



燕山大学
YANSHAN UNIVERSITY

硕士学位论文

MASTER'S DISSERTATION

论文题目 基于 Cartographer 和 RBPF 的室内 SLAM 技
术研究

作者姓名 姬少英

学科专业 测试计量技术及仪器

指导教师 金梅 副教授

2018 年 5 月

中图分类号：TP242.6

学校代码：10216

UDC：621

密级：公开

工学硕士学位论文

基于 Cartographer 和 RBPF 的室内 SLAM 技术研究

硕 士 研 究 生：姬少英

导 师：金梅 副教授

申 请 学 位：工学硕士

学 科 专 业：测试计量技术及仪器

所 属 学 院：电气工程学院

答 辩 日 期：2018 年 5 月

授 予 学 位 单 位：燕山大学

A Dissertation in Measurement technology and instruments

RESEARCH ON INDOOR SLAM TECHNOLOGY
BASED ON CARTOGRAPHER AND RBPF

by Shaoying Ji

Supervisor: Associate Professor Mei Jin

Yanshan University

May, 2018

燕山大学硕士学位论文原创性声明

本人郑重声明：此处所提交的硕士学位论文《基于 Cartographer 和 RBPF 的室内 SLAM 技术研究》，是本人在导师指导下，在燕山大学攻读硕士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。本声明的法律结果将完全由本人承担。

作者签字： 日期： 年 月 日

燕山大学硕士学位论文使用授权书

《基于 Cartographer 和 RBPF 的室内 SLAM 技术研究》系本人在燕山大学攻读硕士学位期间在导师指导下完成的硕士学位论文。本论文的研究成果归燕山大学所有，本论文的研究内容不得以其它单位的名义发表。本人完全了解燕山大学关于保存、使用学位论文的规定，同意学校保留并向有关部门送交论文的复印件和电子版本，允许论文被查阅和借阅。本人授权燕山大学，可以采用影印、缩印或其它复制手段保存论文，可以公布论文的全部或部分内容。

保密☐，在 年解密后适用本授权书。

本学位论文属于

不保密☐。

(请在以上相应方框内打“√”)

作者签名： 日期： 年 月 日

导师签名： 日期： 年 月 日

摘 要

智能机器人技术属于人工智能的一个分支，它融合了数学、自动化、机器机械结构等传统理论，它的发展也必将伴随着视觉、深度学习等人工智能的前沿学科实现共同进步。实现移动机器人运动的“自主”是服务机器人领域的发展方向，而即时定位与地图构建(SLAM)技术是机器人“自主”的前提，但是目前的 SLAM 算法面临着所建地图精度低、实时性差、各种算法应用场景混乱等问题，造成 SLAM 并未实现真正的产品化。针对现有问题，本文主要研究了 SLAM 算法建图定位的精度以及 SLAM 算法应用场景分类的问题。

本文首先在研究 RBPF 和 Cartographer 算法原理的基础上，并对 RBPF 算法的实时性进行了相应的改进。对 RBPF 算法、改进的 RBPF 算法和 Cartographer 算法所建地图的精度进行了仿真分析，得出了改进后的 RBPF 算法的实时性与精度均优于标准的 RBPF 算法的结论；仿真结果还证明了改进的 RBPF 算法在建图速度上优于当前最流行的 Cartographer 算法，且在小型场景中的建图精度与 Cartographer 算法相差极小的结论。为进一步验证改进的 RBPF 算法的定位精度，在现实环境中设计实验比较改进的 RBPF 算法与 Cartographer 算法的定位精度。

其次，针对实验中所使用的移动机器人，研究了机器人的差速运动模型以及两种计算轨迹的方法：动态窗口法和两轮差速法。实验表明，在实验中所使用的移动机器人行驶距离为 20 米的条件下，使用动态窗口法和两轮差速法计算机器人轨迹时，动态窗口法的累计误差略小的结论，选择使用动态窗口法计算机器人的里程信息。

最后，针对 Cartographer 和改进的 RBPF 两种算法的定位精度，搭建了基于 ROS 的 SLAM 算法的移动机器人实验平台，实现了移动机器人的自主运动、建图定位以及实时监控等功能。实验分别采集了两种算法在短路径、长路径和复杂环境下的定位偏差，结果表明 Cartographer 算法在三种环境下的定位精度均具有一定优势，结合改进的 RBPF 算法在建图速度上的优势，提出了在建图时采用改进的 RBPF 算法，定位时采用 Cartographer 算法的建议。

关键词：SLAM；Cartographer 算法；RBPF 算法；移动机器人；ROS

Abstract

As one of the branch of artificial intelligence, intelligent robot technology integrates traditional theories such as mathematics, automation, machine mechanics, etc. Its achievement will be along with the frontier disciplines of artificial intelligence such as vision and deep learning. The realization of the "autonomy" of the mobile robot movement is the shape of things to come, and mobile robot simultaneous localization and mapping (SLAM) is the premise of the robot's "autonomy". However, the current SLAM technology faces a series of problems such as low accuracy of the constructed map, poor real-time performance, and confusion of application scenarios of various algorithms, resulting in SLAM not realizing productization. In view of the existing problems, the accuracy of SLAM algorithm and the application scene of SLAM algorithm are studied in this paper.

Firstly, the principles of RBPF and Cartographer algorithm are studied in this paper, and the real-time performance of RBPF algorithm is improved accordingly. The accuracy of the maps constructed by the RBPF algorithm, the improved RBPF algorithm and the Cartographer algorithm is simulated and analyzed. It is concluded that the real-time performance and accuracy of the map created by the improved RBPF algorithm is better than that of the standard RBPF algorithm. The simulation results also prove that the improved RBPF algorithm is superior to the current most popular Cartographer algorithm in the speed of building maps, and the difference of the accuracy in the small scene between the improved RBPF algorithm and cartographer algorithm is very small. In order to further verify the positioning accuracy of the improved RBPF algorithm, the experimental of the positioning accuracy of the improved RBPF algorithm and the Cartographer algorithm is designed in the real enviroment.

Secondly, for the mobile robot used in the experiment, the differential motion model of the robot and two methods for calculating trajectory are stuided: dynamic window method and two-wheel differential method. Experiments show that when the distance traveled by the mobile robot used in the experiment is 20 meters and dynamic window

method and two-wheel differential method are used to calculate the robot trajectory, the cumulative error of the dynamic window method is slightly smaller. the dynamic window method is used to calculate the robot's mileage information.

Finally, aiming at positioning accuracy of the Cartographer algorithm and the improved RBPF algorithm, a mobile robot experimental platform based on ROS and SLAM algorithm is set up. The functions of autonomous movement, map building and positioning, the real-time monitoring of mobile robot are realized. In the experiments, the deviations of the map positioning using the two algorithms in short-path, long-path and complex environments are collected respectively. The results show that the improved RBPF algorithm has the same accuracy as the Cartographer algorithm in both short-path and long-path environments. The Cartographer algorithm has an advantage in a complex environment. Combined with the advantage of the improved RBPF algorithm in the speed of map building, an improved RBPF algorithm is proposed for building maps, the Cartographer algorithm is recommended for positioning in a complex environment, and the improved RBPF algorithm and the Cartographer algorithm in all can be used in a small and single-structured environment.

Keywords: SLAM; Cartographer; RBPF ; Mobile robot; ROS

目 录

摘 要.....	I
Abstract.....	II
目 录.....	V
第 1 章 绪 论.....	1
1.1 课题研究的背景、目的和意义.....	1
1.2 机器人 SLAM 算法的国内外发展现状.....	3
1.2.1 移动机器人的国内外发展现状.....	3
1.2.2 SLAM 算法研究现状.....	6
1.3 本文主要研究内容.....	9
第 2 章 机器人同时定位与建图算法研究.....	12
2.1 SLAM 方法简述.....	12
2.2 RBPF 算法与地图构建.....	12
2.2.1 基于贝叶斯的 SLAM 问题描述.....	13
2.2.2 Rao-Blackwellized 变换.....	14
2.2.3 改进的 RBPF 算法.....	16
2.3 Cartographer 算法与地图创建.....	18
2.3.1 Cartographer 方法简述.....	18
2.3.2 Cartographer 算法.....	18
2.4 仿真结果分析.....	22
2.5 本章小结.....	24
第 3 章 地面移动机器人运动模型分析.....	26
3.1 差分轮里程计算原理.....	26
3.2 差速驱动机器人运动模型.....	26
3.3 动态窗口法计算运动轨迹.....	27
3.4 左右轮差速计算运动轨迹.....	29
3.5 里程误差分析.....	30
3.6 本章小结.....	34
第 4 章 基于 ROS 的移动机器人 SLAM 系统实现.....	35
4.1 ROS 系统概念.....	35
4.2 移动机器人 SLAM 系统实现.....	35
4.2.1 系统构建及传感器选择.....	35
4.2.2 移动机器人 SLAM 系统中的数据传输.....	38

4.3 ROS 分布式系统部署	40
4.4 本章小结	42
第 5 章 实验结果与分析	43
5.1 SLAM 建图实验环境选取	43
5.2 数据关联过程	43
5.3 地图定位偏差的实验及分析	48
5.4 本章小结	58
结 论	59
参考文献	61
攻读硕士学位期间承担的科研任务与主要成果	66
致 谢	67

第1章 绪论

1.1 课题研究的背景、目的和意义

现代社会中，人们对于许多危险、复杂或者枯燥的工作已经产生了厌烦，期望能找到一种工具来替代人们完成这些任务，因此机器人的概念及技术应运而生^[1]。移动机器人是指在地面上使用“轮子”来活动的机器人，它既不同于传统的固定式机器人，也不像人型机器人拥有复杂的运动学原理，在机器人中备受青睐，是实现智能工厂、智能生产和智能制造^[2]的关键环节。移动机器人集合了环境感知、行为控制、动态决策规划等功能^[3]，因其灵活的运动机制与对环境良好的适应性，在许多领域都得到了应用：例如家庭式的扫地机器人，工厂物流中搬运物资的自动搬运机器人，科学考察中探测各种信号的探测机器人，教育教学中应用的教学机器人，军用领域中的武装机器人和侦察机器人等^[4-8]。

移动机器人是在机械工程、嵌入式、传感器、人工智能与计算机以及自动控制等学科的基础上蓬勃发展起来的，它属于一种跨学科的研究成果^[9,10]。移动机器人在各行各业都有其需求与应用前景，它的分类方式也多种多样，目前主流的分类方式有以下几种：(1) 根据移动机器人的移动方式可分为：轮式、履带式、足式等移动机器人和爬行类机器人等；(2) 根据机器人的工作环境可分为室内移动机器人和室外移动机器人；(3) 根据控制结构可分为：水平结构机器人与垂直结构机器人以及混合结构机器人；(4) 根据移动机器人的功能可分为^[11,12]：救援机器人、清洁机器人等。

当今社会正是机器人技术加速发展的时期，面临着一系列的机遇与挑战。机器人的快速发展与逐步趋于复杂化，人们已经不满足于重复“造轮子”这种低效率的工作，转而寻求一种通用于所有机器人的方式与平台，可以实现代码的复用与模块化的愿望越来越强烈，在这种情况下，开源机器人操作系统(Robot Operating System, ROS)应运而生^[13]，现今整个机器人领域都已展开了学习与使用 ROS 的风潮。ROS 系统是由 Willow Garage 公司开发的一款开源机器人操作系统，它在代码复用思想的基础上，使用节点对各种功能模块进行分割，开发者无需改动代码或只需要做极小的改动，关联对应节点的信息，就可以实现相应的功能，有效避免了开发者重复的书写代码工作。ROS 还利用很多现有的著名开源项目代码^[14,15]，例如 Player 项目中的驱动、运动控制及仿真；Open CV 中的视觉算法；Open RAVE 的规划算法等。除

此之外,ROS 可以不断的进行社区的维护升级,极为适合使用者在平台上开发应用。

导航是移动机器人实现“自主”和“智能”的关键技术,是移动机器人最基本的功能之一,不同于传统的固定式机器人,移动机器人在复杂环境下具有自我规划、自动适应能力^[16]。实现移动机器人的自主导航需要解决三个基本问题:我在哪里?我要去哪里?如何去?即:定位、目标识别和避障路径规划问题^[17]。导航的目标是在无人干涉的情况下,移动机器人可以通过自身携带的传感器求出与环境的相对位姿,根据已知地图信息进行到达目标位置的全局规划,移动机器人在行驶时,同时感知周围环境信息,与已知地图信息做对比,实现自我定位,并进行局部路径规划,绕开离散的障碍物,使移动机器人安全行驶到目标位置。

实现导航功能的前提之一就是创建周围环境地图。SLAM(Simultaneous Localization and Mapping, 即时定位与地图创建)^[18]是指机器人在未知环境中创建地图时对自身位置不确定,因此机器人需要在创建地图的同时,利用所建地图对自身所处的位置进行定位,进而在已知周围环境地图和自身位置的情况下实现自主导航运动。移动机器人只有在知道自身位置和周围环境时才能“自由”地活动,因此 SLAM 是其实现“真正自主”运动的前提,在机器人导航、控制、避障和路线规划等方面都有着重要的研究意义。SLAM 可以描述为一个“鸡生蛋,蛋生鸡”的问题^[19,20]:机器人实现精确定位的前提是其已知周围的环境地图,在此基础上机器人才能利用周围与环境地图中的相似场景来逐渐校正自己的位置,而获取周围环境地图的前提则是机器人需要知道自己此时所处位置。

移动机器人的机构灵活、稳定,能适应各种恶劣环境,在各行各业都有应用需求,而创建未知环境地图是实现机器人导航功能的前提^[21]。本文在 ROS 系统的基础上,研究移动机器人的 SLAM 技术。移动机器人平台执行人类发布的控制指令、进行环境感知以及运行 SLAM 算法程序,并将信息通过无线网络反馈给笔记本电脑,笔记本电脑执行的工作为可视化移动机器人感知到的信息以及发布远程任务指令。

本文着重研究了 RBPF 算法和 Cartographer 算法这两种 SLAM 方法,并在 ROS 分布式系统的基础上搭建了移动机器人实验平台,实现了地面移动机器人的自主移动、远程控制以及建图定位等,并对改进的 RBPF 算法和 Cartographer 算法所建地图的精度进行了比较分析,证明了改进的 RBPF 算法和 Cartographer 算法在实际中应用的特性,为实际应用中选用合适的 SLAM 算法提供了实验依据。

1.2 机器人 SLAM 算法的国内外发展现状

1.2.1 移动机器人的国内外发展现状

1920 年，卡雷尔·恰佩克在《罗萨姆的机器人万能公司》一书中首次提出了机器人这一概念。紧接着，在 1939 年，美国的西屋电气公司推出了机器人 Elektro，Elektro 属于家用型，可以进行简单的行走、说出简单的单词，甚至还可以抽烟，受当时技术限制，Elektro 是由电缆控制的。1954 年，美国人乔治德沃尔研制出了可编程的机器人，极大地提高了机器人的通用性与灵活性。1959 年，“工业机器人之父”德沃尔与约瑟夫·英格伯格联手推出了世界上第一台工业机器人，开启了机器人在工业领域的应用^[22-25]。

20 世纪 60 年代早期，人们开始在机器人上安装各种传感器，目的是提高机器人的可操作性。托莫维奇和博尼在“灵巧手”上安装了压力传感器，来获得外界力的感觉，进而完成各种工作。麦卡锡在机器人中引入了摄像头，与 MIT 一起制造出了世界上第一台拥有“视觉”的机器人，它实现了对积木识别与定位^[26]。1965 年，约翰·霍普金斯大学研制出了机器人 Beast，它通过声纳和光电管做到识别周围环境，进而对自身所处位置进行校正^[27]。

从 20 世纪 60 年代中期开始，人们逐渐认识到了带有传感器的机器人在实现机器人识别定位领域的优势，因此各国纷纷开始成立机器人实验室，开始研究带传感器、对外界有“感觉”的机器人，其中以美国的斯坦福大学和麻省理工学院以及的英国爱丁堡大学为代表。自此，机器人领域开始朝着人工智能方向发展^[28]。

20 世纪 60 年代末，美国的斯坦福研究所研制了首台具有人工智能的移动机器人 Shakey，它能够实现对环境的感知，并且对环境建模，以此实现自身行为的规划，完成指定任务^[29]。Shakey 装备有测距仪、摄像机和碰撞传感器，它可以在人的指挥下发现积木并进行抓取。Shakey 的控制中心是两台计算机，但限于当时的计算机技术，它们的运算速度较慢且拥有庞大的体积。1970 年，前苏联研制出世界上第一辆实现“无人驾驶”，可自主移动的月球车，用于考察月球表面环境^[30]。同一时代，美国喷气推进实验室也研制出了月球车并用于行星探测^[31]，月球车上装备有摄像机、激光传感器和触觉传感器。

20 世纪 90 年代，移动机器人技术在经过了 30 多年的发展之后得到了很大提高。美国国家航空和宇宙航行局(National Aeronautics and Space Administration, NASA)在

1997 年向火星发射了机器人 Sojourner rover, Sojourner rover 成功着陆后, 在火星上总共行驶了 90 多米, 拍摄了 500 多幅照片, 帮助科研人员实现了对火星表面的岩石成分的分析。Sojourner rover 的成功应用开启了移动机器人技术的一个新的发展方向, 表明机器人可以代替人类完成一些危险、枯燥以及肮脏的工作, 展现了机器人在需求上的无限潜力^[32]。

20 世纪 90 年代末期, 世界各国以及国际机构都加大了对机器人研究的投入, 并产生了一系列成果。

1998 年, 丹麦的乐高公司推出了一种“机器人套件”, 极大地降低了机器人技术的入门、学习及研究的门槛, 让机器人的制造变得像搭积木一样简单而又能任意拼装, 乐高公司后来推出了乐高 NXT 机器人, 促使机器人的初步成果开始走入了个人世界, 使得机器人的研究不再是大型研究机构及国际组织的专利。乐高机器人 (Lego Mindstorms) 是在集合了可编程主机、电动马达、传感器、Lego Technic 部分 (齿轮、轮轴、横梁、插销) 等硬件结构上建造而成, 简洁而实用。最近的版本是 2013 年上市的 Lego Mindstorms EV3^[33]。

1999 年, 位于日本的索尼公司开启了娱乐机器人的市场, 首次推出了犬型机器人爱宝(AIBO), AIBO^[34]当时便获得了人们的极度喜爱, 火爆地销售一空, 这标志着娱乐型机器人开始走进了普通家庭。不仅如此, 爱宝拥有着极其出色的智能技术水平, 并以此被机器人足球世界杯(Robocup)比赛委员会采用, 成为了 Robocup 比赛的专用机器人^[35]。

2002 年, 美国 iRobot 公司研制出了扫地机器人 Roomba^[36], Roomba 不仅能自主进行路径规划, 避开障碍物设计到达目标点的前进路线, 此外, 在电量不足的时候还能自动地驶向充电座进行自主充电操作, 这些特点使得 Roomba 成为了当时最受欢迎的服务机器人之一。到 2018 年, iRobot 公司已经推出第九代的智能机器人 Roomba 980^[37], 它利用 iAdapt 情景规划智能导航与 vSLAM 视觉运算处理技术, 实现连贯、高效地导航, 同时在其地图上创建可视化地标便于在清洁时跟踪位置。

2006 年 6 月, 微软推出了 Microsoft Robotics Studio^[38], 自此开始, 模块化、平台统一化成为了机器人的又一个发展方向。

21 世纪之后, 人们对移动机器人技术要求更高的自主性、适应性以及可交互性。移动机器人的应用领域也主要集中在了以下 3 个领域: (1) 空间探索。比如美国 2011 年向火星发射的好奇号探索机器人, 以及中国在 2013 年及之后向月球发射的嫦娥号

系列的无人探测器，它们都用于对外太空的科研探测；(2) 服务机器人。服务机器人主要用于人类的日常生活中，比如目前应用广泛的扫地机器人和 2016 年国网大量采购的巡检机器人等。(3) 无人驾驶汽车。无人驾驶汽车也属于移动机器人范畴，可完全自主运行和精确定位。2015 年，谷歌公开了其研发的无人驾驶汽车的原型，引发了无人驾驶领域研究的热潮，2018 年百度无人驾驶汽车春晚上亮相。

除了移动型机器人外，仿人机器人的研究也得到了迅速的发展。包括本田公司的机器人 ASIMO 以及阿尔德巴兰公司机器人 Nao，伊朗德黑兰大学 Surena3，以及波士顿动力公司与美军合作开发的 Atlas，它们都能进行双足行走，应用了人体运动学和力学技术，在此基础上，仿人机器人可以做出更复杂的动作，进而可以提高完成的困难任务的概率^[39-41]。



图 1-1 移动机器人

国内的研究学者在上世纪末期才开始研究移动机器人技术，受到国家的大力支持和资金投入，因此，虽然国内在该领域起步较晚，但许多移动机器人的研究项目取得了较好的成果。

2002 年，清华大学研制出一种多功能的可用于室外的智能移动机器人 THMR.V^[42]。THMR.V 不仅可以实现在校园环境中的低中速全自主行驶、侦察及遥控驾驶，它还可以在实现对高速公路上的车道线的快速视觉检测和完成部分的辅助驾驶等功能。

上海交通大学研制出了 211AMCTB、Frontier-ITM 等^[43-44]。211AMCTB 是一种移动机构的实验平台，采用了可以跨过障碍物的关节轮式结构，应用的目的领域是复杂的室外环境。Frontier-ITM 是一种高性能自主移动机器人，曾多次获得国内外机器人比赛的冠军，并且还作为中国的参赛队首次参加了 Robocop 中型组比赛，另一方面 Frontier-ITM 具有良好的稳定性、开放性和可扩展功能，适用于智能机器人研究和教学中。

除此之外，中科院自动化所研制的 CASIA—I 智能移动机器人集成了多种传感器

感知外界环境，具有视觉、语音识别与会话等多种功能^[47]。沈阳自动化研究所研制的自动输送小车已经开始投入生产中，另外自动化所还开始投入研制“蛇形机器人”和“多功能排险防爆机器人”等^[45]。在移动机器人技术研究领域的这些成就推动了我国移动机器人领域的发展，如今，中国的各行各业都在加紧对机器人技术以及人工智能方面的研究，我国在这些领域的发展水平已经不输于世界的先进水平，甚至在某些方面的成果已经领先于世界，移动机器人技术正在成为我国在人工智能领域发展、创新及应用的一个突出方面。

随着移动机器人技术逐步走向成熟，自主导航作为其基本功能之一，代表着移动机器人的性能好坏，而人们对机器人性能的要求也必然会越来越高，这对于 SLAM 技术既是机遇，也是一种挑战，人们必然会倾尽全力将 SLAM 做到尽善尽美，尽快完成商品化路程，因此对于 SLAM 技术的研究变得必要起来。

1.2.2 SLAM 算法研究现状

作为导航领域的一种基础技术，SLAM 由来已久，上世纪的核潜艇海底定位系统就隐现出了 SLAM 的雏形，到今天，SLAM 已经走进了人们的视野，人们也开始认识到 SLAM 的重要性。特别是几年前小米扫地机器人的出现，更是让人们都认识了 SLAM，而 SLAM 与三维视觉的结合(VSLAM)，又大幅降低了 SLAM 的成本，逐渐展现出无与伦比的发展潜力。

“定位、定向、测速、授时”，是人们研究了千年，到现在还没能得到完美解决的问题。最开始的时候，古人依靠夜观天象或者使用司南来分辨南北，实现简单的定向；到元代时期，才华横溢的中国人发明了牵星术，通过使用牵星板来测量星星并据此实现对纬度的估计，令人叹为观止；1964 年美国开始使用 GPS 进行定位，成为定位史上历史性的一步，其中，军用的 P 码可以达到 1-2 米级精度，大众使用的 CA 码能够实现 5-10 米级的精度；后来，人们为了突破 P 码封锁以及追求更高的定位定姿精度，提出了很多方法提升 GPS 定位的精度，例如 RTK 的实时相位差分技术，可以将定位的精度提升到厘米级别^[46,47]，大致解决了在室外确定位姿的问题；2015 年以来，我国开始布置全球北斗卫星导航系统，预计 2020 年可完成。

但是，对于室内来说，GPS 和北斗定位就无能为力了。为了实现在室内的确定位姿的问题，学者们提出了多种技术及方法，其中，SLAM 技术脱颖而出。解决导航与定位问题的传统的主要方法主要有惯性导航、星光制导、卫星导航和它们的组

合导航技术，而 SLAM 导航定位属于无源导航范畴，可以实现独立设备的自主导航定位。

Smith Self 和 Cheesman^[48,49]最早提出 SLAM 问题，将机器人的定位和构建地图这两个过程结合在了一起。1987 年，Smith Self 和 Cheesman 提出基于卡尔曼滤波的 SLAM 算法。基于卡尔曼滤波器的 SLAM 算法主要是通过卡尔曼滤波器对机器人的位姿信息不断的估计和更新，卡尔曼滤波器有两个输入，一个是来自激光测距的观测数据，另一个是机器人自身轮子携带的里程计数据，卡尔曼滤波器利用里程计估计机器人所走轨迹以及机器人此时的位姿，激光测距采集的观测数据作为机器人的真实位姿加入卡尔曼滤波器的计算中，不断校正机器人对自身位置的估计，使估计量不断接近真实状态。扩展卡尔曼滤波器(EKF)的工作原理与卡尔曼滤波器相同，只不过将卡尔曼滤波器中的状态转换函数和线性观测模型替换成了可微函数，EKF 方法有效估计了地图路标与机器人位姿相关联的后验概率，应用更加广泛，但需要对机器人运动模型和传感器噪声做出合理假设，不当的假设会导致滤波器发散^[50]。相比卡尔曼滤波器及基于其一系列的改进算法，粒子滤波器(Particle Filter, PF)因为不受线性高斯系统的限制，因此适用于任何非线性或非高斯的动态系统，在此基础上，Doucet 等提出基于 Rao-Blackwellized 的粒子滤波器(RBPF)的方法^[51]，其中，粒子滤波器中的每个粒子是一种运动轨迹，拥有各自的地图信息，Griseti 等人改进了粒子的建议分布，并引入自适应重采样机制减少了算法的计算量，提高了该算法的实时性。ROS 系统引入了 Griseti 等人改进后的算法，稍加修改，编写成了一个叫作 Gmapping 的程序包，人们只需要利用激光雷达和 Turtlebot 就可以创建出高精度的二维地图，但该 SLAM 功能包的实时性仍然有待提高。2016 年 5 月，谷歌开源了 Cartographer 2D 和 3D 的软件库，算法融合了多种传感器数据(例如：雷达、惯性传感器和摄像机)来同步计算传感器的位置和传感器周围的环境地图。Cartographer 是一种实时 SLAM 算法，它降低了使用雷达数据计算闭环约束的计算要求，在建立成千上万平方米的大面积地图时，可以同时提供全面实时优化的结果。由于 ROS 集成和支持多源代码，Cartographer 已经支持在多种基于 ROS 的机器人平台上使用^[52]。

SLAM 的流程大致分为前端和后端。前端主要研究相邻时刻之间机器人位姿的变换关系，在机器人相邻时刻测得的观察信息之间匹配特征点，去除噪声，得到传感器的位置和姿态，同时可以加入惯性测量单元进行滤波融合，从而获得更加准确的位姿数据与信息。SLAM 前端所得结果存在累计误差，时间越长，累积误差越大，

而后端存在的目的就是为了解除前端所得匹配结果的累积误差，对地图进行优化，使地图信息更加准确，优化的方法主要有滤波(EKF、UKF、PF)的方法或 TORO、G2O 等优化理论进行优化，得到最佳的估计位姿。因为滤波器具有稳态增长快的特点，而 EKF、PF 又需要频繁地求逆操作，对计算机的压力很大，而基于图的 SLAM 优化，以关键帧为节点，关键帧之间的转换关系(如仿射变换矩阵等)作为节点之间的连接线，据此绘制出一张具有点和边的变换图形，在保证精度的同时，降低了计算量，因此人们已经开始慢慢地从传统的滤波理论走向了图优化理论^[53-55]。

实现 SLAM 算法并应用于实际中需要考虑以下几个方面：

(1) 地图表示问题^[56]。栅格地图、拓补地图、特征地图和位姿地图等都是 SLAM 创建地图的表现形式，地图也分为稀疏型和密集型，分别适应不同的场景和需求。

(2) 信息感知问题。感知环境信息的(超声波、红外线、激光雷达、视觉等)传感器各有其优势与缺陷。例如摄像头感知的环境范围比激光雷达小，但获得的信息更丰富，成本也更低。

(3) 数据关联问题。市场上存在的观测类传感器多种多样，不同传感器获取信息的方式、信息的数据类型以及坐标系的表达方式等方面都不相同，需要通过数据关联及融合来获得较准确的信息。

(4) 定位与构图问题^[57]。这是 SLAM 算法的核心问题，目的是为了实现机器人位姿的估计和创建环境地图。定位与构图首先需要建立合理的物理模型，在此基础上才能实现对位姿的较好的估计，以及对所建地图的后续优化等，这都是数学方面的问题。

除此之外，SLAM 还存在其他问题，如回环检测，探索问题，以及绑架问题等。

目前市场上用于 SLAM 的传感器主要分为两大类^[58]：激光雷达和摄像头。传感器和需求的不同导致 SLAM 分为了激光 SLAM(2D 和 3D)和视觉 SLAM(Sparse、semiDense、Dense)两类，图 1-2 列举了几种常用的雷达品牌和型号以及各种结构的摄像头。激光雷达分为单线激光雷达和多线激光雷达，其中多线激光雷达扫描的范围更大，可扫描到立体的障碍物。各种激光雷达的角分辨率和精度各不相同，Hokuyo、SICK 以及 Slamtech 和北醒光学是做 SLAM 常用的几个品牌。VSLAM(Visual Simultaneous Localization and Mapping)是指使用视觉传感器来实现 SLAM 的技术^[59]，视觉传感器的品种很多，包括单目视觉、单目结构光、双目视觉、双目结构光以及 ToF 等几大类，它们的原理都是获取 RGB 和深度信息。无论是激光 SLAM 还是视觉

Rao-Blackwellized 粒子滤波算法和 Cartographer 算法,详细分析了它们实现的原理与推导过程,对 RBPF 算法进行了实时性和建议分布方面的改进,并对其进行仿真,分析了这两种算法的地图创建的速度与精度;(3) 阐述了移动机器人中常用的差速驱动机器人的计算里程的运动模型和方法,通过实验证明了使用动态窗口法计算机器人轨迹的累积误差略小于两轮差速的方法;(4) 搭建了基于 ROS 系统的移动机器人实现 SLAM 的软件与硬件平台,并选择了实际的应用环境,实现了在实际场景中的移动机器人的远程控制、建图定位以及远程监控等功能;(5) 通过实验证明了改进的 RBPF 算法与 Cartographer 算法在短路径、长路径的环境下的定位偏差几乎相同,以及 Cartographer 算法在复杂环境下的定位精度具有优势的结论,提出了在小型和单一结构的环境中优先选择改进的 RBPF 算法,较大以及复杂结构环境环境中优先选择 Cartographer 算法进行建图定位导航的建议。

以下是本文中各个章节的描述与安排:

第 1 章主要叙述了 SLAM 研究的背景目的和意义,阐述了国内外关于移动机器人的发展历史以及相关研究和其现状,介绍了 SLAM 算法研究的发展现状和 SLAM 算法在实际场景中面临的问题。

第 2 章描述了当前 SLAM 算法的主要组成部分及其概况,研究了基于贝叶斯准则的 SLAM 问题的描述,以及基于贝叶斯 SLAM 问题提出的 RBPF 算法,采用了两种改进 RBPF 算法实时性的方法;简要介绍了 Cartographer 算法的定位方法及过程,研究了 Cartographer 算法的推导原理及算法的主要流程;采用网络上开源的数据集,对 RBPF 和 Cartographer 这两种算法进行仿真,对两种算法的地图误差进行了分析。

第 3 章主要研究了差速驱动的移动机器人的运动模型和里程计算方法,里程计算在 SLAM 中扮演着重要的角色,里程计数据与传感器观测到的信息相互验证才能得出较准确的地图及自身位姿信息。本章主要描述了两轮差速驱动的运动模型估计运动轨迹的方法,并通过实验对比它们的累积误差。

第 4 章首先介绍了 ROS 系统的组成部分和在机器人应用领域的优势,在此基础上采用了合适的硬件设备搭建了移动机器人的硬件平台,研究并实现了基于 ROS 系统的 SLAM 软件结构的构建,介绍了 ROS 系统的跨计算机的分布式系统的部署方法,实现了移动机器人的远程控制、建图定位以及远程监控等功能。

第 5 章研究了移动机器人 SLAM 平台的数据关联方式,选择了实验场景和实验方式,分别在移动机器人上运行改进的 RBPF 算法和 Cartographer 算法,创建了实验

环境的地图，并通过实验对这两种 SLAM 算法所创建的地图进行了定位精度的比较分析，证明了改进的 RBPF 算法和 Cartographer 算法在不同的环境中的特性。

第 2 章 机器人同时定位与建图算法研究

2.1 SLAM 方法简述

如图 2-1 所示，SLAM 技术是一个跨领域的技术，它包含了定位、地图创建和路径规划的部分领域，即 SLAM 是这几个领域交叉融合形成的新的领域，它的主要目的就是在定位的同时完成周围环境地图的创建，进而利用创建的地图完成路径规划及运动控制。

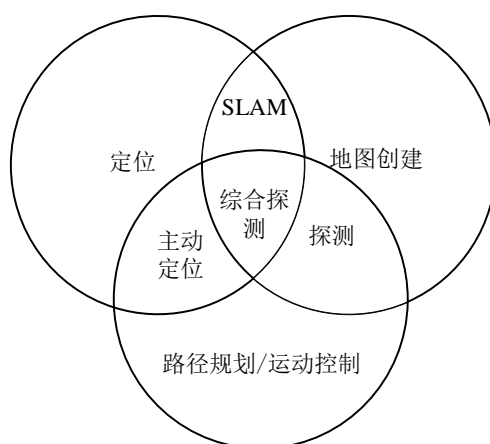


图 2-1 SLAM 的定义

移动机器人 SLAM 主要分为三个部分：创建数学模型、扫描匹配和闭环检测。数学建模属于 SLAM 的基础理论方面，数学模型与实际情况匹配情况决定着系统的性能是否良好，SLAM 的扫描匹配和闭环检测两个部分也依赖于 SLAM 模型的创建方法，目前 SLAM 问题的解决方法主要以概率估计为主，分为基于卡尔曼滤波(KF)的算法和基于粒子滤波(PF)的算法；扫描匹配部分，又称为 SLAM 的前端，包括对传感器数据的预处理和时间帧扫描匹配以及地图的拼接，在此基础上，可以初步完成地图的创建工作；地图的具体形式也是研究点之一，常见的有路标地图、度量地图、拓扑地图。闭环检测，即后端优化，需要机器人对曾到达的场景具有再识别的能力，如果识别成功，那么机器人就会对两次到达的场景的地图进行闭环匹配，这样可以显著地减小机器人 SLAM 前端的累积误差，目前最常用的是基于图(Graph-Based)的优化。

2.2 RBPF 算法与地图构建

2.2.1 基于贝叶斯的 SLAM 问题描述

SLAM 中, x_t 表示在 t 时刻机器人的位姿, 它是一个三维向量 (x, y, z) , 其中 x 和 y 表示机器人在坐标系下的位置, z 则表示机器人自身旋转的角度, 序列 $X_t = \{x_0, x_1, x_2, \dots, x_T\}$ 表示一段时间内的位姿(即轨迹), 只有初始位姿 x_0 是已知量。里程计用于测量两个相邻时刻之间的位置关系, 即 u_t 是 $t-1$ 时刻到 t 时刻的位置变化, 那么序列 $U_t = \{u_1, u_2, u_3, \dots, u_T\}$ 则用来表示机器人相邻时刻之间的轨迹变化。机器人周围的环境地图使用 m 来表示, m 描述了周围障碍物的位置, 通常假设环境地图为静态, 距离传感器测量并建立了 m 与机器人位姿 x_t 之间的关系, 即序列 $Z_t = \{z_1, z_2, z_3, \dots, z_T\}$ 。SLAM 研究的问题就是在已知里程计信息和观测数据的基础上, 计算环境地图 m 和机器人轨迹 X_t 。

因此, SLAM 需要建立 2 个数学关系^[61]: 描述 u_t 与 x_{t-1} 和 x_t 间的数学模型和描述 z_t 与 m 和 x_t 间关系的数学模型。这两个模型可以看成一种概率分布, 其中 $p(x_t | x_{t-1}, u_t)$ 表示机器人在 $t-1$ 时刻的位姿 x_{t-1} 和机器人 $t-1$ 时刻到 t 时刻的里程数据 u_t 已知的情况下, 机器人 t 时刻的位姿 x_t 的概率分布; 同理, $p(z_t | x_t, m)$ 表示机器人的环境地图 m 和机器人 t 时刻的位姿 x_t 已知的情况下, 观测数据 z_t 的概率分布, 之后通过贝叶斯准则计算机器人位置 x_t 的概率分布:

$$Bel(x_t) = \eta P(z_t | x_t, m) \int P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \quad (2-1)$$

贝叶斯网图描述 SLAM 问题如图 2-2 所示。

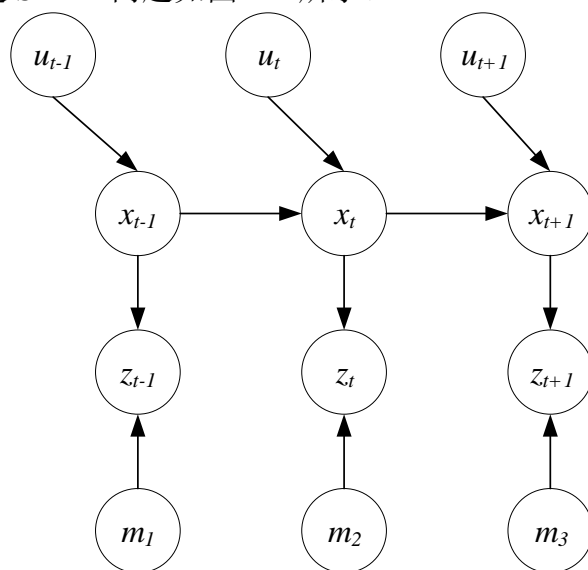


图 2-2 贝叶斯 SLAM 问题描述

2.2.2 Rao-Blackwellized 变换

由于 PF(Particle Filter, 粒子滤波)在高维状态空间中采样时效率低下,所以在某些状态模型中的状态向量可以在一定条件下用其它方法(如解析)求得。若状态模型中存在条件线性高斯模型,则可选择使用 Kalman 滤波器计算地图的条件后验分布,而其它部分仍使用 PF,得到了混合滤波器 RBPF(Rao-Blackwellized Particle Filter)^[62-64],这为在 SLAM 问题中应用粒子滤波器提供了可能。Montemerlo 等人在 2002 年实现了在 SLAM 中首次应用 Rao-Blackwellised 粒子滤波器,并为它取名 FastSLAM 算法,该算法将 SLAM 问题分解成了两部分,分别是机器人的位姿估计以及基于机器人位姿的环境地图估计,即使用 PF 估计运动轨迹,使用 EKF 估计环境特征。RBPF 融合了 EKF 和 PF 的优点,在降低计算复杂度的同时又有较好的鲁棒性。基于 SLAM 问题的 Rao-Blackwellized 变换可以描述为:

$$p(x_{t_r}, m | z_{t_r}, u_{t_r-1}) = p(m | x_{t_r}, z_{t_r}) \cdot p(x_{t_r} | z_{t_r}, u_{t_r-1}) \quad (2-2)$$

式(2-2)中: x 表示机器人的运动轨迹, m 表示所建地图, z 表示传感器观测到的环境信息, u 表示接收到的里程计数据, 式(2-2)把联合后验概率分解为了轨迹的后验概率 $p(x_{t_r} | z_{t_r}, u_{t_r-1})$ 和给定轨迹下地图的后验概率 $p(m | x_{t_r}, z_{t_r})$, 使用粒子滤波算法计算 $p(x_{t_r} | z_{t_r}, u_{t_r-1})$, 在 $p(x_{t_r} | z_{t_r}, u_{t_r-1})$ 的基础上使用卡尔曼滤波算法计算 $p(m | x_{t_r}, z_{t_r})$ 。

(1) 粒子滤波部分

本文中所用的粒子滤波是一种基于马尔可夫链蒙特卡罗(Markov Chain Monte Carlo, MCMC)的算法^[65], 它用粒子集表示概率的方法来模拟状态空间, 进而估计未知状态。粒子滤波中, 每个粒子都代表了机器人一种可能的轨迹, 它的实现过程可以简述为: 若机器人在已知地图中定位, 那么机器人首先需要在所有地图中均匀地播洒粒子, 这时每个粒子代表着机器人可能的位置, 之后随着机器人的运动, 每个粒子也做相应的运动, 这段时间里, 里程计返回的里程信息和激光雷达或视觉所观察到的环境信息分别与各个粒子的轨迹和地图信息相匹配, 得到机器人位置的最优分布, 机器人继续运动, 最优分布范围不断缩小, 最后计算出机器人的精确位置。简单地说, 在粒子滤波中, 每个粒子都代表着一种轨迹, 计算每个粒子的概率可以有效的得出机器人现如今最可能的位置。

本文中的 RBPF 算法主要用以下三个步骤实现所用的粒子滤波部分(符号 $P_t^{(i)}$ 表示第 i 个粒子):

第一步，蒙特卡罗：从先验分布中采样：

$$x_t^{(i)} = P(x_t | x_{t-1}^{(i)}) \quad (2-3)$$

得到 $x_t^{(i)}$ ，粒子 $P_t^{(i)}$ 中包括机器人轨迹 $\mu_{t|t-1}^{(i)}$ 和该位置周围的地图 $\Sigma_{t|t-1}^{(i)}$ 信息。

第二步，重要性采样：计算 $P(x_{1:t} | z_{1:t}, u_{1:t-1})$ 。将先验分布作为重要性函数(或建议分布)： $\pi(x_{1:t} | z_{1:t}, u_{1:t-1}) = P(x_t | x_{t-1})$ ，根据重要性函数计算单个粒子的重要性权值 w_t^i ，如下式所示：

$$w(x_{1:t}) = \frac{p(x_{1:t} | z_{1:t}, u_{1:t-1})}{\pi(x_{1:t} | z_{1:t}, u_{1:t-1})} \quad (2-4)$$

对上式所得的粒子的重要性权值进行归一化：

$$w_t^{(i)} = \frac{w(x_{1:t}^{(i)})}{\sum_{j=1}^N w(x_{1:t}^{(j)})} \quad (2-5)$$

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(z_t, u_t | z_{1:t-1}, u_{1:t-2}, x_t^{(i)}) \quad (2-6)$$

其中， $p(z_t, u_t | z_{1:t-1}, u_{1:t-2}, x_t^{(i)})$ 是似然函数。

第三步，重采样：舍弃低权值粒子，只保留高权值的粒子。

由归一化权值 $w_t^{(i)}$ 与采样粒子 $\{x_{1:t}^{(i)}, m^{(i)}\}$ 可以计算 $p(x_t, m | z_{1:t}, u_{1:t-1})$ ，如用 N 个粒子 $\{x_{1:t}^{(i)}, m^{(i)}, w_t^{(i)}\}_{i=1}^N$ 表示地图与路径的后验分布：

$$P(x_{1:t}, m | z_{1:t}, u_{1:t-1}) = \sum_{i=1}^N w_t^{(i)} \delta_{x_{1:t}, m}^{(i)}(x_t, m) \quad (2-7)$$

其中， $\delta_{(\cdot)}(\cdot)$ 表示 Dirac 函数， N 个粒子 $\{x_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ 的路径的后验概率密度：

$$P(x_{1:t} | z_{1:t}, u_{1:t-1}) = \sum_{i=1}^N w_t^{(i)} \delta_{x_{1:t}}^{(i)}(x_{1:t}) \quad (2-8)$$

由上两述可得地图的后验概率：

$$P_N(m | z_{1:t}, u_{1:t-1}) = \sum_{i=1}^N w_t^{(i)} p(m | z_{1:t}, u_{1:t-1}, x_{1:t}^{(i)}) \quad (2-9)$$

使用卡尔曼滤波器对上式进行计算。

(2) 卡尔曼滤波部分

每个粒子都要经过卡尔曼滤波，每个卡尔曼滤波器都对应着相应环境特征。在使用粒子滤波采样 $x_t^{(i)}$ 后，使用扩展卡尔曼滤波器(EKF)计算 $\mu_t^{(i)}$ 和 $\Sigma_t^{(i)}$ 更新不同位置的粒子，计算后验地图，下列式(2-10)到式(2-14)即为 EKF 更新地图均值和方差的实现过程：

$$\mu_{t|t-1}^{(i)} = A(x_t^{(i)})\mu_{t-1|t-1}^{(i)} + F(x_t^{(i)})u_t \quad (2-10)$$

$$\Sigma_{t|t-1}^{(i)} = A(x_t^{(i)})\Sigma_{t-1|t-1}^{(i)}A(x_t^{(i)})^T + B(x_t^{(i)})B(x_t^{(i)})^T \quad (2-11)$$

$$S_t^{(i)} = C(x_t^{(i)})\Sigma_{t|t-1}^{(i)}C(x_t^{(i)})^T + D(x_t^{(i)})D(x_t^{(i)})^T (z_t^{(i)}, u_{t|t-1}^{(i)}) = C(x_t^{(i)})\mu_{t|t-1}^{(i)} + G(x_t^{(i)})u_t \quad (2-12)$$

$$\mu_t^{(i)} = \mu_{t|t-1}^{(i)} + \Sigma_{t|t-1}^{(i)}C(x_t^{(i)})^T S_t^{-1(i)}((z_t, u_t) - (z_{t|t-1}^{(i)}, u_{t|t-2}^{(i)})) \quad (2-13)$$

$$\Sigma_t^{(i)} = \Sigma_{t|t-1}^{(i)} - \Sigma_{t|t-1}^{(i)}C(x_t^{(i)})^T S_t^{-1(i)}C(x_t^{(i)})\Sigma_{t|t-1}^{(i)} \quad (2-14)$$

其中： $\mu_{t|t-1} \triangleq E\{m_t | z_{1:t-1}, u_{1:t-2}\}$ ， $\mu_t \triangleq E\{m_t | z_{1:t}, u_{1:t-1}\}$ ， $\Sigma_{t|t-1} \triangleq \text{cov}(m_t | z_{1:t-1}, u_{1:t-2})$ ，
 $\Sigma_t \triangleq \text{cov}(m_t | z_{1:t}, u_{1:t-1})$ ， $S_t \triangleq \text{cov}((z_t, u_t) | z_{1:t-1}, u_{1:t-2})$ 。

2.2.3 改进的 RBPF 算法

本文中对 RBPF 算法进行了相应改进。本文针对 RBPF 算法的计算量大的问题，提出了一种提高 RBPF 算法实时性的方法，利用粒子的重要性权值大小适量减少卡尔曼滤波部分的重复计算。原始的 RBPF 算法对每一个存在的粒子都要计算和更新其概率分布 $p_t^{(i)}$ ，不断地重复计算 $\mu_t^{(i)}$ 、 $\Sigma_t^{(i)}$ ，以及逆矩阵 $S_t^{-1(i)}$ ，导致计算量特别大。在粒子滤波中拥有高权值的粒子可能会在一些位置相互聚集，而 RBPF 变换中的粒子处于离散状态，这些离散的粒子群在空间中的存在范围是有限的，因此，相同粒子群中的粒子的地图(例如： $\mu_t^{(i)}$ 和 $\Sigma_t^{(i)}$)必然是相近的，可以选取其中一个粒子为代表来计算 $\mu_t^{(i)}$ 和 $\Sigma_t^{(i)}$ ，此时，RBPF 算法中卡尔曼滤波器的运算量将少与实际的粒子数 N 。

改进 RBPF 的卡尔曼滤波器的算法流程：

for $p_t^{(1)}$ 更新均值 $\mu_t^{(1)}$ 和方差 $\Sigma_t^{(1)}$

$$\mu_t^{(1)} \leftarrow \mu_{t|t-1}^{(1)} + \Sigma_{t|t-1}^{(1)}C(x_t^{(1)})^T S_t^{-1(1)}((z_t, u_t) - (z_{t|t-1}^{(1)}, u_{t|t-2}^{(1)}))$$

$$\Sigma_t^{(1)} \leftarrow \Sigma_{t|t-1}^{(1)} - \Sigma_{t|t-1}^{(1)}C(x_t^{(1)})^T S_t^{-1(1)}C(x_t^{(1)})\Sigma_{t|t-1}^{(1)}$$

for $i = 2$ to N do

if $p_t^{(i)} = p_t^{(i-1)}$ then

$$\mu_t^{(i)} \leftarrow \mu_t^{(i-1)}, \Sigma_t^{(i)} \leftarrow \Sigma_t^{(i-1)}$$

else

$$\mu_t^{(i)} \leftarrow \mu_{t|t-1}^{(i)} + \Sigma_{t|t-1}^{(i)}C(x_t^{(i)})^T S_t^{-1(i)}((z_t, u_t) - (z_{t|t-1}^{(i)}, u_{t|t-2}^{(i)}))$$

$$\Sigma_t^{(i)} \leftarrow \Sigma_{t|t-1}^{(i)} - \Sigma_{t|t-1}^{(i)}C(x_t^{(i)})^T S_t^{-1(i)}C(x_t^{(i)})\Sigma_{t|t-1}^{(i)}$$

end if

end for

选择性更新 $\mu_t^{(i)}$ 和 $\Sigma_t^{(i)}$ ，在粒子数较多时(如： $100 \leq N \leq 1000$)，明显可以较少计算机时间和性能的消耗，而当粒子数较少的时候(如： $N \leq 10$)，这种改进方法优势并不明显，不过在实际的 SLAM 应用场景中，粒子的数目都是成百上千的，所以这种改进方法有它的可取性。

采用式(2-15)的粒子滤波的建议分布，它通过舍去一些低权重粒子，减少粒子数量的方法来减小计算量。下式是建议分布的间距：

$$p(x_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_t) \approx \frac{p(z_t | m_{t-1}^{(i)}, x_t)}{\int_{x' \in L^{(i)}} p(z_t | m_{t-1}^{(i)}, x') dx'} \quad (2-15)$$

$$L^{(i)} = \{x | p(z_t | m_{t-1}^{(i)}, x) > \varepsilon\} \quad (2-16)$$

改进的 RBPF 算法采用自适应重采样技术，主要分为先验采样，计算粒子权值，自适应重采样，更新地图等几个步骤。

改进的算法流程图如图 2-3 所示。

图 2-3 中的算法流程如下： $t-1$ 时刻，采样阶段， $x_{t-1}^{(i)} \sim P(x_{t-1} | x_{t-2}^{(i)})$ ，得到无权重粒子 $\{p_{t-1}^{(i)}, N^{-1}\}$ ，计算粒子的重要性权值，得到带权重的粒子 $\{p_{t-1}^{(i)}, w_{t-1}^{(i)}\}$ ；重采样阶段，只保留高权重粒子，得到 $\{p_{t-1}^{(i)}, N^{-1}\}$ ，卡尔曼滤波此时只更新不同位置的粒子，减少 $\mu_t^{(i)}$ 、 $\Sigma_t^{(i)}$ 的计算量； t 时刻重复采样 $x_t^{(i)} \sim P(x_t | x_{t-1}^{(i)})$ ，得到 $\{p_t^{(i)}, N^{-1}\}$ 。

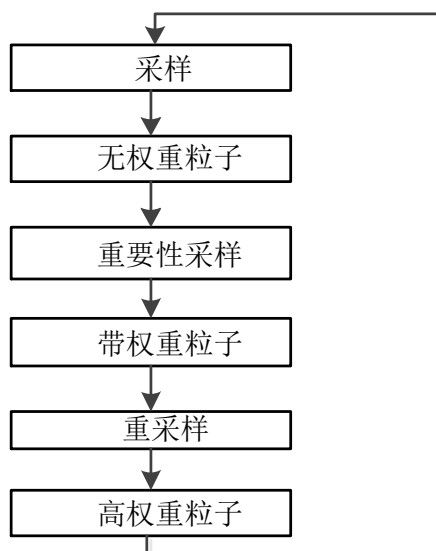


图 2-3 改进 RBPF 算法建议分布方法流程图

2.3 Cartographer 算法与地图创建

2.3.1 Cartographer 方法简述

如图 2-4 所示，使用 Cartographer 算法绘制客厅的地图的过程可分为以下几个步骤：

- (1) 将激光测距仪放入房子的中间，并在纸上标出当前位置 X；
- (2) 测量传感器到任意一个墙壁的距离；
- (3) 在纸上画出代表墙壁的直线，并标出位置 X 和墙壁之间的距离；
- (4) 测量到另外一条墙壁的距离，重复(3)步骤；
- (5) 移动到房间的另一个位置；
- (6) 重复(2)，(3)，(4)步骤，确定当前位置。

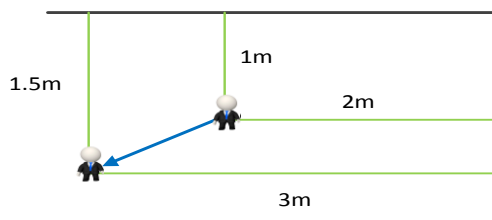


图 2-4 室内 Cartographer 建图方法

2.3.2 Cartographer 算法

2D SLAM 中，位姿 ξ 由平移 (x, y) 和旋转 ξ_θ 组成，由激光雷达扫描得到。

Cartographer 算法优于其他算法之处在于同时在局部地图和全局地图优化了位姿 $\xi = (\xi_x, \xi_y, \xi_z)$ 。

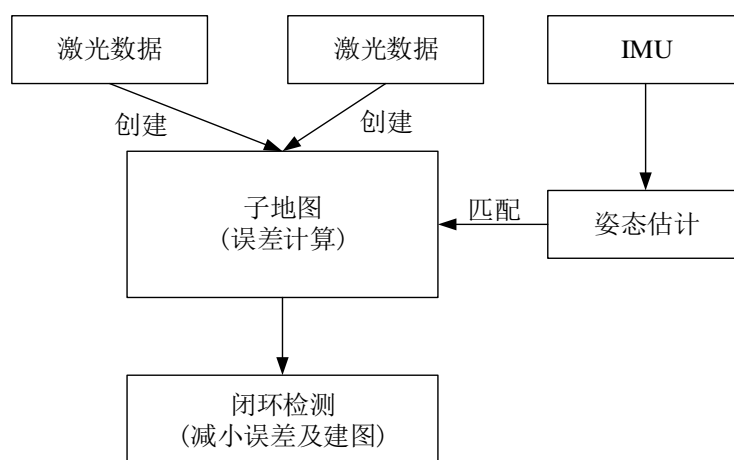


图 2-5 Cartographer 算法流程

如图 2-5 所示, 创建局部地图时, 每个扫描都匹配得到一个子地图, 称为 Submap M, 使用非线性优化将扫描数据与 Submap 进行匹配, 这个过程被称为扫描匹配(Scan matching)。创建全局地图时, 随着时间的推移, 扫描匹配存在累积误差, 扫描次数越少, 累积误差越小, 而创建大量 Submap 需要更大的空间, 采取稀疏姿态调整(Sparse Pose Adjustment)的方法, 优化所有扫描器的位姿和 Submap。当 Submap 不再变化, 除相对位姿外, 扫描器和 Submap 成对进入闭环。扫描匹配器在后台运行, 一旦找到好的匹配, 对应的相对位姿就加入到优化中, 从而去除累积误差。

(1) 扫描匹配

Submap 是扫描所得信息与 Submap 坐标系不断匹配的迭代过程。以 $o \in \mathbf{R}^2$ 为原点, 扫描点的信息写作 $\mathbf{H} = \{h_k\}_{k=1, \dots, k}, h_k \in \mathbf{R}^2$, 扫描器位姿 ξ 以转换系数 T_ξ 表示, 扫描点信息通过 T_ξ 转换到 Submap 坐标系:

$$T_\xi p = \underbrace{\begin{pmatrix} \cos \xi_\theta & -\sin \xi_\theta \\ \sin \xi_\theta & \cos \xi_\theta \end{pmatrix}}_{R_\xi} p + \underbrace{\begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}}_{t_\xi} \quad (2-17)$$

连续扫描周围环境, 建立子地图, 子地图采用概率网格模式 $M: rZ \times rZ \rightarrow [p_{\min}, p_{\max}]$, 这表示将给定分辨率为 r 的离散网格点映射到一个值, 该值表示网格点被阻塞的概率。对于每个网格点, 所有最接近该网格的点以一个像素表示, 如图 2-6 所示。

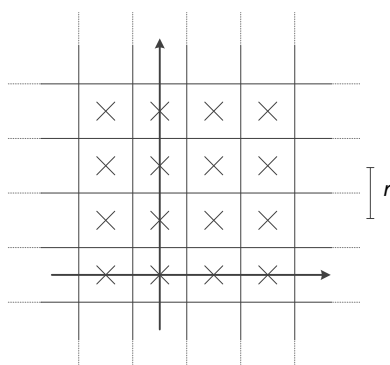


图 2-6 概率地图与相应像素

将扫描器放入到概率网格中进行扫描, 得到一组击中点(hits)和一组不相交的穿过点(misses), 如图 2-7 所示。击中点以最近的网格点表示, 穿过点是指除击中点外, 与扫描器和扫描点之间的射线相交的像素相关的网格点。处于击中集和穿过集这两个集合中的以前未被观测到的网格点, 将会被分配一个概率 p_{hit} 或 p_{miss} 。如果网格点 x 被观察到, 更新击中和穿过的概率(odds):

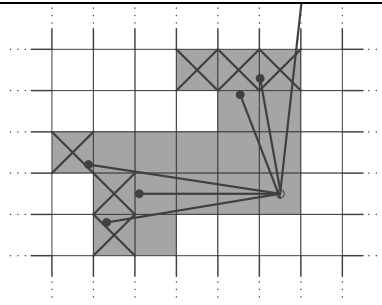


图 2-7 穿过点与击中点

$$\text{odds}(p) = \frac{p}{1-p} \quad (2-18)$$

$$M_{\text{new}}(x) = \text{clamp}\left(\text{odds}^{-1}\left(\text{odds}(M_{\text{old}}(x)) \cdot \text{odds}(p_{\text{hit}})\right)\right) \quad (2-19)$$

扫描器放入 submap 前,使用 Ceres 扫描匹配器优化扫描器相对 submap 的位姿 ξ 。扫描匹配器用于寻找能使 submap 中扫描点的概率最大化的扫描器位姿,可以看作是非线性最小二乘问题:

$$\arg \min_{\xi} \sum_{k=1}^K \left(1 - M_{\text{smooth}}\left(T_{\xi} h_{\xi}\right)\right)^2 \quad (2-20)$$

其中 T_{ξ} 将 h_{ξ} 从扫描器坐标系转换到 Submap 坐标系, 函数 $M_{\text{smooth}}: R^2 \rightarrow R$ 是局部 Submap 中概率的平滑。局部优化需要良好的初步估计, 平滑函数的数学优化精度通常比网格分辨率高。

(2) 闭环优化

扫描器的相对位姿存储在内存中, 用于闭环优化。与扫描匹配一样, 闭环优化也是一个非线性最小二乘问题:

$$\arg \min_{\Xi^m, \Xi^s} \frac{1}{2} \sum_{ij} \rho\left(E^2\left(\xi_i^m, \xi_j^s; \Sigma_{ij}, \xi_{ij}\right)\right) \quad (2-21)$$

其中 $\Xi^m = \{\xi_i^m\}_{i=1, \dots, m}$ 和 $\Xi^s = \{\xi_j^s\}_{j=1, \dots, n}$ 分别为 Submap 和扫描器在世界坐标系中的位姿, 在给定约束下对它们进行优化, 这些约束的形式为相对位姿 ξ_{ij} 和相关协方差矩阵 Σ_{ij} 。对于成对的 Submap i 和扫描器 j , ξ_{ij} 描述了扫描器在所匹配的 Submap 坐标系下的相对位姿, 协方差矩阵 Σ_{ij} 可以使用 Ceres 协方差估计特征(2-10)进行估计。

为完成优化, 使用公式(2-21)搜索像素精确的最优匹配:

$$\xi^* = \arg \max_{\xi \in W} \sum_{k=1}^K M_{\text{nearest}}\left(T_{\xi} h_{\xi}\right) \quad (2-22)$$

其中, W 是搜索窗口, M 首先将自身参数近似到最近的网格点, M_{nearest} 是子地图 M 中参数的最近网格点到对应像素(R^2)的扩展, 其精度依赖公式(2-20)。

选择正确步长可以提高 ξ^* 的计算效率。若最大扫描范围 $d_{\max} = \max_{k=1,\dots,K} \|h_k\|$ 内的扫描点的移动距离不超过单个像素宽度 r ，使用余弦定理，计算角步长 δ_θ ：

$$\delta_\theta = \arccos\left(1 - \frac{r^2}{2d_{\max}^2}\right) \quad (2-23)$$

给定搜索线窗口和角窗口大小，计算搜索的步数：

$$w_x = \left\lceil \frac{W_x}{r} \right\rceil, w_y = \left\lceil \frac{W_y}{r} \right\rceil, w_\theta = \left\lceil \frac{W_\theta}{\delta_\theta} \right\rceil \quad (2-24)$$

得到以 ξ_0 为中心的有限集 w 组成的搜索窗口，即：

$$w = \left\{ \xi_0 + (rj_x, rj_y, \delta_\theta j_\theta) : (j_x, j_y, j_\theta) \in \bar{w} \right\} \quad (2-25)$$

其中， $\bar{w} = \{-w_x, \dots, w_x\} \times \{-w_y, \dots, w_y\} \times \{-w_\theta, \dots, w_\theta\}$ 。

窗口搜索方法选择分支定界法，它可以高效地在较大的搜索窗口内计算 ξ^* 。它将概率子集表示为树上的节点，其中根节点 ω 代表所有可能解，每个节点的子节点组成父节点的分区，共同代表同一个概率集合，叶节点单独表示一个可能解。分支定界法能够精确寻找窗口内得分最高的节点 c ，记为 $score(c)$ 。

分支定界法分为节点选择、分支法则和计算上界三个步骤。首先使用深度优先搜索(DFS)的方法快速评估许多叶节点，计算每个子节点的得分，优先访问得分最高的子节点。引入分数阈值，避免将效果不好的匹配加入闭循环的约束，降低算法对节点选择或查找最初的启发式解的依赖。

概率树中的每个节点都可以写作一个整数元组 $c = (c_x, c_y, c_\theta, c_h) \in Z^4$ ，节点高度为 c_h ，最大平移 $2^{c_h} \times 2^{c_h}$ ，且有特定旋转量。

$$\bar{w}_c = \left(\left\{ (j_x, j_y) \in Z^2 : \begin{matrix} c_x \leq j_x < c_x + 2^{c_h} \\ c_y \leq j_y < c_y + 2^{c_h} \end{matrix} \right\} \times \{c_\theta\} \right), \bar{w}_c = \bar{w}_c \cap \bar{w} \quad (2-26)$$

其中，根节点是一组覆盖搜索窗口的高度为 h_0 的初始节点： $C_0 = \bar{w}_{0,x} \times \bar{w}_{0,y} \times \bar{w}_{0,\theta} \times \{h_0\}$ ， $\bar{w}_{0,x} = \{-w_x + 2^{h_0} j_x : j_x \in Z, 0 \leq 2^{h_0} j_x < 2w_x\}$ ， $\bar{w}_{0,y} = \{-w_y + 2^{h_0} j_y : j_y \in Z, 0 \leq 2^{h_0} j_y < 2w_y\}$ ， $\bar{w}_{0,\theta} = \{j_\theta \in Z : -w_\theta \leq 2^{h_0} j_\theta \leq w_\theta\}$ ；

叶节点的高度为 $c_h = 0$ ，对应的可能解：

$$w \ni \xi_c = \xi_0 + (rc_x, rc_y, \delta_\theta c_\theta) \quad (2-27)$$

\bar{w}_c 搜索区域的边界大于 \bar{w}_c ：

$$score(c) = \sum_{k=1}^K \max_{j \in \bar{w}_c} M_{nearest} \left(T_{\xi_j} h_k \right) \geq \sum_{k=1}^K \max_{j \in \bar{w}_c} M_{nearest} \left(T_{\xi_j} h_k \right) \geq \max_{j \in \bar{w}_c} M_{nearest} \left(T_{\xi_j} h_k \right) \quad (2-28)$$

计算内部节点得分:

$$score(c) = \sum_{k=1}^K M_{precom}^{c_h} (T_{\xi_c} h_k) \quad (2-29)$$

$$M_{precom}^h(x, y) = \max_{\substack{x' \in [x, x+r(2^h-1)] \\ y' \in [y, y+r(2^h-1)]}} M_{nearest}(x', y') \quad (2-30)$$

其中, ξ_c 为叶节点, 预计算网格 M_{precom}^h 与 $M_{nearest}$ 有相同的像素结构, 每个像素中存储从其开始的 $2^h \times 2^h$ 的像素窗口中的最大值。

2.4 仿真结果分析

在 slam benchmarking 官网上下载开源数据集, 对两种 SLAM 算法: Cartographer 算法与 RBPF 算法进行仿真分析, 过程如图 2-8 与图 2-9 所示。

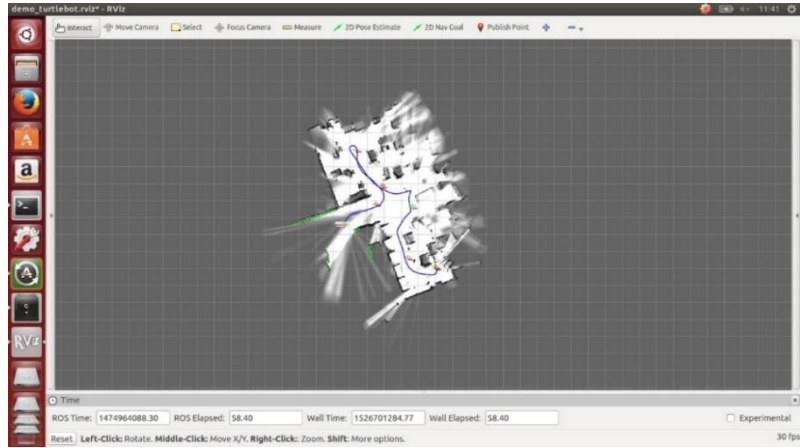


图 2-8 使用数据集对 SLAM 算法仿真过程

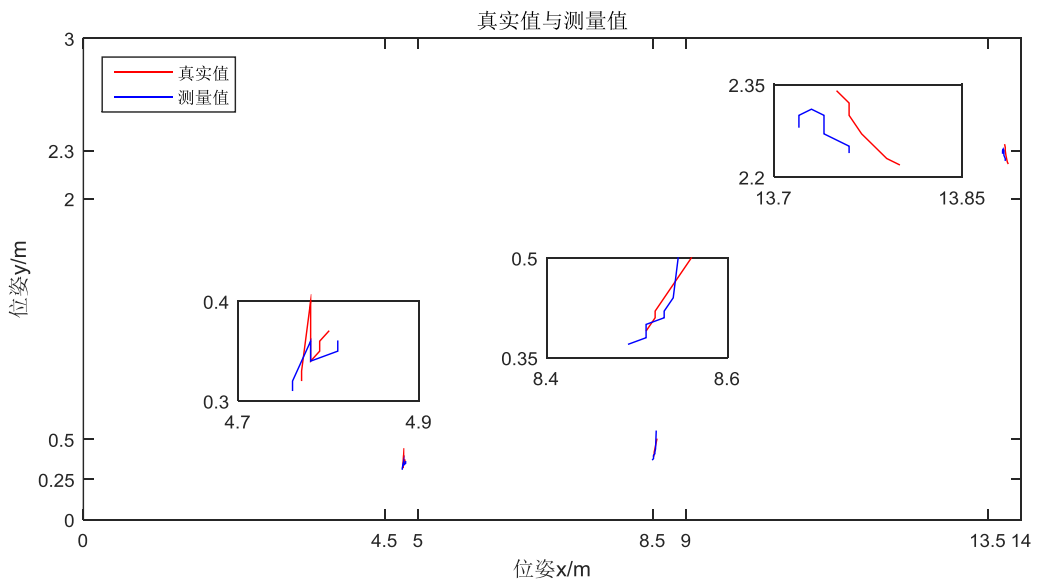


图 2-9 真实值与测量值之间误差

数据集中机器人的位姿作为 SLAM 算法的输入，SLAM 在创建地图的过程中会输出移动机器人在各个时刻的位姿及周围环境地图，选取几组相同位置的机器人的位姿与数据集中的真实数据，求出它们之间的误差，以此作为 SLAM 算法的精度，实现对 SLAM 算法进行仿真分析。

表 2-1 即为对本文中的 SLAM 算法：Cartographer 算法与 RBPF 算法进行仿真所得的地图精度。该表是 SLAM 算法创建地图时的输出位姿与地图中的实际位置之间的误差。

表 2-1 用数据集分析 Cartographer 算法与 RBPF 算法 单位(米)

数据集	RBPF	改进的 RBPF	Cartographer
Aces	0.068	0.052	0.038
Intel	0.067	0.049	0.023
MIT Killian Court	0.088	0.058	0.040
MIT CSAIL	0.037	0.030	0.032
Freiburg bldg 79	0.053	0.047	0.045
Freiburg hospital (local)	0.812	0.614	0.108
Freiburg hospital (global)	8.53	6.41	5.224

从表 2-1 中可以得出，机器人在同一速度和同一环境下行驶，改进后的 RBPF 算法比改进之前的算法所创建地图的精度要高，说明在改进 RBPF 算法的实时性的同时，提高了 RBPF 算法创建地图时的精度。



(a) Intel 实验室

(b) MIT 走廊环境

图 2-10 Radish 数据集中地图环境

以 Intel 实验室和 MIT 走廊环境的误差结果为例分析 Cartographer 算法、RBPF 算法以及改进的 RBPF 算法创建的地图的精度。Intel 实验室属于典型的办公场所的

地图,其中充满了各种的障碍, Cartographer 算法在该环境中的建图误差是 0.023 米,已经处于极为精确的状态,建图速度为 10 分钟。RBPF 算法的精度在 0.067 米,误差为 7 厘米,所花费的时间为 12 分钟,改进后的 RBPF 算法在该环境中误差为 0.049 米,所花费的时间为 8 分钟,相比未改进的 RBPF 算法的地图的速度快,误差小,在 5 厘米以内。

同理, MIT Killian Court 是大型闭合的走廊环境地图,在该环境下, Cartographer 算法与改进后的 RBPF 算法的误差均有所增加, RBPF 算法的精度为 9 厘米,建图时间为 55 分钟,改进的 RBPF 算法的精度为 6 厘米,建图时间为 43 分钟,在速度与精度上优于未改进的 RBPF 算法。此时, Cartographer 算法的精度为 4 厘米,建图时间为 47 分钟,在 500 米 \times 500 米的环境内,改进的 RBPF 算法的建图速度优于 Cartographer 算法,两者之间精度的差别几乎可以忽略不计。

Freiburg hospital 的数据集属于室外环境下的 SLAM,从该数据集的仿真结果可以看出,室外环境下, Cartographer 算法的局部误差为 0.108 米,远远低于 RBPF 算法和改进的 RBPF 算法的局部误差。说明了在室外环境下,适合采用误差偏小的 Cartographer 算法进行 SLAM。

综上所述,可以得出两点结论:(1)改进后的 RBPF 算法比未改进的 RBPF 算法的建图速度更快,地图精度更高,误差更小,因此本文后续实验中将采用改进后的 RBPF 算法与 Cartographer 算法进行现实环境中的地图定位偏差分析;(2)在大型的、复杂环境中, Cartographer 算法的精度明显高于改进的 RBPF 算法。而在小型环境中, Cartographer 算法的地图精度与改进的 RBPF 算法的地图精度相差并不大,都处于厘米级,因此,本文的后续实验中的将重点研究小型室内环境下, Cartographer 算法的地图精度与改进的 RBPF 算法的地图精度的差别。

2.5 本章小结

本章首先分析了 SLAM 算法的基本方法和流程,在此基础上研究了当前最流行的两种 SLAM 算法: RBPF 算法和 Cartographer 算法,分析了这两种算法的理论原理和算法流程。RBPF 算法是基于蒙特卡罗-粒子滤波和扩展卡尔曼滤波的算法,并提出了改进其实时性的方法:改进建议分布和减少卡尔曼滤波循环次数。Cartographer 算法使用了 Submap 的方法,实时进行定位与建图, Submap 在短时间内足够精确,当没有新的扫描点加入时,自动进入后端全局优化的过程,使用分支定界

(branch-and-bound)法和预计算 Submap 的 Grids 实现全局优化。最后，本章利用开源框架中的数据集对 RBPF 算法、改进的 RBPF 算法和 Cartographer 算法进行了仿真分析，结果表明改进的 RBPF 算法的建图速度与地图精度均高于未改进的 RBPF 算法，改进的 RBPF 算法在速度上优于 Cartographer 算法，精度上相差无几。本文在后续实验中使用改进的 RBPF 算法代替未改进的 RBPF 算法与 Cartographer 算法进行地图定位偏差的比较。

第3章 地面移动机器人运动模型分析

3.1 差分轮里程计算原理

航迹推演是移动机器人的基本问题，它属于移动机器人的底层应用程序，它可以估计机器人位姿，描述了移动机器人速度、左右轮速度与移动机器人轨迹之间的转换关系。定位导航与普通的方向控制都要用到航迹推演。本章主要介绍了机器人差速驱动模型中两种计算里程(odom)的方法。

差速驱动属于两轮驱动系统，轮子分别安装在机器人两边，有各自的执行器，机器人的运动轨迹是每个轮子的运动矢量和。差速驱动的优点在于机构简单，只需要两个驱动执行机构，且可以原地旋转。

计算里程的目的主要是为了估计机器人的运动轨迹，在 SLAM 中，已知机器人的里程信息，就可以初步估计出机器人在地图中的位姿以及当前速度信息。本文中所使用的机器人采用差分的运动控制方式。差分轮的运动并不是连续的，它存在最小的运动步长， Δ 是差分轮 N 分之一。假设移动机器人在水平面上行走，其中心作为移动机器人的坐标位置，移动机器人的坐标系遵循右手法则^[67-69]。

3.2 差速驱动机器人运动模型

图 3-1 是两轮差速驱动的移动机器人的运动模型。

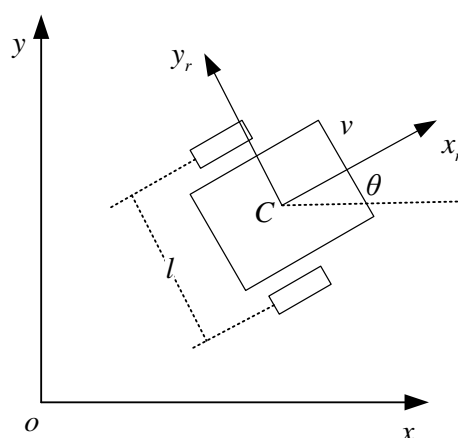


图 3-1 两轮差速移动机器人运动模型

如图 3-1 所示， C 点是两轮差速移动机器人的中心位置，坐标为 (x_c, y_c) 。此时，

机器人的位姿为： $P = (x_c, y_c, \theta)^T$ 。

由刚体的运动学知识可知差速驱动的移动机器人的运动方程：

$$\begin{pmatrix} v \\ w \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{l} & -\frac{1}{l} \end{pmatrix} \begin{pmatrix} v_r \\ v_l \end{pmatrix} \quad (3-1)$$

$$\begin{pmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \quad (3-2)$$

其中， v 是机器人中心处的线速度， w 是机器人中心处的角速度， v_r 和 v_l 分别是移动机器人左右轮的速度， θ 是机器人中心旋转的角度， l 是移动机器人两个轮子之间的轮距。

假设移动机器人的运动是纯滚动的，不存在滑动，那么应该满足下列条件：

$$\dot{y}_c \cos \theta - \dot{x}_c \sin \theta = 0 \quad (3-3)$$

根据实际情况中的采样频率，将式(3-2)改写成以下形式：

$$\begin{cases} x(k+1) = x(k) + v(k)T_s \cos \theta(k) \\ y(k+1) = y(k) + v(k)T_s \sin \theta(k) \\ \theta(k+1) = \theta(k) + w(k)T_s \end{cases} \quad (3-4)$$

其中， T_s 是机器人的采样时间，即 k 时刻到 $k+1$ 时刻所经历的时间。

3.3 动态窗口法计算运动轨迹

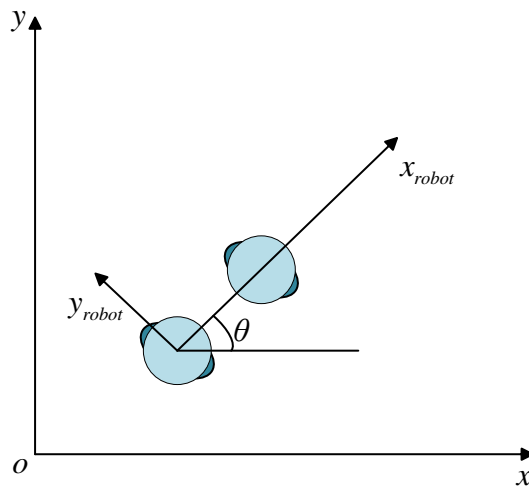


图 3-2 机器人运动图解

如 3-2 图所示，若机器人不能全向移动，只能前进和旋转(v_t, w_t)的距离，在两

个相邻时刻内，机器人的运动距离较短，因此可以将之看作直线，即沿机器人坐标系 x 轴移动了 $v_t \cdot \Delta t$ ，投影到世界坐标系，即可得到两个时刻内机器人在世界坐标系中移动的位置 Δx 和 Δy ，如式(3-5)所示。

$$\begin{cases} \Delta x = v \Delta t \cos(\theta_t) \\ \Delta y = v \Delta t \sin(\theta_t) \end{cases} \quad (3-5)$$

将上式累加求和，即可求得一段时间内的轨迹：

$$\begin{cases} x = x + v \Delta t \cos(\theta_t) \\ y = y + v \Delta t \sin(\theta_t) \\ \theta_t = \theta_t + w \Delta t \end{cases} \quad (3-6)$$

若机器人是全向移动的，那么机器人坐标轴 y 轴上也有速度，将该速度也投影到世界坐标系：

$$\begin{cases} \Delta x = v_y \Delta t \cos\left(\theta_t + \frac{\pi}{2}\right) = -v_y \Delta t \sin(\theta_t) \\ \Delta y = v_y \Delta t \sin\left(\theta_t + \frac{\pi}{2}\right) = v_y \Delta t \cos(\theta_t) \end{cases} \quad (3-7)$$

那么机器人的运动轨迹就可以用机器人两个方向的速度累加求和来表示：

$$\begin{cases} x = x + v_x \Delta t \cos(\theta_t) - v_y \Delta t \sin(\theta_t) \\ y = y + v_x \Delta t \sin(\theta_t) + v_y \Delta t \cos(\theta_t) \\ \theta_t = \theta_t + w \Delta t \end{cases} \quad (3-8)$$

ROS 中的机器人轨迹的局部规划就是使用的公式(3-8)。

根据运动模型，机器人不断地采样速度就可以推算轨迹，但受机器人本身限制和环境限制，所采样的速度存在也存在如下限制：

(1) 受本身限制，移动机器人存在最大速度和最小速度：

$$V_m = \{v \in [v_{\min}, v_{\max}], w \in [w_{\min}, w_{\max}]\} \quad (3-9)$$

(2) 移动机器人的电机力矩有限，所以存在最大加减速限制：

$$V_a = \{(v, w) | v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t] \cap w \in [w_c - \dot{w}_b \Delta t, w_c + \dot{w}_a \Delta t]\} \quad (3-10)$$

其中， v_c 、 w_c 是机器人当前速度。

(3) 为使移动机器人能在碰到障碍物之前停下来，在最大减速条件下，速度存在一个范围：

$$V_a = \{(v, w) | v \leq \sqrt{2 \cdot \text{dist}(v, w) \cdot \dot{v}_b} \cap w \leq \sqrt{2 \cdot \text{dist}(v, w) \cdot \dot{w}_b}\} \quad (3-11)$$

其中， $\text{dist}(v, w)$ 为速度轨迹 (v, w) 上离障碍物最近的距离。

3.4 左右轮差速计算运动轨迹

动态窗口法的应用场景是在已知机器人线速度和角速度，对机器人的运动轨迹进行推算，然而若采用左右轮差速的驱动方式，将无法直接应用动态窗口法来计算机器人的运动轨迹，因此本小节主要介绍通过左右轮速度推导计算得到机器人运动轨迹的方法。

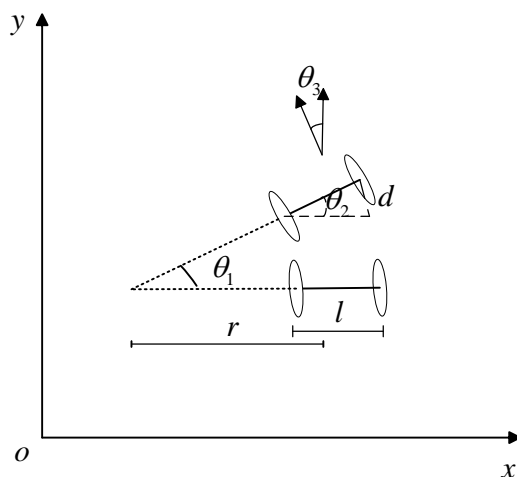


图 3-3 差速驱动机器人运动模型

图 3-3 所示是移动机器人在两个相邻时刻的位姿，其中 r 是移动机器人所做圆弧运动的半径， θ_1 是所做圆弧运动的角度， θ_3 是两相邻时刻移动机器人航向角的差， l 是机器人左右轮的轮间距， d 是机器人在相邻时刻内右轮比左轮多走的路程。

其中，移动机器人在相邻时刻内前进的路程为 s ：

$$s = \left(v_l + \frac{d}{2} \right) \Delta t = \left(\frac{v_r + v_l}{2} \right) \Delta t \quad (3-12)$$

因此，移动机器人线速度等于左右轮速度的平均：

$$v = \frac{v_r + v_l}{2} \quad (3-13)$$

移动机器人在相邻时刻内的旋转的角度 θ_1 可以根据弧度计算公式得到：

$$\theta_1 \approx \frac{d}{l} = \frac{(v_r - v_l) \cdot \Delta t}{l} \quad (3-14)$$

已知移动机器人相邻时刻前进的路程和角度，可计算出移动机器人的当前位姿，如式(3-15)所示：

$$\begin{cases} x_t = x_{t-1} + s \cdot \cos \theta_1 \\ y_t = y_{t-1} + s \cdot \sin \theta_1 \\ \theta_t = \theta_{t-1} + \theta_1 \end{cases} \quad (3-15)$$

此外，根据移动机器人的左右轮速度还可以计算出它的角速度和所做圆弧运动的半径。

根据图 3-3 中的几何关系，可知机器人在两个时刻内的航向角变化量 θ_3 与 θ_1 相等，即移动机器人航向角变化的角度与其绕运动轨迹的圆心旋转的角度相等：

$$\theta_3 = \theta_2 = \theta_1$$

由于相邻时刻时间间隔很短，角度变化量 θ_2 很小，使用下面的近似公式：

$$\theta_2 \approx \sin(\theta_2) = \frac{d}{l} = \frac{(v_r - v_l) \cdot \Delta t}{l} \quad (3-16)$$

机器人绕运动轨迹圆心运动的角速度 w ：

$$w = \frac{\theta_1}{\Delta t} = \frac{(v_r - v_l)}{l} (\text{rad} / \text{s}) \quad (3-17)$$

因此移动机器人圆弧运动的半径：

$$r = \frac{v}{w} = \frac{l \cdot (v_r + v_l)}{2(v_r - v_l)} \quad (3-18)$$

半径无穷大时，此时，移动机器人做直线运动。

3.5 里程误差分析

里程计计算机器人的轨迹时存在累积误差，所以只依靠里程计计算机器人位置时，与实际误差会相差越来越大。下面通过实验说明累积误差的增长过程和动态窗口法与两轮差速法在累积误差中的表现。

设计两组实验，一组采用动态窗口法计算机器人轨迹，另一组是使用左右轮差速计算机器人轨迹。

选择一个大于 10 米的长走廊，选定起点(0, 0)，从起点开始，在走廊里每向前 1 米做一个标记，共做 11 个标记。在没有地图的情况下，控制移动机器人以 0.2 米每秒的速度向前移动，每次移动的距离 1 米，依次经过这些标记点。当机器人行驶了 10 米后，驱使机器人旋转，向出发点前进，仍然以 0.2 米每秒的速度，每次前进的距离为 1 米，依次经过标记点。

机器人每次移动 1 米，机器人每次移动后，记录机器人距离相应标记点的距离差，作为机器人此时位置与标记点之间的误差。每组实验进行 3 次，选取其中含奇异值最小的一组数据进行分析，如表 3-1 和表 3-2 所示。

将表 3-1 与表 3-2 中的数据绘制成曲线图，结果分别如图 3-4 与图 3-5 所示。

表 3-1 动态窗口法计算轨迹的累积误差

单位(米)

误差	Δx	Δy	$\Delta z = \sqrt{\Delta x^2 + \Delta y^2}$
1	0.003	0	0.003
2	0.005	0	0.005
3	0.012	0.005	0.013
4	0.015	0.006	0.016
5	0.021	0.007	0.022
6	0.025	0.010	0.027
7	0.030	0.011	0.032
8	0.035	0.012	0.037
9	0.042	0.013	0.044
10	0.049	0.014	0.051
11	0.060	0.019	0.063
12	0.063	0.012	0.065
13	0.066	0.012	0.067
14	0.068	0.017	0.070
15	0.070	0.017	0.072
16	0.073	0.021	0.076
17	0.078	0.022	0.081
18	0.082	0.022	0.085
19	0.085	0.023	0.088
20	0.088	0.027	0.092

由于实验时机器人的轨迹为直线，主要在 x 轴上运动，所以运动时 y 轴的累积误差小于 x 轴的累积误差，标准差主要取决于机器人 x 轴的误差。

表 3-2 左右轮差速计算轨迹的累积误差

单位(米)

误差	Δx	Δy	$\Delta z = \sqrt{\Delta x^2 + \Delta y^2}$
1	0.004	0	0.004
2	0.005	0.003	0.006
3	0.012	0.007	0.014
4	0.014	0.008	0.016

续表 3-2

误差	Δx	Δy	$\Delta z=\sqrt{\Delta x^2+\Delta y^2}$
5	0.021	0.009	0.022
6	0.025	0.011	0.027
7	0.032	0.013	0.034
8	0.035	0.015	0.038
9	0.042	0.016	0.045
10	0.049	0.018	0.052
11	0.059	0.020	0.062
12	0.062	0.016	0.064
13	0.065	0.016	0.067
14	0.068	0.020	0.071
15	0.071	0.021	0.074
16	0.075	0.017	0.077
17	0.078	0.025	0.082
18	0.082	0.026	0.086
19	0.088	0.023	0.091
20	0.093	0.024	0.096

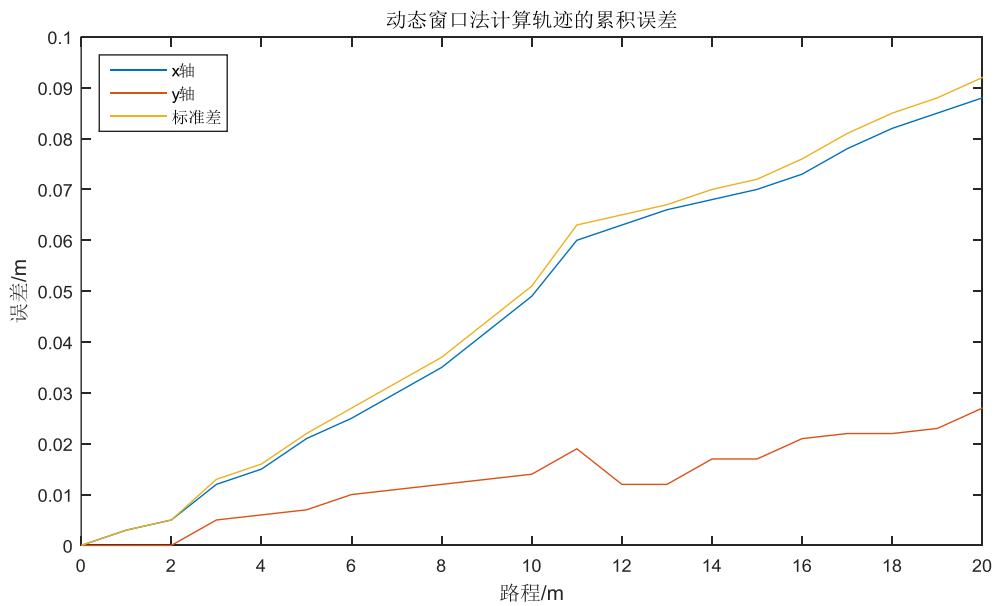


图 3-4 动态窗口法计算轨迹的累积误差

从表 3-1 和图 3-4 中可以看出,随着时间的增长,机器人与实际位置的误差越来越大,尤其是机器人在 x 轴方向(即机器人正前方)的累积误差增加明显逐渐增大,这就是机器人的累积误差。

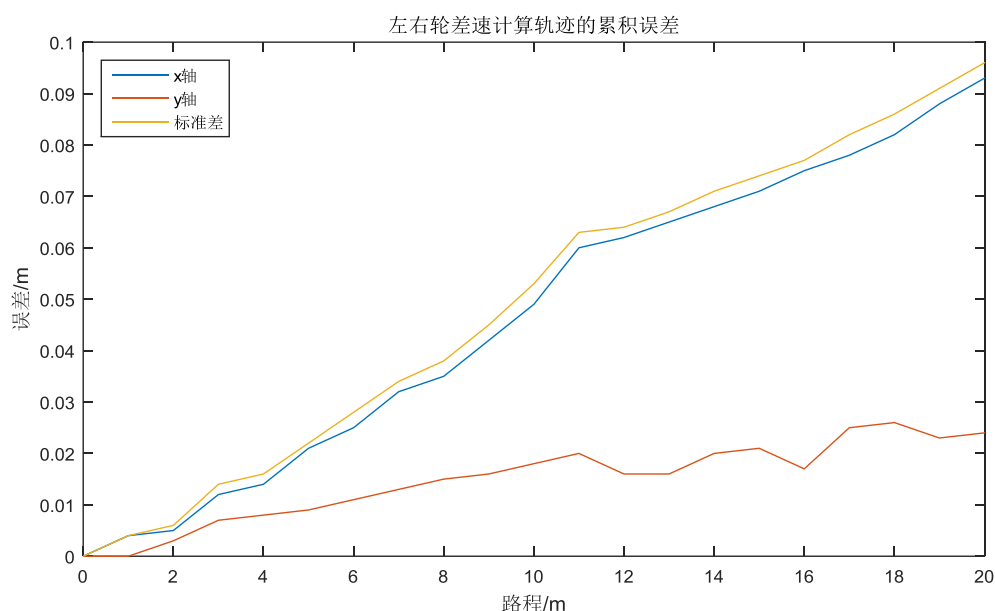


图 3-5 左右轮差速计算轨迹的累积误差

表 3-2 和图 3-5 是使用左右轮差速计算机器人运动轨迹的实验采集的累积误差数据,及绘制的相应的曲线图。

从图 3-5 中可以看出,在 20 次到达目标点时,左右轮差速计算轨迹的累积误差逐步增大,且累计误差依赖于 x 轴上的累积误差。

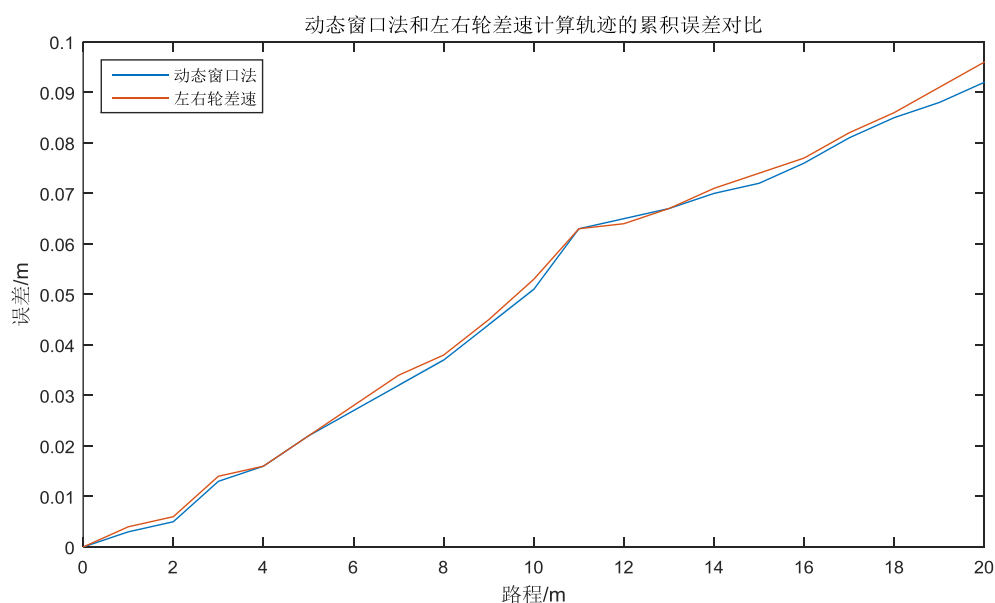


图 3-6 动态窗口法与左右轮速度计算累积误差对比

图 3-6 是左右轮差速计算机器人位姿的累计误差与使用动态窗口法计算机器人位姿的累积误差的对比。

如图 3-6 所示,使用动态窗口法进行机器人的轨迹计算的累积误差,与使用左右轮差速的方法轨迹计算的累积误差,两者误差相差不多,基本上一直处于相同状态。

从图 3-6 中还可以看出,机器人在 10 米以内时,机器人的累积误差增长缓慢,在 11 米时,机器人的累积误差增长加快,这是由于机器人旋转 360 度造成的里程计累积误差增大,在 SLAM 实验时应该尽量避免是移动机器人旋转 360 度,以免对 SLAM 建图及定位的精度造成影响。

综上所述,使用左右轮差速计算机器人轨迹的累计误差,与使用动态窗口法进行机器人的轨迹计算的累积误差相差不多,动态窗口法的精度略高于左右轮差速法。

本文中选择使用动态窗口法计算里程信息,作为机器人的里程数据,与激光雷达的数据一起输入到 SLAM 算法中,进行地图创建的实验。

3.6 本章小结

本章主要研究了两轮差速驱动的移动机器人的运动模型中的轨迹计算问题。本章首先利用刚体运动学分析了差速驱动的移动机器人的运动模型。之后介绍了两种方法估计移动机器人运动轨迹的方法。动态窗口法已知机器人的线速度和角速度,利用机器人局部坐标系与全局坐标系之间的转换关系,将机器人的局部运动量投影到全局坐标系中,从而计算机器人行走的轨迹。两轮差速法是根据左右轮速度计算出机器人的线速度、角速度和机器人圆弧运动的半径,估计机器人的里程信息,进而估计机器人的运动轨迹。实验测得这两种方法的累积误差,其中,动态窗口法计算机器人的运动轨迹的表现良好,可以作为里程计信息,输入到后续 SLAM 算法中,为得到精确的地图数据做准备。

第 4 章 基于 ROS 的移动机器人 SLAM 系统实现

4.1 ROS 系统概念

ROS 系统一般分为：计算图级、文件系统级和社区级三个级别。程序运行时，系统中的所有进程和数据处理过程，都可以通过一种点对点的网络形式显示出来，这就是计算图级，计算图级包括节点(node)、消息(message)、主题(topic)和服务(service)。文件系统级是 ROS 中程序文件的组织结构，使开发人员能有效的管理节点、服务、消息等，例如单独设计的可执行文件在运行时进行的数据连接的过程可以封装到包(Packages)和堆(Stacks)中。以下是 ROS 中几个重要概念的解释：

节点：一个实现一定功能的完整的系统可以由很多节点组成，节点也可以被称为“功能模块”。当两个相关联的节点同时运行时，可以很方便地进行数据的关联，实现端对端的通讯。

消息：节点之间通过消息进行通讯，消息是一种数据结构，它支持标准的数据类型如整型，浮点型，布尔型等。

话题：消息传递的方式，节点可以在一个给定的话题中发布或订阅消息，发布者和订阅者并不了解彼此的存在。

服务：服务是节点的一种补充，适用于不适合节点模式的同步传输模式，包括请求与回应。

ROS Master：使所有节点有条不紊执行的控制器，用于查找其他节点，交换消息或调用服务等。

包：包是 ROS 的软件的组织方式，短小精悍，易于使用，有利于软件的重复使用。节点、依赖库和配置文件等组成的功能模块即是一个包。

堆：堆是各种功能包的集合，它实现的是一个完整模块的所有功能，比如“navigation stack”实现的则是导航的所有功能。同时堆也是 ROS 发布软件的方式。

4.2 移动机器人 SLAM 系统实现

4.2.1 系统构建及传感器选择

搭建移动机器人硬件平台，所需的设备有：笔记本电脑、工控主机、移动机器

人底盘、雷达等。

移动机器人的底盘采用的 Kobuki 底盘,它是 Yujin 公司开发的移动机器人平台,机器人接口控制板有 32 个内置传感器,两个驱动轮,110 度/秒单轴陀螺仪,另外开放式接口可以直接实现对机器人的移动、声音、输入传感器的操作。移动机器人底盘的运动方式采用差速驱动方式,有四种运动方式:原地旋转,此时机器人左右轮速度相等,方向相反;直线前进或后退,此时机器人左右轮的速度大小和方向均相同;曲线前进或后退,此时机器人左右轮速度大小不等,方向相同;转弯运动,此时机器人左右轮速度的大小、方向均不同。



图 4-2 Kobuki 底盘

采用 NVIDIA 系列的 Jetson TK1 作为工控主机,Jetson TK1 是一款微型版的超级电脑,它的系统是经过 GPU 与 CUDA 加速的计算密集型系统,面向计算机视觉、机器人技术、医疗和更多领域,它还支持 OpenGL 和 Tegra 加速的 OpenCV,以及对摄像头和其他外围产品的硬件支持。本文在 Jetson TK1 上装载了 ubuntu 系统和 ROS 系统,布置了需要在工控主机上运行的机器人底盘驱动和雷达的驱动,并对这些驱动设置了开机自启。



图 4-3 Jetson TK1 开发板

移动机器人平台上搭载的雷达选用 Hokuyo 雷达,它的扫描频率高、稳定性强,能够为移动机器人提供全面且准确的环境信息。Hokuyo 雷达与工控机之间采用 USB 口的方式进行通讯。



图 4-4 Hokuyo 雷达

其中笔记本电脑与移动机器人上的工控机采用无线的方式进行通讯，工控主机与移动机器人底盘控制器之间采用 USB 口进行通讯。

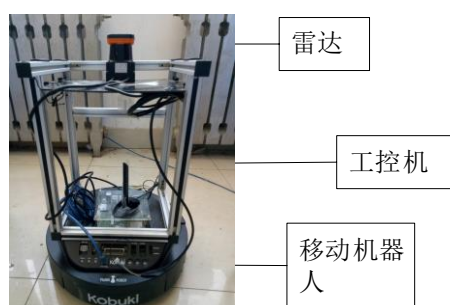


图 4-5 硬件机器人平台

图 4-5 是搭建的基于二维激光雷达的硬件机器人平台。如图所示，kobuki 机器人底盘性能稳定，行走平稳，左右两轮差速驱动模式，通过控制两轮的转速实现移动机器人的直线行走、转弯运动和原地旋转等。移动机器人底盘接收工控主机的运动指令，并转化为对电机的控制信号，同时将里程计的信息反馈到工控机 Jetson TK1 上。平台上搭载 Hokuyo 激光雷达，激光雷达扫描平面 270° ，用于采集周围环境障碍物的距离信息，SLAM 可根据激光雷达的扫描信息计算周围的环境信息，为建图与定位提供原始数据支持。

图 4-6 是机器人硬件平台的系统结构图。图中机器人通过串口与工控主机连接，将工控主机的指令转化为对电机的控制信号，并将陀螺仪和里程计采集到的数据送往工控主机，实现机器人的基本控制功能和里程计信息的处理。激光雷达和深度相机将采集到的位姿数据和距离信息传送到 Jetson TK1，Jetson TK1 对信息进行融合、计算，得出周围环境地图及确定自身位姿，并将地图和位姿信息传送到后台监控区，实现周围地图及机器人位置的可视化。Jetson TK1 作为工控主机，通过无线路由的方式接入网络，与后台监控区进行信息交换。后台监控区接受 Jetson TK1 发送的信息，显示 SLAM 可视化结果，并远程控制机器人。

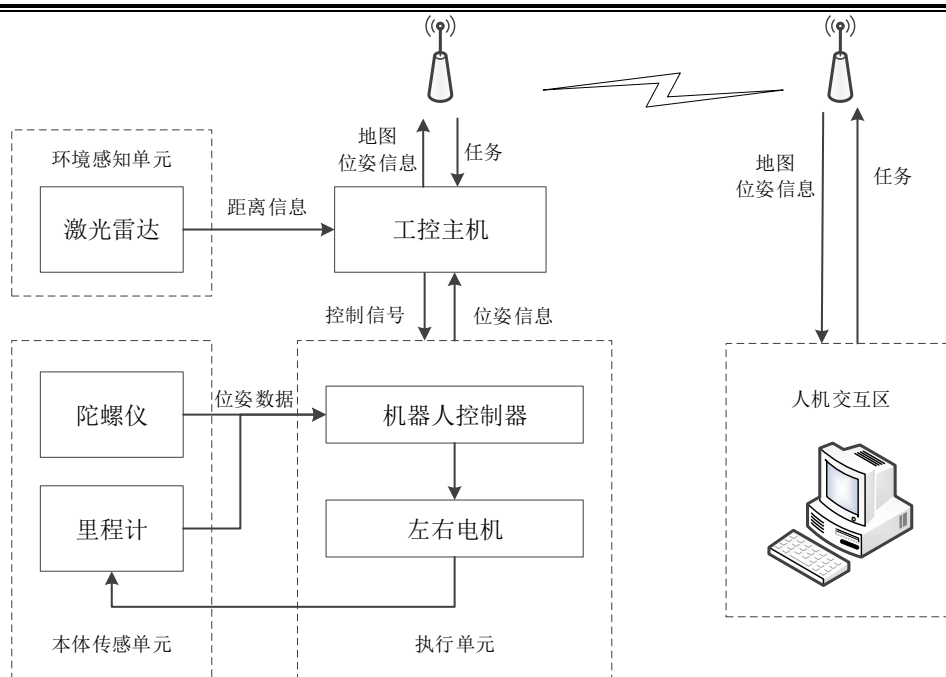


图 4-6 硬件系统结构图

Jetson TK1 的主要功能就是用于处理底层部分采集到的数据和将上级计算机的任务分配到各个底层模块。Jetson TK1 是移动机器人的主机，实现了机器人的自主运动，且 Jetson TK1 与笔记本电脑同时运行 ROS 系统，实现分布式系统，各自执行特定的任务，降低了对硬件处理器的要求，也有助于多机器人的任务规划。

4.2.2 移动机器人 SLAM 系统中的数据传输

搭建的基于 ROS 系统的移动机器人 SLAM 的软件系统可分为三个部分：底层驱动、远程通讯和 SLAM 建图。

底层驱动实现了软件与硬件之间的数据连接以及信息互换，是系统中最基础的部分。底层驱动包括机器人底盘控制器的驱动程序、雷达的驱动程序等，它们负责启动相应的硬件设备，接收上层程序对底层硬件的控制指令，以及底层硬件反馈回上层程序的信息等，是上层程序与硬件之间的连接转换器。

远程通讯是移动机器人必需的功能之一，ROS 属于分布式操作系统，具有实现分布式计算的能力。部署 ROS 分布式系统后，远程通讯模块在 ROS 中则表现为了最基本的话题发布与订阅，这使得实现机器人远程操控功能，甚至多机器人控制变得极其容易。

SLAM 建图是 SLAM 算法的在机器人上的实现，它负责从底层驱动中接收其需

要的雷达扫描信息和里程信息，这些数据经过 SLAM 算法的计算后，得出周围环境的地图信息。SLAM 建图是本文中所搭建的移动机器人的核心，它实现了在激光雷达对周围环境的扫描定位的基础上，对机器人自身位姿及周围环境的识别。

此外，搭建的基于 ROS 系统的移动机器人 SLAM 的软件系统中还包括其它模块，比如：使用 rviz 实现对建图过程及结果的可视化，用于监控建图过程；使用 ROS 系统中专门的地图服务器实现从 SLAM 建图模块所得的环境地图信息到二维栅格地图格式的转换及保存等。

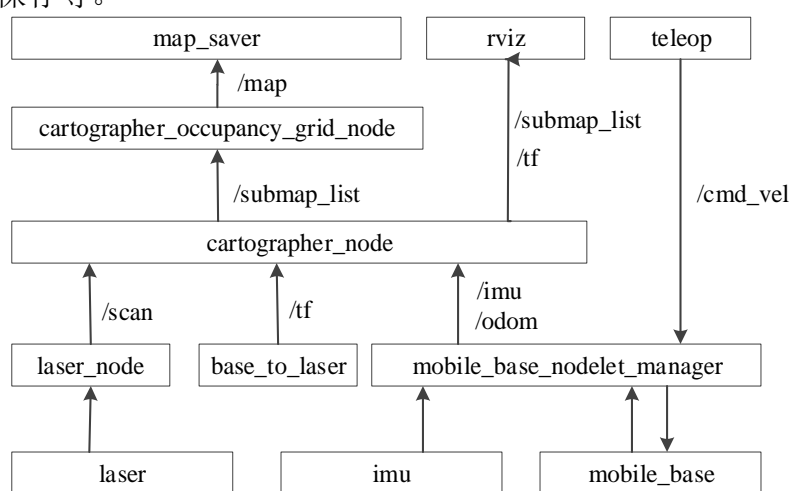


图 4-7 节点之间关系

图 4-7 中所示的是各个软件模块程序在 ROS 中的表现形式及它们之间的数据关联。图 4-7 中的方框里的是 ROS 中的节点名称，一个节点就是一个程序模块，每个节点占用了一个线程；方框之间的箭头指向表明了节点之间的数据传输方向，箭头旁边的是节点发布或订阅的话题名称，箭头指向的节点订阅了箭头尾端的节点发布的话题；话题中包含了一种或多种消息，每个话题都有其特定的消息格式，比如话题 `cmd_vel` 的内容包括 `header`、`time` 以及线速度和角速度等，其中线速度和角速度是一种消息类型。

如图 4-7 所示，节点 `mobile_base_nodelet_manager` 是机器人底盘节点的一个重要管理中枢，它订阅了计算机发布的速度信息、imu 和里程计的信息，实现了与机器人底盘驱动的通信。

节点 `mobile_base` 是底盘驱动节点，它订阅了 `cmd_vel`，根据 `cmd_vel` 向电机发送速度数据，机器人底盘对这些数据进行响应，完成相应的运动任务。节点 `base_to_laser` 发布了机器人与激光传感器之间的坐标转换关系，用于确定机器人在地图上的位置。

节点 `laser_node` 发布了激光数据，用于创建地图。节点 `cartographer_node` 订阅了激光传感器的扫描信息、结合 `imu` 的速度里程信息，计算出周围环境地图，并将其作为话题发布出去。

节点 `cartographer_occupancy_grid_node` 订阅了 `cartographer_node` 发布的地图信息，将其转换为网格地图，用于保存到计算机中，同时 `rviz` 也订阅了地图信息，用于可视化显示和实时监控建图进度。

在搭建的机器人平台中，`/teleop` 节点和 `rviz` 节点位于远程计算机上，`/teleop` 节点读取操作者的输入数据，并发布底盘驱动节点所订阅的话题 `/cmd_vel`，驱动节点都位于工控机上。因为无线数据传输存在一定的带宽频率限制，而 SLAM 需要实时采集大量的传感器数据，所以算法部分位于工控机上，这样也更有助于实现多台计算机的同时工作。

4.3 ROS 分布式系统部署

在实验中，移动机器人本体上搭载一块英伟达系列的 Jetson TK1 作为工控机，上面运行底层驱动的部分。因为导航、SLAM、视觉这些模块都需要采集传感器的实时数据，所以会在工控机上运行，因此对工控机机的运算速度与内存的要求较高。工控机与远程计算机之间通过无线网络连接，这种方式可以很方便地实现多个机器人之间的相互通信。ROS 分布式系统^[69-70]可以在多个不同计算机上同时运行，部署分布式系统的方法过程及要求简述如下：

(1) 选定主从机，节点管理器(ROS Master)运行在主机上，同一个 ROS 系统中只允许运行一个 Master，本文中选定 Jetson TK1 作为主机，远程计算机作为从机。

(2) 分别在主从机中的 `/etc/hosts` 文件中加入对方系统的名字及 IP 地址。每台计算机都会在网络中广播自己的计算机名，而这样做是为了在网络中更快更方便地找到对方系统，添加方法如下：

从机中添加：“192.168.0.202 ubuntu”，ubuntu 是主机 Jetson TK1 中 Linux 系统的名字，192.168.0.202 是主机的 IP 地址；

主机中添加：“192.168.0.1 dream”，dream 是从机 Linux 系统的名字，192.168.0.1 是从机的 IP 地址。

同时在 `.bashrc` 文件中需要配置 ROS 系统的环境变量 `ROS_MASTER_URI`，指主机的 IP 地址，使所有的节点都能准确地找到 master 在系统中的位置，配置方式如下：

从机配置后：

```
export ROS_MASTER_URI=http://192.168.0.202:11311
```

```
export ROS_HOSTNAME=192.168.0.1
```

主机配置后：

```
export ROS_MASTER_URI=http://192.168.0.202:11311
```

```
export ROS_HOSTNAME=192.168.0.202
```

其中，第一行指定主机的 ip 地址，第二行指定此时所在分系统的 ip。

(3) 确定两台计算机之间是否可以相互通信：

主机运行：ping 192.168.0.1；从机运行：ping 192.168.0.202；

若能 ping 通，则表明网络状况良好。

(4) 主从机之间的网络连接设置好后，测试系统是否配置成功。首先，在主机上运行 roscore 命令，启动节点管理器。

(5) 在 Jetson TK1 上运行 rospy_tutorials 包中的 listener 节点，在远程计算机上运行 talker 节点，查看 Jetson TK1 中的 listener 节点是否接收到本地计算机 talker 节点发送的消息，若能收到，则表明从远程计算机到 Jetson TK1 的通信没有问题。同理，在 Jetson TK1 上运行 talker 节点，在远程计算机上运行 listener 节点，查看远程计算机中的 listener 节点是否接收到 Jetson TK1 中 talker 节点发送的消息，若能收到，则表明从 Jetson TK1 到远程计算机的通信没有问题。此时两台计算机之间的 ROS 分布式系统部署成功。

在 ROS 系统运行过程中，Jetson TK1 主要运行机器人及其传感器的底层驱动节点，发布传感数据及订阅速度信息驱动机器人底盘运动，远程计算机上的 teleop、rviz 等节点直接订阅底层的传感信息进行可视化以及发布任务命令。这种分布式系统设计方式减少了计算机的内存负担，保证了信息的同步，在利于系统实时性的实现。



图 4-8 移动机器人 SLAM 平台实验过程

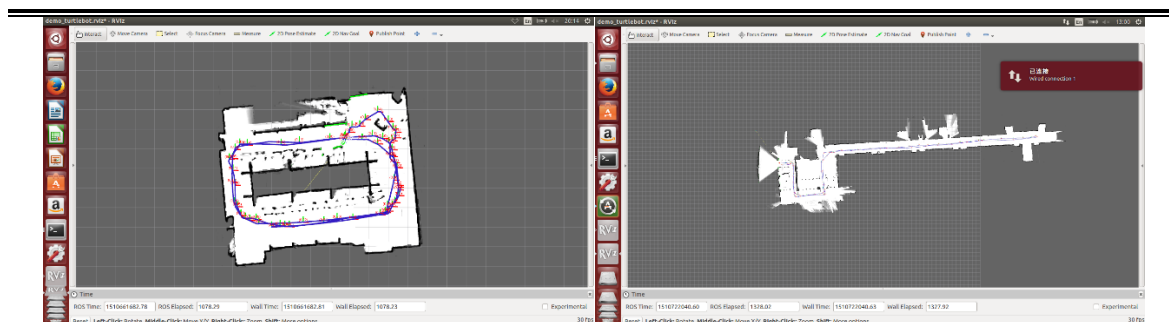


图 4-9 移动机器人 SLAM 过程

图 4-8 和图 4-9 即为分布式系统成功后，在移动机器人的实验平台上在运行 SLAM 时的实验过程及监控制图时的截图。

4.4 本章小结

本章分析了 ROS 系统的特点、组成结构和应用现状，并在此基础上使用移动机器人、Sick 雷达和 Jetson TK1 搭建了基于移动机器人的 SLAM 实验的硬件平台，部署了基于 ROS 系统的软件系统。Jetson TK1 作为工控机位于移动机器人本体上，负责启动运行底层驱动部和 SLAM 算法，Jetson TK1 与远程计算机之间通过无线网络连接，这种方式可以很方便地实现多个机器人之间的相互通信，远程计算机负责可视化及保存地图创建的结果。最后，本章在各模块的数据关联方式及方法的基础上，部署实现了 ROS 实现分布式系统。

第 5 章 实验结果与分析

5.1 SLAM 建图实验环境选取

鉴别一个算法的好坏不仅仅需要对它进行仿真分析，还需要将它运行在现有的硬件设备以及实际的环境中，这样才能得出适合当前现实的结果。鉴于学校的实际情况，选取了会议室作为实验环境，如图 5-1 所示。



图 5-1 SLAM 地图创建实验环境选取

会议室属于小型的结构鲜明的闭合的 SLAM 实验环境，在这种实验环境中 SLAM 算法较容易取得良好的效果。

5.2 数据关联过程

本节在上一节启动实验平台及 SLAM 算法的基础上，对实验过程中的数据类型及各驱动程序及算法之间的数据关联过程进行详细研究及分析。

(1) 激光雷达 laser 数据。

数据类型：LaserScan/Scan。这是 laser 传感器输出的数据类型，是传感器发出的激光在一个固定平面内探测到的障碍物的激光点与传感器之间距离信息的集合。另外该数据类型中还包括该点到传感器的角度及其时间等信息。下面列出的即为监听到的话题/scan 的数据：

header:

seq: 0

stamp:

```

secs: 1486613351
nsecs: 108051866

frame_id: laser
angle_min: -2.35513907051
angle_max: 3.92559274101
angle_increment: 0.0174532923847
time_increment: 2.27658802032e-06
scan_time: 0.000817295105662
range_min: 0.15000000596
range_max: 8.0
ranges:[2.7060000896453857,3.611999988555908,3.609999895095825,inf,2.490000
009536743,2.4809999465942383,...]
intensities: [47.0,47.0,47.0,0.0,47.0,47.0,0.0,47.0,0.0,47.0,47.0,0.0,47.0,0.0,47.0,47.0,
47.0,47.0,0.0,0.0,...]

```

其中数据段 `frame_id`: “laser”表示的是该数据是在 laser 的坐标系下测得的数据；数据段 `angle_min` 与 `angle_max` 表示激光雷达的扫描角度。

数据段 `angle_increment`: 0.0174532923847 是激光雷达的增量，计算方法为： $\text{angle_increment} = (\text{angle_max} - \text{angle_min}) / (\text{double})(\text{node_count} - 1)$ ，其中 `node_count` 为 360，表明雷达每扫描一圈有 360 个点数据。

数据段 `scan_time` 表示雷达扫描一圈所需要的时间，数据段 `time_increment` 表示时间的增量，计算方法为： $\text{time_increment} = \text{scan_time} / (\text{double})(\text{node_count} - 1)$ 。

数组 `ranges` 中有 360 个数据，即扫描到的点到传感器的距离，其中 “inf” 的意思是 infinite，即表示该点的距离无法测量或者无穷大。数组 `intensities` 表示每个测量点的精确度。

数据段 `range_min` 和 `range_max` 是激光雷达的固定参数，表示该雷达的测量的距离范围。

(2) 里程计 odom 数据。

数据类型：`nav_msgs/Odometry`。该数据是机器人底盘轮子反馈来的霍尔信号，提供比较精确的机器人的移动信息，利用相应的算法就能够计算出当前机器人的位置及速度估计。下面即为监听到的里程计话题/odom 的数据：

```
header:
  seq: 52
  stamp:
    secs: 1517210385
    nsecs: 749312977
  frame_id: odom
child_frame_id: base_footprint
pose:
  pose:
    position:
      x: 0.0
      y: 0.0
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: -0.00034906584331
      w: 0.9999999939077
  covariance: [0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.7976931348623157e+308, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.7976931348623157e+308, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.7976931348623157e+308, 0.0, 0.0, 0.0, 0.0, 0.0, 0.05]
twist:
  twist:
    linear:
      x: 0.0
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
```

z: 0.0

其中,数据段 `frame_id:odom` 表示该数据中的位置及速度信息都是在里程计坐标系 `odom` 下产生的。

(3) SLAM 算法发布的 map 数据。

header:

其中数据段 `frame_id`: `map` 表示获得的地图数据是基于 `map` 坐标系的；数据段 `resolution` 是地图的分辨率；数据段 `width` 和 `height` 表示地图的宽度和高度。

数组 `data` 中的数据即为二维的栅格地图信息，用 0-100 内的数字表示，-1 则表示未知。

图 5-2 Cartographer 和 RBPF 所建地图的解释文件

(4) 各坐标系之间的转换。

laser 坐标系: 以激光雷达为中心的坐标系, 观测到的数据也是在激光雷达坐标

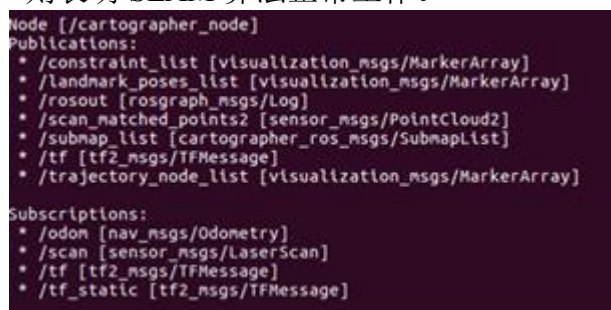
系下的距离信息，需要转换为以机器人中心为原点的坐标系。

odom 坐标系：一般以机器人中心为原点的坐标系，需要严格校准与 laser 坐标系之间的转换关系，才能保证传感器采集的数据与机器人反馈回来的数据在同一个坐标系下，保证 SLAM 地图创建的精度。

map 坐标系：以机器人在 SLAM 建图时的出发点为原点的坐标系，X 轴为机器人正前方，Y 轴在机器人的左方。

本文中以 odom 坐标系为局部坐标系，map 坐标系为全局坐标系对 SLAM 地图创建的结果进行分析。

以 Cartographer 为例，SLAM 算法启动后，查询 cartographer_node 是否工作正常，若结果如图 5-3 所示，则表明 SLAM 算法正常工作。



```

Node [/cartographer_node]
Publications:
  * /constraint_list [visualization_msgs/MarkerArray]
  * /landmark_poses_list [visualization_msgs/MarkerArray]
  * /rosout [roscpp_msgs/Log]
  * /scan_matched_points2 [sensor_msgs/PointCloud2]
  * /submap_list [cartographer_ros_msgs/SubmapList]
  * /tf [tf2_msgs/TFMessage]
  * /trajectory_node_list [visualization_msgs/MarkerArray]
Subscriptions:
  * /odom [nav_msgs/Odometry]
  * /scan [sensor_msgs/LaserScan]
  * /tf [tf2_msgs/TFMessage]
  * /tf_static [tf2_msgs/TFMessage]
  
```

图 5-3 cartographer_node 发布和订阅的话题

图 5-3 是 Cartographer 启动后发布和订阅的话题。如图 5-3 所示，cartographer_node 订阅了 4 个话题的消息，分别是里程计消息/odom，激光雷达消息/scan，坐标系转换/tf 和/tf_static。cartographer_node 发布了机器人的轨迹约束/constraint_list，子地图/submap_list 等，其中/submap_list 由节点 cartographer_occupancy_grid_node 转化为栅格地图，发布话题/map。

5.3 地图定位偏差的实验及分析

在所搭建的硬件平台和软件系统的基础上，设计了三组实验来比较改进的 RBPF 算法和 Cartographer 算法的定位精度。

首先使用改进的 RBPF 算法和 Cartographer 算法对会议室进行了建图，结果如图 5-4 所示。

可以看出，图 5-4 中 Cartographer 算法和改进的 RBPF 算法所建会议室地图的构成几乎没有分别，说明 Cartographer 算法和 RBPF 算法在小的室内环境中都适用，精度几乎没有差别。

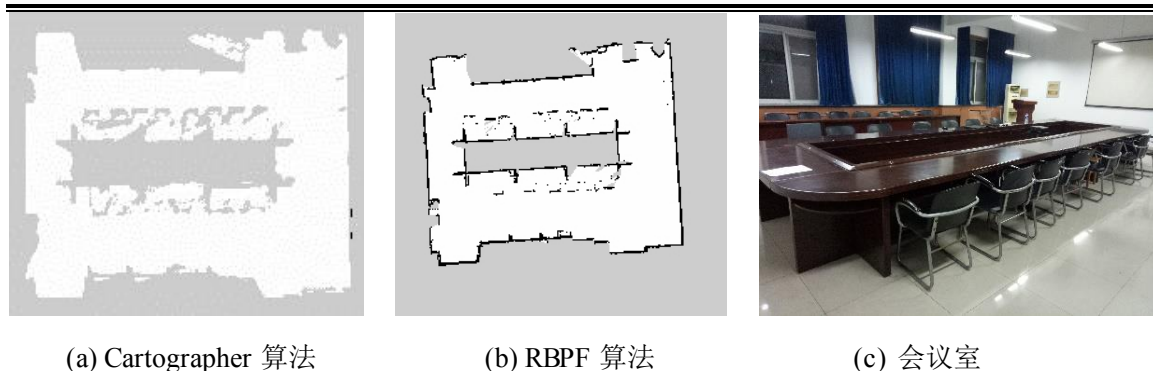


图 5-4 所建会议室地图

如图 5-4 所示，(a)，(b)分别与(c)图中的真实环境相对照，可以看出其中的障碍物(椅子腿)都能扫到并在地图中显示出来，表示 Cartographer 算法和改进的 RBPF 算法所建地图的精度表现良好。

下面设计三组实验来具体计算 Cartographer 算法和改进的 RBPF 算法在所建地图上的定位偏差。

(1) 第一组实验。

第一组实验分为两次实验。分别使用改进的 RBPF 算法 Cartographer 算法测量移动机器人在各自所建地图上行驶较短路径时的地图定位偏差。

在会议室中选择机器人建图时的起点标记为移动机器人的原点位置，并分别测量出移动机器人原点前方 1 米、2 米的位置，分别作为移动机器人的目标点 $(0, 0)$ 、 $(1, 0)$ 、 $(2, 0)$ 。在地图上驱使机器人以 0.2 米/秒的速度分别在这三个目标点之间来回运动，记录下移动机器人停止运动时的位置，计算此位置与相应目标点的误差。

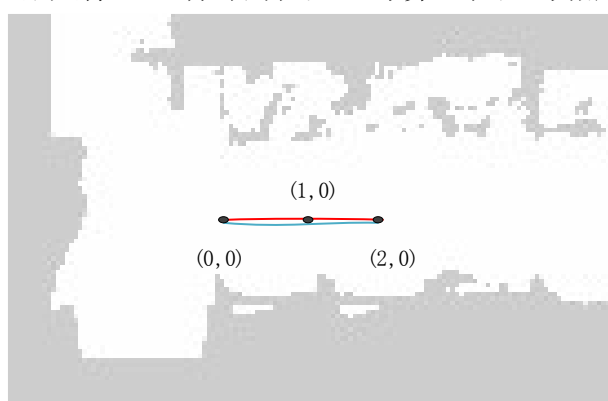


图 5-5 短路径下机器人目标点及轨迹

图 5-5 是机器人目标点及在实验中行走的轨迹。在地图上标记 $(0, 0)$ ， $(1, 0)$ ， $(2, 0)$ ，驱使机器人在这三个点之间重复运动，为防止累积误差的影响，机器人每做一次往返运动后，重启机器人，使误差清零。

实验时，每次机器人从原点位置移动到 2 米位置再回到原点时，误差重置为 0，该实验选取的路径较短，用于分析移动机器人在行走较短路程时的误差。表 5-1 所示的是使用改进的 RBPF 算法进行定位实验所采集的实验数据，移动机器人到达了每个目标点 20 次，选出其中的 7 组数据进行分析。

表 5-1 改进 RBPF 算法地图定位偏差			单位(米)
目标点	(0,0)	(1,0)	(2,0)
1	(-0.05,0.10)	(0.10,0)	(-0.05,0)
2	(0,0.030)	(0.10,0.02)	(0.08,0)
3	(0.05,0.02)	(-0.06,0)	(0.06,0.10)
4	(0.01,0.01)	(-0.05,0.01)	(0.03,0)
5	(-0.06,0.05)	(0.03,0.01)	(0,0.02)
6	(0.08,0.05)	(0.10,0.20)	(0.10,0)
7	(-0.08,0.05)	(0.08,0.05)	(-0.08,0.05)

从表 5-1 中可以看出移动机器人在地图上移动时的定位比较精确，误差都处于厘米级别。为进一步分析机器人到达各个目标点之间的误差，计算表 5-1 中机器人到达各目标点的误差的均值，如表 5-2 所示。

表 5-2 改进 RBPF 算法各目标点定位偏差平均值			单位(米)
目标点	(0,0)	(1,0)	(2,0)
平均值	(0.046,0.043)	(0.074,0.041)	(-0.053,0.024)

从表 5-2 中可以明确看出移动机器人的在地图上定位的误差在 8 厘米以下。

使用 Cartographer 算法进行定位实验所采集的实验数据如表 5-3 所示，移动机器人到达了每个目标点 20 次，选出其中的 7 组数据进行分析。

表 5-3 Cartographer 算法地图定位偏差			单位(米)
目标点	(0,0)	(1,0)	(2,0)
1	(0,0.05)	(-0.05,0)	(-0.02,0.05)
2	(-0.05,0.10)	(0,0.05)	(-0.05,0.10)
3	(-0.05,0.10)	(0.05,0)	(0,0.10)
4	(0.02,0)	(0,-0.01)	(0,0.10)
5	(-0.05,0)	(0.02,0.01)	(0,0.05)

续表 5-3

目标点	(0,0)	(1,0)	(2,0)
6	(0.03,0.01)	(-0.08,0.03)	(0.05,0)
7	(0.02,0.05)	(-0.02,0.01)	(-0.05,0)

从表 5-3 中可以看出,使用 Cartographer 算法,移动机器人在地图上移动时的定位偏差均处于 8 厘米以内,与改进的 RBPF 算法的定位误差大致相同,说明在短路径下, Cartographer 算法与改进的 RBPF 算法的地图定位的精度大致相同。实验中,机器人到达各个目标点的地图定位偏差的平均值如表 5-4 所示。

表 5-4 Cartographer 算法各目标点定位偏差平均值 单位(米)

目标点	(0,0)	(1,0)	(2,0)
平均值	(0.024,0.044)	(0.031,0.016)	(0.024,0.05)

综合第一组实验中的两次实验结果,对比分析机器人在地图上行驶路程较短时, Cartographer 算法和 RBPF 算法的地图定位偏差。

取表 5-1 和表 5-3 中的两次实验中的地图各偏差数据的标准差,绘制 Cartographer 算法和改进的 RBPF 算法的对比分析图,如图 5-6 所示。图 5-6 中分别抛除了表 5-1 和表 5-3 中的异常点,留下了 20 组数据,绘制成了折线图。

从图 5-6 中的数据中可以初步看出改进的 RBPF 算法和 Cartographer 算法的地图定位的偏差基本上维持在 10 厘米以下,改进的 RBPF 算法在第 3-10 次测量时的定位偏差高于 Cartographer 算法,大约相差 3 厘米。

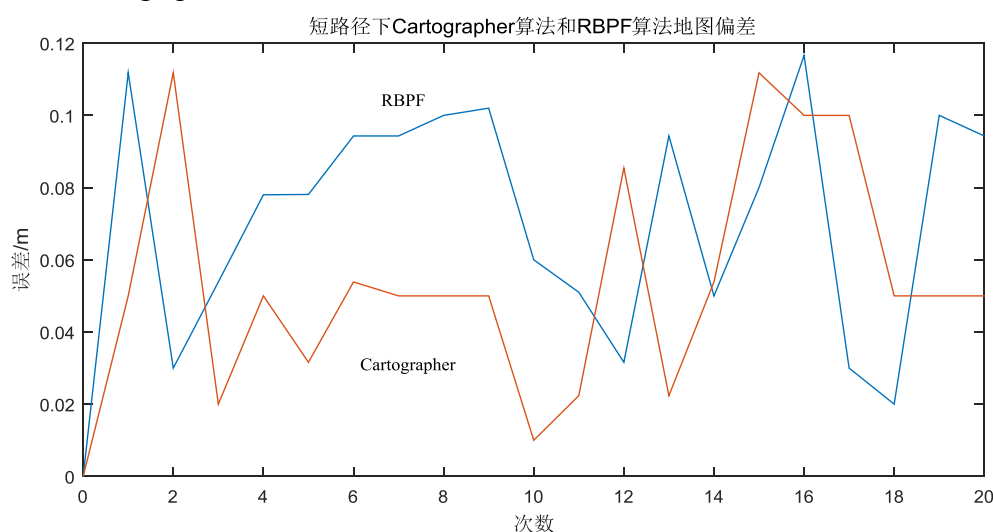


图 5-6 短路径下 Cartographer 和改进 RBPF 地图定位偏差对比

为准确分析改进的 RBPF 算法和 Cartographer 算法的地图定位偏差, 计算两次实验中采集到的各个点的误差, 分别计算改进的 RBPF 算法和 Cartographer 算法的地图定位偏差的平均值和标准差, 结果如表 5-5 所示。

从表 5-5 中可以看出, 使用改进 RBPF 算法的机器人的地图定位偏差的平均值大于使用 Cartographer 算法的机器人的地图定位偏差的平均值, 但两者的标准差相差不多。可以得出结论, 短路程下, 机器人使用 Cartographer 算法创建地图和定位的精度与改进的 RBPF 算法定位的精度相差在 1 厘米。

表 5-5 Cartographer 和改进 RBPF 地图定位精度对比 单位(米)

SLAM 算法	RBPF	Cartographer
平均值	(0.0577,0.0362)	(0.0291,0.0391)
标准差	0.0330	0.0200

第一组实验是将机器人行驶短路程的情况下所得数据与结果。下面第二组实验驱使机器人行驶较长的路程时, 对地图的定位精度进行分析。

(2) 第二组实验。

第二组实验分为两次实验, 分别用于测量移动机器人在 Cartographer 算法和改进的 RBPF 算法所建地图上, 机器人行驶较长路径时的地图定位偏差。以机器人建图时的出发点为原点(0, 0), 选取机器人前方 8 米远的位置作为目标点(8, 0), 用于对机器人长路径下的地图定位偏差的分析。在会议室中标记移动机器人开始建图的原点位置, 并测量出移动机器人原点前方 8 米的位置, 作为移动机器人的目标点(8, 0)。驱使机器人从原点出发, 在地图上给定目标点, 以 0.2 米/秒的速度驱使机器人向目标点行驶, 当机器人停下时, 记录机器人此时的位置与目标点之间的偏差。

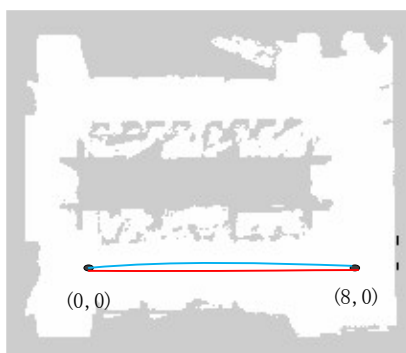


图 5-7 长路径下机器人目标点及轨迹

图 5-7 是实验中机器人在地图中的目标点及相应轨迹。在地图上标记(0, 0)点和

(0, 8)点, 实验中驱使机器人在这两点之间做往复运动, 每次往复运动完成后, 重启机器人, 使误差清零。

该实验用于分析移动机器人在 Cartographer 算法和改进的 RBPF 算法所建地图中行驶较长路径时的定位误差。

表 5-6 较长距离时 Cartographer 和改进 RBPF 地图定位精度

单位(米)

SLAM 算法	RBPF		Cartographer	
目标点	(0,0)	(8,0)	(0,0)	(8,0)
1	(0.15,0)	(0.10,0.11)	(-0.05,0)	(-0.10,-0.10)
2	(0.05,0.05)	(0.15,0.04)	(-0.05,0.1)	(0,-0.05)
3	(0.1,0)	(0.10,0.08)	(0.1,0.1)	(0.10,0.06)
4	(0.02,0.1)	(0.05,0.03)	(0.1,0)	(0,0.05)
5	(0.06,0)	(0.05,0.02)	(0.06,0)	(0.04,0)
6	(0.08,0)	(0.06,0.12)	(0.08,0)	(0.06,0.03)
7	(0.07,0)	(0.09,0.05)	(0.06,0)	(0.07,0.04)
平均值	(0.09,0.025)	(0.085,0)	(0.043,0.028)	(0.024,0.004)

如表 5-6 所示。实验中, 移动机器人到达目标点 20 次, 选出其中的 7 组数据进行分析, 并求出误差的平均值。

从表 5-6 中可以看出, 在使用改进的 RBPF 算法进行实验时, 此时移动机器人到达目标点(8, 0)时的平均误差为 0.085 米, 大于第一组实验中机器人在(0, 0), (1, 0), (2, 0)点之间行驶(较短路程)时的平均误差, 当机器人行驶较长的路程时, 机器人本身是存在累积误差的, 但此累积误差是经过传感器的校正后遗留下的累积误差, 误差比单纯的里程计计算的累积误差要小的多, 处于厘米级别。

在使用 Cartographer 算法进行实验时, 此时移动机器人到达目标点(8, 0)时的平均误差为 0.043 米, 大于第一组实验中机器人使用 Cartographer 算法在(0, 0), (1, 0), (2, 0)点之间行驶(较短路程)时的平均误差, 小于使用改进 RBPF 算法进行建图定位的误差, 一方面说明了 Cartographer 算法在机器人行驶较长的路程时, 机器人存在的累积误差比改进 RBPF 算法的累积误差小, 另一方面说明了 Cartographer 算法对传感器的校正力度比改进 RBPF 算法大。

绘制较长距离时 Cartographer 和改进 RBPF 地图定位精度对比图, 结果如图 5-8

所示。

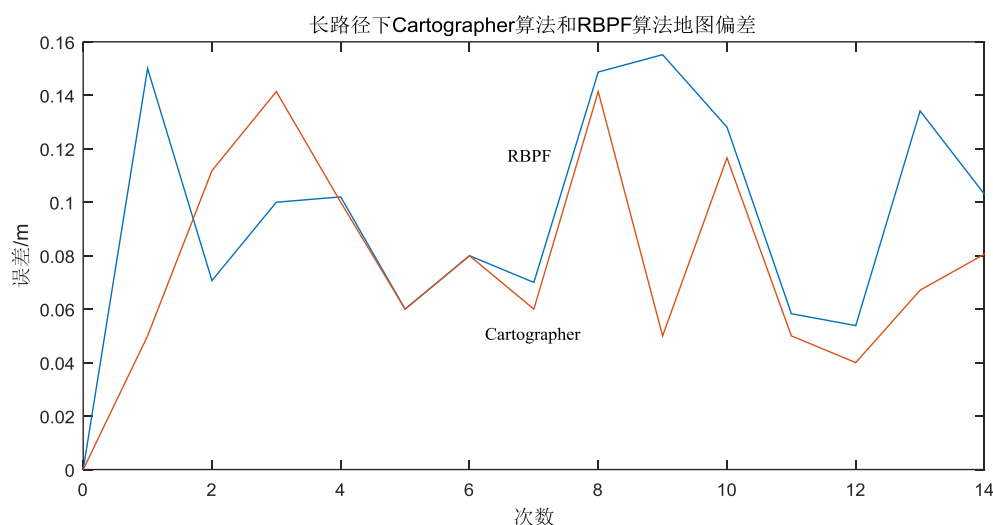


图 5-8 较长距离时 Cartographer 和改进 RBPF 地图定位精度对比

从图 5-8 中可以看出机器人直线行驶较长距离时，Cartographer 算法的地图定位偏差与改进 RBPF 算法的地图定位偏差均在 0.15 米以内、0.09 米上下波动。可以得出结论，机器人在长路径行驶下，Cartographer 算法的地图定位精度与改进的 RBPF 算法的地图定位精度总体相差不大。

(3) 第三组实验

第三组实验分为两次实验，分别为移动机器人在 Cartographer 算法和改进的 RBPF 算法所建地图上存在较多障碍物时，机器人在地图上的定位偏差。

首先选取机器人建图时的出发点为原点，机器人左前方(3.67 米，0.8 米)为目标点，目标点与原点在地图上做出标记，在目标点周围放置两个障碍物(纸盒)。在地图上指定目标点，驱使机器人自动从原点(0, 0)出发，向(3.67, 0.8)点之间移动，在停止实验之前，控制移动机器人多种方式多次向地图上目标点移动，记录机器人移动到地图上的目标点时，机器人在实际中的位置，计算它们与目标点之间的差值。

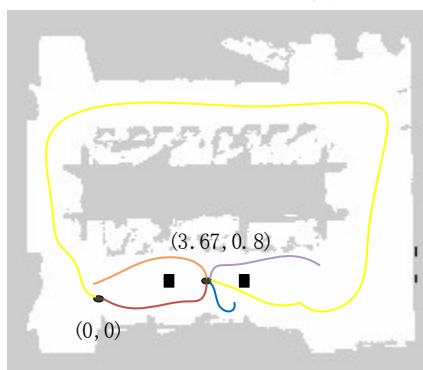


图 5-9 复杂环境下机器人多种轨迹设计

图 5-9 中是本实验中主要采用的几种轨迹。在地图上标记目标点(3.67, 0.8)，驱使机器人从(0, 0)点出发，以各种方式到达目标点。图 5-9 中圆形的点代表机器人的目标点与原点，黑色方块代表障碍物，彩色的线代表驱使机器人到达目标点可行驶的几个主要轨迹。

表 5-7 复杂环境下 Cartographer 和改进 RBPF 地图定位偏差 单位(米)

SLAM 算法	RBPF	Cartographer
1	(0.05,0.05)	(0.02,0)
2	(0.04,0.05)	(0.06,0.01)
3	(0.08,0)	(-0.03,0.02)
4	(0.07,0.02)	(0.02,0)
5	(0.15,0.05)	(-0.02,0.01)
6	(0.14,0.04)	(-0.04,0.02)
7	(0.1,0.02)	(-0.02,0.05)
8	(0.05,0.05)	(0,0.03)
9	(0.08,0.05)	(-0.02,0.05)
10	(0.08,0.02)	(-0.06,0)
11	(0.03,0.02)	(-0.02,0.02)
12	(0.1,0)	(0.1,0.03)
13	(0.08,0)	(0.06,0)
14	(0.09,0.02)	(0.12,0.03)
15	(0.13,0.04)	(-0.01,0.04)
16	(0.09,0.03)	(-0.1,0)
17	(0.11,0.02)	(-0.02,0.06)
18	(0.07,0.03)	(0.02,0.02)
19	(0.09,0)	(-0.02,0.02)
20	(0.1,0.03)	(0.03,0)

本组实验记录了机器人在多障碍物、多路径情况下的地图定位偏差，可作为 Cartographer 算法和改进的 RBPF 算法在复杂环境下的定位偏差比较。本组实验中记录了两次实验中 Cartographer 算法和改进的 RBPF 算法地图定位偏差各 20 组数据，

如表 5-7 所示。

从表 5-7 中可以看出 Cartographer 算法的地图定位偏差明显小于改进的 RBPF 算法的地图偏差。

为使数据结果更加直观,求取表 5-7 中的每个数据的 x 轴偏差与 y 轴偏差的平方差,将这些方差值绘制成折线图,如图 5-10 所示。

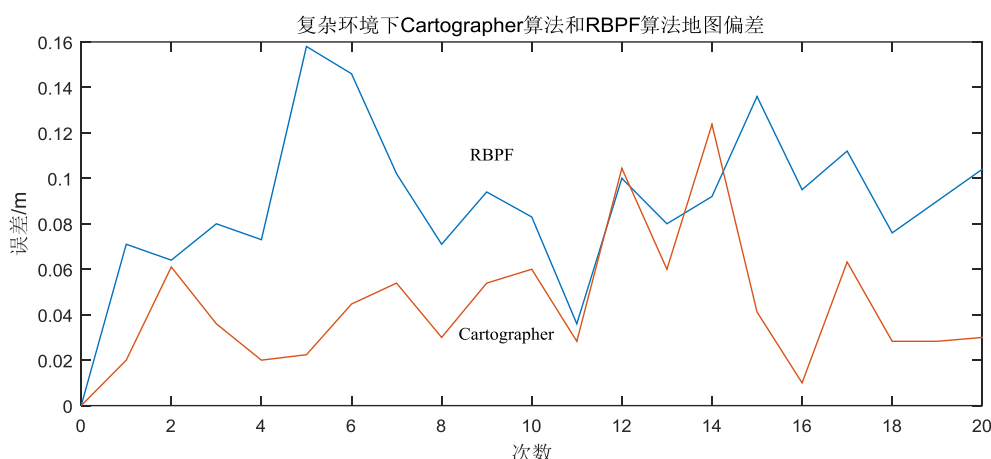


图 5-10 复杂环境下 Cartographer 和 RBPF 地图定位偏差

从图 5-10 中可以看到改进 RBPF 算法的地图偏差明显高于 Cartographer 算法的地图定位偏差。

从图 5-10 中得出, Cartographer 算法的定位偏差基本上维持在 0.1 米以内, 在 0.04 米上下波动; 而改进 RBPF 算法的定位偏差基本上维持在 0.16 米以内, 在 0.09 米上下波动。综上所述, 可以得出以下结论, 在复杂环境下 Cartographer 算法地图定位的精度优于 RBPF 算法。

将记录的 Cartographer 和改进 RBPF 的 20 组偏差数据做平均, 得出表 5-8。

表 5-8 Cartographer 和改进 RBPF 地图定位偏差平均值 单位(米)

SLAM 算法	RBPF	Cartographer
平均值	(0.0865,0.027)	(-0.0015,0.0205)
标准差平均值	0.0932	0.0460

从表 5-8 中可以得出与图 5-8 绘制的折线图相似的结果, 改进 RBPF 地图定位偏差大约是 Cartographer 地图定位偏差的两倍。

表 5-8 的平均值中, Cartographer 的 x 轴的偏差远远小于 RBPF 的 x 轴偏差, 这是由于在 Cartographer 在机器人 x 轴方向的地图定位偏差有正有负, 说明在此实验中, Cartographer 算法不存在累积误差或累积误差极小。

与第二组实验中的 Cartographer 地图定位偏差的平均值相比, 第三组实验表明, 复杂环境下, Cartographer 算法对机器人的累积误差有校正的功能, 在驱使机器人到达目标点时, 使机器人的位置一直位于目标点实际位置的周围, 且偏差较小; 而改进 RBPF 算法在复杂环境下的偏差的平均值与机器人在长路径下的偏差的平均值相差不多, 说明改进 RBPF 算法对机器人的累积误差也存在校正, 但效果不如 Cartographer 算法好。

(4) 三组实验对比分析。

三组实验中的数据结果对比如表 5-9 和表 5-10 所示。

取三组实验中每组误差数据的平均值, 绘制改进 RBPF 和 Cartographer 地图定位的误差的对比, 如表 5-9。

表 5-9 三组实验中改进 RBPF 和 Cartographer 地图定位误差平均值 单位(米)

SLAM 算法	RBPF	Cartographer
第一组实验(短路径)	(0.0577,0.0362)	(0.0291,0.0391)
第二组实验(长路径)	(0.0875,0.0125)	(0.0335,0.016)
第三组实验(复杂环境)	(0.0865,0.027)	(-0.0015,0.0205)

在复杂环境下, Cartographer 算法的地图定位偏差的平均值小于前两组实验中 Cartographer 算法的地图定位偏差的平均值, 说明 Cartographer 算法在复杂环境下有较好的适应性。

取三组实验中每组误差数据的平方差的均值, 绘制单次改进的 RBPF 和 Cartographer 地图定位的误差的对比, 如表 5-10 所示。

表 5-10 三组实验中改进 RBPF 和 Cartographer 地图定位误差的平方差均值 单位(米)

SLAM 算法	RBPF	Cartographer
第一组实验(短路径)	0.0330	0.0200
第二组实验(长路径)	0.1009	0.0778
第三组实验(复杂环境)	0.0932	0.0460

从上面的三组实验中可以发现改进 RBPF 算法和 Cartographer 算法在不同的实验条件下, 它们的各自的地图定位精度的表现也各不相同。从表 5-9 和 5-10 中可以得到以下五个结论。

结论一: 在短路径下, RBPF 和 Cartographer 算法的地图定位偏差都较小, 在

4 厘米以下，必要时可以忽略不计。

结论二：短路径环境和长路径环境下，改进的 RBPF 和 Cartographer 两种算法的定位偏差分别相差 1 厘米和 2 厘米，此时，Cartographer 算法与改进的 RBPF 算法的地图定位偏差几乎相同。

结论三：Cartographer 算法在第二组长路径实验中的误差值远高于第一组和第三组实验的误差值，说明当机器人向前行驶较长路径时，Cartographer 算法匹配不到足够的特征点与地图进行匹配，进而导致地图定位的偏差增大。所以在实际中做导航定位时应避免使机器人一直以直线的方式前进。

结论四：在复杂环境下，改进的 RBPF 算法的地图定位偏差与长路径下的地图定位偏差相差不多，表示改进 RBPF 算法地图定位中对累积误差的校正有限。

综上所述，证明了改进的 RBPF 算法在短路径和长路径环境下的精度相对比较精确，Cartographer 算法在复杂环境中的建图定位具有明显的优越性。结合第二章中的仿真实验结果，建议在建图时优先选择改进的 RBPF 算法，小型且结构较单一的环境中改进的 RBPF 算法与 Cartographer 算法均可以用来实现定位，复杂环境中优先选择 Cartographer 算法进行定位导航。

5.4 本章小结

本章中选取了两处 SLAM 实验的环境，并以 Cartographer 算法为例，分析了 SLAM 算法的数据关联过程，并在启动后查阅了 `cartographer_node` 发布和订阅的话题，显示一切正常运行。

本章通过三组实验，实现移动机器人在 Cartographer 算法和改进 RBPF 算法所建地图上的定位偏差的对比分析，实验中分别对短路径、长路径以及复杂环境中的 Cartographer 算法和改进 RBPF 算法的地图定位偏差进行了详细的数据图表及折线图的分析，以此计算了 Cartographer 算法和改进 RBPF 算法两种算法所建地图的精度。从实验中得出了，改进的 RBPF 算法在短路径和长路径下的定位较精确，以及 Cartographer 算法在复杂环境中的定位具有优势的结论。

结 论

本文首先介绍了关于移动机器人以及 SLAM 理论及成果研究的现状及发展, 研究了当前流行的 Cartographer 算法和 RBPF 算法的基本原理和实现流程, 并使用数据集对这两种算法进行了仿真。本文分析了移动机器人的运动模型及其计算轨迹的方法, 设计实验采集了动态窗口法和两轮差速法在现有机器人平台上的积累误差, 对实验结果进行了分析和对比, 采取了合适的运动模型计算及机器人的运动轨迹。本文搭建了以移动机器人为基础的硬件平台, 小型计算机 Jetson TK1 作为车载主机, 笔记本电脑作为可视化平台, 采用 Hokuyo 激光雷达和里程计实现 SLAM 建图与导航, 部署了基于 ROS 的分布式系统, 研究了 SLAM 的两种算法在 ROS 系统上的实现过程, 设计实验分析了 Cartographer 算法和改进 RBPF 算法两种算法的优势、缺陷和较适用场景。证明了改进的 RBPF 算法在建图速度上具有优势, 以及在短路径和长路径的环境下的定位较精确, Cartographer 算法在复杂环境中的定位具有一定优势的结论。

本文在 Cartographer 和 RBPF 两种算法的理论研究的基础上, 以搭建实验平台, 得出实验结果, 并对结果进行分析研究为重点, 主要进行了以下几方面的工作:

(1) 在研究基于粒子滤波器的 RBPF 算法的基本理论的基础上, 针对其计算繁琐、实时性差等缺点, 提出了改进其实时性的方法。本文研究了 Cartographer 算法的推导过程和流程分析, 包括算法的前端 Submap 的创建, 匹配、优化, 以及后端算法对全局地图的匹配优化方法与流程。最后通过网络上开源的数据集对这两种算法进行了仿真分析, 结果证明了改进的 RBPF 算法的误差较小以及建图速度较快的结论, 在 500 米 \times 500 米的环境中, 改进的 RBPF 算法的误差与 Cartographer 算法的误差相差 2 厘米, 几乎可忽略不计, 选择改进的 RBPF 算法与 Cartographer 算法进行实验比较分析。

(2) 分析了本文中所使用的移动机器人的差速运动模型和轨迹计算的方法。设计实验证明了使用差速移动机器人计算轨迹的累积误差与使用动态窗口法计算机器人的累积误差相差不多, 动态窗口法的精度高于左右轮差速的方法, 选择动态窗口法进行移动机器人位姿计算, 作为实验中 SLAM 的输入的里程信息。

(3) 设计了基于 ROS 的移动机器人 SLAM 系统, 包括移动机器人模型的设计及

选择,系统结构的设计以及传感器的选择,以及基于 ROS 的 SLAM 系统中各个传感器数据的数据传输与关联的设计,SLAM 移动机器人实验平台的分布式系统部署等。实现了笔记本电脑的速度控制机器人移动、使用 SLAM 算法创建地图和导航、笔记本电脑远程监控地图过程等。

(4) 分别针对两种算法所建地图及其机器人在地图中定位的误差,分别使用改进的 RBPF 算法和 Cartographer 算法创建了会议室的环境地图,并且在此基础上设计了三组实验:短路径、长路径以及复杂环境下的地图定位的偏差实验,对三组实验中所得到的地图定位偏差数据进行了相互的对比分析,得出了 Cartographer 算法与改进的 RBPF 算法在三种环境下的特性与各自的优势,证明了 Cartographer 算法在复杂环境中的建图定位的优越性,结合改进的 RBPF 算法在建图速度上的优势,建议在建图时采用改进的 RBPF 算法,复杂环境中定位时采用 Cartographer 算法。

经过对实验过程、实验结果的分析,可以作出以下改进:

(1) 在 Cartographer 算法中可以引入 IMU,提高角度测量的精度,进一步提高建图定位的精度。

(2) 受实际环境所限,本文中所选实验环境都比较小,下一步应选取大型的、具有结构性特征的实验环境进行 SLAM 建图实验,这样更便于分析 Cartographer 算法在后端优化方面的优势。

参考文献

- [1] 孙博雅. 移动机器人 SLAM 技术[J]. 电子技术与软件工程, 2018(2): 95-95.
- [2] 宋楚轩. 室内移动机器人的定位导航技术[J]. 中国新通信, 2018, 20(02): 73.
- [3] 张长勇, 王兴财, 步亚, 等. 移动机器人搬运物料目标定位优化仿真[J]. 计算机仿真, 2018, 35(02): 257-261.
- [4] 李磊, 叶涛, 谭民, 等. 移动机器人技术研究现状与未来[J]. 机器人, 2002, 24(5): 475-480.
- [5] 赵航, 刘玉梅, 卜春光, 等. 扫地机器人的发展现状及展望[J]. 信息与电脑(理论版), 2016(12): 167-168.
- [6] Davison A J, Reid I D, Molton N D, et al. MonoSLAM: real-time single camera SLAM.[J]. IEEE Trans Pattern Anal Mach Intell, 2007, 29(6): 1052-1067.
- [7] Endres F, Hess J, Engelhard N, et al. An evaluation of the RGB-D SLAM system[C]. IEEE International Conference on Robotics and Automation. IEEE, 2012: 1691-1696.
- [8] 贺伟, 梁昔明. 未知环境中移动机器人 SLAM 问题的研究进展[J]. 微计算机信息, 2005(3): 179-180.
- [9] Yan X. Herding nerds on your table: Nerd Herder, a mobile augmented reality game[C]. Proceedings of 30th ACM Conference on Human Factors in Computing Systems, 2012: 1351-1356.
- [10] Gerkey B P. Sold!: Auction methods for multi-robot coordination[C]. IEEE Transactions on Robotics and Automation, 2002, 18(5): 758-768.
- [11] 季秀才, 郑志强, 张辉. SLAM 问题中机器人定位误差分析与控制[J]. 自动化学报, 2008, 34(3): 323-330.
- [12] 石杏喜, 赵春霞. 基于概率的移动机器人 SLAM 算法框架[J]. 计算机工程, 2010, 36(2): 31-32.
- [13] Kramer J, Scheutz M. Development environments for autonomous mobile robots: A survey[J]. Autonomous Robots, 2007, 22(2): 101-132.
- [14] Straszhheim T, Gerkey B, Cousins S. The ROS Build System [ROS Topics][J]. Journal of Science & Medicine in Sport, 2011, 18(2): 19-23.
- [15] Okada, Kei. ROS(Robot Operating System)[J]. Journal of the Robotics Society of Japan, 2012,

- 30(9): 830-835.
- [16] 陈卓, 苏卫华, 安慰宁, 等. 移动机器人 SLAM 与路径规划在 ROS 框架下的实现[J]. 医疗卫生装备, 2017, 38(2): 109-113.
- [17] 陈白帆, 蔡自兴, 袁成. 基于粒子群优化的移动机器人 SLAM 方法[J]. 机器人, 2009, 31(6): 513-517.
- [18] Dissanayake M W M G, Newman P, Clark S, et al. A solution to the simultaneous localization and map building (SLAM) problem[J]. IEEE Trans Ra, 2001, 17(3): 229-241.
- [19] Geiger A. Are we ready for autonomous driving? The KITTI vision benchmark suite[C]. IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2012: 3354-3361.
- [20] Montemerlo M, Thrun S, Roller D, et al. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges[C]. International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers Inc. 2003: 1151-1156.
- [21] 涂刚毅, 金世俊, 祝雪芬, 等. 基于粒子滤波的移动机器人 SLAM 算法[J]. 东南大学学报(自然科学版), 2010, 40(1): 117-122.
- [22] 王田苗, 陶永, 陈阳. 服务机器人技术研究现状与发展趋势[J]. 中国科学:信息科学, 2012, 42(9): 1049-1066.
- [23] 史兵, 段锁林, 李菊, 等. 基于无线传感器网络的室内移动灭火机器人系统设计[J]. 计算机应用, 2018(1): 284-289.
- [24] 余洪山, 王耀南. 基于粒子滤波器的移动机器人定位和地图创建研究进展[J]. 机器人, 2007, 29(3):281-289.
- [25] Montemerlo M, Thrun S, Roller D, et al. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges[C]// International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers Inc. 2003:1151-1156.
- [26] Em. O. Univ.-Prof. Dipl.-Ing. Dr. techn. Dr.hc. mult. Peter Kopacek. Development trends in robotics[J]. E & I Elektrotechnik Und Informationstechnik, 2016, 49(29): 36-41.
- [27] Hirai K, Hirose M, Haikawa Y, et al. The development of Honda humanoid robot[C]. IEEE International Conference on Robotics and Automation, 1998. Proceedings. IEEE, 1998(2): 1321-1326.
- [28] Mayhew D, Bachrach B, Rymer W Z, et al. Development of the MACARM - a novel cable robot

- for upper limb neurorehabilitation[C]. International Conference on Rehabilitation Robotics. IEEE, 2016: 299-302.
- [29] Manti M, Pratesi A, Falotico E, et al. Soft assistive robot for personal care of elderly people[C]. IEEE International Conference on Biomedical Robotics and Biomechatronics. IEEE, 2016: 833-838.
- [30] 安峰,陈强,查艳芳,等.基于 RGB_D 相机的视觉里程计设计与实现[J].电子测量技术,2018,41(03):139-142.
- [31] Deng H, Jia Y, Zhang Y. 3DRobot: automated generation of diverse and well-packed protein structure decoys[J]. Bioinformatics, 2016, 32(3): 378-387.
- [32] Kang B B, Lee H, In H, et al. Development of a polymer-based tendon-driven wearable robotic hand[C]. IEEE International Conference on Robotics and Automation. IEEE, 2016: 3750-3755.
- [33] Costa V, Duarte M, Rodrigues T, et al. Design and development of an inexpensive aquatic swarm robotics system[C]. Oceans. IEEE, 2016: 1-7.
- [34] Casasent D P. A modular real-time vision system for humanoid robots[J]. Proceedings of SPIE - The International Society for Optical Engineering, 2012, 8301(1): 29.
- [35] Krishnan A B, Aswath S, Udupa G. Real Time Vision Based Soccer Playing Humanoid Robotic Platform[J]. 2014: 1-8.
- [36] Sung J Y, Grinter R E, Christensen H I. Pimp My Roomba: designing for personalization[J]. Chi09 Acm, 2009: 193-196.
- [37] Bellifemine F, Poggi A, Rimassa G. Developing multi - agent systems with a FIPA - compliant agent framework[J]. Software Practice & Experience, 2015, 31(2): 103-128.
- [38] Jackson J. Microsoft robotics studio: A technical introduction[J]. Robotics & Automation Magazine IEEE, 2007, 14(4): 82-87.
- [39] Pickem D, Glotfelter P, Wang L, et al. The Robotarium: A remotely accessible swarm robotics research testbed[C]. IEEE International Conference on Robotics and Automation. IEEE, 2017: 1699-1706.
- [40] Sadedel M, Yousefi-Koma A, Khadiv M, et al. Adding low-cost passive toe joints to the feet structure of SURENA III humanoid robot[J]. Robotica, 2016: 1-23.
- [41] Feng S, Whitman E, Xinjilefu X, et al. Optimization based full body control for the atlas robot[C]. Ieee-Ras International Conference on Humanoid Robots. IEEE, 2014: 120-127.

- [42] 张鹏飞, 何克忠, 欧阳正柱, 等. 多功能室外智能移动机器人实验平台——THMR-V[J]. 机器人, 2002, 24(2): 97-101.
- [43] 张国良, 汤文俊, 敬斌, 等. 基于机器人运动模型的 EKF-SLAM 算法改进[J]. 计算机测量与控制, 2012, 20(4): 1064-1066.
- [44] Watanabe S, Hajima T, Sudo K, et al. MIROC-ESM 2010: model description and basic results of CMIP5-20c3m experiments[J]. Geoscientific Model Development, 2011, 4(2): 845-872.
- [45] 陆政, 韩昊一, 纪良, 等. 基于 ROS 的 UR 机器人离线编程系统设计与开发[J]. 计算机仿真, 2017, 34(9): 309-313.
- [46] 刘基余. GPS 卫星导航定位原理与方法[M]. 科学出版社, 2003:20-25.
- [47] Ganskopp D. Manipulating cattle distribution with salt and water in large arid-land pastures: a GPS/GIS assessment[J]. Applied Animal Behaviour Science, 2001, 73(4): 251-262.
- [48] Smith, R.C. and P. Cheeseman, On the Representation and Estimation of Spatial Uncertainty. International Journal of Robotics Research, 1986. 5(4): 56--68.
- [49] 林志林, 张国良, 王峰, 等. 一种基于 VSLAM 的室内导航地图制备方法[J]. 电光与控制, 2018(1):98-103.
- [50] 简毅, 高斌, 张月. 一种室内扫地机器人全遍历路径规划方法研究[J]. 传感器与微系统, 2018(1):32-34.
- [51] 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.
- [52] Hess W, Kohler D, Rapp H, et al. Real-time loop closure in 2D LIDAR SLAM[C]. IEEE International Conference on Robotics and Automation. IEEE, 2016: 1271-1278.
- [53] 高翔, 张涛. 视觉 SLAM 十四讲[M]. 北京: 电子工业出版社, 2017:70-78.
- [54] Thrun S. Probabilistic robotics[J]. Communications of the Acm, 2005, 45(3): 52-57.
- [55] Himstedt M, Frost J, Hellbach S, et al. Large scale place recognition in 2D LIDAR scans using Geometrical Landmark Relations[C]. Ieee/rsj International Conference on Intelligent Robots and Systems. IEEE, 2014: 5030-5035.
- [56] Elfes A. Using occupancy grids for mobile robot perception and navigation[J]. Computer, 2002, 22(6): 46-57.
- [57] Mejri S, Gagnol V, Le T P, et al. Dynamic characterization of machining robot and stability analysis[J]. International Journal of Advanced Manufacturing Technology, 2016, 82(1-4):

- 351-359.
- [58] Fortino G, Guerrieri A, Lacopo M, et al. An Agent-Based Middleware for Cooperating Smart Objects[C]. International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer Berlin Heidelberg, 2013: 387-398.
- [59] 王依人, 邓国庆, 刘勇, 等. 基于激光雷达传感器的 RBPF-SLAM 系统优化设计[J]. 传感器与微系统, 2017, 36(9):77-80.
- [60] 曹天扬, 蔡浩原, 方东明, 等. 基于视觉内容匹配的机器人自主定位系统[J]. 光电工程, 2017, 44(05): 523-533.
- [61] 孙曼晖, 杨绍武, 易晓东, 等. 基于 GIS 和 SLAM 的机器人大范围环境自主导航[J]. 仪器仪表学报, 2017, 38(3): 586-592.
- [62] 蔡云飞, 唐振民, 赵春霞. 一种改进的多机器人协作实时 FastSLAM 算法[J]. 计算机研究与发展, 2012, 49(4): 763-769.
- [63] 罗元, 余佳航, 汪龙峰, 等. 改进 RBPF 的移动机器人同步定位与地图构建[J]. 智能系统学报, 2015(3): 460-464.
- [64] 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.
- [65] Gasparini M. Markov chain monte carlo in practice [M]. Markov chain monte carlo in practice. Chapman & Hall, 1996: 9236-9240.
- [66] Kümmerle R, Steder B, Dornhege C, et al. On measuring the accuracy of SLAM algorithms[J]. Autonomous Robots, 2009, 27(4): 387.
- [67] 李建勇, 刘雪梅, 李雪霞, 等. 基于 ROS 的开源移动机器人系统设计[J]. 机电工程, 2017, 34(2): 205-208.
- [68] 范永, 谭民. MRCS 中机器人控制体系框架结构[J]. 控制与决策, 2000, 15(3): 325-328.
- [69] 宋鑫坤, 陈万米, 徐昱琳, 等. 非结构化场景下移动机器人 FastSLAM 应用研究[J]. 计算机技术与发展, 2010, 20(2): 95-98.
- [70] 白亮亮, 平雪良, 陈盛龙, 等. 分布式移动机器人控制系统设计与实现[J]. 机械设计与制造, 2015(10): 180-183.
- [71] Cantelli L, Mangiameli M, Melita C D, et al. UAV/UGV cooperation for surveying operations in humanitarian demining[C]. IEEE International Symposium on Safety, Security, and Rescue Robotics. IEEE, 2013: 1-6.

攻读硕士学位期间承担的科研任务与主要成果

（一）参与的科研项目

- [1] 基于 MEMS 技术的无人机航姿估计及目标跟踪算法研究.燕山大学基础研究专项课题（16LGA007）,2016.10-2019.9 负责人：金梅

（二）发表的学术论文

- [1] 金梅, 姬少英, 张立国,等. 基于惯性跟踪的手臂运动及脑波一致性分析[J]. 高技术通讯, 2017(7):646-656.

致 谢

随着研究生生活的结束，我也即将告别这所我待了 7 年的学校，告别这座待了 7 年的城市。在秦皇岛市的燕山大学里，我结识了很多，也告别过很多人，有过得意与欢笑，也有过失意与泪水，我始终记得在我最彷徨失措的时候，是这所学校给予了我依靠与力量，让我重整旗鼓，再度前进。如今我也将告别这所学校，在此，我向所有关心我的老师和同学致以最诚挚的谢意！

首先要感谢的就是我的导师金梅老师和张立国老师，是他们给予我学业上的引导和关怀。他们在我的三年研究生生活中引导我学习的方向，特别是在论文帮助我，不断地督促我，监督我，并且帮助我，使我能按时完成毕业论文。在他们的帮助下，我自己在各个方面的水平都有了很大的提高，在此，我向他们致以我最衷心的感谢。

其次要感谢本专业的各位老师，感谢他们对我学业上的教导和生活上的帮助，感谢他们将知识和经验毫无保留地传达给我。感谢实验室的师兄和师弟师妹们，也感谢王松、张伟亚、郭文龙等我同届的同学，感谢在实验室的陪伴与交流指导，我从他们身上学习到了很多。感谢我的室友冯影、杨雪莹，感谢刘婷婷、刘玲妃她们在生活上对我的包容与照顾。

最后要感谢的是我的父母，感谢他们对我的养育与包容，感谢他们一直在我的求学之路上给予的理解与支持。父母之爱子，则为之计深远，他们一直以来默默的付出与不求回报，深深地影响着我，一直鞭策着我不断前行。

再次对关心我、爱护我的人致以我真诚的祝愿与感谢！