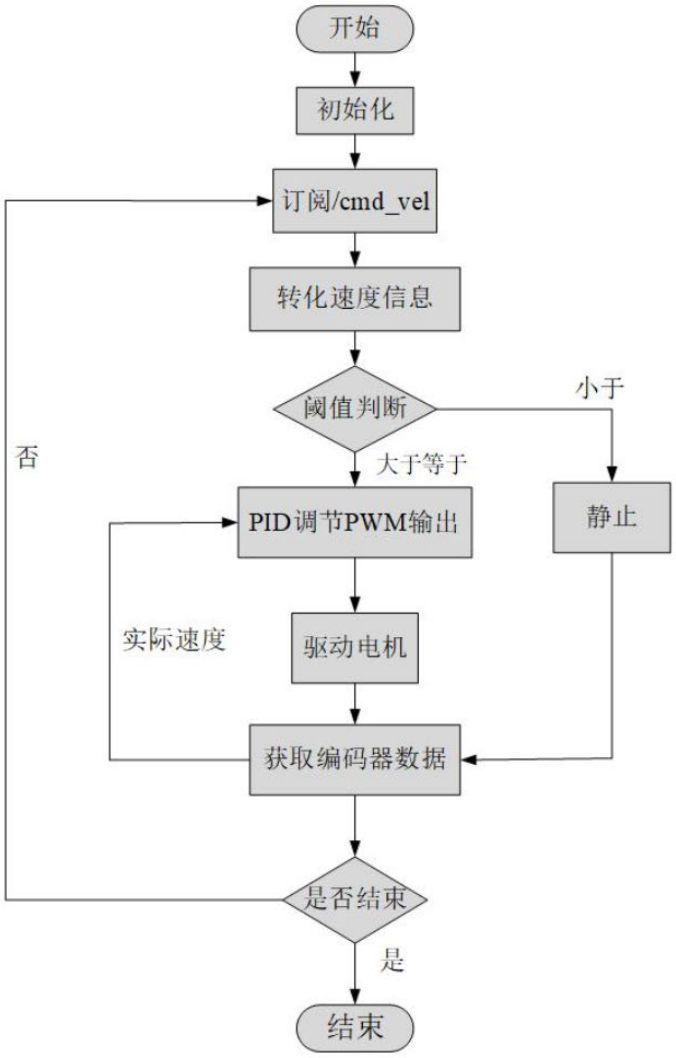


图 2-5 底盘运动控制节点算法流程图

底盘运动控制节点的算法流程如图 2-5 所示。在系统初始化完成后，底盘运动控制节点开始执行主要的算法流程，该流程包括以下几个步骤。

（1）接收/cmd_vel 话题中的速度信息，获取待控制的底盘运动指令。

（2）根据底盘的动力学模型和运动控制算法，将底盘运动指令转化为适合下位机的控制信号。

（3）将转化后的控制信号通过串口通信协议发送给下位机。

（4）下位机接收并解析控制信号，执行相应的底盘运动控制动作。

通过这样的算法流程，底盘运动控制节点能够实现根据系统要求控制底盘的运动。速度信息的来源可以根据具体的应用场景和需求进行灵活选择，包括算法处理的结果、PC 端手动指定的速度或导航算法生成的速度。

2.3.2 同步建图节点设计

在陌生的室内环境中，为了实现智能车的有效行驶，定位和导航是必不可少的。为了建立室内环境的地图，我们可以利用 SLAM 算法来处理传感器获取的数据。SLAM 算法通过将传感器数据与智能车的位姿信息相结合，实现环境地图的构建。智能车可以利用这个环境地图进行导航和避障操作。

本智能车采用基于粒子滤波的 SLAM 算法，即 Gmapping 算法。图 2-6 展示了 Gmapping 算法在实际运行中的结构。

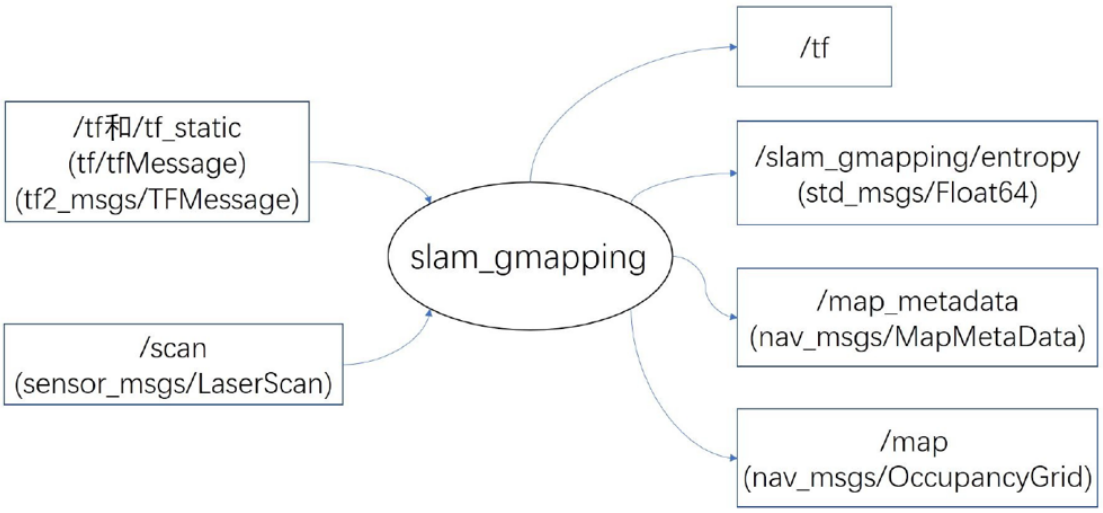


图 2-6 Gmapping 算法运行节点结构

中心位置的节点是 slam_gmapping 节点，该节点负责执行整个 Gmapping SLAM 算法的工作。

输入：

（1）/tf 和/tf_static：这些话题用于传输坐标变换信息，以支持多个坐标系之间的关系。其中，/tf 话题用于传输动态坐标变换信息，而/tf_static 话题用于传输静态坐标变换信息。您提到的两个重要的坐标变换是机器人底盘和激光雷达之间的变换，以及机器人底盘和里程计原点之间的变换。

（2）/scan：这个话题用于传输激光雷达的数据，类型可以是 sensor_msgs 或 LaserScan。激光雷达数据是 SLAM 算法中的关键输入，通过对环境进行扫描，获取障碍物的距离和位置信息。

输出：

（1）/tf：这个话题用于输出坐标变换信息，特别是 map_frame 和 odom_frame 之间的变换。这个变换描述了地图坐标系和里程计坐标系之间的关系，以便在导航过程中进行定位。

（2）/slam_gmapping/entropy：这是一个 std_msgs/Float64 类型的话题，用于反映机器人位姿估计的分散程度。熵是衡量位姿估计不确定性的指标，较高的熵表示位姿估计的不确定性较大。

（3）/map：这个话题用于输出由 slam_gmapping 建立的地图。地图是基于激光雷达数据和位姿估计生成的环境模型，包含了障碍物、空间结构和特征等信息。

（4）/map_metadata：这个话题用于输出地图的相关信息，包括地图的尺寸、分辨率、原点等。

地图建立好以后，智能车就可以利用已经建立好的地图进行导航了。对于智能车的导航功能，基于 ROS 的导航功能包提供了一个完整的系统框架。以下是导航系统框架图（图 2-7）。

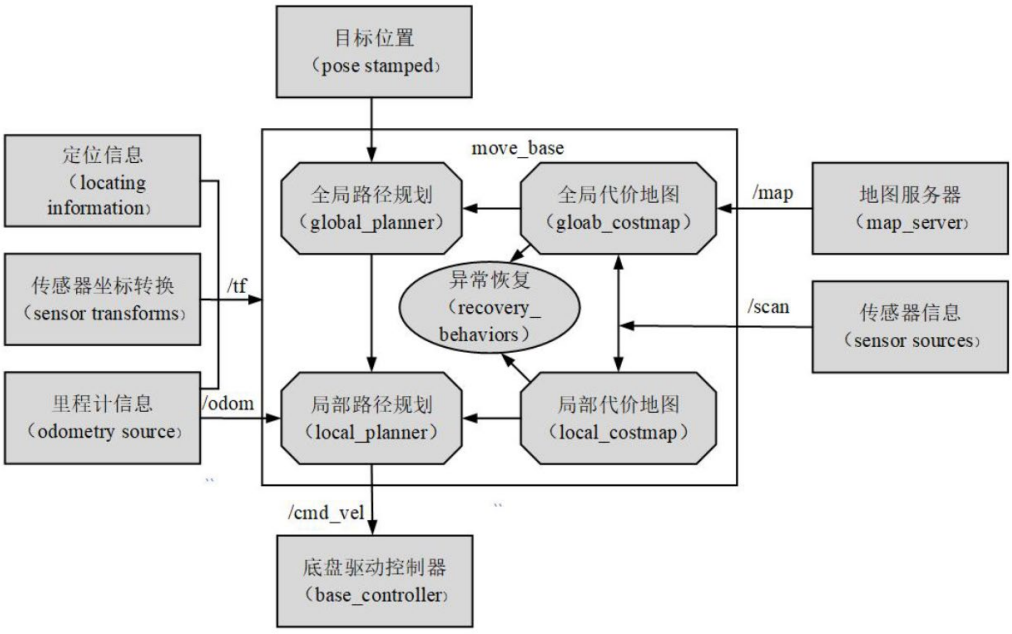


图 2-7 导航系统框架图

智能车导航系统启动后，在 PC 端启动 rviz 来可视化导航过程。通过 rviz 的界面加载已经建立的地图，并使用"2D Nav Goal"按钮选择智能车的目标位置，从而启动导航过程。导航系统框架图如图 2-7 所示。

（1）定位信息：定位是确定智能车在地图中的位姿的过程。通过在 rviz 中点击"2D Pose Estimate"按钮，然后在地图中用鼠标左键选取一个点，并拖动鼠标指定方向后松开，可以指定智能车当前的正方向。定位信息将被用于与地图进行匹配，从而更新雷达数据

与地图的对应关系。

（2）里程计信息：里程计信息是通过传感器坐标转换后将智能车的坐标转换到地图坐标系中的过程。里程计可以提供智能车在地图中的相对位置和移动信息。通过将里程计信息与定位信息相结合，可以更准确地确定智能车在地图中的位置。

（3）地图服务器：地图服务器的作用是在已知地图信息的情况下选择需要导航的地图。它可以加载已经建立好的地图文件，并提供给导航系统使用。地图服务器可以根据实际需求选择不同的地图，以适应不同的导航场景。

（4）动态避障：导航系统通过实时获取的传感器信息，特别是激光雷达的二维激光数据，可以实现动态避障功能。激光雷达可以扫描周围环境，检测障碍物的位置和距离，并根据障碍物的信息进行路径规划和调整，以避免与障碍物发生碰撞。

通过上述步骤和功能，智能车导航系统能够在已建立好的地图上进行自主导航，实现从起始位置到目标位置的路径规划和动态避障，提高智能车的自主性和导航能力。

在智能车的导航系统中，全局路径规划和局部路径规划是两个重要的步骤，分别使用全局代价地图和局部代价地图进行路径规划。

（5）全局路径规划：在全局路径规划中，当选择了导航目标点后，根据地图信息进行全局规划，确定智能车的整体行驶路线。一般常用的全局路径规划算法有 A*算法和 Dijkstra 算法，它们通过搜索地图上的节点和路径来找到一条最优或最短路径。全局代价地图提供了地图中各个区域的代价信息，例如障碍物的位置、通行能力等，以帮助路径规划算法做出合理的决策。

（6）局部路径规划：在智能车行驶过程中，如果通过二维激光数据检测到障碍物信息，就需要进行局部路径规划来避免碰撞。局部路径规划器（如 DWA 和 TEB 路径规划器）根据当前的传感器数据和车辆状态，在局部代价地图中规划出一条安全的、可行的路径。这样智能车可以根据局部路径进行调整和避障，保证行驶的安全性。

底盘运动控制器通过订阅局部路径规划节点发布的/cmd_vel 消息获取底盘运动的速度信息，然后根据这些速度信息来控制智能车的运动，实现路径跟踪和导航功能。整个过程中，全局路径规划和局部路径规划相互配合，使智能车能够在复杂的环境中安全、高效地导航。

2.4 本章小结

在本章中，我们首先对 ROS 系统的特点、通信机制和文件系统进行了分析，然后完成了智能车软件系统的总体框架设计。通过对智能车功能模块的划分，我们规划设计了总体功能节点。接下来，针对每个具体的功能节点，我们进行了智能车软件节点的设计。通过本章的工作，我们完成了智能车软件系统的设计，为后续的实施和开发奠定了基础。

第三章　SLAM 与路径规划

在室内 GPS 信号较弱，没有路标信号参考的陌生环境下，要想实现小车的自主导航功能，准确地知道小车在陌生环境中的位置以及小车姿态是实现这一功能的前提。在确定小车在陌生环境中的位置后，让小车到达某一预定位置，还需要根据已建立的环境地图规划到达目的地的最优路径。上一章介绍了小车的控制系统的搭建，本章节将介绍实现自主导航、避障等功能的关键技术，包括 SLAM（同时定位与建图）技术和路径规划的相关算法。

3.1 SLAM 概述

SLAM（Simultaneous Localization and Mapping，即同时定位与建图）是一种用于在未知环境中同时实现自主定位和地图构建的技术。SLAM 技术在机器人领域具有重要意义，它使机器人能够在没有先验地图或 GPS 信息的情况下，通过传感器数据来感知和理解环境，并实现自身的定位和地图生成。

SLAM 技术的核心挑战是解决机器人在未知环境中的自我定位和地图生成的问题。为了实现这一目标，SLAM 算法通常分为预处理、匹配、地图融合（地图更新）。

（1）预处理：在使用激光雷达数据进行后续处理之前，通常需要对原始数据进行预处理，以优化数据质量并剔除一些有问题的数据。这个预处理过程有助于提高数据的准确性和可靠性，从而为后续的算法和任务提供更好的输入。

（2）匹配：匹配是 SLAM 中的重要步骤，它的目标是将当前局部环境的点云数据与已建立地图进行匹配，以找到其在地图上的对应位置。匹配过程是 SLAM 中的一个关键环节，它为后续的定位和建图提供了准确的位置信息。

（3）地图融合：地图融合是 SLAM 过程中的一个重要步骤，它将当前时刻激光雷达获取的新数据与已建立的地图进行融合，从而更新地图的信息。地图融合的目标是将新数据与原始地图进行拼接，使得地图能够反映环境的变化。

图 3-1 展示了地图融合的过程。在每一时刻，激光雷达会获取到新的点云数据，这些数据需要与已建立的地图进行匹配，确定其在地图上的位置。然后，根据匹配结果，将新数据与地图进行融合，更新地图的信息。

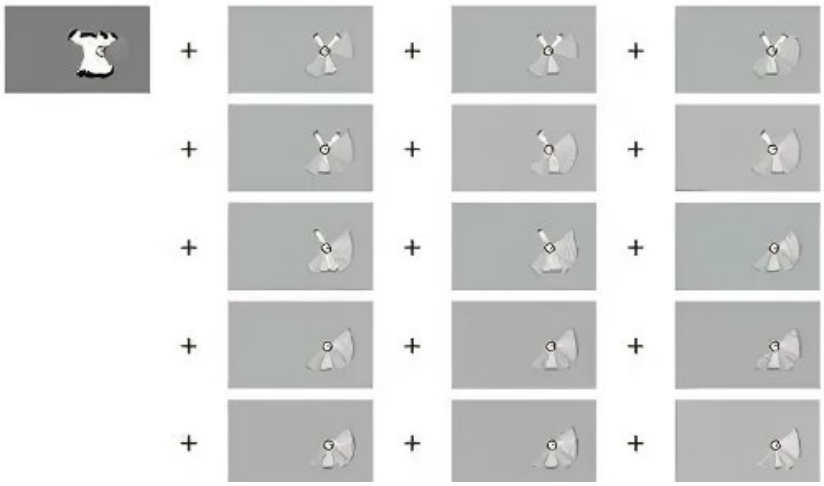


图 3-1 地图融合过程

机器人通过不断移动和感知环境，获取新的激光雷达数据并与地图进行匹配和融合。通过重复执行这个过程，地图会逐渐扩展，障碍物和环境特征会逐渐准确地被反映在地图中。图 3-2 所示的格栅地图是通过逐步执行数据融合过程得到的最终结果，它反映了机器人所处环境的结构和特征，为机器人的自主导航和决策提供重要的参考依据。

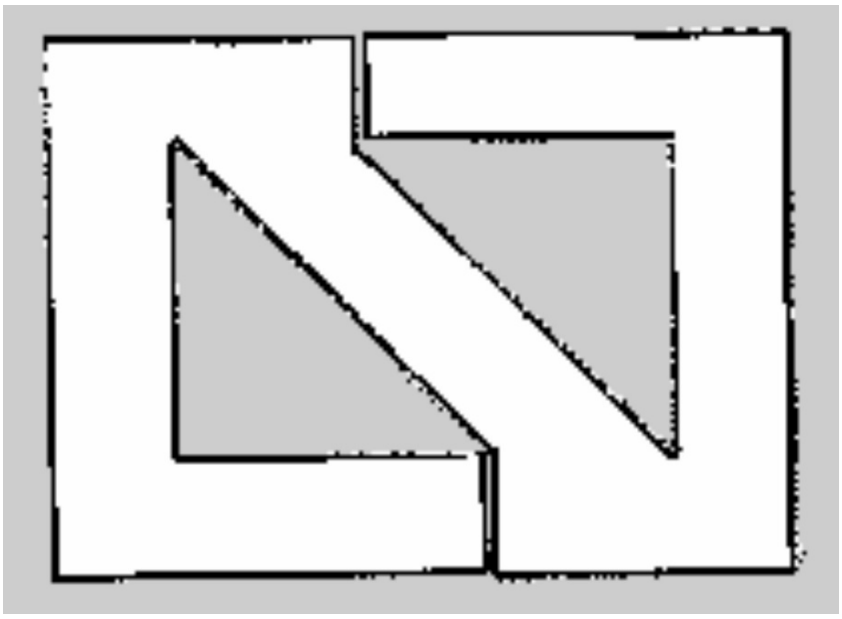


图 3-2 格栅地图

3.2 Gmapping 算法

结合已有的激光雷达的性能以及室内的应用场景，本文使用了应用比较广泛的 Gmapping 算法进行智能车的定位与建图。Gmapping 算法基于 RBPF（Rao-Blackwellized Particle Filters）技术，通过同时定位与建图的方法实现地图的构建。

在 SLAM 问题中，需要同时估计机器人的位姿和建立环境的地图，但位姿估计和地

图建立之间存在着一种相互依赖的关系。一方面，为了得到准确的位姿估计，通常需要有一个准确的地图作为参考。机器人的位姿估计是通过将传感器数据与地图进行匹配来实现的，如果地图不准确或不完整，位姿估计的精度将受到影响。另一方面，为了建立一个好的地图，需要有准确的位姿信息作为地图的基准。如果位姿估计不准确，将会导致地图中的障碍物位置和机器人轨迹的不准确，从而影响地图的质量和可用性。

为了解决 SLAM 问题中位姿估计和地图建立之间的相互依赖关系。FastSLAM 算法采用了 RBPF（Rao-Blackwellized Particle Filters）方法，将 SLAM 问题分解为机器人定位和地图构建两个子问题。在 FastSLAM 算法中，机器人定位问题是通过粒子滤波器来解决的。粒子滤波器使用一组粒子来表示机器人的位姿，每个粒子都有一个权重，表示其对位姿的置信度。通过不断更新粒子的权重，可以逐步优化机器人的位姿估计。这样，机器人的定位问题被分解为对粒子集合的更新和重采样。另一方面，地图构建问题则是在已知机器人位姿的情况下进行的。当机器人的位姿已经确定时，可以利用传感器数据来构建地图。在 FastSLAM 算法中，每个粒子都维护了一个地图的部分信息，通过在粒子级别上构建地图，可以得到整体的地图信息。因此，地图构建问题可以看作是对每个粒子进行地图更新的过程。

在 FastSLAM 算法中，机器人轨迹估计问题使用了粒子滤波方法，其中每个粒子包含了机器人的轨迹和对应的环境地图。然而，这种方法可能面临两个问题：内存爆炸和粒子耗散与多样性丢失。

第一个问题是内存爆炸，当环境较大或机器人的里程计误差较大时，为了得到准确的估计，需要增加更多的粒子。然而，每个粒子都需要存储机器人的轨迹和地图信息，这会占用大量的内存资源。

第二个问题是粒子滤波中的重采样操作可能导致粒子的耗散和多样性的丢失。重采样是为了确保当前粒子的有效性，但它可能会使一些粒子在重采样过程中被丢弃，导致粒子集合中的多样性减少。这可能导致滤波器集中在局部最优解附近，而忽略了其他潜在的解。

为了解决这些问题，Gmapping 提出了两种针对性的解决方法：

（1）降低粒子数量：通过减少粒子的数量可以减轻内存压力。Gmapping 算法使用了一种基于栅格地图的近似方式，可以有效降低所需的粒子数量，从而减少内存占用。

（2）缓解粒子消散：为了避免粒子的耗散和多样性的丢失，Gmapping 采用了自适应的重采样策略。该策略会根据粒子的权重分布情况，对粒子集合进行重采样操作，以保持粒子的多样性，并尽量避免耗散现象的发生。

通过降低粒子数量和缓解粒子消散，Gmapping 算法能够在实际应用中更好地处理 SLAM 问题，并取得较好的定位和地图建立效果。这些技术的引入使得 FastSLAM 算法

在实践中变得可行，克服了其理论上的局限性。

3.3 全局路径规划

路径规划是通过考虑已知地图和特定准则（如最短时间或最短距离）来规划最佳路径，使移动小车能够避开障碍物并以最优的方式到达目标点。这意味着智能小车会根据预设目标，在环境中评估障碍物的位置，并通过智能算法确定最佳的行进路径。路径规划的目标是确保移动小车安全、高效地到达目标位置。

在本课题中，移动小车的全局路径规划是基于已知的格栅地图进行的。通过代价函数，在已知地图中为机器人选择一条安全路径，使其能够到达指定目标。其中，我们采用了 Dijkstra 算法作为路径规划的基础。Dijkstra 算法最初由荷兰科学家 Edsger Wybe Dijkstra 于 1956 年提出，它利用广度优先搜索的方式解决带权有向图的单源最短路径问题。这个算法存在多个变体，原始版本的 Dijkstra 算法用于寻找两个顶点之间的最短路径，但更常见的变体是将一个顶点固定为源节点，然后找到该顶点到图中所有其他节点的最短路径，从而形成一个最短路径树。

Dijkstra 算法通过逐步扩展距离最短的节点来构建最短路径树，同时计算出每个节点到起始点的最短距离。这个过程包括以下步骤。

（1）初始化：将起始点的最短距离设置为 0，其他节点的最短距离设置为无穷大。

（2）选择最短距离节点：从未标记的节点中选择一个距离最短的节点，标记该节点为已访问。

（3）更新邻居节点距离：对于当前节点的所有邻居节点，计算通过当前节点到达邻居节点的距离，并更新邻居节点的最短距离。

（4）重复步骤 2 和步骤 3，直到所有节点都被访问或者目标节点的最短距离被确定。

（5）构建最短路径：从目标节点开始，按照每个节点的最短距离逐步回溯，直到回溯到起始节点，形成最短路径。

通过 Dijkstra 算法，可以找到起始点到目标点的最短路径，并且在已知的格栅地图中考虑了每个节点之间的权重或代价。这样可以确保移动小车在全局路径规划中选择安全的路径，并尽快到达目标位置。

经典的 Dijkstra 算法的时间复杂度为 O(n^2)，其中 n 表示图中的顶点数量。随着顶点数量的增加，算法的时间复杂度也会增加，并且循环次数会急剧上升，导致效率下降。此外，Dijkstra 算法在求解过程中确实会产生大量的临时路径，并且需要维护一个距离表和一个标记表来记录节点的距离和是否被访问过。这些临时路径和数据结构的维护会占用较大的内存空间，特别是在处理大规模图时可能会消耗大量的内存。

BFS 算法类似于 Dijkstra 算法，但引入了启发式搜索的概念。它通过选择离目标节点

最近的节点来扩展，利用启发式函数快速导向目标节点，从而加速搜索过程。然而，BFS 算法无法保证找到最短路径，但在大规模图或复杂环境中具有较高的搜索效率。选择适当的路径规划算法应考虑具体应用需求和图的特性。

A*算法是一种启发式搜索算法，它综合了 BFS 和 Dijkstra 算法的优点。通过引入启发式函数来评估节点到目标的代价，A*算法能够快速导向目标节点并找到一条最优路径。它通过综合考虑节点的实际代价和启发式函数的估计值，选择具有最小综合代价的节点进行扩展，从而在保证最优路径的同时提高搜索效率。这使得 A*算法在路径规划和图搜索等应用中广泛受到使用，并成为一种常用的启发式搜索算法。A*算法流程图如图 3-3 所示。

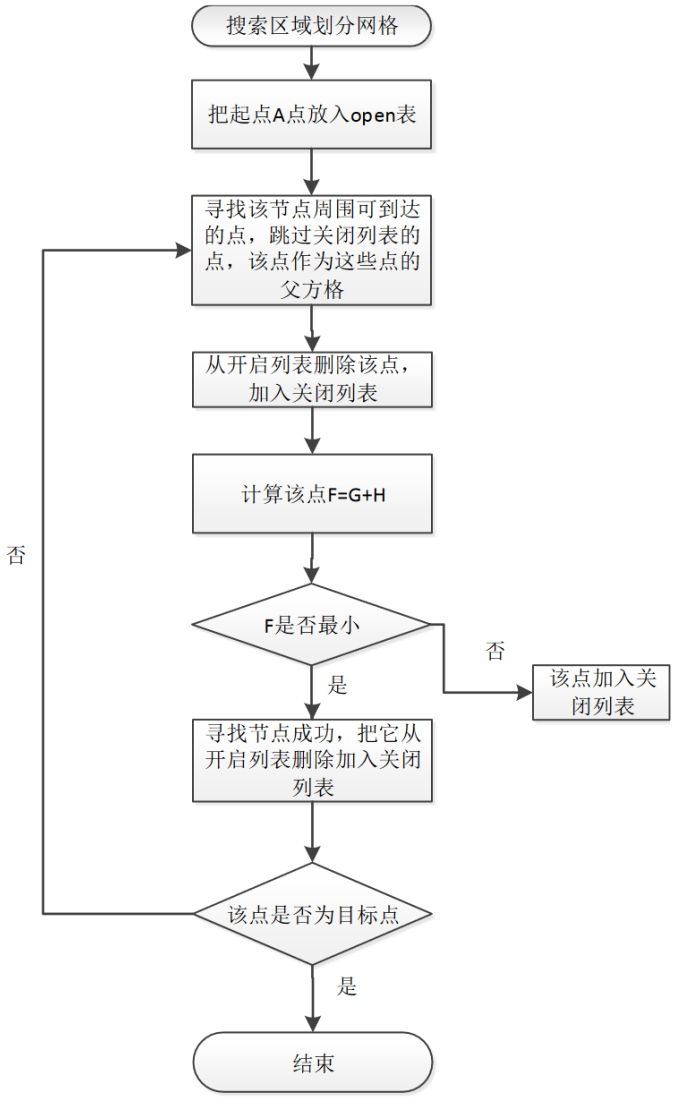


图 3-3 A*算法流程图

通过使用 Matlab 进行仿真，可以实现 Dijkstra 算法和 A*算法的路径搜索。首先，创建一个 10x10 的网格地图，其中绿色表示起始点，黄色表示目标点，黑色表示障碍物。

然后，使用启发式函数选择离目标点最近的相邻框进行搜索，并将已搜索的框标记为红色，下一次搜索的框标记为蓝色。重复此过程直到到达黄色区域。最后，找到最短路径，该路径将作为解。

根据图 3-4 中展示的 Dijkstra 算法的搜索范围和最终路径规划，以及图 3-5 中展示的 A*算法的搜索范围和路径规划，可以明显观察到 A*算法相比于 Dijkstra 算法具有更小的搜索范围和更快的搜索速度。因此，在本课题中，我们选择 A*算法作为全局路径规划算法，以实现更高效的路径规划。

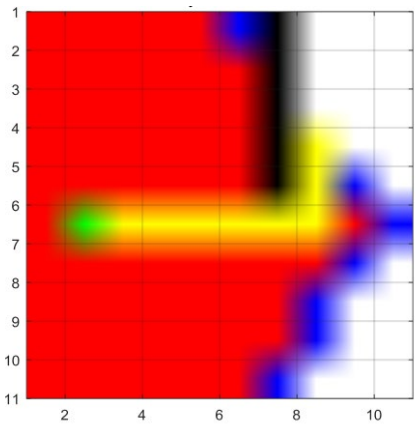


图 3-4 Dijkstra 算法仿真图

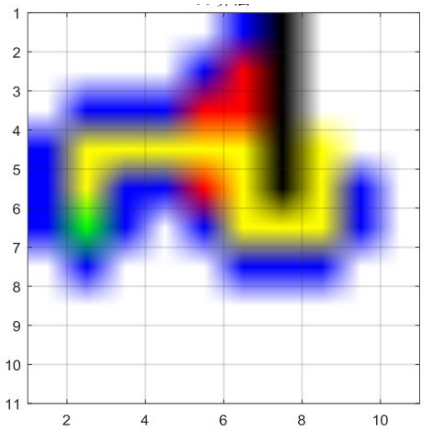


图 3-5 A*算法仿真图

3.4 局部路径规划

为了赋予移动小车自主避障的能力，在机器人 SLAM 与路径规划系统中，我们采用了动态窗口法（DWA）作为局部路径规划算法。DWA 算法通过定义机器人的状态空间窗口，考虑速度和加速度等因素，在局部环境中搜索合适的速度命令，以生成安全的局部路径。通过模拟多个速度命令并评估其安全性和优劣，DWA 算法选择最佳速度命令，实现了实时避障和灵活路径调整的功能。选择 DWA 算法可以确保移动小车快速响应障碍物并实现高效的局部路径规划。

动态窗口法是一种基于机器人动态特性和环境感知的路径规划方法。它通过根据机器人的运动能力和感知到的环境信息，生成一系列动态窗口，每个窗口代表机器人可能的速度和转向范围。然后，通过评估每个窗口的轨迹，选择具有最佳轨迹代价的窗口作为机器人的局部路径。这样，移动小车可以根据实时的环境变化，灵活地调整自身的速度和转向，以避免障碍物并沿着安全的路径向目标点移动。

采用 DWA 作为局部路径规划算法的机器人具有较高的灵活性和适应性，能够在复杂环境中快速响应并做出决策。这使得移动小车能够安全、高效地导航并完成任务。移动机器人运动模型如图 3-6 所示。

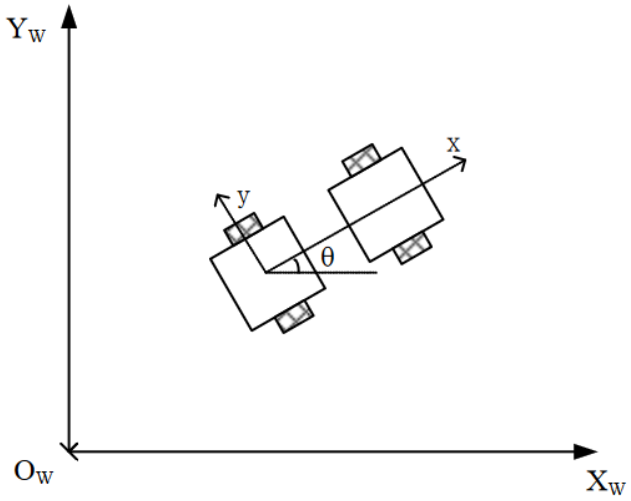


图 3-6 机器人直线运动模型

移动小车沿 y 轴方向速度为 0，移动小车的速度 v 即为机器人坐标系下 x 轴方向速度，由此可得机器人在世界坐标系下航迹推演公式为公式（3-1）。

$$\begin{cases} x = x + v\Delta t\cos(\theta) \\ y = y + v\Delta t\sin(\theta) \\ \theta = \theta + \omega\Delta t \end{cases} \tag{3-1}$$

在移动机器人的轨迹推演中，我们使用航迹推演公式来预测机器人在一定时间内的运动轨迹。通过采样多组速度并计算评价函数，我们可以选择最优轨迹及对应速度来驱动机器人前进。然而，我们还需考虑机器人的动力学限制、环境中的障碍物、地形和地图限制以及传感器的限制等因素，以确保生成的轨迹在限制范围内并能避免碰撞或进入危险区域。这些限制因素包括：

（1）机器人动力学限制：机器人的物理特性和动力学参数会对其运动轨迹产生影响。例如，最大速度、最大加速度以及最大转弯半径等参数都需要在速度采样过程中进行考虑，以确保生成的轨迹在机器人的可操作范围内。

（2）环境障碍物限制：机器人在运动过程中必须避开环境中的障碍物，以确保安全导航。因此，在采样速度时，需要考虑到障碍物的位置和形状，以及机器人与障碍物之间的最小安全距离，从而避免发生碰撞。

（3）导航目标限制：机器人的导航目标也可能对其轨迹产生限制。例如，目标位置可能位于无法直接到达的区域，或者存在特定的进入条件。因此，在速度采样时，需要考虑如何使机器人能够在满足目标要求的同时找到最优的轨迹。

DWA 算法基本思路如下：

（1）获取机器人当前状态，包括位置和速度。

（2）在速度空间中进行采样，生成多组候选速度。

（3）对于每组候选速度，利用航迹推演公式预测机器人在一定时间内的轨迹。

（4）对每条轨迹，评估其与目标之间的距离、与障碍物之间的安全距离以及平滑度等指标，计算出一个评价函数值。

（5）根据评价函数值，选择具有最优评价的速度，作为机器人的下一步移动速度。

（6）根据选择的速度，控制机器人实际移动，并更新机器人的状态。

（7）重复执行步骤 2 到步骤 6，以不断调整机器人的速度和轨迹，实现局部路径规划和避障。

在每个时间步骤中，持续监测环境和机器人状态，根据需要进行路径修正和调整，以确保机器人能够安全、高效地导航到目标位置。

3.5 本章小结

本章节详细介绍了本课题所使用的相关算法，包括 SLAM 概述、Gmapping 算法、全局路径规划和局部路径规划。SLAM 的重点是介绍了成熟的 Gmapping 算法，用于实现机器人的建图和定位。全局路径规划采用了广泛应用的 A*算法，用于在构建的格栅地图上获取机器人到指定目标点的最优路径。局部路径规划使用了动态窗口法，实现了机器人在局部环境中的路径规划和避障。这些算法为顶层控制实现提供了可靠的算法基础，使得智能小车能够实现自主导航和避障功能。

第四章　仿真与自主导航实验

以上章节详细介绍了智能小车的实时定位与建图算法以及路径规划算法，并确定了本课题所采用的路径规划算法。本章将进行仿真实验和真实实验，对移动机器人在室内环境下的性能进行评估和分析。通过仿真实验和真实实验的结果，以验证算法的可行性。

在本章节中，首先将在 Gazebo 仿真平台上搭建物理模型，然后在 ROS 系统中部署 SLAM 和导航算法来验证算法在该搭建小车上的可行性。最后将该算法移植到智能车的上位机中，在真实的实验环境下进行建图以及自主导航实验，验证算法部署到实车上后实际效果。接下来将分别从仿真的实车实验进行介绍。

4.1 Gazebo 仿真平台搭建

4.1.1 整车与环境搭建

为了简化整车模型并保留关键部分，在 Gazebo 仿真平台搭建了图 4-1 所示的模型。模型中保留了传感器部分，包括激光雷达和 IMU，它们分别由黑色圆柱和立方块表示。为了增加实验的真实性，在激光雷达参数中添加了噪声，并在 IMU 参数中添加了高斯噪声。控制部分包括舵机转向和四个车轮的阻尼，同时考虑了前后车轮的传动属性。为了验证实验的准确性，我们在 Gazebo 中建立了仿真环境，如图 4-2 所示，以模拟真实场景下的移动机器人行驶情况。

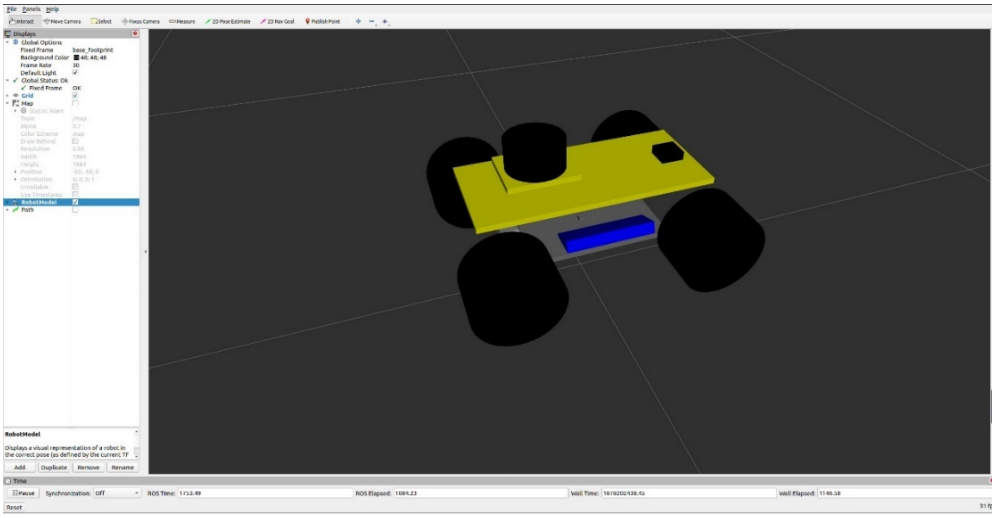


图 4-1 Gazebo 中整车模型

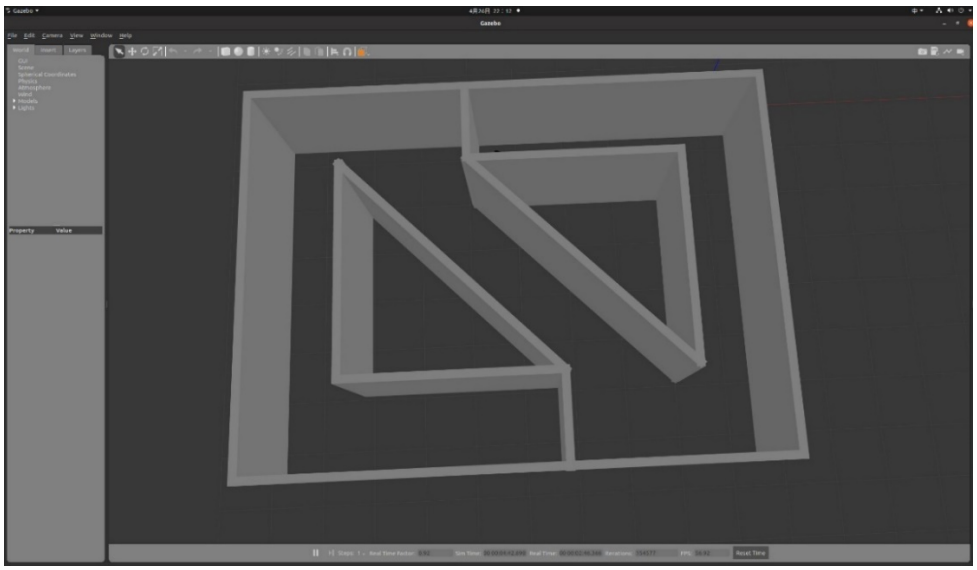


图 4-2 Gazebo 中环境搭建

4.1.2 在仿真环境中创建地图

基于以上 Gazebo 中建立的整车模型和搭建的模拟环境，进行仿真建图。首先对 Gmapping 配置文件进行配置如图 4-3 所示。

```
<launch>
  <param name="use_sim_time" value="true"/>
      <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
        <remap from="scan" to="scan"/>
        <param name="base_frame" value="base_footprint"/><!--底盘坐标系-->
        <param name="odom_frame" value="odom"/> <!--里程计坐标系-->
        <param name="map_update_interval" value="5.0"/>
        <param name="maxUrange" value="16.0"/>
        <param name="sigma" value="0.05"/>
        <param name="kernelSize" value="1"/>
        <param name="lstep" value="0.05"/>
        <param name="astep" value="0.05"/>
        <param name="iterations" value="5"/>
        <param name="lsigma" value="0.075"/>
        <param name="ogain" value="3.0"/>
        <param name="lskip" value="0"/>
        <param name="srr" value="0.1"/>
        <param name="srt" value="0.2"/>
        <param name="str" value="0.1"/>
        <param name="stt" value="0.2"/>
        <param name="linearUpdate" value="1.0"/>
        <param name="angularUpdate" value="0.5"/>
        <param name="temporalUpdate" value="3.0"/>
        <param name="resampleThreshold" value="0.5"/>
        <param name="particles" value="30"/>
        <param name="xmin" value="-50.0"/>
        <param name="ymin" value="-50.0"/>
        <param name="xmax" value="50.0"/>
        <param name="ymax" value="50.0"/>
        <param name="delta" value="0.05"/>
        <param name="llsamplerange" value="0.01"/>
        <param name="llsamplestep" value="0.01"/>
        <param name="lasamplerange" value="0.005"/>
        <param name="lasamplestep" value="0.005"/>
      </node>

      <node pkg="rviz" type="rviz" name="rviz" />
      <!-- 可以保存 rviz 配置并后期直接使用-->
      <!-- <node pkg="rviz" type="rviz" name="rviz" args="-d $(find my_nav_sum)/rviz/gmapping.rviz"/> -->

</launch>
```

图 4-3 Gmapping 配置文件

在 Gazebo 中进行建图。首先启动 Gazebo 仿真环境，然后启动地图绘制的 launch 文件，其中包含了 Gmapping 算法功能包、仿真模型的配置文件等，启动键盘控制节点，用于控制小车运动建图，在 rviz 中添加组件，显示栅格地图。最后，就可以通过键盘控制

Gazebo 中的机器人运动，同时，在 rviz 中可以显示 Gmapping 发布的栅格地图数据了，下一步，还需要将地图单独保存。使用 Gmapping 算法对仿真环境的建图效果如图 4-4 所示。

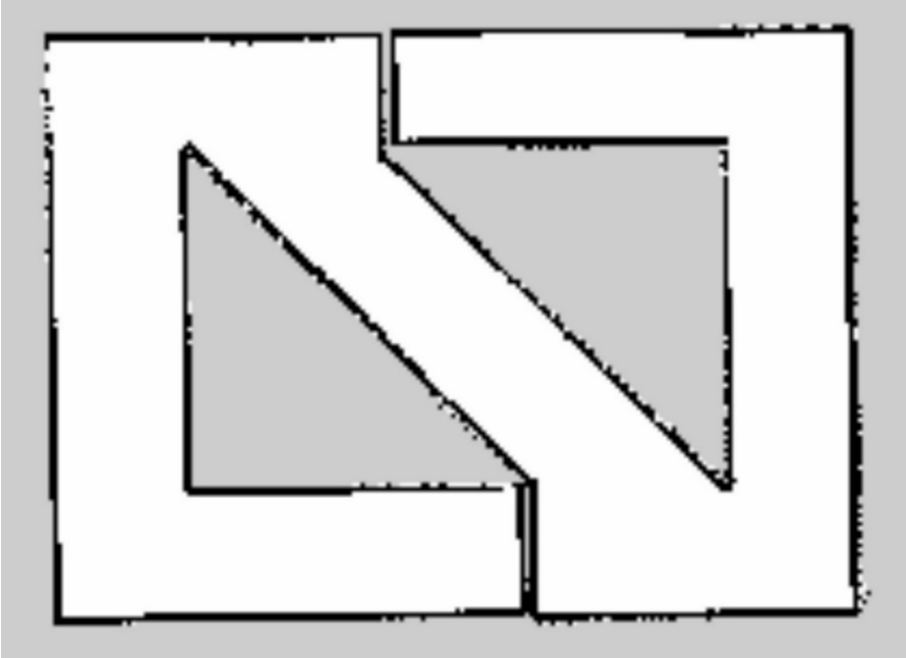


图 4-4 仿真环境建图效果

4.2 Gazebo 自主导航仿真测试

本实验使用了 Gazebo 和 rviz 两个可视化工具进行自主导航仿真，如图 4-5 所示。在仿真环境中，将移动小车模型导入 Gazebo 物理模型仿真中，以模拟小车在真实世界中的运动。而在右侧的 rviz 控制仿真中，将移动小车需要接收和发布的话题可视化出来，方便监测和调试机器人的状态和行为。通过这两个工具的联合使用，能够全面地模拟和观察移动小车的自主导航过程。

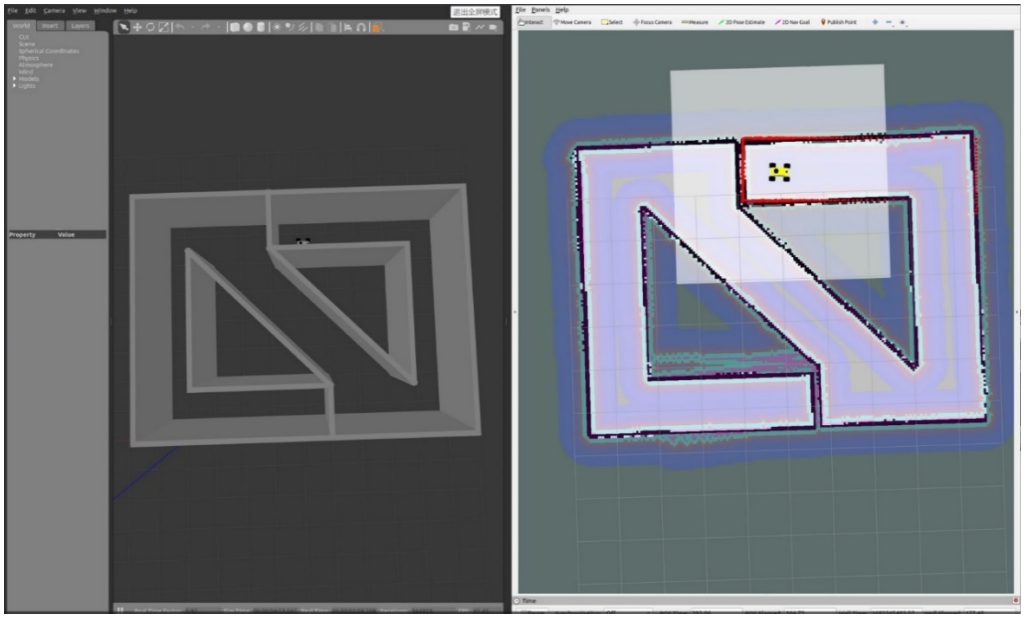


图 4-5 Gazebo 物理仿真与 rviz 可是化

通过在 Rviz 中使用 2D NavGoal 工具，在地图中发布目标指令，如图 4-6 所示的红色箭头。当接收到指令后，整车启动 Move_base 节点，进行全局和局部路径规划，并进行测试。图 4-7 中的绿色线代表 A*算法生成的全局路径，它是一条光滑的路径从起点到目标点，并且能够绕过障碍物。红色线表示动态窗口法生成的局部路径规划结果，虽然只是一小段路径，但反映了智能小车当前的局部行驶路径。

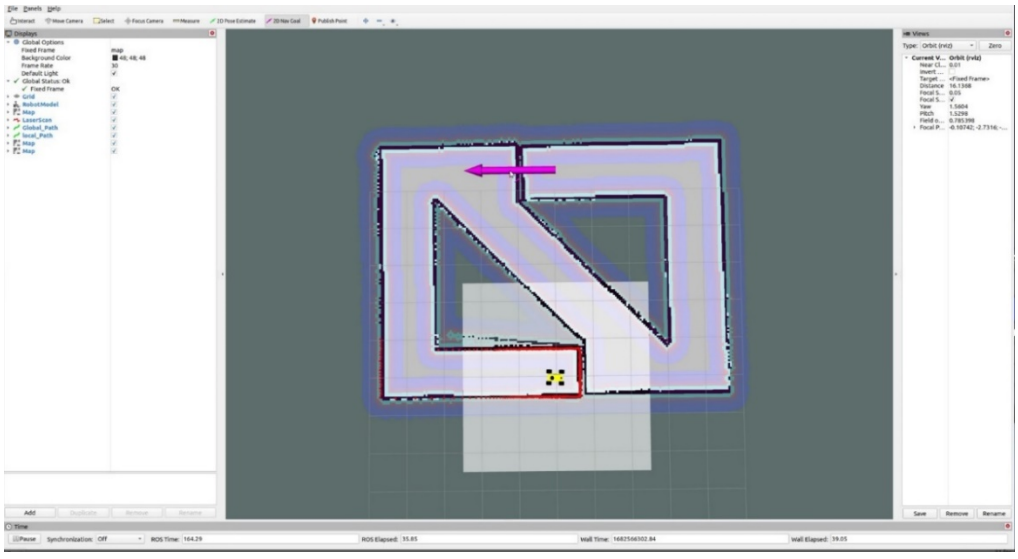


图 4-6 设置目标点

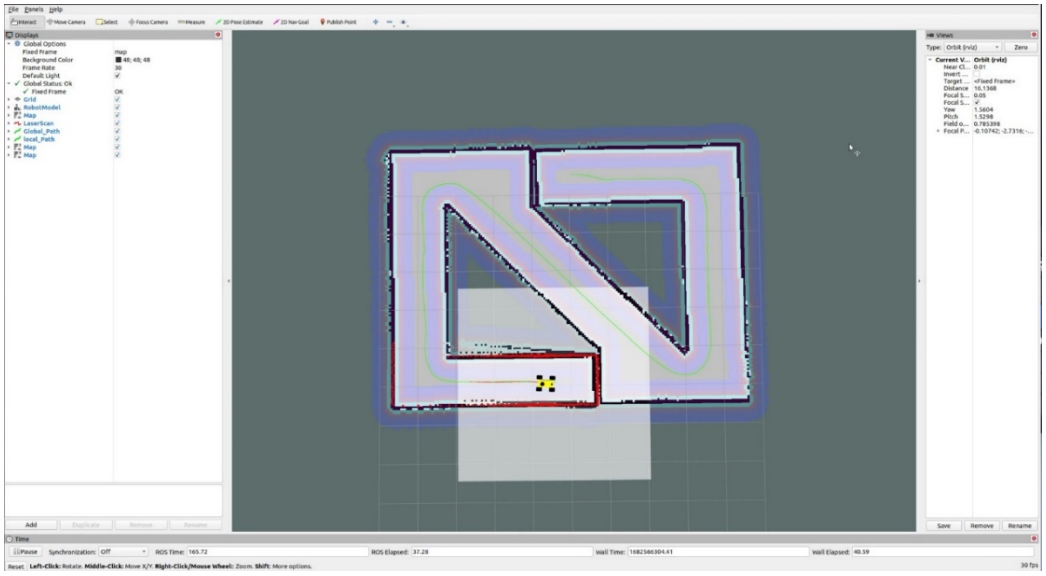


图 4-7 路径规划

现在仿真环境中添加障碍，如图 4-8 所示，而这些障碍物是在建图后添加，所以并没有出现在格栅地图信息中，因此，从全局路径规划的绿色曲线可以看出并没有避开障碍物。

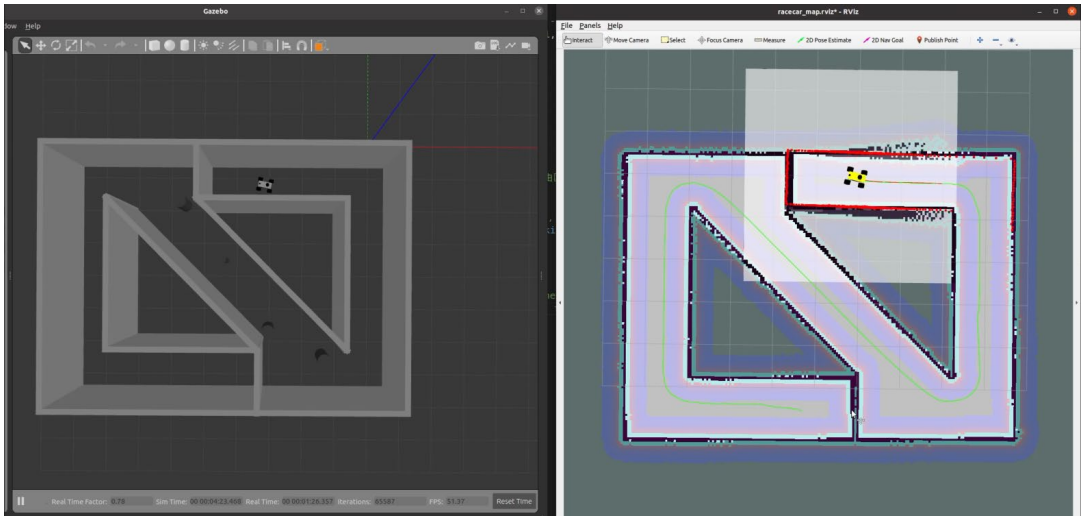


图 4-8 避障仿真环境

当没有遇到建图后添加到仿真环境中的障碍物时，智能车将继续沿着全局规划的路径行进，当智能车移动到障碍物附近时，由激光雷达感知智能车周围的障碍物信息，通过局部路径规划器调整智能车的行进路径，如图 4-9 所示的红色曲线，进而绕开障碍物。

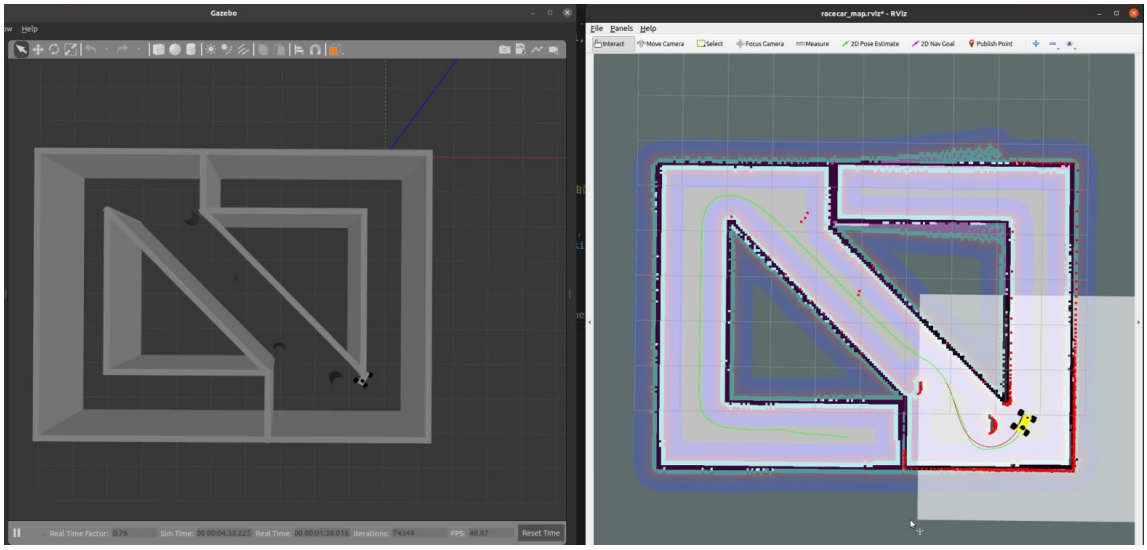


图 4-9 仿真避障

下面是仿真的演示视频生成的二维码，其中图 4-10 为自主导航仿真视频，图 4-11 为自主避障仿真视频，如需观看，请使用手机扫码即可。

图 4-10 自主导航仿真视频

图 4-11 自主避障仿真视频

4.3 自主导航实验

在上一小节中，在 Gazebo 环境中建立了整车模型，并在搭建的模拟环境中进行了建图以及路径规划仿真，通过仿真验证了算法在搭建小车系统上应用的可行性。接下来在搭建的真实环境中，对本课题中的智能车进行自主导航实验。

4.3.1 移动小车硬件架构

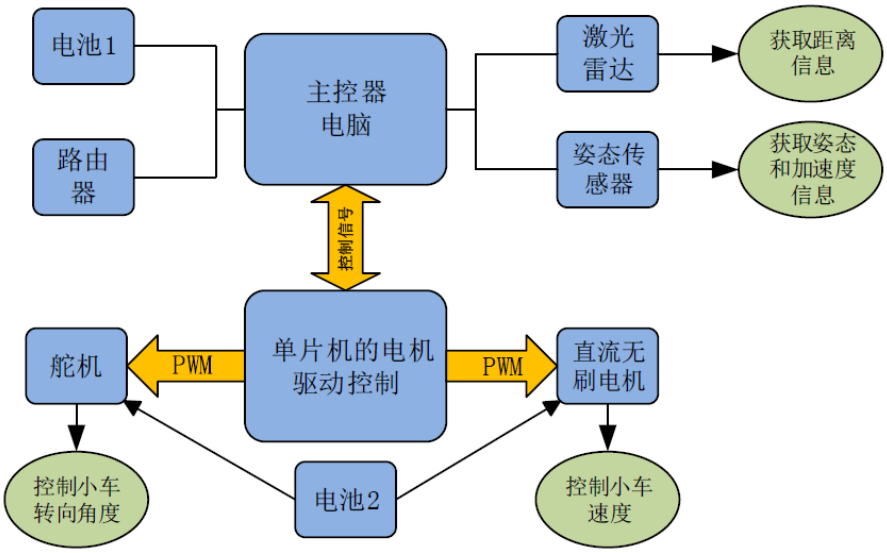


图 4-12 硬件架构

智能车的硬件系统主要由以下组成部分构成：主控电脑、激光雷达传感器、姿态传感器、直流无刷电机、舵机、单片机和路由器。整个硬件架构框架如图 4-12 所示。主控电脑是整个系统的核心，负责控制和协调各个硬件设备的工作。激光雷达传感器用于获取车辆周围物体的距离信息。姿态传感器用于获取车辆的姿态和加速度信息。直流无刷电机作为驱动装置，用来控制车辆的前进和后退。舵机则用于控制车辆的转向。单片机作为下位机的控制器，接收主控电脑传输的控制信号，并进一步控制直流无刷电机和舵机。为了方便远程控制，车辆还配备了路由器。组装完成后的智能小车如图 4-13 所示。

图 4-13 智能小车

由于试验室的场地有限，分别搭建了两种试验环境来测试智能车的自主导航和避障的能力，图 4-14 为搭建的自主导航实验环境，图 4-15 为搭建自主避障实验环境。

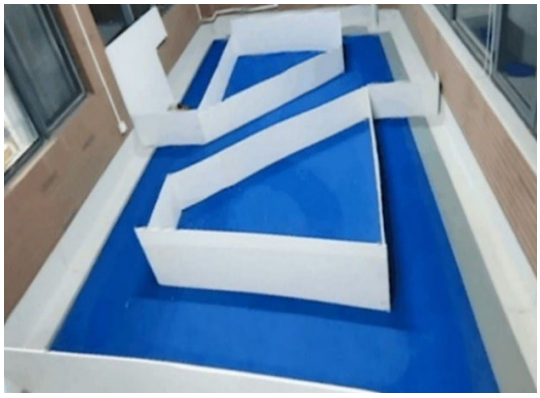


图 4-14 自主导航实验环境

图 4-15 自主避障实验环境

完成实验环境以及智能小车的搭建后，接下来对智能小车在实验环境中进行自主导航实验。

4.3.2 根据实验环境创建地图

将智能小车的上位机与 PC 端建立 SSH 连接，以便于建图过程中使用 PC 端键盘控制小车移动。启动底盘控制节点，以及建图相关的 launch 文件。在 PC 端启动 Rviz 三维可视化界面，用于实时观看建图效果。启动键盘控制节点，控制小车沿着实验环境移动，随着小车的移动，小车顶部的激光雷达实时的对小车周围的环境信息进行采集。图 4-16 为实际实验环境。根据实际环境创建的格栅地图如图 4-17 所示。通过观察智能小车在搭建的实验环境中建图效果可以看出，创建的格栅地图与真实环境较为一致，并且与前面小结中使用 Gmapping 算法进行仿真的效果一致，验证所搭建小车使用 Gmapping 算法在室内建图的可行性。

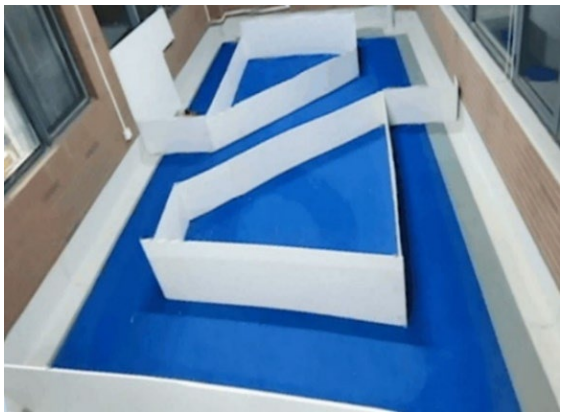


图 4-16 实验的真实化境

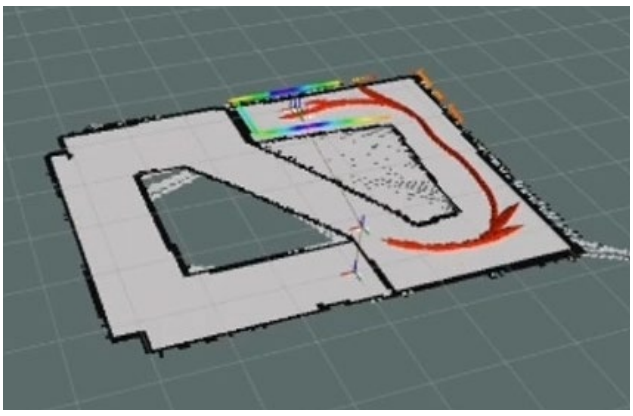


图 4-17 根据实际环境创建的地图

4.3.3 路径规划以及自主导航

通过前面的 SLAM 算法已经对实验环境创建了格栅地图，使用匹配算法，将激光雷达所感受到的自身周围的环境信息与地图特征相匹配，从而确定小车在地图中的位置，启动 rviz 三维可视化软件，就可以看到小车在所创建地图中的位置。在 rviz 界面上选中 2D Nav Goal，然后在显示的栅格地图界面选中小车移动的目标点，那么 rviz 将这一目标点在地图中的所在位置发送给路径规划算法的输入节点，结合地图中的障碍物信息，通过算法运算，将自动规划一条通往目标点的路径，如图 4-18 中绿色曲线，我们将其称之为全局路径规划。

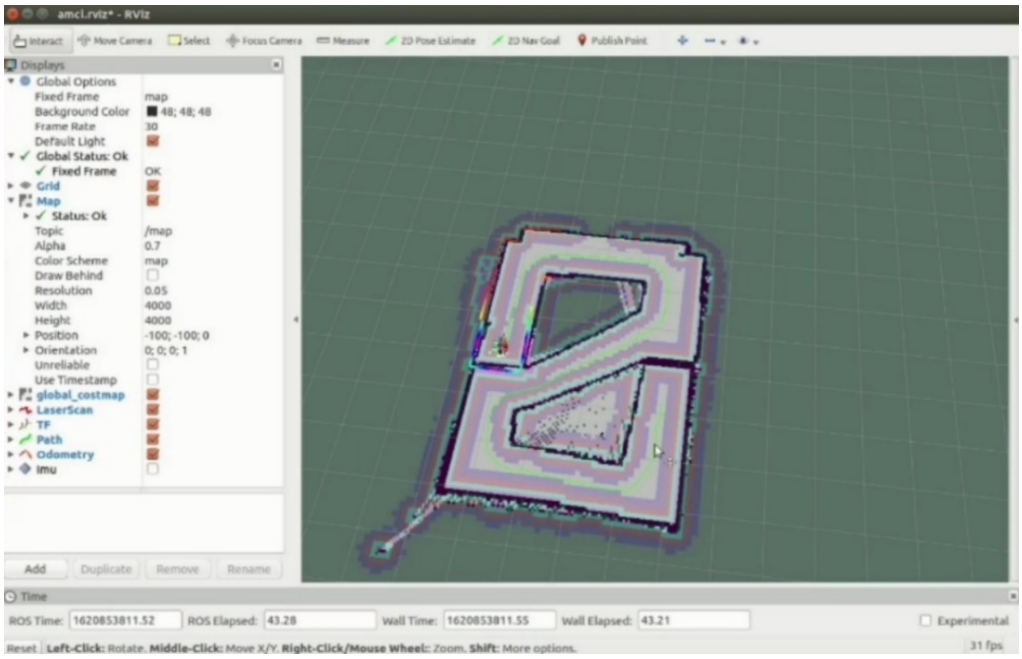


图 4-18 路径规划

另外，在小车行进的过程，如有障碍物未在静态地图中标记，如继续按照全局路径规划的路径行进，则有撞击障碍物的危险，因此需要在小车行进的过程中根据小车周围的环境信息重新规划行进路径来躲避障碍物，我们将其称之为局部路径规划，如图 4-19 中的小车附近较短的红色曲线。

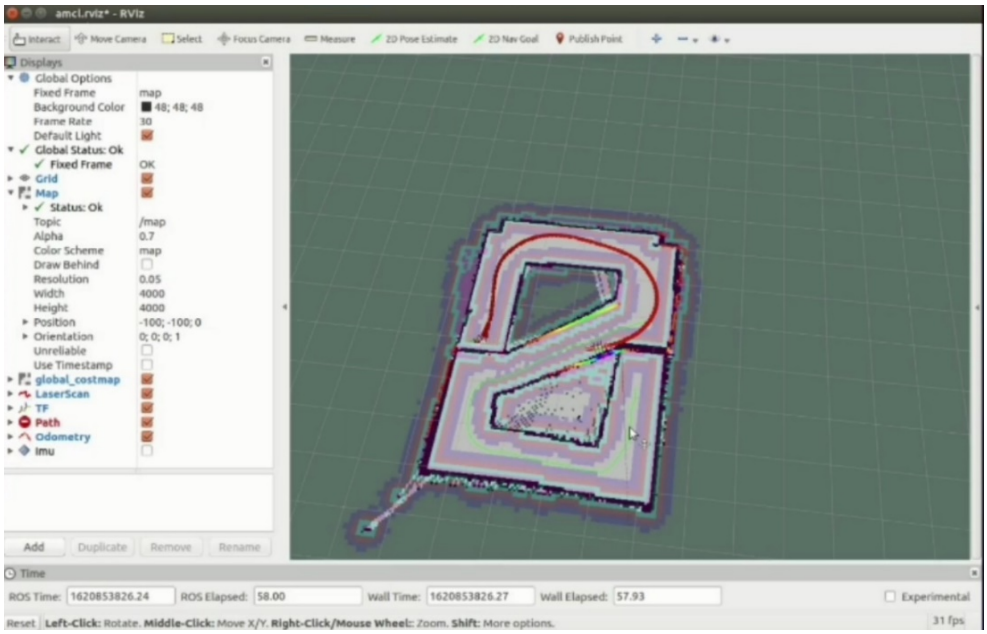


图 4-19 沿着规划路径行进

通过实车在所搭建的环境试验，发现小车可以按照所规划的全局路径到达目标点，从图可以看出行驶的轨迹较为光滑，因此小车在沿着轨迹行进的过程也较为平稳。验证了路径规划算法在室内自主导航的可行性。图 4-20 为智能车在搭建的实际环境下自主导航以及避障的演示视频生成的二维码，如需观看，请使用手机扫码即可。

图 4-20 智能车导航与避障演示视频二维码

4.4 本章小结

本章节首先在 ROS 系统中使用 Gazebo 仿真平台建立了阿克曼小车和环境模型，并进行了 A*算法全局路径规划和 DWA 局部路径规划的自主导航仿真实验。随后，在真实环境中进行了自主导航研究实验与分析。实验结果表明，所设计的阿克曼小车能够结合自身控制系统，在全局路径规划和实时避障的基础上实现自主导航任务的成功完成。

第五章　总结与展望

5.1 总结

人工智能技术的发展推动了智能车功能的不断扩展和多样化，同时也拓展了其应用领域。从火星车、工业制造、国防安全到家庭智能服务、智能物流等不同领域，智能车的应用场景越来越广泛。本文设计的智能车采用阿克曼转向的底盘结构，ROS 系统的软件设计能够缩短开发周期、方便后续的调试维护。同时，本智能车还开发了同时定位与建图系统和自主导航系统，能够广泛应用于餐厅、工厂、仓库等室内场景，提高工作效率，降低人工成本，具有实用价值。

（1）对于智能车的软件设计部分，充分利用了 ROS 系统的特点，完成了基于 ROS 系统的智能车软件总体架构设计。

（2）对于智能车的 SLAM 与路径规划的设计部分，首先总体介绍了 SLAM 算法。结合智能车的应用场景，采用了应用较为广泛的 Gmapping SLAM 算法设计了智能车同步定位建图系统，其次分别介绍全局路径规划算法和局部路径规划算法，并对全局路径规划算法中的 Dijkstra 算法和 A*算法进行了比较，最终选择了较优的 A*算法，而局部路径规划中使用了应用较为成熟的 DWA 算法。

（3）在完成软件系统设计后，首先在 ROS 系统中利用 Gazebo 仿真平台搭建了阿克曼小车和环境模型，并进行了 A*算法全局路径规划和 DWA 局部路径规划的自主导航仿真实验。随后，搭建了实验环境，并进行了智能车地图构建实验和自主导航实验。实验结果表明，本文所设计的智能车具备在室内陌生环境中建立地图、进行导航定位和避障的功能。

5.2 展望

本文介绍了一款基于 ROS 机器人操作系统平台开发的室内智能小车，该小车具备自主导航功能，并利用激光雷达的 SLAM 技术和路径规划算法来在室内无 GPS 信号的未知环境中进行导航。尽管本文在这个领域取得了一些成就，但还存在一些改进的空间。未来的工作包括：

（1）融合传感器可以改善激光雷达的缺点，提高实时性。

（2）进一步研究和改进粒子滤波器可以提高机器人 SLAM 系统的实时性。

（3）完善系统程序的编写和调试可以在满足应用需求的同时提高运行效率。

通过融合传感器、改进粒子滤波器、优化系统程序编写和调试等方法，可以进一步提高机器人 SLAM 系统的实时性，以满足实际应用需求。

参考文献

[1] 刘文之，危双丰. 基于激光雷达的 SLAM 和路径规划算法研究与实现[D].哈尔滨工业大学,2018.

[2] Grisetti G, Stachniss C, Burgard W. Improved techniques for grid mapping with rao-blackwellized particle filters[J]. IEEE transactions on Robotics, 2007, 23(1): 34-46.

[3] 李猛钢. 煤矿救援机器人导航系统研究[D]. 徐州: 中国矿业大学, 2017.

[4] H. He, Y. Jia and L. Sun. Simultaneous Location and Map Construction Based on RBPF-SLAM Algorithm[C]. 2018 Chinese Control And Decision Conference (CCDC). Shenyang, 2018: 4907-4910.

[5] 李昀泽. 基于激光雷达的室内机器人 SLAM 研究[D]. 华南理工大学, 2016.

[6] J. Danping, D. Guangxue, W. Nan, et al. Simultaneous Localization and Mapping based on Lidar[C]. 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019:5528-5532.

[7] X. Zhao, Z.Wang, C. Huang, et al. Mobile robot path planning based on an improved A* algorithm[J]. ROBOT, 2018,40(06): 903-910, 2018.

[8] 吴鹏, 桑成军, 陆忠华,等.基于改进 A*算法的移动机器人路径规划研究[J].计算机工 程与应用, 2019, 55(21):227-233.

[9] R. Fareh, M. Baziyad, T. Rabie,et al.Enhancing Path Quality of Real-Time Path Planning Algorithms for Mobile Robots: A Sequential Linear Paths Approach[J]. in IEEE Access, 2020,8: 167090-167104.

[10] 胡春旭. ROS 机器人开发实践[M]. 机械工业出版社, 2018.

[11] 王珊珊.轮式移动机器人控制系统设计[D].南京：南京理工大学，2013.

[12] 李昌杰.智能移动机器人控制系统设计研究[D].陕西：长安大学，2012.

[13] 徐曙.基于 SLAM 的移动机器人导航系统研究[D].武汉：华中科技大学，2014.

[14] 程姜荣. 基于 ROS 系统的智能车设计[D].上海工程技术大学,2020.

[15] 孙炜,吕云峰, 唐宏伟等.基于一种改进 A*算法的移动机器人路径规划[J].湖南大学学报(自然版),2017,44(4):94-101.

[16] 刘毅.移动机器人路径规划中的仿真研究[J].计算机仿真，2011，28(06)：227-230.

[17] 孙玉梁.移动机器人室内即时地图构建与自主导航[D].大连理工大学,2011.

[18] 齐春辉.基于 ROS 的室内移动机器人导航技术研究[D].河北工业大学,2017.

[19] 吴欣. ROS 下移动机器人激光雷达地图构建与路径规划研究[D].西安理工大学,2021.

[20] Thale S P, Prabhu M M, Thakur P V, et al. ROS based SLAM implementation for Autonomous navigation using Turtlebot[C]//ITM Web of conferences. EDP Sciences, 2020, 32: 01011.

[21] 张杰.移动机器人路径规划研究[D].上海交通大学,2014.

[22] 顾幸方,陈晋音.移动机器人未知环境避障研究[J].传感器与微系统,2011,30(05):16-20.

附录

1.Gmapping 算法参数配置

```
<launch>
  <param name="use_sim_time"   value="true"/>
    <node pkg="gmapping"   type="slam_gmapping"   name="slam_gmapping"   output="screen">
      <remap from="scan" to="scan"/>
      <param name="base_frame"   value="base_footprint"/>                    <!--底盘坐标系-->
      <param name="odom_frame"   value="odom"/>                              <!--里程计坐标系-->
      <param name="map_update_interval" value="5.0"/>
      <param name="maxUrange" value="16.0"/>
      <param name="sigma" value="0.05"/>
      <param name="kernelSize" value="1"/>
      <param name="lstep" value="0.05"/>
      <param name="astep" value="0.05"/>
      <param name="iterations" value="5"/>
      <param name="lsigma" value="0.075"/>
      <param name="ogain" value="3.0"/>
      <param name="lskip" value="0"/>
      <param name="srr" value="0.1"/>
      <param name="srt" value="0.2"/>
      <param name="str" value="0.1"/>
      <param name="stt" value="0.2"/>
      <param name="linearUpdate" value="1.0"/>
      <param name="angularUpdate" value="0.5"/>
      <param name="temporalUpdate" value="3.0"/>
      <param name="resampleThreshold" value="0.5"/>
      <param name="particles" value="30"/>
      <param name="xmin" value="-50.0"/>
      <param name="ymin" value="-50.0"/>
      <param name="xmax" value="50.0"/>
      <param name="ymax" value="50.0"/>
      <param name="delta" value="0.05"/>
      <param name="llsamplerange" value="0.01"/>
      <param name="llsamplestep" value="0.01"/>
      <param name="lasamplerange" value="0.005"/>
      <param name="lasamplestep" value="0.005"/>
    </node>

   <!--启动 rviz（三维可视化界面）-->
    <node pkg="rviz" type="rviz" name="rviz" />

</launch>
```

2.在 Gazebo 中加载仿真环境与智能车模型的 launch 文件

```
<?xml version="1.0"?>
<launch>
  <!-- 设置 launch 文件的参数 -->
    <arg name="paused"   default="false"/>
    <arg name="use_sim_time"   default="true"/>
    <arg name="gui"   default="true"/>
    <arg name="headless"   default="false"/>
    <arg name="debug"   default="false"/>
  <!--模型车的位置不能修改-->
     <arg name="x_pos"   default="-4.15"/>
     <arg name="y_pos"   default="-1.16"/>
     <arg name="z_pos"   default="0"/>
  <!--运行 gazebo 仿真环境-->
     <include file="$(find gazebo_ros)/launch/empty_world.launch">
          <arg name="debug"   value="$(arg debug)" />
          <arg name="gui"   value="$(arg gui)" />
          <arg name="paused"   value="$(arg paused)"/>
          <arg name="use_sim_time"   value="$(arg use_sim_time)"/>
          <arg name="headless"   value="$(arg headless)"/>
          <arg name="world_name"   value="$(find racecar_description)/world/lab3.world"/>
     </include>
  <!-- 加载机器人模型描述参数 -->
      <param  name="robot_description"  command="$(find xacro)/xacro  --inorder '$(find
      racecar_description)/urdf/racecar.urdf.xacro' "/>
  <!--运行 joint_state_publisher 节点，发布机器人关节状态-->
     <node  name = "robot_state_publisher"  pkg = "robot_state_publisher"  type = "robot_state_publisher">
     <param  name="publish_frequency"  type="double"  value="20.0" />
     <param  name="use_tf_static "  type="bool"  value="false" />
     <remap  from="/joint_states"  to="/racecar/joint_states"/>
     </node>
     <node    pkg="joint_state_publisher"    type="joint_state_publisher"    name="joint_state_publisher"
output="screen" />
  <!-- 在 gazebo 中加载机器人模型-->
```

```
    <node  name="urdf_spawner"  pkg="gazebo_ros"  type="spawn_model"  respawn="false"
    output="screen"  args="-urdf -model mycar -param robot_description -x $(arg x_pos) -y $(arg
    y_pos) -z $(arg z_pos)"/>
  <!-- 加载控制器 -->
    <node  name="controller_spawner"  pkg="controller_manager"  type="spawner"  respawn="false"
output="screen" ns="/racecar"
      args="joint_state_controller
      left_rear_wheel_velocity_controller
      right_rear_wheel_velocity_controller
      left_front_wheel_velocity_controller
      right_front_wheel_velocity_controller
      left_steering_hinge_position_controller
      right_steering_hinge_position_controller  ">
    </node>
  <!-- controller param yaml 控制参数配置 -->
    <rosparam file="$(find racecar_description)/config/racecar_control.yaml" command="load"/>
  <!-- ZJ controller command node 开启控制节点 -->
    <node name="ZJ_controller_cmd" pkg="zj_pkg" type="ZJ_controller_cmd.py" output="screen">
    </node>
  <!-- rf2o 激光雷达里程计-->
    <include file="$(find rf2o_laser_odometry)/launch/rf2o_laser_odometry.launch">
    </include>
  <!-- ekf IMU 与激光雷达数据融合 -->
    <include file="$(find robot_pose_ekf)/launch/robot_pose_ekf.launch">
    </include>

</launch>
```

3.智能车自主导航的 launch 文件

```
<?xml version="1.0"?>
<launch>
  <!-- 启动仿真环境和小车模型 -->
    <include file="$(find racecar_description)/launch/racecar.launch"></include>
  <!-- 启动 rviz 三维可视化软件 -->
    <node name="rviz" pkg="rviz" type="rviz"
          args="-d $(find racecar_description)/rviz/racecar_map.rviz" required="true">
```

```
    </node>
  <!-- 启动地图服务节点，加载已创建的格栅地图 -->
    <node name="map_server" pkg="map_server" type="map_server"
          args="$(find racecar_description)/map/nava4.yaml" >
    </node>
  <!-- 配置 amcl 重定位参数 -->
    <include file="$(find racecar_description)/launch/includes/amcl.launch.xml"/>
  <!-- 配置导航参数 -->
    <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <!-- 配置代价地图相关参数 -->
      <rosparam file="$(find teb_local_planner_tutorials)/cfg/carlike/costmap_common_params.yaml"
command="load" ns="global_costmap" />
      <rosparam file="$(find teb_local_planner_tutorials)/cfg/carlike/costmap_common_params.yaml"
command="load" ns="local_costmap" />
      <rosparam file="$(find teb_local_planner_tutorials)/cfg/carlike/local_costmap_params.yaml"
command="load" />
      <rosparam file="$(find teb_local_planner_tutorials)/cfg/carlike/global_costmap_params.yaml"
command="load" />
      <rosparam file="$(find teb_local_planner_tutorials)/cfg/carlike/teb_local_planner_params.yaml"
command="load" />
    <!-- 配置路径规划相关参数 -->
      <!-- 配置全局路径规划 -->
        <param name="base_global_planner" value="global_planner/GlobalPlanner" />
        <param name="planner_frequency" value="10.0" />
        <param name="planner_patience" value="5.0" />
      <!-- 配置局部路径规划 -->
        <param name="base_local_planner" value="teb_local_planner/TebLocalPlannerROS" />
        <param name="controller_frequency" value="10.0" />
        <param name="controller_patience" value="15.0" />
        <param name="clearing_rotation_allowed" value="false" />
    </node>
  <!-- 配置导航文件 -->
    <node name="ZJ_play"  pkg="zj_pkg"  type="ZJ_play.py"/>

</launch>
```

致　谢

时光荏苒，四年的本科生涯即将画上句号，大学期间忙碌而又充实的生活让我收获很多，同时也成长了很多。我在此向那些在本科论文的研究和撰写过程中给予我支持和帮助的人们表示衷心的感谢。

首先，我要感谢我的指导教师吕冬慧老师。感谢您在整个研究过程中对我的悉心指导和专业知识的分享。您的经验和建议对我的论文起到了重要的推动和指引作用。

此外，我要感谢学校和学院提供的实验设备和资源，为我提供了良好的研究环境。感谢吕冬慧老师对我本科期间在本课题研究上的支持和鼓励。

我还要感谢我的同学和朋友们，特别是那些在研究中给予我帮助和支持的同学。你们的讨论和建议对我的研究起到了积极的促进作用。

最后，我要感谢我的家人和亲人们。感谢你们对我一直以来的支持和鼓励，让我有信心克服困难并完成这篇论文。

再次表达我对以上人们的真诚感谢。你们的支持和帮助是我能够完成这项研究的重要动力。

主要研究成果

获奖情况：

2020 年中国智能机器人大赛无人驾驶智能车项目一等奖

第十六届“挑战杯”中国银行天津市大学生课外学术科技作品竞赛一等奖