東北大學 NORTHEASTERN UNIVERSITY

硕士学位论文

THESIS FOR MASTER'S DEGREE

论文题目 基于分布式计算的3D SLAM研究

作　　者 赵家兴
学　　院 信息科学与工程学院
专　　业 模式识别与智能系统
指导教师 吴成东 教授

备　　注

二〇一六年十二月

分类号_____ 密级 _____

UDC _____

学 位 论 文

基于分布式计算的 3D SLAM 研究

作 者 姓 名 ： 赵家兴

指 导 教 师 ： 吴成东 教授

东北大学信息科学与工程学院

申请学位级别： 硕士 学 科 类 别 ： 工学

学科专业名称： 模式识别与智能系统

论文提交日期： 2016 年 12 月 论文答辩日期： 2016 年 12 月

学位授予日期： 2017 年 1 月 答辩委员会席： 魏颖

评阅人：王晓哲 、李孟歆

东 北 大 学

2016 年 12 月

A Thesis in Pattern Recognition and Intelligent System

Research on 3D SLAM Based on Distributed Computing

By Zhao Jiaxing

Supervisor: Professor Wu Chengdong

Northeastern University

December 2016

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：赵家兴

日　　期：2016年12月

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 □　　一年□　　一年半□　　两年☑

学位论文作者签名：赵家兴　　导师签名：吴成东

签字日期：2016年12月　　签字日期：2016年12月

摘 要

机器人 SLAM 通常采用激光雷达、声呐等传感器，其获取的信息量较少并且形式单一，很难适应复杂的环境，而且目前 SLAM 技术大多采用集中式架构，存在计算量大、复杂度高、容错性能差的问题。与此同时，传统的机器人开发模式针对每一项具体任务进行事先编程，经济成本高昂且机器人不能共享资源，严重障碍了智能机器人的发展应用。将机器人与云计算相结合，把云服务器端作为开发平台，机器人可以从云服务器平台获取相关知识，在拓展能力范围的同时极大地降低开发成本。本文研究基于分布式计算的机器人 3D SLAM，依托云计算平台实现移动机器人在室内环境的 3D 构图，具有积极的科学意义和应用价值。

针对室内环境 3D SLAM 算法，本文采用特征方法建立了算法流程。改进了误匹配算法，利用词袋模型解决了闭环检测问题。为了提高算法的执行效率，同时减少传输压力，本文采取了关键帧策略。为提高建图效率，解决 SLAM 集中式架构缺点，提出基于分布式文件系统（HDFS）的 SLAM 架构。在移动机器人导航方面，本文建立了移动机器人动态模型，研究了蒙特卡罗定位算法、A*路径规划算法、局部避障的动态窗口算法等。为避开 3D 障碍物，提出了 2.5D 地图模型，通过改进 A*算法，可以快速地寻找出一条最优路径，并应用于实际的室内机器人导航。

为验证本文理论研究的效果，基于 Kinect、ROS 与 iRobot Create 机器人构建了实验系统，依次对移动机器人的室内自主三维建图、避开 3D 障碍物、局域网的实时 3D SLAM 和云端分布式架构构图进行测试。实验结果表明，本文设计的室内移动机器人三维建图、自主导航及分布式计算方案具有良好的效果，可以快速地构建环境的三维地图，充分展现了分布式计算的 3D SLAM 的优势。

关键词：机器人 3D SLAM；Kinect 相机；分布式计算；云计算；3D 避障

Abstract

Robot SLAM usually uses laser radar, sonar and other sensors, the amount of information to obtain less and a single form.It is difficult to adapt to complex environments. There are computational complexity and fault-tolerant performance Poor problem due to most of the current SLAM technology using centralized architecture.At the same time, the traditional robot development model is pre-programmed for each specific task. The economic cost is high and the robot can not share the resources which hinders the development and application of the intelligent robot. The combination of robot and cloud computing that the cloud server-side as a development platform, the robot can be obtained from the cloud server platform, knowledge, ability to expand the scope of the same time greatly reduce development costs.In this thesis, 3D SLAM based on distributed computing is studied and realize the 3D composition of mobile robot in the indoor environment by relying on the cloud computing platform, which has positive scientific significance and application value.

Aiming at the 3D SLAM algorithm in indoor environment, this thesis uses the feature method to build the algorithm flow. In order to improve the efficiency of the algorithm, meanwhileand,to reduce the transmission pressure, a key frame strategy is adopted to solve the closed-loop detection problem by using the bag model. In order to improve the efficiency of drawing and solve the shortcomings of SLAM centralized architecture, this thesis proposes a SLAM architecture based on distributed file system (HDFS). In the aspect of mobile robot navigation, this thesis establishes the dynamic model of mobile robot, and studies Monte Carlo localization algorithm, A * path planning algorithm, dynamic window algorithm of local obstacle avoidance and so on. In order to avoid the 3D obstacle, a 2.5D map model is put forward. Through the improved A * algorithm, an optimal path can be found quickly and applied to the actual indoor robot navigation.

In order to verify the effectiveness of this thesis, an experimental system based on Kinect, ROS and iRobot Create robot was constructed. In order to validate the 3D model of mobile robots, 3D automatic 3D SLAM and 3D distributed architecture are proposed. The experimental results show that the three-dimensional mapping, autonomous

navigation and distributed computing of the indoor mobile robot are effective and can be used to construct the 3D map of the environment, which can fully show the advantages of 3D SLAM in distributed computing.

Key words: Robot 3D SLAM; Kinect Camera; Distributed Computing; Cloud Computing; 3D Obstacle Avoidance

目 录

独创性声明 I
摘 要 II
Abstract III
第 1 章 绪 论 1
1.1 研究背景与意义 1
1.2 国内外研究现状 2
1.2.1 国外研究现状 2
1.2.2 国内研究现状 4
1.3 主要研究内容及组织结构 5
1.3.1 论文研究内容及文章节安排 5
第 2 章 室内环境下视觉 3D SLAM 算法研究 7
2.1 视觉 3D SLAM 算法概述 7
2.2 成像与针孔摄像机模型 8
2.2.1 成像原理 8
2.2.2 针孔摄像机模型 9
2.2.3 深度相机工作原理 11
2.3 特征点提取与匹配 12
2.3.1 SIFT 算法 12
2.3.2 SURF 算法 16
2.3.3 特征点匹配算法及改进 18
2.3.4 实验结果及分析 20
2.4 位姿估计 22
2.4.1 姿态估计分类 22
2.4.2 2D 到 3D 位姿估计 22
2.5 闭环检测与全局优化 24

2.5.1 BoVW 模型与闭环检测 24
2.5.2 图优化 27
2.6 实验结果及分析 29
2.7 本章小结 30
第 3 章 移动机器人导航算法研究 31
3.1 机器人动态模型建立 31
3.2 导航地图表达 33
3.2.1 Cost_2d 地图 34
3.2.2 Octomap 地图 36
3.2.3 2D 地图与 3D 地图融合的 2.5D 地图 37
3.2.4 基于粒子滤波的定位算法 39
3.3 路径规划研究概述 41
3.3.1 全局路径规划 41
3.3.2 Dijkstra 算法 42
3.3.3 A*路径规划算法 42
3.3.4 基于蚁群算法的改进 A*算法 43
3.3.5 实验结果与分析 47
3.4 局部避障路径规划 48
3.4.1 机器人局部避障的动态窗口算法 48
3.4.2 实验结果与分析 50
3.5 本章小结 51
第 4 章 基于分布式架构的云计算系统 53
4.1 云计算 53
4.1.1 基本概念与分类 53
4.1.2 云计算的优势 54
4.2 基于分布式架构的云计算系统 54
4.2.1 基于分布式架构的云计算系统架构设计 55
4.2.2 ROS 操作系统及消息机制 56
4.2.3 HDFS 集群 56

4.2.4 MAP/Reduce 框架 57
4.3 关键帧选取策略 58
4.3.1 关键帧选取过程 58
4.3.2 实验结果与分析 59
4.4 基于 Socket 的网络传输机制 59
4.4.1 OSI 七层网络通信模型 59
4.4.2 TCP 数据传输机制 61
4.5 本章小结 62
第 5 章 移动机器人系统实验及分析 63
5.1 基于 ROS 的移动机器人平台 63
5.1.1 移动机器人软件平台 63
5.1.2 移动机器人硬件平台 63
5.1.3 移动机器人系统总体架构 64
5.2 基于 ROS 移动机器人导航系统 65
5.2.1 系统坐标变换 66
5.2.2 Navigation 栈构架 67
5.2.3 实验结果及分析 68
5.3 室内机器人导航实际实验 68
5.3.1 移动机器人自主三维建图实验 68
5.3.2 移动机器人的 3D 避障实验 70
5.4 基于分布式架构的视觉 SLAM 实验 71
5.4.1 基于分布式机制的局域网实时视觉 SLAM 实验 71
5.4.2 基于分布式架构的云计算视觉 SLAM 实验 72
5.5 本章小结 73
第 6 章 总结与展望 75
6.1 总结 75
6.2 展望 75
参考文献 77

致谢 81

第1章 绪 论

1.1 研究背景与意义

在发生矿难、地震等灾害的救援任务中，往往会形成高温、毒气、坍塌等危险环境，使得救援人员难以进入。移动机器人可以取代救援人员执行相关任务，减少人员伤亡，周围环境的3D信息不但有助于机器人更清楚地了解自身所处的环境信息，而且可以实现与环境的交互。在过去的十年中，伴随计算机计算能力的增强与计算机视觉的发展，用视觉传感器采集环境信息加之图像处理技术,已成为同时定位与建图（SLAM）[1]的重要技术。2010 年微软推出的 Kinect 深度相机不仅可以获取环境的纹理信息，而且还可以获取环境的深度信息[2]，其低廉的价格、更高的频率、更小的重量使其在机器人领域得到了广泛的应用。近年来，机器人发展极为迅速，但因为智力程度低、成本高等问题遏制了其推广应用。而且完全依赖本地的计算资源进行实时求解，会导致计算开销大、执行速度慢。伴随云计算技术[3]的日益成熟稳定，把云计算技术和机器人学结合起来，将是攻克以上难题的思路，而且目前已成为机器人领域的研究热点。

在自主定位和地图构建等自主移动机器人应用领域，激光雷达传感器在距离测量、障碍检测精度上具有显著的优势，曾经在自主导航系统之中[4]得到了广泛应用。然而，激光雷达的体积和价格针对大多数机器人应用场合来说是不易接纳的。目前，惯性导航技术相对成熟，由于其微惯性测量单元（ MIMU）体积小、重量轻、价格相对低廉，已经被普遍应用于自主导航系统之中 [5]。但是，由于其航位递推解算方式会导致误差不可避免地积累。与激光雷达、MIMU 等相比，视觉传感器可以廉价得得到更丰富的环境信息。基于视觉传感器导航的研究方向主要分为单目、双目及多目方向。单目相机帧序列难以计算物体的深度信息，导致场景深度的尺度模糊，这已然成为单目视觉导航系统[6]面对的的难题。而双目以及多目视觉系统，不仅体积、质量相对较大，而且由于其复杂的标定工艺，导致造价昂贵。微软 Kinect 深度相机的出世，克服了单目、双目等相机中存在的深度难以获得、体积大、质量重、价格昂贵等一系列问题[7]。

云机器人[8]是云计算与机器人学的结合，机器人本体不但不必具备超强计算的能力而且不需要存储所有资料信息，只是在需要的时候通过网络去连接相关服务器获得所需信息。作为一个全新概念在机器人应用领域，云机器人的重要性是能够提高机器人的自我学习能力并减少机器人的研发周期。将视觉 SLAM 部分运算分布到云计算平台，对于机器人领域是个崭新富有挑战的课题，对机器人领域的发展具有重大意义。云机器人架构模型如图 1.1 所示。

图 1.1 云机器人架构模型
Fig 1.1 Cloud robot architecture model

本课题针对分布式计算的视觉 SLAM 问题开展研究。在云计算、ROS、机器人软硬件平台支持下，结合计算机视觉理论和机器人概率学理论，研究在室内未知环境下基于图像特征点的视觉 SLAM 及移动机器人自主导航问题，最终设计和实现基于分布式计算的视觉 SLAM 的系统及移动机器人自主导航系统。

1.2 国内外研究现状

1.2.1 国外研究现状

(1) 云机器人研究现状与发展动态

1998 年，东京大学科学家提出的远程大脑(remote brain)的概念[9]是将机器人与外部计算机相连接，后来这一概念被云机器人概念更进一步深化，将与网络连接探求更加廉价的计算方法。在 Humanoids 2010 会议上，Google 人工智能科学家兼任卡耐基梅隆大学教授 JamesKuffner 创造的性提出了云机器人(cloud robotics)的概念[10],将机器人的信息存储在云端服务器上，并允许机器人在必要时通过互联网从云端服务器获取这些信息，引起广泛讨论。

自 2010 年提出云机器人概念之后，云机器人已成为国内外重要的研究方向。2011 年，RoboEarth 计划[11]被欧洲科学家启动，计划利用互联网建立一个庞大的开源数据库，让世界上的机器人可以轻松地访问和更新信息。图 1.2 展示了 RoboEarth 的四个机器人共享云端的多种知识源和数据源，在医院协作共同照顾病人。同年，法国 Aldebaran Robotics 公司研发的 NAO 机器人，在位于意大利的一家儿童医院里尝试依赖云设施实施视频监控、语音交互等功能，提高与患者的互动交互能力。新加坡 A-Star 公司的 ASORO(A-Star Social Robotics Laboratory)实验室正在研制的云计算基础设施，能呈现环

境 3D 地图并达到机器人迅速同步定位与地图构建(Simultaneous localization and mapping,SLAM)的目的[12]。

图 1.2 四个机器人在医院协作共同照顾病人

Fig 1.2 The four robots collaboratively working together to help patients in a hospital.

(2) 基于 Kinect 的视觉 SLAM 的研究现状与发展动态

在 Kinect 发布不久，华盛顿大学与微软实验室，开发了基于图优化算法—TORO[13,14]的视觉 SLAM（Simultaneous localization and mapping）系统。该 SLAM 方法通过用 SIFT 特征匹配方法，得到初始估计（当前帧相对于第一帧的位姿），然后使用 ICP[15]方法进行点云匹配来改善初始估计。德国 Freiburg 大学提出了 RGBD-SLAM 算法[16]，为了提高实时性使用了 Hog-man 图优化算法[17]，同时采用了 SURF 特征进行对应点匹配。因为图优化方法采用了全部可用信息，其精度比滤波器方式高很多。图 1.3 展示了 Freiburg 大学的室内 RGB-D SLAM 算法效果图。Felix Endres 等提出了一种利用 g2o[18](a General Framework for GraphOptimization) 图优化框架的 3D-SLAM 方法 。Engelhard N[19]提出了采用 Kinect 深度传感器生成物体和室内场景的彩色三维地图的 RGB-D SLAM 系统，该系统首先利用 RANSAC 算法[20]估计帧与帧之间的相对转换，接着采用改进的 ICP 算法来提高初始估计，最后使用姿态图来优化结果，从而输出一个全局一致感知环境的用彩色点云表示的三维地图。Kainz[21]提出了一种能够减少干扰的物理装置和校准过程方法，并扩展改进了 KinectFusion [22]算法的应用，使其能够同时操作多个 Kinect 设备，同时也克服了深度测量的系统误差。Chen[23]设计了一个基于 GPU 图像显卡存储有

效和分层的数据结构，这种结构支持具有精细几何细节的大规模场景实时 SLAM。2015 年，Felix Endres 等人[24]构建了基于 RGB-D 传感器的三维 SLAM 系统，采用从彩色图提取视觉要点的方法，利用深度图在三维空间定位这些关键点，然后使用 RANSAC 算法估计相关的关键点之间的转换，最后利用非线性最优化方法来优化位姿图。

图 1.3 Freiburg 大学的室内 RGB-D SLAM
Fig 1.3 The university of Freiburg indoor RGB- D SLAM

1.2.2 国内研究现状

(1) 云机器人研究现状与发展动态

与欧美、日本、新加坡等相比，国内对云机器人领域的研究并不多。北京理工大学的赵连翔等人[25]设计了机器人云操作平台，将机器人的资源和服务存储在云端，设计了包含资源层、管理层和服务层的 3 层结构，实现了机器人的遥控操作。山东大学陈宏兴、周风余等[26]创造了一种云服务机器人计算平台 SOA 接口层模型设计方法。2014 年，中国科学技术大学的陈小平教授团队与美国卡内基-梅隆大学伟罗莎（ManuelaVeloso）教授团队共同合作研究云机器人相关技术，通过云服务平台进行远程合作与知识共享实验。在实验中，位于中国合肥的中国科大“可佳”（KeJia）机器人与位于美国匹兹堡的卡内基-梅隆“可宝”（CoBot）机器人通过云服务平台，可佳向可宝提供自动规划服务，可宝向可佳提供大数据分析服务，基于这些知识共享，可佳与可宝最终合作完成了在室内指定位置取到矿泉水并送达指定“客人”手中的任务。图 1.4 展示了中科大与卡内基-梅隆的合作实验。

图 1.4 中科大与卡内基-梅隆的合作实验
Fig 1.4 USTC and Carnegie Mellon university cooperation experiment

（2）视觉 SLAM 的研究现状与发展动态

目前，国内专门针对视觉 SLAM 研究相对较少，关注点主要集中在基于激光雷达的 SLAM 技术。视觉 SLAM 的研究虽然在我国起步比较晚，但经过国内高校学者不断探索和研究，提出了新的方法和见解，在理论和实践研究上取得了积极成果。

南开大学张展宇学者[27]提出了利用 SURF[28]（Speeded up Robust Feature）来进行路标的检测与匹配，提高系统的速率及匹配的稳定性。同时，还优化了 SLAM 地图中特征地标的管理策略和检测方法，解决了被遮挡特征不能准确进行观测的问题。陈晓明分[29]析了基于 Kinect 获取的深度信息进行 SLAM 技术研究，针对实现过程中深度数据含有的大量噪声问题，采用联合双边滤波器来解决深度信息的高噪声干扰问题，根据噪声特性改进了双边滤波算法，弥补了缺失的深度信息。

1.3 主要研究内容及组织结构

1.3.1 论文研究内容及文章节安排

本文研究基于分布式计算的视觉 SLAM 及机器人室内自主导航，涉及视觉 3D SLAM 算法、机器人定位算法、路径规划算法、避障算法、云计算、Kinect 深度相机、ROS 软件平台、iRobot Creat 机器人硬件平台等。其中在局域网与云端实现了 3D SLAM 算法及移动机器人能够进行自主探索未知环境。构建未知环境地图和避开 3D 障碍物，需要研究理论算法及其实现软件开发及系统硬件构建。

根据所研究的内容，论文章节安排如下：

第 1 章：阐述论文的研究背景、目的及意义，分析了国内外对云机器人、视觉 SLAM 问题的研究现状与发展趋势，以及本文研究内容及章节安排。

第 2 章：阐述室内环境下 3D 视觉 SLAM 的基本原理，重点研究了基于深度相机的稀疏特征点法的 3D SLAM 算法基本流程，如特征点检测与描述子生成、特征匹配、运动估计、闭环检测、全局姿态优化等，其中改进了匹配算及用词袋模型进行闭环检测。并做了实际实验验证算法效果。

第 3 章：在蒙特卡罗定位算法的框架下，对机器人进行建模，深入研究室内机器人精确自主导航的方法，包括导航地图表达、全局路径规划与局部避障路径规的实现，提出 2D 与 3D 地图融合的 2.5D 导航地图模型并改进了基于 A*的路径规划算法，并且做了仿真实验与实际实验进行算法可行性验证。

第 4 章：在分布式计算框架下，建立通信模型，阐述云计算的优势、数据由本地传上云端的网络传输机制、关键帧选取策略等，在此基础上构建基于分布式架构的云计算系统，将系统的 SLAM 算法分布到云平台上计算。

第 5 章：在基于 ROS 软件系统平台下进行自主移动机器人的总体设计与实现，开展实际实验，验证未知环境自主 3D 构图算法效果、避开 3D 障碍物并到达目的地效果与局域网实时 3D SLAM 与云端分布式 3D SLAM 效果。

第 6 章：对本论文的工作进行回顾和总结，并对课题的进一步研究做出展望。

全文结构如图 1.5 所示。

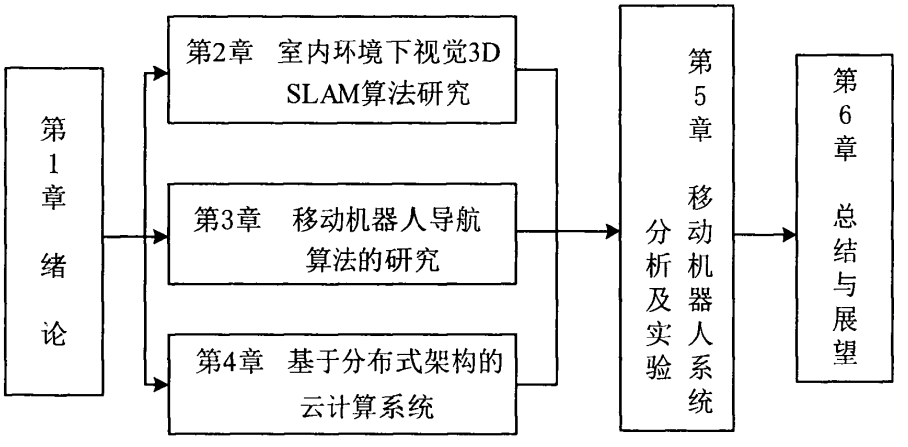


图 1.5 论文结构图

Fig. 1.1 Structure diagram of thesis

第2章 室内环境下视觉 3D SLAM 算法研究

室内环境的特点决定了 GPS 等技术，甚至在一些不具备明显位置标识和物理提示物的极端环境中，基于激光雷达的 SLAM 技术也失去了实用性。如何在这样的环境中快速准确地获取相机位姿信息，成为 SLAM 系统所需解决的一个关键科学问题。视觉 SLAM 的方法，是用相机作为传感器计算机器人的运动与构建地图。

2.1 视觉 3D SLAM 算法概述

在未知环境中，对自主机器人来说定位与建图是相辅相成的。准确的环境模型有助于实现机器人精确定位，而精确定位有利于建立更加完备的环境模型，因此，机器人的自主建图与定位是密不可分的，同时定位与建图（SLAM）已经成为机器人自主导航领域的热门主题之一。传统的 SLAM 一般是通过激光、声呐等进行环境信息获取，这些传感器获取的信息不仅在形式上显得十分单一，而且由于所获取的信息量不充分，致使机器人难以在复杂多变的环境中执行任务。视觉传感器的出现就恰好可以满足价格低廉、信息量大的要求，再加上计算机视觉的领域的飞速发展，基于视觉传感器的 3D 构图更加具备研究的价值。只依靠二维图像方式得到场景信息，因为缺乏真实的空间距离信息，难以真实地反映三维空间。现今，通常利用结构光取得场景的三维信息，传感器如微软的 Kinect 等。

3D SLAM 算法流程图如图 2.1 所示，具体算法步骤为：

(1) 依靠 RGB-D 相机所获取的 RGB 图，选择合适的特征提取算法进行特征点提取与描述；

(2) 根据 RGB-D 相机所获取彩色图对应的深度图，进行点云二次采样获取点云数据；

(3) 进行特征点匹配，考虑到特征点匹配过程中出现的一系列误匹配问题，引入 RANSAC 算法解决误匹配，并建立转换；

(4) 利用视觉词袋模型（BoVW）进行闭环检测，采用大闭环闭环检测方法，当前帧不仅和上一帧进行比较，还要和先前的帧比对，形成检测大闭环，进行全局优化。

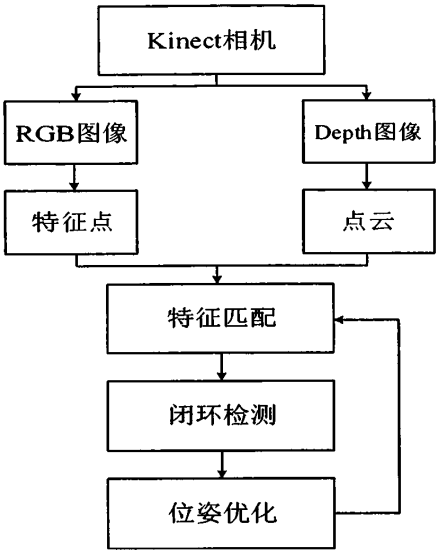


图 2.1 3D SLAM 算法研究框图

Fig 2.1 The diagram of 3D SLAM algorithm research

2.2 成像与针孔摄像机模型

2.2.1 成像原理

图像是由相机拍摄的真实场景形成的，成像原理是光线透过相机透镜投影到图像传感器上。图像是由 3D 场景到 2D 平面上，因此场景与它的成像之间存在重要的联系。来自场景的光线是通过正面孔径被相机捕获，光线击中位于相机背面的图像平面（或图像传感器）。此外，镜头是用来聚焦来自不同场景元素的光线。图 2.2 说明了这个过程。

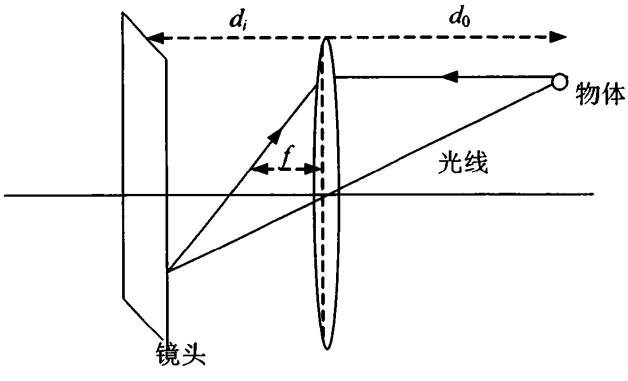


图 2.2 相机成像原理

Fig 2.2 The principle of camera imaging

在这里，d_0表示观察对象到镜头的距离，d_i表示镜头到图像平面的距离，f 表示镜头的焦距。这些量在薄透镜方程的关系如下：

$$\frac{1}{f}=\frac{1}{d_0}+\frac{1}{d_i} \tag{2.1}$$

2.2.2 针孔摄像机模型

在计算机视觉中，2.2.1 节的相机模型可以简化。首先，将相机的孔径视作无穷小，理论上忽略透镜的影响将不会对成像造成影响。因此只考虑中心位置的射线。其次由于大多数情况d_0远远大于d_i，可以假设图像平面位于焦距处。最后，系统的几何结构可以看出，平面上的图像是颠倒的。通过简单的将图像平面放置在镜头前，我们可以得到相同的笔直图像。这在物理上是不可行的，但从数学角度来看，完全等价，这个简化模型被称作针孔摄像机模型。它的表现形式如图 2.3 所示。

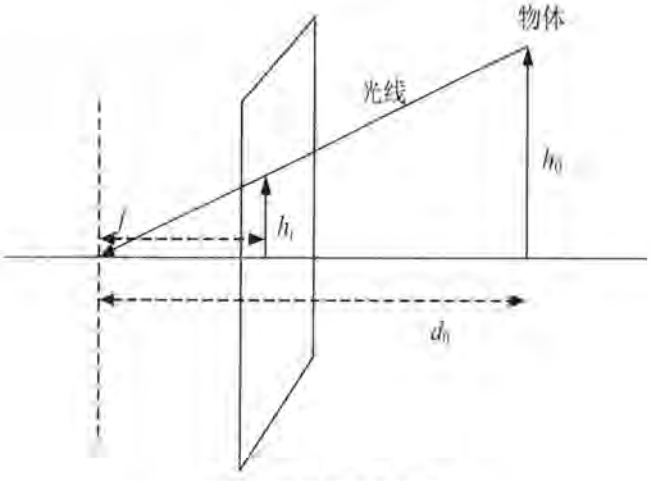


图 2.3 针孔摄像机模型

Fig 2.3 The model of Pin-hole camera

通过这个模型以及相似三角形的法则，我们可以推导出基本的投影等式：

$$h_i=f\frac{h_0}{d_0} \tag{2.2}$$

物体高度为h_0图像中的尺寸h_i，因此与它离真实相机的距离h_0成反比，这个关系使我们能够预测三维场景中点在图像中的位置。

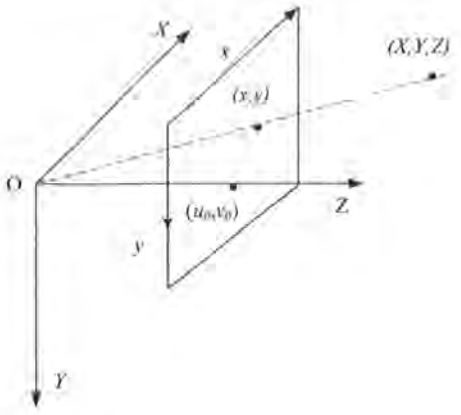


图 2.4 空间点与图像点映射关系

Fig 2.4 The mapping between the point in world space and the pixel in the image .

每一个完整的点云，由 r、g、b、x、y、z 一共 6 个分量组成，其中 r、g、b 代表其颜色信息，x、y、z 代表其空间位置。颜色信息主要由彩色图像纪录，空间位置信息由图像和相机模型联合计算出来。由图 2.4 知，空间点 $[x,y,z]$与它的像素坐标 $[u,v,d]$（d 指深度数据）之间的对应关系是：

$$u=\frac{x\cdot f_x}{z}+c_x \tag{2.3}$$

$$v=\frac{y\cdot f_y}{z}+c_y \tag{2.4}$$

$$d=z\cdot s \tag{2.5}$$

其中，f_x，f_y 指相机在 x,y 两个轴上的焦距，c_x，c_y 指相机的光圈中心（图像中心可能不在原点，c_x，c_y 表示 c 在图像平面上的位置），s 指深度图的缩放因子。通常，将 f_x，f_y，c_x，c_y 这四个参数定义为相机的内参矩阵 C。在给定内参之后，用下式（2.6）来描述每个点的空间位置与像素坐标的关系：

$$s\cdot\begin{bmatrix}u\\v\\1\end{bmatrix}=C\cdot\left(R\cdot\begin{bmatrix}x\\y\\z\end{bmatrix}+t\right) \tag{2.6}$$

其中，R 和 t 是相机的姿态，R 表示旋转矩阵，t 表示平移矢量。

将式 (2.6)表达成矩阵形式，即将世界坐标系中的点[x,y,z]映射到图像坐标系[u,v]。从三维立体空间到图像坐标系的坐标变化为：

$$s\cdot\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & u_0\\0 & f_y & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}R & t\\0^T & 1\end{bmatrix}\begin{bmatrix}x & y & z\end{bmatrix} \tag{2.7}$$

u_0 和 v_0 表示图像物理坐标系的原点在图像像素坐标系中的位置。

2.2.3 深度相机工作原理

2009 年 6 月，Microsoft 发布了“Project Natal”体感控制器，2010 年 Microsoft 正式将其更名为“Kinect”并推向市场。Kinect 以每秒 30 帧的速率得到图像流，同时取得周围环境的深度图像。通过结合红外 CMOS 传感器和彩色摄像头，可以将目标物的三维信息投放到屏幕当中。获取深度图像包括两个传感器，分别是红外线发射器和红外 CMOS 摄像头，红外发射器负责发射信号，红外 CMOS 摄像头负责接收信号。Kinect 实体如图 2.5 所示。

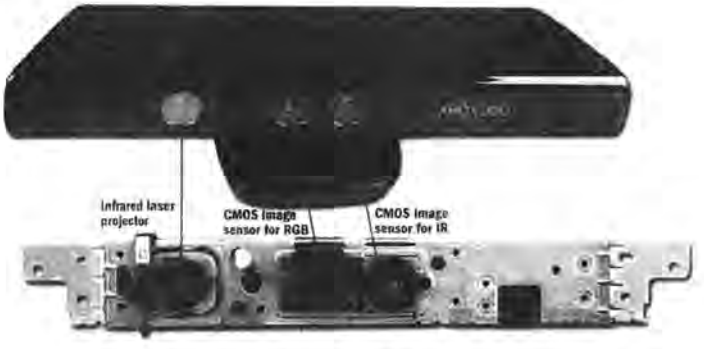


图 2.5 Kinect 深度相机

Fig 2.5 The depth camera of Kinect

Kinect 采用 Primesensor 的光编码(Light Coding)技术[30]，Light Coding 技术理论是凭借连续光（近红外线）对测量空间进行编码，通过感应器读入编码的光线，然后由芯片运算进行解码，最终取得有深度信息的图像。其具体方法是利用光源照明给需要测量的空间进行光学编码，使用的光源叫做激光散斑（激光照射到物体上呈现随机衍射斑点），激光散斑随机性很高，它会根据距离的差异变换不同的图案，因而空间中任意两处的散斑图像都不尽相同。一旦这般结构光被打到空间里，空间就相当于被做了标记，此刻将一个物体放进该空间，只需要测量物体表面的散斑图案，就能计算出该物体的位置。所以整个空间内的激光散斑图案需要在使用之前借助光源标定记录，光源标定的方法是通过在空间中选择参考平面，参考平面的数量可以根据场景的大小和复杂度来选择。首先在空间投射激光散斑，然后把空间中的

散斑图案都记下来，当需要对空间中的物体位置进行测量的时候，获取一幅待测量的散斑图案，与所有保存下来的参考图依次做互相关运算，得到一系列的相关度图像，而空间中出现物体的地方就会在相关图像上表现出峰值，把峰值叠加，再通过插值运算后就得到了整个场景的三维形状了。参考平面越密，测量越准确，精度就越高。

2.3 特征点提取与匹配

作为视觉 SLAM 系统中的特征点提取算法，需要具备下列特点：

(1) 稳定性

对于每一个特征点，特征提取算法应该提取出与尺度、旋转、光照无关的特征。

(2) 高效性

作为 SLAM 系统的组成部分的特征提取算法，应该能实时执行，尽可能的缩短执行时间，提高的执行效率。

(3) 鲁棒性

特征提取方法应具有较高的抗干扰能力，尽可能降低特征的误检率。

(4) 可区别性

特征提取算法应该赋予每个特征点必要的描述子，以区别与其它特征点的不同。

本文使用 Kinect 获取图像信息来提取特征点，基于以上原则，本文采用 SURF 提取图像中的特征点的方法，该方法在机器视觉与图像处理领域较为常用也相对成熟。为接下来的匹配工作中提高匹配的正确率，还要将提取出来的特征生成描述子。本文使用了 SURF 描述器来描述特征，其具有尺度不变性、旋转不变性、光照不变性等优点。

2.3.1 SIFT 算法

图像在拍摄时与目标物体距离的不同导致目标物体在不同视角有不同的尺寸，当尝试在不同图像之间匹配特征时，通常面临尺度变化的难题。这个难题可以通过尺度不变的特征，即每个特征点都关联着对应的尺寸因子来解决。

David.Lowe 在 2004 年提出了一种对图像平移等变换保持不变性的基于尺度空间的图像局部特征。这种方法被称为尺度不变特征变换（Scale Invariant Feature Transform），简称为 SIFT[31]算法。具体算法如下：

(1) 构建尺度空间

特征点的检测是一般是在多尺度空间完成的，为了保持特征具有尺度不变的

特性。

二维图像的尺度空间可以定义为：

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{2.8}$$

其中$G(x,y,\sigma)$是尺度可变高斯函数：

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.9}$$

(x, y)表示图像的像素位置，σ表示尺度空间因子。

高斯金字塔[32]就是对图像进行降采样同时使用高斯平滑所构建出来的，高斯金字塔模型如图 2.6 所示，其尺度系数与降采样系数也都是成倍增大的。高斯金字塔一般有 o 组（一幅图像生成 o 组图像）、s 层(一组图又包含了 s 层图像)，则有：

$$\sigma(s) = \sigma_0 \times 2^{s/S} \tag{2.10}$$

组内与组间尺度定义成：

$$2^{i-1}(\sigma, k\sigma, k^2\sigma, \ldots k^{n-1}\sigma), k = 2^{\frac{1}{s}} \tag{2.11}$$

其中，i 表示金字塔组的个数，n 表示每一组的层数。

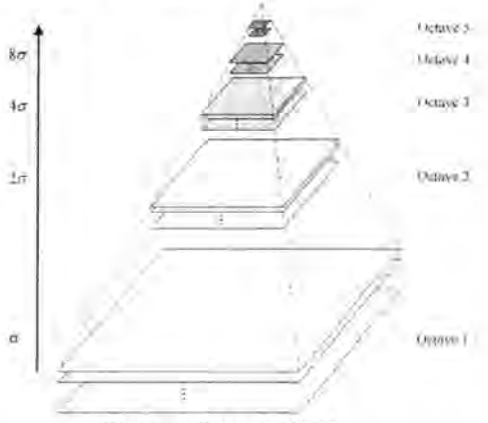


图 2.6 高斯金字塔模型

Fig 2.6 The model of Gauss Pyramid

尺度规范化的 LOG 算子：

$$\nabla^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \tag{2.12}$$

LOG 算子与高斯核函数的关系：

$$LOG(x,y,\sigma) = \sigma^2\nabla^2 G \approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{\sigma^2(k-1)} \tag{2.13}$$

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{2.14}$$

其中 k-1 是一个常数值，并不会影响到极值点的求解。这里不妨引入一种新的算子 DOG：

$$D(x,y,\sigma) = [G(x,y,k\sigma) - G(x,y,\sigma)] * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \quad (2.15)$$

LOG 算子与 DOG 算子的函数分布图如图 2.7 所示，DOG 算子在运算过程中在速度上更具有优势。

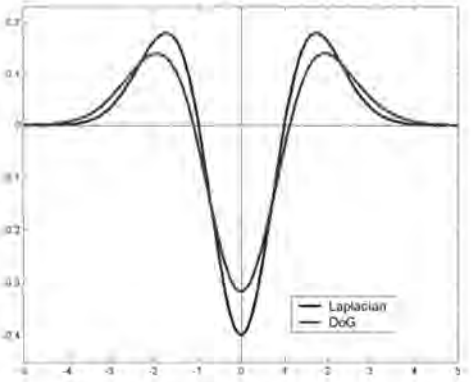


图 2.7 高斯拉普拉斯和高斯差分的比较

Fig 2.7 Comparison of Goss Laplace (LOG) and Gauss difference (DOG)

(2) 关键点定位

中心的检测点要和它 8 个相邻点以及上下相邻的 9×2 个点，一共 26 个点进行对比，选择灰度值最大的点作为候选点。然后将离散空间的候选极值点利用插值三维二次函数方法精准定位关键点的位置和尺度，而且还要继续利用近似 Harris Corner 检测器除掉低对比度的候选特征点及不稳定的边缘响应点。

(3) 关键点方向分配

通过为关键点分配参数方向，确保描述子对图像的旋转不变性。像素点的梯度表示为:

$$gradI(x,y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \quad (2.16)$$

梯度幅值：

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.17)$$

梯度方向：

$$\theta(x,y) = \tan^{-1}\left[\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right] \quad (2.18)$$

(4) 关键点描述

由统计的 4*4*8=128 个梯度信息作为该关键点的特征向量的构成。关键点描述子生成步骤如下：

① 图像区域的半径如下表示：

$$radius = \frac{3\sigma_{oct} \times \sqrt{2} \times (d+1)+1}{2}, d = 4 \tag{2.19}$$

σ_{oct}是关键点所在组的尺度。

② 将坐标转换成关键点主方向，如图 2.8 所示。

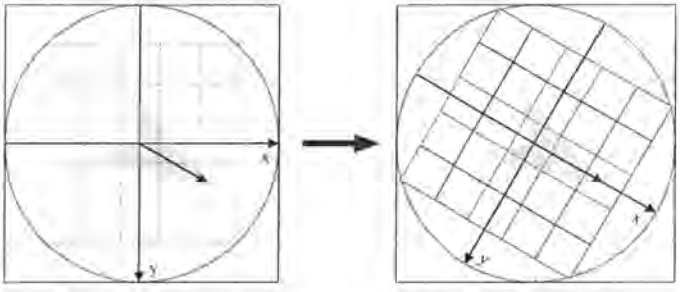


图 2.8 关键点方向迁移示意图

Fig 2.8 Schematic diagram of the key points in the direction of migration

旋转角度后新坐标为：

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.20}$$

③ 为了生成方向直方图，要求解每个像素点的梯度幅值和方向，然后再用幅度乘以高斯参数。

④ 首先获取每一个梯度方向的直方图，然后统计各自的累计值，最后就可以形成一个种子点。然后再在下一个区域进行之前步骤，一共需要生成 16 个种子点。

⑤ 归一化处理描述子向量。为了除掉光照的影响，须对特征向量进行归一化处理，由于图像每一点的梯度是邻域像素相减得到，因此图像灰度值的整体漂移也能除去。

SIFT 算法特征提取与描述子的生成如图 2.9 所示。

图 2.9 SIFT 特征提取与描述子生成

Fig 2.9 SIFT feature extraction and descriptor generation

2.3.2 SURF 算法

SURF[31](Speeded up Robust Feature)算法是对 SIFT 算法的一种改进算法，主要是在算法的实现效率上进行改进，改进后该算法比 SIFT 算法运行更快。在特征检测中，在构建尺度空间时 SURF 没有降采样过程，而且 SURF 描述子大部分基于强度的差值，计算更快捷，描述子维度是 SIFT 的一半，因此速度比 SIFT 快。为保证 SLAM 算法实时性，本文特征提取与描述采用 SURF 算法。

SURF 特征提取与特征描述子生成步骤如下：

(1) 构建 Hessian 矩阵

首先对每个像素求算 Hessian 矩阵，该矩阵测量一个函数的局部曲率特性，定义如下：

$$H\big(f(x,y)\big)=\begin{pmatrix}\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x\partial y}\\ \frac{\partial^2 f}{\partial x\partial y} & \frac{\partial^2 f}{\partial y^2}\end{pmatrix} \tag{2.21}$$

其中，H是由函数$f(z,y)$的偏导数组成，H矩阵判别式为：

$$\det(\mathrm{H})=\frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2}-\left(\frac{\partial^2 \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{y}}\right)^2 \tag{2.22}$$

$$H(x,\sigma)=\begin{bmatrix}L_{xx}(x,\sigma) & L_{xy}(x,\sigma)\\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma)\end{bmatrix} \tag{2.23}$$

核函数$G(t)$由式(2.25)表示，$g(t)$是高斯函数，该方法可以用来计算图像中每个像素的H的行列式值，最后可以使用该值来判别特征点。

$$L(X,t) = G(t) * I(X) \tag{2.24}$$

$$G(t) = \frac{\partial^2 g(t)}{\partial x^2} \tag{2.25}$$

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.26}$$

(2) 构建尺度空间

在 SURF 算法中，由于图像核的大小的增加，需要对尺度空间的多层次图像进行处理。在金字塔构建过程中，金字塔内图像大小保持不变，只改变滤波器的大小。

(3) 关键点方向分配

以特征点为中心，建立一个以 6s 为半径的圆形区域。以 60 度为一个单位，分别计算在 y，z 方向上的 Haar 小波，同时赋予高斯权重，使越靠近特征点的响应越贡献大，而越远离特征点的响应贡献越小。然后把扇形范围内的响应全部加和就可以得到新的矢量，最后选取矢量最长的方向作为特征的主方向。关键点方向分配示意图如图 2.10 所示。

图 2.10 关键点方向分配示意图

Fig 2.10 Schematic diagram of the key points in the direction of distribution

(4) 特征描述子生成：

将坐标旋转至关键点主方向，旋转角度后新坐标变成：

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.27}$$

描述子向量周围区域的尺寸设为尺度因子的 20 倍，即 20σ。将方形区划分成 4×4大小的子区域，计算每个5×5子区域的d_x响应和d_y响应（核的尺寸为 2σ），加和全部这些响应，提取每个子区域 4 个描述子的值：$[\sum d_x \quad \sum d_y \quad \sum|d_x| \ \sum|d_y|]$，由于存在$4 \times 4 = 16$个子区域，所以总共有 64 个描述子的值，为了赋予靠近的像素值更多的重要性，核响应按照中心位于特征点位置的高斯函数进行加权（$\sigma = 3.3$）。

获取每个梯度方向的直方图，然后统计各自的累计值，最后就可以形成一个种子点。然后再在下一个区域进行之前步骤，一共需要生成 16 个种子点。对描述子向量进行归一化处理。SURF 算法特征提取与描述子的生成如图 2.11 所示。

图 2.11 SURF 特征提取与描述子生成

Fig 2.11 SURF Feature extraction and descriptor generation

2.3.3 特征点匹配算法及改进

优质的描述子可以通过简单的距离测量进行比较（如欧式距离），两个特征点越相似，对应的特征向量也越接近。步骤主要为：起先提取每帧图像中的特征点，接着生成描述子向量，接下来用第一幅图像中的每个特征描述子向量与第二幅图像中的描述子向量进行比对，两个向量的欧式几何距离最近也就是得分较高的一对描述子将视为那个特征的最佳匹配。

目前广泛应用的特征匹配算法有穷举法匹配，BBF[33]查询法匹配和 KD-Tree[34]搜索法匹配。

穷举法也叫暴力匹配，是依次遍历两幅图像对应特征点，求出每个对应点的欧式几何距离。

BBF(Best Bin First)算法查询路径上的节点排序，为了确保优先检索包括最近邻节点，需要回溯优先级最高的节点。BBF 与此同时还设定了运行超时制约时间，当处于优先级队列中的全部节点都已经被搜索了，算法就会返回目标并且设置为最近邻节点。

KD-Tree 是一种二叉树数据结构，其中的每一个节点都象征着一个数据空间。设数据的维度是 N 维，在其中选择数据中方差最大的维度，将数据划分为左子空

间和右子空间。它的构建流程图如图 2.12 所示。

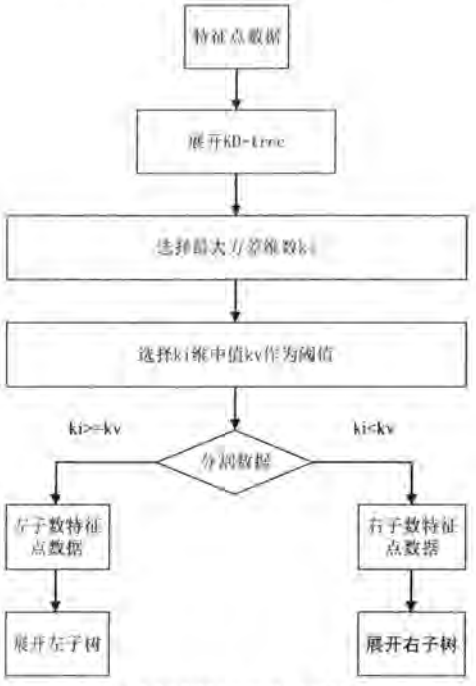


图 2.12 KD-Tree 构建流程

Fig 2.12 Building process of KD-Tree

为提高匹配速度，本文采用 KD-Tree 搜索法匹配方法。

在基于稀疏特征点的视觉 SLAM 中，特征匹配环节中常常会存在误匹配现象，这样将导致计算得到的位姿精度低，将造成位姿估计失败。所以，须要去除这些误匹配点。

本文采取两种算法融合的方法去除误匹配。首先采用最小比例法统计所有匹配对的匹配距离，计算其中的最小匹配距离然后将匹配距离大于最小匹配距离一定倍数的匹配对剔除。这样很大程度的减少了匹配点对的数量，但处理后仍然会留下一些错误的匹配。

RANSAC[22]算法（Random Sample Consensus,随机抽样一致）算法核心思想是使尽量多的点接近拟合的直线。假设观测数据既包括局内点也包括局外点，而且局内点在直线附近，局外点远离直线。由于最小二乘的方法是尽量适应包含局外点在内的全部点，所以在这个假设下，简单的最小二乘法不能找到仅仅适应于局内点的直线。但是，RANSA 算法能得出一个概率足够高的且能仅仅适用局内点的直线。

RANSAC 算法用于去除图像之间误匹配的原理是探寻一个矩阵大小为 3*3 的最佳单应性矩阵 H，致使符合该矩阵的数据点个数最多。其对应关系如下：

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.28}$$

其中，通常令 $h_{33}=1$，(x,y) 表示目标图像点位置，(x',y') 表示场景图像点位置，S 表示尺度参数。

算法步骤：

(1) 从匹配数据集中随意选出 4 个不共线样本，计算得出单应性矩阵 H，记为模型 M;

(2) 利用模型 M 对全部数据进行测试，并计算全部的数据与这个模型的投影误差，如若误差小于阈值则加入内点集 I;

(3) 如若当前内点集元素个数大于最优内点集 I_{best} 个数，则要令 $I_{best}=I$，同时也要更新迭代次数 k;

(4) 如迭代次数大于 k，则退出；否则，迭代次数要加上 1，之后并重复上述步骤。

最优模型要满足对应的代价函数最小，代价函数定义为：

$$\sum_{i=1}^{n}((x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}})^2 + (y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}})^2) \tag{2.29}$$

注：迭代次数 k 是不停更新而不是稳定不改变的，其公式为：

$$k = \frac{\log(1-p)}{\log(1-w^m)} \tag{2.30}$$

其中，P 表示置信度，一般取固定值 0.995，w 表示内点的比例，m 表示在计算模型时所需要的最少样本个数，此模型 m 取 4。

2.3.4 实验结果及分析

针对矿电楼 208 采集两帧图像，正常匹配效果如下图 2.13 所示。先采用最小比例法剔除误匹配，匹配效果如下图 2.14 所示。接下来用 RANSAC 算法进一步消除误匹配，最终的匹配效果如下图 2.15 所示。通过图 2.13、图 2.14 与图 2.15 比对，明显看出本文采用的最小比例方法与 RANSAC 算法相融合的算法，可以有效消除误匹配，提高匹配精度。

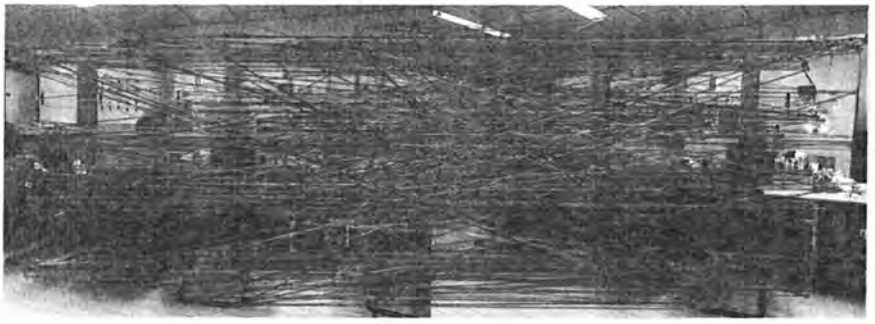


图 2.13 特征匹配

Fig 2.13 Feature matching

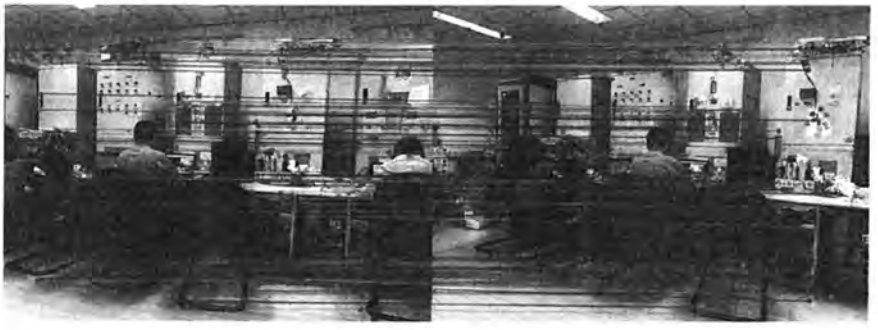


图 2.14 最小比例法消除误匹配

Fig 2.14 The least proportion method to eliminate the false match

图 2.15 利用 RANSAC 算法进一步消除误匹配

Fig 2.15 RANSAC algorithm to eliminate the error matching

表 1 特征点匹配信息

Table 1 Feature point matching information

特征	左图特征点个数	右图特征点个数	匹配点对	时间
SIFT 特征	962	905	240	1.849s
SURF 特征	992	967	158	0.752s

由表 1 可以看出，在匹配相同图像时，SURF 算法与 SIFT 算法相比，SURF 算法在速度方面较有优势，大约是 SIFT 的 2.5 倍；SURF 算法提取的特征点个数也多于 SIFT 算法提取的特征点个数；SIFT 算法匹配的点对数比 SURF 算法占优势。为保证 SLAM 算法的的实时性和地图精度本文采用 SURF 算法作为特征提取与匹配的算法。

2.4 位姿估计

姿态估计问题是指确定目标物体的方位指向问题。本文研究的主要是视觉姿态估计问题，该姿态可以通过一个旋转和平移变换从一个参考姿态变换到另一个姿势，这种旋转变换可以用不同的方式表示，例如一个旋转矩阵或四元数。

2.4.1 姿态估计分类

基于视觉的姿态估计问题，依据算法的差异一般划分为基于学习估计问题与基于模型的估计问题。

基于学习的估计依靠机器学习[35]的方式，从训练样本中学习得到二维观测与三维姿态之间的对应关系，最终学习所获取的结果作为对样本的姿态估计。但是，要想获取比较准确地结果，就必须有足够密集的样本，但是所需要样本的数量是随状态空间的维度呈现指数级增加的，针对高维状态空间，几乎不可能得到所必须的密集采样。

基于模型的估计依靠物体的特征点，从提取的物体特征中凭借某种几何模型来表达物体的结构和形状，在模型和图像之间建立起对应关系，最终完成物体空间姿态的估计。

2.4.2 2D 到 3D 位姿估计

Perspective-n-Point[36]简称 PnP 问题，通过给定相机内参和一组 3D 点及对应的 2D 投影点，来确定相机的位置和方向。

由给定的 n 个点的 2D/3D 对应点对 $\{P_i, m_i\}_{i=1}^{n}$，建立线性约束方程，求解摄像机姿态及标定摄像机。任意空间点 P_i 与像点 m_i 之间的投影成像关系如式：

$$\lambda \tilde{m}_i = KRP_i + Kt \tag{2.31}$$

其中 R、t 为相机姿态，K 为相机内参，P_i 为世界坐标系下空间点，$\tilde{m}_i$ 为成像

点 m_i 的齐次坐标。

令 $F = KR$，且 F_i 是 F 的第 i 行。根据式(2.31)，可以得出：

$$\lambda_1 \tilde{m}_1 - \lambda_2 \tilde{m}_2 = F(P_1 - P_2) \tag{2.32}$$

将式（2.32）的深度信息消去，整理得：

$$P_{12}{}^T F_1^T - \mu_2 P_{12}{}^T F_3^T - \mu_{12}\lambda_1 = 0 \tag{2.33}$$

$$P_{12}{}^T F_2^T - v_2 P_{12}{}^T F_3^T - v_{12}\lambda_1 = 0 \tag{2.34}$$

同理，空间点 P_1 与 $P_3 ... P_n$ 等组成矢量也可分别得到(2.33)，(2.34)两式，则组成 2(n−1)个关于未知数 X 的 9 元线性约束方程：

$$\mathrm{AX} = \lambda_1 b \tag{2.35}$$

其中：

$$A = \begin{bmatrix} P_{12}{}^T & 0 & -\mu_2 P_{12}{}^T \\ 0 & P_{12}{}^T & -v_2 P_{12}{}^T \\ \vdots & \cdots & \vdots \\ P_{1n}{}^T & 0 & -\mu_n P_{1n}{}^T \\ 0 & P_{w1n}{}^T & -v_n P_{1n}{}^T \end{bmatrix}$$

$$X = [F_1, F_2, F_3]^T$$

$$0 = [0, 0, 0]$$

$$b = [\mu_{12}, v_{12} ... \mu_{1n}, v_{1n}]^T$$

每个矢量 P_{ij} 与对应像点可组成 2 个独立方程，根据 n 的不同， 系数矩阵 A 的秩不同，由最小二乘法或线性理论求出未知数 X 的解，建立约束求得 F，利用矢量运算，从 F 中分解出相机姿态 R 和 t。实验数据如表 2 所示。

表 2 相机姿态 R 和 t 数据

Table 2 Camera pose R and T Data

姿态	R	t
数据	[-0.0248601839305800 4;0.0352015444938 41;0.0151139337611071]	[0.0158486715238912 1;0.0085423887827 33614;0.0074704329939 66038]

2.5 闭环检测与全局优化

闭环 (Loop closure) 检测是 SLAM 的基础问题之一，用以判断机器人当前的位置是否是之前已经经历过的历史环境区域。累积误差是视觉里程计（新来的数据与上一帧进行匹配，估计其运动，然后再把运动累加起来）中不可避免的，后续的相机姿态依赖着前面的姿态。要确保地图的精准，必须要确保每次匹配都准确无误，而这是不容易实现的。为解决这个问题，不单单考虑两帧的信息，而要把全部的信息都考虑进去，这就是闭环检测与全局优化问题。

2.5.1 BoVW 模型与闭环检测

闭环检测的关键依赖有效的场景模型。BoVW[42,43]是一种非常有效的场景建模方法，它通过提取图像局部特征，并对其聚类构建视觉字典树 。场景中任何一幅图像都可以用视觉字典中的“单词集合”来表示。将当前获取的图像和之前数据集中图像进行比对，找出相似度高的匹配这就是基于图像的闭环检测系统。

机器人在现实中的运动中常常路经之前走过的地方，依靠基于视觉词典(Bag of Visual Words,BoVW) [37,38]的闭环检测(Loop Closure)环节可以有效降低机器人运动过程中的累积误差。回环检测可以说是一种检测观测数据相似性的算法，一旦机器人检测出闭环随即对姿态进行求解。

词袋模型（BoW）[39]最初是文件表示法的一种在信息检索领域中。其基本思想是对于一个文本，忽略其语序和语法，只将其看做是一些独立的词汇的集合。此模型被用在文本分类中，将文档表示成特征矢量，即表示成顺序无关的关键词的组合，通过统计文档中关键词出现的频率来进行匹配。

视觉词典（Bag of Visual Words,BoVW）是将词袋模型应用到计算机视觉领域，对图像进行结构化描述，利用 BoVW 表示图像。其思想是通过提取图像特征，将其整合成视觉单词，从而得到表示图像的关键词，并将图像特征空间转化成离散的视觉字典。然后对待分类图像进行相同的处理，将新的图像特征映射到视觉字典中最近邻视觉词汇中，再通过计算视觉字典间距离获取图像的相似度，从而完成闭环检测。

视觉词典的生成流程如下图 2.16 所示：

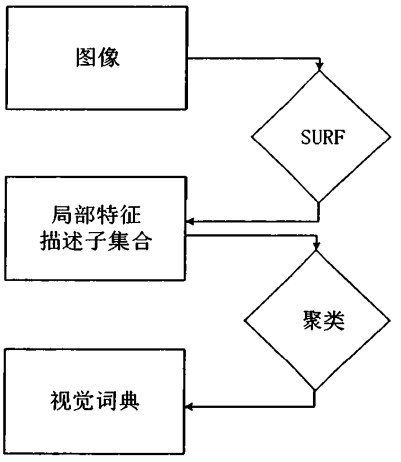


图 2.16 视觉词典的生成流程

Fig 2.16 The generation process of visual dictionary

首先提取图像特征并生成描述子，这样每幅图像由一连串视觉单词组成，于是将图像 I_u 表示为一个 n 维的描述符集合 $D: I_u \rightarrow \{d_1 ... d_n\}$。每个 SURF 特征点 d_i 都与视觉字典中的一个视觉单词 $\hat{d}_i$ 有关联，视觉字典表示为：$V = \{\hat{d}_1 ... \hat{d}_{|V|}\}$。视觉字典 V 通过 BoVW 建模方式，聚类相似的描述符，使得每一个 SURF 描述向量都成为一个息息相关的视觉词表。

建立好视觉字典之后，通过 K-D 树[39]对群集进行中心化，并对所有描述符量子化，实现对群集的简化。

图像 I_u 是由各异的权重 w_i 的的视觉词汇 $\hat{d}_i$ 组成的，权重 w_i 是整个图像中单词的频率。每个词汇的权重表达如下式:

$$w_i = \log_{10}(\frac{N}{n_i}) \tag{2.36}$$

式中，N 代表所有图像的数目，n_i 代表 d_i 中含有的图像的数目。要是视觉字典含有 $|V|$ 个各异的视觉词汇，则图像的矢量可表示为：$\vec{I}_i = [u_1 ... u_{|V|}]^T$。

其中，词汇权重定义如下：

$$u_i = \begin{cases} w_i & \text{如果} d_i \in I_u \\ 0 & \text{否则} \end{cases} \tag{2.37}$$

在字典树的构建环节中，每个 word 在全部的训练样本中的频率被称为叶，而且频率与区分度成反比关系，频率的计算定义如下：

$$\mathrm{idf(i)}=\log(\frac{N}{n_i}) \tag{2.38}$$

如果有必要在字典树中加入一幅新图像时，依据 *Hamming* 距离[40]，图像中提取的描述子必将会从字典树的根节点逐步朝下达到叶子节点，叶子节点的频率定义如下：

$$\mathrm{tf(i,I_t)}=\frac{n_iI_t}{nI_t} \tag{2.39}$$

此式中，n_iI_t代表 word 显出的次数，nI_t代表描述子的总数目；每个叶节点保存了 inverse index（倒排挡索引），即叶节点的图像I_t的 ID 。word 描述 vector 中第i维的值为：

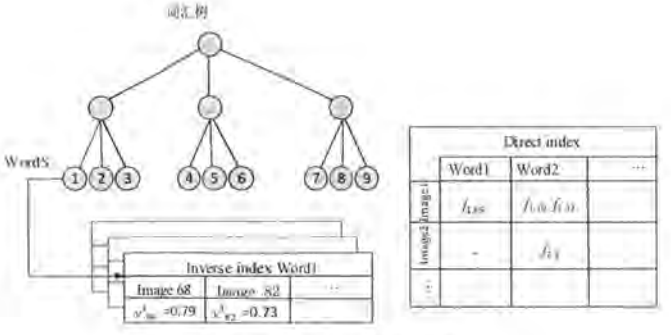
$$v_it=\mathrm{tf}(i,I_t)\times i\mathrm{df}(i) \tag{2.40}$$

针对一个图像描述符全部，重复上面的操作，即可获得每个单词的值，这些值代表了矢量图像。

然后通过计算两幅图像的余弦距离获可得测量相似度。相似度计算公式如下：

$$s(v_1,v_2)=1-\frac{1}{2}\left|\frac{v_1}{\|v_1\|}-\frac{v_2}{|v_2\|}\right| \tag{2.41}$$

两幅图像的得分与两幅图像的相似度成正比。在树中，除了存留了 inverse index，还存留了 direct index (直接索引)，原理示意图如图 2.17 所示，direct index 连接了特征之间的对应关系，使两幅图像可以便利的进行特征搜索及计算两帧之间的位姿关系。



2.17 视觉词典树生成示意图

Fig 2.17 Sketch map of visual dictionary tree generation

为去除相似度的大小和字典树、图像的联系，归一化是个有效地方法。归一化相似度计算公式如下：

$$\eta(v_t, v_{t_j}) = \frac{s(v_t, v_{t_j})}{s(v_t, v_t - \Delta t)} \tag{2.42}$$

其中，$v_t - \Delta t$ 表示上一帧图像。式（2.42）表达如果当前帧和前一帧是极相似的，则可以用前一帧的相似度去处理字典树中图像的相似度。

当 $s(v_t, v_t - \Delta t)$ 较小时，即代表机器人做旋转运动，导致总体得分较高，设置阈值 α，可化解此种不利因素的影响，如若当前帧和前一帧相似度小于 α，将不做闭环检测。

BoVW 字典创建与闭环检测实验数据如图 2.18 所示。实验样本数量为 5 幅场景图片编号为 0、1、2、3、4，聚为 9 类，树深为 3，总共生成 674 个视觉词汇，两两计算，大于 0.1 分视为形成闭环，场景 0 与场景 2，场景 1 与场景 3 形成了闭环。

图 2.18 BoVW 字典创建与闭环检测实验数据

Fig 2.18 BoVW dictionary creation and closed loop detection experimental data

2.5.2 图优化

图优化[15,16,20]是处理视觉 SLAM 问题常用的方法之一。与滤波方法处理 SLAM 相对比，其不同之处在于它是先将所有的数据记录下来，构建完地图之后最后再一次性矫正地图。基于图优化方法的 SLAM 通过构建一张位姿图(pose graph)，图的节点代表机器人的位姿或者路标，节点之间的边表示传感器测得的位姿约束信息，通过对图进行求解可以求得机器人的位姿信息。

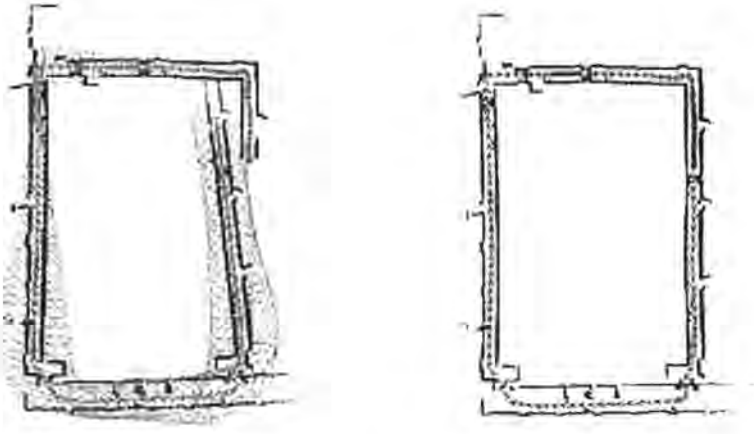
图的组成包括节点和边，在图优化的 SLAM 中，一个节点就是机器人现在所处的状态位姿，边的就是有这些位姿状态之间的关系构成。例如 t 时刻与 t+1 时刻

这两个相邻时间之间位姿运动就构成边。当地图构建完成之后，为了满足机器人位姿关系之间的各种约束，就需要不断的改变机器人的位姿。最后可以将图优化分解为两个步骤：

(1) 构建图，机器人采集信息，并把自己的位姿状态作为节点，各时刻之间的位姿关系作为边。

(2) 优化图，机器人为了满足约束关系不断更新自己的位姿状态。

图优化过程：首先获得数据，然后把位姿状态设置成地图的节点。将位姿之间的关系设置为边。地图构建完成之后，机器人不断更新节点满足各个边的约束，最后就能得到的优化后的地图。优化效果对比图如图 2.19 所示。



(a)优化前的图　　(b)过优化后的图

图 2.19 优化效果对比图

Fig 2.19 Contrast diagram on optimization effect

在 SLAM 过程中，获得最优轨迹可以转化成求解并优化机器人的位姿，使得下式误差平方函数最小：

$$F(x) = \sum_{<i,j>\in C} e(x_i, x_j, z_{ij})^T \Omega_{ij}\, e(x_i, x_j, z_{ij}) \tag{2.43}$$

$$x^* = argmin F(x) \tag{2.44}$$

其中，x_i表示图顶点的参数向量，z_{ij}和Ω_{ij}分别代表x_i和x_j之间约束关系的期望和信息矩阵，$e(x_i, x_j, z_{ij})$是向量误差函数，表示参数块x_i和x_j满足约束z_{ij}的程度。Ω_{ij}代表边的信息矩阵，实质是误差的权重矩阵，由测量协方差矩阵的逆得来。

为了简化，将误差函数写为：

$$e(x_i, x_j, z_{ij}) \overset{\text{def}}{=} e_{ij}(x_i, x_j) \overset{\text{def}}{=} e_{ij}(x) \tag{2.45}$$

假设初始参数估计x^0已知，式（2.44）的数值解可以通过 Gauss-Newton 算法[41]和 L-M（Levenberg-Marquardt）算法[42]进行求解。求解方法是将误差函数在初始估计x^0附近进行一阶 Taylor 展开。

$$e_{ij}\left(x_i^0 + \Delta x_i, x_j^0 + \Delta x_j\right) = e_{ij}(x^0 + \Delta x) \simeq e_{ij} + J_{ij}\Delta x \tag{2.46}$$

J_{ij}是误差函数$e_{ij}(x)$在x^0附近的雅克比矩阵，e_{ij}代替$e_{ij}(x^0)$。将式(2.46)代入式(2.43)可得:

$$F_{ij}(x^0 + \Delta x) = c_{ij} + 2b_{ij}^T\Delta x + \Delta x^T H_{ij}\Delta x \tag{2.47}$$

其中，b_{ij}、Δx、e_{ij}都表示列向量，c_{ij}是一个数值。

将式(2.47)带入式(2.43)，可将误差平方和函数重写为：

$$F(x^0 + \Delta x) = c + 2b^T\Delta x + \Delta x^T H\Delta x \tag{2.48}$$

其中，$c = \sum c_{ij}$，$b = \sum b_{ij}$，$H = \sum H_{ij}$

求解式(2.48)，令求其最小，方法是求解一阶导并使其为 0：

$$H\Delta x^* = -b \tag{2.49}$$

H 是系统的信息矩阵，系统的解是初始值加上这个增量。

$$x^* = x^0 + \Delta x^* \tag{2.50}$$

Gauss-Newton 法一直迭代(2.48)线性化，(2.49)求解和(2.50)更新，直至收敛。在每次迭代中，之前解都被用作线性点和初始解。

L-M 法是 G-N 法的非线性变体，引入阻尼因子来控制收敛速度。

$$(H + \lambda I)\Delta x^* = -b \tag{2.51}$$

其中，λ表示阻尼因子。如果新误差相对之前误差变小，在下次迭代过程中阻尼因子λ变小；反之，λ变大。

2.6 实验结果及分析

为验证算法的整体性能，本文利用 Kinect 深度相机作为传感器，遥控机器人对某楼 208 室进行了 3D SLAM 实验，真实环境如图 2.20 所示，3D SLAM 效果图如图 2.21 所示，其中粉红色线表示相机定位轨迹，点云表示地图。

图 2.20 某楼 208 的真实环境
Fig 2.20 The real environment of a building 208

图 2.21 某楼 208 3D SLAM 图
Fig 2.21 The image of 3D SLAM of a building 208

2.7 本章小结

本章首先阐述了室内环境下 3D 视觉 SLAM 算法步骤，对 SIFT 算法、SURF 算法进行了理论推导并做了对比试验，并改进了匹配方法很有效的消除误匹配。阐述了姿态估计、闭环检测的词袋模型、图优化理论并做了相关实验，最后验证了整个算法的有效性，开展了实际试验。

第3章 移动机器人导航算法研究

自主移动机器人是指没有人工引导的结构化或非结构化环境中能完成预期任务的机器人。路径规划是其实现自主导航的关键，也是移动机器人自主性的重要体现。本章阐述了导航地图的基本理论、路径规划的基本理论、基于粒子滤波的定位算法、基于动态窗口的局部避障算法等。

3.1 机器人动态模型建立

第二章详细介绍了基于深度相机的 SLAM 算法，为此，为将该理论应用于实际的 irobot 机器人，首先本节先建立差动驱动的 irobot 机器人动态模型。机器人的运动主要由左轮和右轮电动机转速决定。在此分别将机器人几何中心的直线位移量和绕中心垂直的旋转量这两个衍生变量看作主导变量。在此，假设机器人不会同时进行旋转运动和平移运动。

建模时，应注意机器人总是沿圆弧运行。曲率为零表示线性平移运动，，而曲率半径为零表示旋转运动。

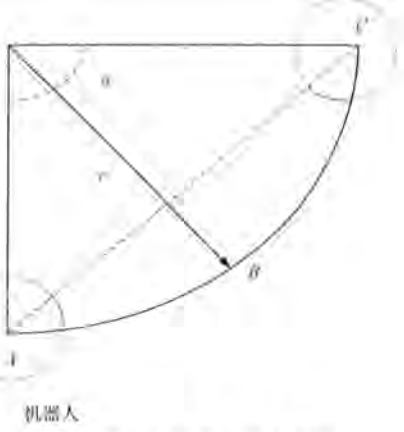


图 3.1 机器人动态模型

Fig 3.1 The dynamic model of robot

由于图 3.1 给出了机器人运动模型机制，此时：

$$\widehat{ABC} = s \tag{3.1}$$

$$\theta = \frac{s}{r} = K_1(\text{右轮旋转量-左轮旋转量}) \tag{3.2}$$

$$s = (K_2/2)(\text{右轮旋转量}+\text{左轮旋转量}) \tag{3.3}$$

由式（3.2）和式（3.3）可得s，同时根据车轮编码器读数可直接获得θ。由此得：

$$r = \frac{s}{\theta} \tag{3.4}$$

$$\overline{AC} = 2r\sin\frac{\theta}{2} \tag{3.5}$$

当机器人的初始位置φ时，可将AC分解为x分量和y分量。因此可得：

$$dx = 2r\sin\frac{\theta}{2}\cos\varphi \tag{3.6}$$

$$dy = 2r\sin\frac{\theta}{2}\sin\varphi \tag{3.7}$$

$$d\varphi = \theta \tag{3.8}$$

在$\theta \to 0^0$情况下建模时，由于$r = \frac{s}{\theta} \to \infty$，因此会产生一个逻辑问题。在此，当$\theta \le 5^0$时，假设$\overline{AC} = s$,对于机器人的直线位移量$D$和旋转量$\theta$，可由下式建立机器人的最终动态模型：

$$\Delta x = D\cos\varphi \tag{3.9}$$

$$\Delta y = D\sin\varphi \tag{3.10}$$

$$d\varphi = \theta \tag{3.11}$$

求解式(3.9)-(3.11)，可得：

$$x_{k+1} = x_k + D\cos\varphi \tag{3.12}$$

$$y_{k+1} = y_k + D\sin\varphi \tag{3.13}$$

$$\varphi_{k+1} = \varphi_k + \theta \tag{3.14}$$

由此，可计算雅各比矩阵和协方差矩阵：

$$\nabla f_u = \begin{bmatrix} \frac{\partial \Delta x}{\partial D} & \frac{\partial \Delta x}{\partial \theta} \\ \frac{\partial \Delta y}{\partial D} & \frac{\partial \Delta y}{\partial \theta} \\ \frac{\partial \Delta \varphi}{\partial D} & \frac{\partial \Delta \varphi}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \tag{3.15}$$

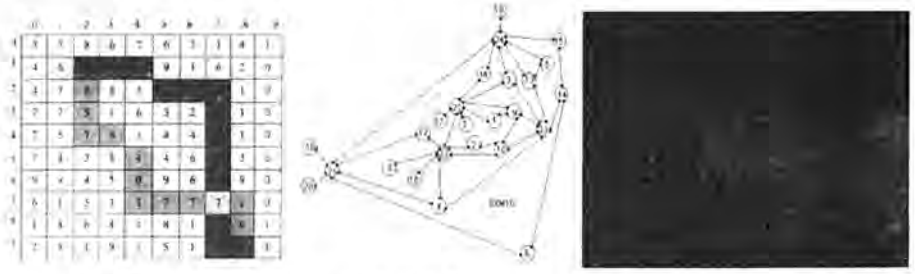
$$\nabla f_{x_v} = \begin{bmatrix} 1 & 0 & -D\sin\varphi \\ 0 & 1 & D\cos\varphi \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$Q = \begin{bmatrix} \sigma D^2 & 0 \\ 0 & \sigma\theta^2 \end{bmatrix} \tag{3.17}$$

式中 $\sigma D = D\times$单位位移的标准偏差，$\sigma\theta=\theta\times$单位旋转的标准偏差。

3.2 导航地图表达

地图的表示方法通常有：栅格地图、拓扑地图、特征地图等等。栅格地图是将环境分割成一连串赋予一个可能数值的栅格，这个数值代表该栅格被占据的概率，这种地图可以方便用来导航。拓扑地图使用节点（每一个地点用一个点来表示）和边（连接相邻的点）来表达环境信息。特征地图就是利用不同的几何结构(如点、线、面）来表达环境，如点云地图。地图表达形式如图 3.2 所示，其中（a）代表栅格地图，（b）代表拓扑地图，（c）代表特征地图。



（a）栅格地图　　（b）拓扑地图　　（c）特征地图

图 3.2 地图的表示方式

Fig 3.2 Representation of a map

SLAM 的目的是估计机器人的运动轨迹并构建精确的环境地图，通常情况下使用优化后的位姿拼接出点云图，完成地图构建。在做完 SLAM 后所表现出来的地图形式主要是点云地图，这种点云地图存在如下缺陷：

(1) 图的形式不紧凑

点云地图一般由几十万个空间点组成，通常以 PCD 格式保存并且文件很大。实际上点云地图没必要这么大，由于它提供了很多冗余的信息，比如光线照射产生的影子等。这样存储会导致点云图浪费了很大的存储空间。

(3) 处理重叠的效果不佳

在点云图的构建过程中，当位姿信息出现误差或者噪声干扰太大，会导致得到

的点云图出现重叠问题，这是无法避免但亟待解决的问题。

(3)难以用于导航

地图的用途在于导航。在具备了地图之后，机器人可以实现快捷的移动。可是，点云地图的特点致使难以做路径规划，是不能完成导航的。为避开 3D 障碍物，并降低计算成本提高导航实时性能，本文采取将 3D 地图障碍物的高度信息映射到 2D 地图上，并在 2D 地图进行路径规划的方法。

3.2.1 Cost_2d 地图

为了克服传感器噪声，引出占据栅格地图（Occupancy Grid Map）[43]的概念，其中每一栅格被赋予固定数值，代表此栅格的占用率。在尺度地图中对于一个点，占据率（Occupancy）指的是要么有障碍物，用P(s=1)代表 Occupied 状态的概率；要么没有障碍物，即用P(s=0)代表 Free 状态的概率，并且两者的和为 1。两个值表示太麻烦，为方便表示点的状态，引入两者的比值，定义如下：

$$odd(\mathrm{s})=\frac{\mathrm{P(s=1)}}{\mathrm{P(s=0)}} \tag{3.18}$$

栅格地图中某点来了一个新测量值之后，假设该点的初始状态为$odd(\mathrm{s})$，测量值来之后，将其更新为：

$$odd(\mathrm{s}|\mathrm{z})=\frac{\mathrm{P(s=1}|\mathrm{z)}}{\mathrm{P(s=0}|\mathrm{z)}} \tag{3.19}$$

根据贝叶斯公式，有：

$$\mathrm{P(s=1}|\mathrm{z})=\frac{\mathrm{P(z}|\mathrm{s=1)P(s=1)}}{\mathrm{P(z)}} \tag{3.20}$$

$$\mathrm{P(s=0}|\mathrm{z})=\frac{\mathrm{P(z}|\mathrm{s=0)P(s=0)}}{\mathrm{P(z)}} \tag{3.21}$$

带入之后，得

$$odd(\mathrm{s}|\mathrm{z})=\frac{\mathrm{P(z}|\mathrm{s=1)}}{\mathrm{P(z}|\mathrm{s=0)}}odd(\mathrm{s}) \tag{3.22}$$

对式（3.22）两边取对数得：

$$\log odd(s|z) = \log \frac{P(z|s=1)}{P(z|s=0)} + \log odd(s) \tag{3.23}$$

$\log \frac{P(z|s=1)}{P(z|s=0)}$被称为测量值的模型（Measurement Model），标记为 lomeas。测量值的模型只有两种：

$$\text{lofree} = \log \frac{P(z=0|s=1)}{P(z=0|s=0)} \tag{3.24}$$

$$\text{looccu} = \log \frac{P(z=1|s=1)}{P(z=1|s=0)} \tag{3.25}$$

而且都是定值。若是用 $\log odd(s)$ 来代表位置 s 的状态 S 的话，更新就可以简化为：

$$S^+ = S^- + \text{lomeas} \tag{3.26}$$

其中，S^+ 代表测量值之后的状态，S^- 代表测量值之前的状态。

此外，一个点的初始状态在没有任何测量值的情况下为：

$$S_{init} = \log odd(s) = \log \frac{P(s=1)}{P(s=0)} = \frac{0.5}{0.5} = 0 \tag{3.27}$$

通过如此建模，简单的加减法运算就能方便的更新地图上一个 2D 点的状态了。信息学院某楼 208 的 Cost_2d 地图，如图 3.3 所示。

图 3.3 某楼 208 Cost_2d 地图

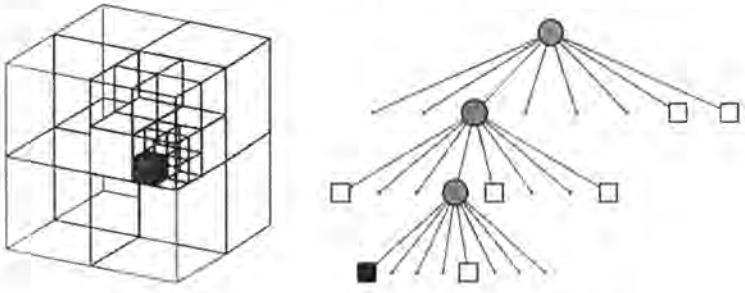
Fig 3.3 The Cost_2d map of a building 208

3.2.2 Octomap 地图

Octomap[44,45]相对于点云图来说，Octomap 用 Octotree (八叉树)来存储地图信息，使得需要的存储空间大大缩减，不仅能够压缩地图而且还能调节分辨率。

(1) 八叉树的表达

Octotree[46]（八叉树）作为描述三维空间的树状结构的一种，它的每一个节点代表一个体积元的立方体，每一个节点都连接到八个节点的体积元，这些体积元加和之后就可以得到父节点的体积。分叉中心通常设置为中心点，八叉树基本结构如图 3.4 所示，(a)八叉树体积结构，(b)八叉树树状结构。



(a)八叉树体积结构　　(b)八叉树树状结构

图 3.4 八叉树基本结构

Fig 3.4 The basic structure of octree

八叉树实质是不断从中心往下分，每次分为八支，直到叶子结点为止的数据结构。每一个小方块都可以用一个值来记录它的属性，即占据或者未被占据，通常会取值在 0~1 范围内的浮点数，这个值越高，那么这个叶子节点就越可能被占据。如图(b)所示，当一个节点的八个子节点的状态相同时，即这些叶子节点都被被占据或者没有被占据时，就可以剔除这个节点。八叉树就是通过这种方式来节省存储空间。

(2) 八叉树的更新

之前已经介绍，八叉树的叶子有一个数值用来表示该叶子是否被占据。通常取值在 0~1 之间的浮点数。由于噪声的干扰，使得这些叶子在不同时刻可能展现出不同的状态，所以，就可以用概率法形式来描述八叉树的更新了。

根据八叉树的推导，假设$t = 1,2,3,\ldots,T$时刻，观测的数据为$z_1, z_2, \ldots, z_T$，那么第 n 个叶子节点记录的信息为：

$$P(n|z_{1:T}) = \left[1 + \frac{1-P(n|z_T)}{P(n|z_T)} \frac{1-P(n|z_{1:T-1})}{P(n|z_{1:T-1})} \frac{P(n)}{1-P(n)}\right]^{-1} \tag{3.28}$$

存在 logit 变换，把一个概率 P 转换到全实数空间 R 上：

$$\alpha = logit(P) = log(\frac{P}{1-P}) \tag{3.29}$$

上式(3.29)是一个可逆变换，则有：

$$\mathrm{P} = logit^{-1}(\alpha) = \frac{1}{1+e^{-\alpha}} \tag{3.30}$$

α叫做 log-odds（更新速率）。并令 L(n)为叶子节点的 log-odds，就有

$$L(n) = log(\frac{P(n)}{1-P(n)}) \tag{3.31}$$

将上式(3.38)代入式（3.31），得：

$$L(n|z_{1:T}) = L(n|z_{1:T-1}) + L(n|z_T) \tag{3.32}$$

另外，再加上一个最大、最小值的限制：

$$L(n|z_{1:T}) = \max(\min(L(n|z_{1:T-1}) + L(n|z_T), l_{max}), l_{min}) \tag{3.33}$$

其中，l_{min}表示更新速率值的最小边界，l_{max}表示更新速率值的最大边界。

经过这样建模，只需要做简单的加减法运算就可以方便的更新地图上一个 3D 点的状态了。信息学院某楼 208 的 Octomap 地图如图 3.5 所示。

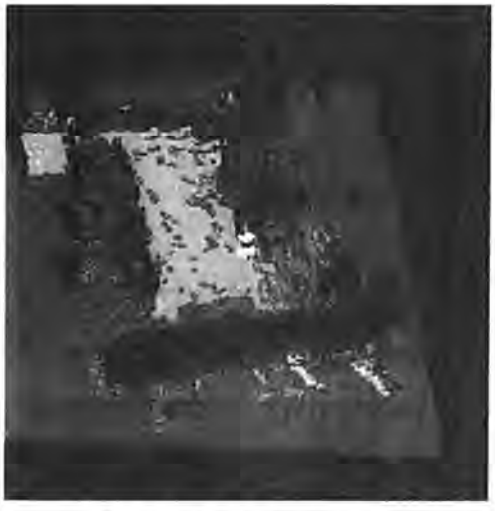


图 3.5 某楼 208 Octomap 地图

Fig 3.5 The Octomap map of a building 208

3.2.3 2D 地图与 3D 地图融合的 2.5D 地图

将以障碍物信息更新的的整个区域从 Octomap 投影到 Cost_2d 地图上。在此，假设机器人/摄像头获取的的 3D 坐标系如图 3.6 所示。假设在任意时刻，该坐标系都与机器人/摄像头的当前位置关联。首先通过坐标变换，将 3D 地图坐标与 2D 地图坐标对齐，然后将投影中心定义为 Z 轴中的原点，最后将障碍物点 Z 轴信息映

射到坐标平面也就是 Cost_2d。

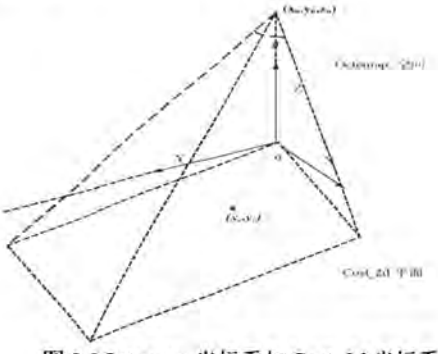


图 3.6 Octomap 坐标系与 Cost_2d 坐标系

Fig 3.6 Octomap coordinate system and Cost_2d coordinate system

根据图 3.6 ，Octomap 中坐标为(x_0, y_0, z_0)，转换为 Cost_2d 坐标系中的二维坐标(x_c, y_c)，即：

$$x_c = \frac{x_0 z_0}{(f\sin\theta + x_0\cos\theta)} \tag{3.34}$$

$$y_c = \frac{z_0(x_0\sin\theta - f\cos\theta)}{(x_0\cos\theta + f\sin\theta)} \tag{3.35}$$

其中，f 是摄像头的焦距，θ是与 2D 平面平夹角。

不断进行上述变换过程，并在地图中更新障碍物信息 Octomap 地图与 Cost_2d 地图融合如图 3.7 所示。Cost_2d 是分层机制的，比如自己扫射的平面地图是一层，映射的障碍物是一层，这样就可以在 Cost_2d 上做路径规划并可以避开 3D 障碍物。Octomap 地图 3D 障碍物映射到 Cost_2d 地图如图 3.8 所示，黑色的点表示障碍物，粉红色表示障碍物的膨胀区域。

图 3.7 Octomap 地图与 Cost_2d 地图融合

Fig 3.7 Octomap map coordinate and Cost_2d map coordinate alignment

图 3.8　Octomap 地图 3D 障碍物映射到 Cost_2d 地图
Fig 3.8 Octomap map 3D obstacles to Cost_2d map

3.2.4 基于粒子滤波的定位算法

机器人在已知地图信息的情况下，要想依靠传感器信息确定自己的位置，需要具备两点，首先，在给定的机器人位置信息的前提下可以计算出和地图的吻合程度；其次，机器人在相邻时刻点的位置的移动很小。依据这两点，本文采纳的定位方法来自于蒙特卡罗定位（Monte Carlo Localization）[47]算法，其算法的核心思想是利用高斯分布表示机器人位置，使用粒子近似描述机器人可能的位置。在算法具体实现中，利用高斯分布撒出的 M 个粒子来具体描述机器人的位置。

具体算法步骤如下图 3.9 所示。

图 3.9 蒙特卡罗定位算法流程
Fig 3.9 Step of localization algorithm based on Monte Carlo

(1) 初始化粒子群

由于初始位置已经给定了确定值，在给出种群大小之后，仅仅需要对 M 个初始位置进行复制以初始化种群。如若初始位置不确定，则可以采用高斯分布的方式初始化粒子群。

(2) 模拟粒子运动

在相邻时刻，假设机器人活动的范围是有限的，则凭借 randn()函数随机生成机器人可能运动到达的位置。

(3) 计算粒子评分

对上次迭代得到的所有粒子，将第m个粒子x_t^m按概率$p(x_t|u_t,x_{t-1}^m)$采样，计算每个被采样粒子的权重,也就是计算传感器定位的障碍物与地图中障碍物的符合个数。当对粒子的评分结束后，得分最高的粒子被选中当作此时刻机器人的位置。

(4) 粒子群重采样

在评分完结后，严重偏离可能位置的粒子评分很低，这部分粒子需要丢弃。传感器读数与地图吻合度 80%以上的粒子评分很高，这部分粒子需要存留，这就是粒子群重采样。将被采样的粒子按和权重成正比的概率重采样。

采样过程中所得到的状态样本x_t^m的分布是对贝叶斯滤波器预测步骤的概率分布$\overline{bel}(x_t)$的逼近。为得到后验概率分布，设状态样本x_t^m的可信度为权重w_t^m。根据$\overline{bel}(x_t)$计算$bel(x_t)$的公式：

$$bel(x_t) = \eta p(z_t|x_t^m)\overline{bel}(x_t) \tag{3.36}$$

其中，η为与状态x_t无关的常量。

因此，表示先验概率分布$\overline{bel}(x_t)$与后验概率分布$bel(x_t)$的关系可以按如下定义权重：

$$w_t^m = p(z_t|x_t^m) \tag{3.37}$$

每个状态样本对应一个权重，就会得到一个临时的样本集$\bar{X}_t$。最终要得到概率密度分布符合$bel(x_t)$的样本。

首次采样时粒子的初始化，若初始位置不确定，则在整个可能的状态空间里等概率的采样粒子；若初始位置确定，则就在确定的位置附近以较高概率采样粒子。由于本文的机器人初始的位置是人为选择一个确定的坐标上，因此采用后者方法。在 ROS 平台下，利用 Stage 仿真器，激光雷达数据作为传感器数据，蒙特卡罗定位仿真图如图 3.10 所示，其中箭头代表 AMCL 返回的各种姿态。

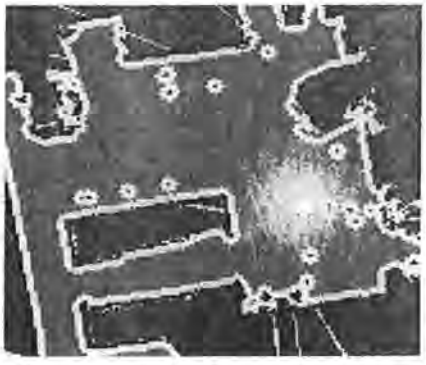


图3.10 蒙特卡罗定位仿真实验

Fig 3.10 The simulation experiment based on Monte Carlo

3.3 路径规划研究概述

在移动机器人导航过程中，确定机器人的当前位置和目标点之间的最优路径是一个重要目标，在这样就能使得机器人在避障的前提下尽可能短时间到达目的地。可分为基于地图的路径规划也称全局路径规划和基于地图构建的路径规划也称局部避障路径规划。

3.3.1 全局路径规划

在基于地图的导航中，系统已具有环境的先验知识，并利用地图信息进行路径规划，使机器人避开静态障碍物在环境中沿一条预期路径运动到达预期目的地。确定最短路径可采用标准的图搜索算法，如 Dijkstra 算法和 A*算法。全局路径规划主要有如下五个部分组成：

(1) 地图

系统需要外界环境的一些内部表征或知识已完成目标驱动的任务。

(2) 数据获取

系统通过传感器取得环境信息。

(3) 特征提取

从采集的输入数据中提取显著特征，如边缘、纹理等。

(4) 路标识别

根据某些预设标准，系统在观测数据的特征和预存的预期路标之间寻找可能存在的匹配信息。

(5) 自定位

计算作为检测路标和之前位置的一个函数的机器人当前位置，然后系统推导出

机器人的运动路径。

3.3.2 Dijkstra 算法

Dijkstra [48]算法用来解决非负权重有向图的单源最短路径问题，是一种改良的广度优先搜索算法。其对一个有权重的图 $G=(V,E)$进行操作，S 被设为起点， G 中所有的顶点的集合用 V 表示。

算法步骤：

(1) 初始时，S 中只有源点，即 $S=\{V\_0\}$，V_0 的距离为 0。U 中则包括了除 V_0 外的其他所以顶点，即：U={其余顶点}，如若 V_0 与 U 中顶点 u 有边的关系，则<u,v>就有权值，倘若 u 不是 v 的出边邻接点，则把<u,v>权值设置为无穷大。

(2) 从 U 中选择一个的顶点 m 距离 V_0 最小，将 m 置入集合 S 中。

(3) m 被设置为新的中间点，调整 U 中顶点之间的距离；如果从点 V_0 到顶点 u 的距离（通过顶点 m）比原来距离（不通过顶点 m）短，那么对顶点 u 的距离值进行调整。

(4) 重复步骤（2）和（3），使得所有顶点都被加入到 S 中。

Dijkstra 是最基本的最短路径算法，也是一种广度优先图搜索算法。由于该算法大的时间和空间复杂度，直接将 Dijkstra 算法应用于移动机器人的导航系统，无法满足实际应用中对实时性的需求。所以在此基础上，出现了基于启发式搜索的路径规划算法诸如 A*算法，借助某种代价函数对相邻节点进行取舍，减小待搜索节点数量，提高算法效率。

3.3.3 A*路径规划算法

A* [49]搜寻算法，是在 Dijkstra 算法基础上，在节点的代价函数中加入入启发函数：

$$f(n) = g(n) + h(n) \tag{3.38}$$

设定 $n=(xn, yn)$为当前节点，即机器人从起点移动到当前点所需要的代价。其目的是使机器人向终点可能的方向移动，来减少不必要的访问位置点，提高算法执行效率。

算法步骤：

(1) 简化搜索区域

将要搜索的区域划分成正方形的格子，这个特殊的方法将搜索区域简化为了二维数组。该数组每一项代表可通过或不能通过的状态。计算出从起点到终点要走过

哪一些方格，就能够得到最终路径。

(2) 路径增量

距离用曼哈顿距离表示，栅格类似于曼哈顿的街道一样，即机器人在栅格地图中只可以沿着每一个栅格边界移动。

(3)路径搜索

① 将起点加入到 Open List；

② 不断执行以下过程： 遍历 Open List，找到 F 值最小的节点，将其设置为处理点。将上述节点移入 Close List。对于该节点周围的 8 个相邻方格，对每一个方格进行如下处理：倘若它无法到达，或者它已经被放入 Close List 中，就剔除它。如果不是，就继续以下操作。倘若在 Open List 找不到上述节点，把需要把它加入至 Open List，并把当前的方格点设置为其父节点，同时记录下这个方格的 F，G 和 H 值。 如果在 Open List 中可以查询到上述节点，就得检查是否需要更新 G 的数值，假如 Open List 排序方式是按 F 值的大小，那么调整后就需要对 Open List 进行再次排序。结束，当终点被加入到了 Open List 中，那么路径就已经得到了；

③ 保存路径：如图 3.11 所示，为从深绿色方块到达红色方块的路径，并绕过中间的障碍物（灰色）。在栅格中写入 F，G，H 的数值，F 在左上角，G 在左下角，H 则在右下角。

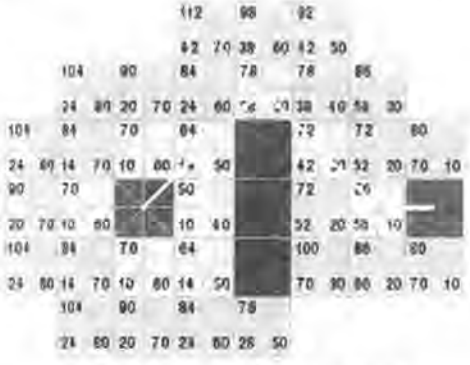


图 3.11 A*算法搜索路径结果

Fig 3.11 Algorithm of A* search path results

3.3.4 基于蚁群算法的改进 A*算法

利用 A*算法可以快速搜索出一条最短路径，但该路径有固有的约束，即每次只能移动一个栅格，并且路径点必须是在栅格中心。这样的路径显然不符合实际需

求，并且路径长度比全局最优路径长度有所增加。因此，要得到全局最优路径，必须要对该路径进行优化，同时应该避免新产生的路径与障碍物发生碰撞。因此，本文设计了一种新的改进算法。

蚁群算法[50]是对真实蚁群协作过程的模拟。机器人的路径规划问题可以看成从蚂蚁巢穴（初始位置）出发，绕过一些障碍物寻找食物（目的地）的过程，只要有足够多的蚂蚁在巢穴附近，这些蚂蚁必将能避开障碍物寻找到一条从巢穴（初始点）到达食物（目的地）的最短路径。

下面就如何利用蚁群算法对已有初始路径进行优化进行详细介绍。

(1) 建立巢穴临近区和食物产生的气味区

找到从起点朝终点方向到障碍物的最近垂直距离d,如图 3.12 所示，以此距离为半径或三角形的垂直高建立扇区或三角区。

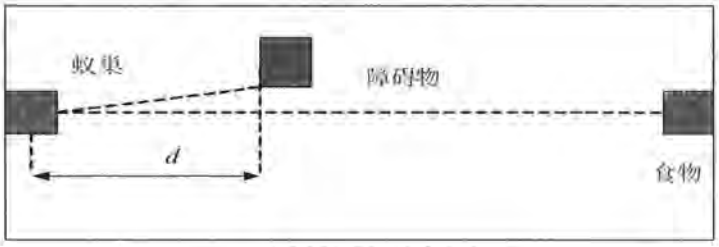


图 3.12 邻近区建立方法

Fig 3.12 The method of neighbor building

(2) 气味区建立

从食物朝起始位置方向直线扫描，没有遇到障碍物之间的区域为气味区。气味区建立示意图如图 3.13 所示。

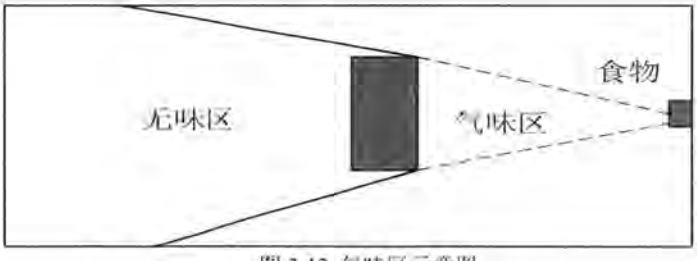


图 3.13 气味区示意图

Fig 3.13 Schematic diagram of odor area

(3) 路径构成

路径由三部分构成：机器人的起始位置到蚁群初始位置的路径、蚂蚁初始位置到蚂蚁进入气味区位置的路径和蚂蚁进入位置到终点位置的路径如图 3.14 所示。分别设为 path0、path1、path2，所以总的路径长度为：

$$L_{path} = L_{path0} + L_{path1} + L_{path2} \tag{3.39}$$

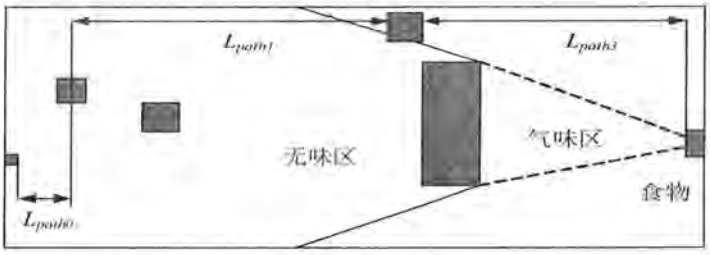


图 3.14 路径构成
Fig 3.14 Path composition

(4) 路径的调整

从开始点 S 出发不断寻找直到找到点 Q，使得 Q 的下一个点与 S 的连线穿过了障碍物，而 Q 以前的点（包括 Q 点）与 S 的连线没有穿越障碍物，连接 Q 与 S，这时 $\overline{SQ}$ 上障碍物最近的一点味 D，则 $\overline{SD}$ 就是要找的路径。下一步设 D 为 S，再在 S 于 G 之间寻找 D，直到 S 点与 G 点重合。所得到的连线即为调整后的路径。显然 $\overline{SD}$ 为 S 到 D 的最短距离，而 $\overline{DG} < \overline{DQ} + \overline{QG}$，所以线段 $\overline{SDG}$ 是沿着曲线 SG 绕过障碍物的最短路径。设总的栅格数位 N，从起始点到终点的直线距离的栅格数为 M，则最坏的时间复杂度为 $O(N^2)$，最好的时间复杂度为 $O(M^2)$。路径的调整示意图如图 3.15 所示。

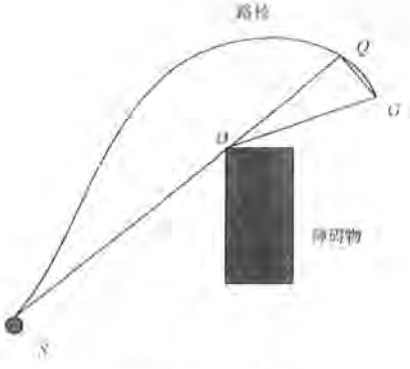


图 3.15 路径调整
Fig 3.15 Path adjustment

(5) 路径的选择

蚂蚁沿食物方向可选择三个行走栅格，如图 3.16 所示，分别编号：0,1,2.每只

蚂蚁根据三个方向的概率选择一个行走方向，移到下一个栅格。在t时刻，蚂蚁k从栅格i沿j（$j\in(0,1,2)$）方向转移到下一个栅格的概率：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{s\in J_k(i)}[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}} & j\in J_k(i)\\ 0 & \text{其他}\end{cases} \tag{3.40}$$

其中$J_k(i)=\{0,1,2\}$。$\tau_{ij}(t)$表示接下来准许选择的栅格集合。α与β分别表示信息素和启发因子的相对重要程度。η_{ij}是启发因子，表示蚂蚁从栅格i沿j（$j\in(0,1,2)$）方向转移到下一个栅格的期望程度，通常取i沿j之间距离倒数，由于栅格距离相等，不妨取1,于是式（3.40）变成：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}}{\sum_{s\in J_k(i)}[\tau_{ij}(t)]^{\alpha}} & j\in J_k(i)\\ 0 & \text{其他}\end{cases} \tag{3.41}$$

如果每一个可选择的方向的转移概率相等则随机选择一个方向，否则根据式（3.41）选择概率最大的方向，作为蚂蚁的下一步行走方向。

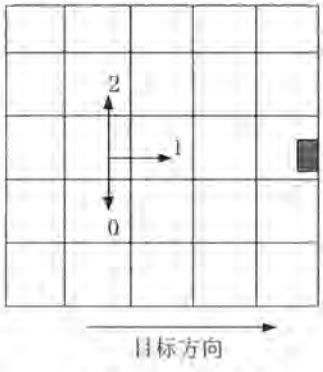


图 3.16 路径方向选择

Fig 3.16 Path direction selection

(6) 信息素的更新

一只蚂蚁在栅格上沿三个方向中的一个方向到下一个栅格，古在栅格设三个信息素，每个信息素更新公式：

$$\tau_{ij}(t+n)=\rho\tau_{ij}(t)+\Delta\tau_{ij} \tag{3.42}$$

$$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{3.43}$$

其中，$\Delta\tau_{ij}$ 表示本次迭代栅格 i 沿 j（$j\in(0,1,2)$）方向信息素的增量。$\Delta\tau_{ij}^{k}$ 表示第 k 个蚂蚁在本次迭代栅格 i 沿 j（$j\in(0,1,2)$）方向信息素的量。用 ρ 表示在某条路径上信息素轨迹挥发后剩余度，ρ 可取 0.9.如果蚂蚁 k 没有经过栅格 i 沿 j 方向到下一个栅格，则 $\Delta\tau_{ij}^{k}$ 值为 0。则：

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k} & \text{蚂蚁}k\text{经}i\text{栅格沿}j\text{方向}\\ 0 & \text{否则}\end{cases} \tag{3.44}$$

其中，Q 是正常数，L_k 代表第 k 个蚂蚁在本次游历中所经过的调整以后的径路长度。

3.3.5 实验结果与分析

针对某楼 208，依靠 ROS 平台，做了实际实验，其中黑色点状物表示障碍物，基于 A*算法的规划路径如图 3.18 所示，。基于蚁群算法的改进 A*算法的规划路径如图 3.19 所示。由实验图 3.18 和图 3.19 对比可以看出，基于蚁群算法的改进 A*算法的规划的路径比基于 A*算法的规划路径更加平滑，在机器人保持线速度 0.3m/s 和角速度 1.0m/s 的情况下，由同一点出发。到达相同目的地，基于 A*算法的路径规划用时 6.7s，基于蚁群算法的改进 A*算法的路径规划用时 7.9s，因此，基于蚁群算法的改进 A*算法的路径规划比基于 A*算法的路径规划改善很多，可以得到全局最优路径。

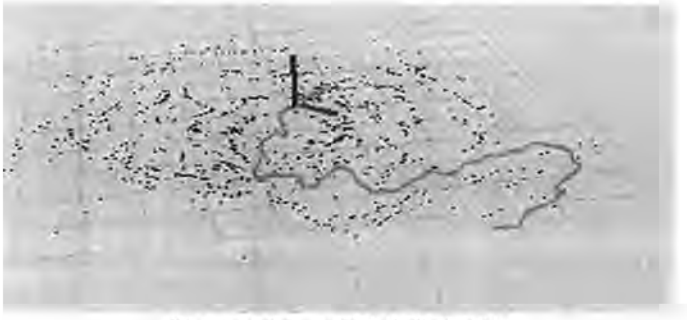


图 3.17 基于 A*算法的路径规划

Fig 3.17 Path planning based on A* algorithm

图 3.18 基于蚁群算法的改进 A*算法的路径规划

Fig 3.18 Path planning of improved A* algorithm based on ant colony algorithm

3.4 局部避障路径规划

基于地图构建的导航试图解决没有环境先验信息的机器人运动问题。机器人首先探索周围环境并建立一个内部描述，之后再利用该内部描述进行导航任务。对于机器人而言，感知区域外的未知环境中障碍物分布状态可以当做一个随机事件。在位置环境下，局部规划应当将有目的性的目标趋向行为与搜索的随机性行为相结合，体现目标及传感器信息对移动机器人行为的综合约束。

3.4.1 机器人局部避障的动态窗口算法

机器人局部路径规划算法根据实时传感器输入信息控制机器人运动，ROS 中主要采用的是动态窗口法（Dynamic Window Approach[51]）。动态窗口算法的基本思想是在速率(v,w)空间中采样多组速率，并在一定采样时间（sim_period）内，模拟机器人的运动轨迹。对得到的这些轨迹进行评价，评分最高的轨迹被选取为最优轨迹，最优路径所对应的速度为最终驱动机器人运动的速度。算法步骤如下：

(1) 机器人运动模型建立

3.1 节已阐述，这里不再赘述。

(2) 速度采样

在速度(v,w)的二维空间中会存在无穷多组速度，但可以将采样速度控制在一定范围内由于机器人自身的限制和环境限制：

① 机器人会受到来自自身最大、最小速率的限制为：

$$V_m = \{v \in [v_{min}, v_{max}], w \in [w_{min}, w_{max}]\} \tag{3.45}$$

② 机器人自身也会受到电机性能的影响：

在 sim_period 周期内，电机力矩的有限致使机器人有最大的加减速限制，在一

定时间范围内存在一个动态窗口，在该窗口内的速率是机器人能够实际达到的速率：

$$V_d = \{(v,w)|v \in [v_c - v_b\Delta t, v_c + v_a\Delta t] \wedge w \in [w_c - w_b\Delta t, w_c + w_a\Delta t]\} \quad (3.46)$$

其中，v_c, w_c是机器人的当前速度。

③ 考虑到机器人的安全方面：

为了防止碰到障碍物，在最大减速度前提下，速率范围限制在：

$$V_a = \{(v,w)|v \leq \sqrt{2 * dist(v,w) * v_b} \wedge w \leq \sqrt{2 * dist(v,w) * w_b}\} \quad (3.47)$$

其中，$dist(v,w)$为速度(v,w)对应轨迹上离障碍物最近的距离，如图 3.19 所示。这个条件在采样一开始是未知的，需要模拟出来机器人轨迹以后，找到障碍物的位置，计算出机器人到障碍物之间的距离，然后看当前采样的这对速度是否在碰到障碍物之前停止，如果能够停止，则这对速度就是可接受的，若是停止不了，则这对速度就得丢弃掉。

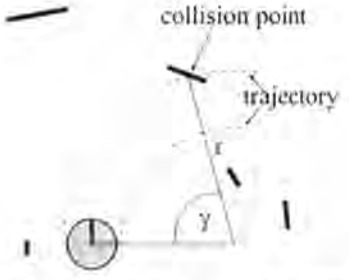


图 3.19 机器人离障碍距离示意图

Fig 3.19 Sketch map of robot distance obstacle distance

(3) 评价函数

① 评价函数

由于有若干条轨迹是可行的，因此采取评价函数的方法对每条轨迹进行评价得分。评价函数定义如下：

$$G(v,w) = \sigma(\alpha * heading(v,w) + \beta * dist(v,w) + \gamma * velocity(v,w)) \quad (3.48)$$

其中，$heading(v,w)$表示角度差距，具体是指机器人在当前设定的速率下，到达模拟轨迹末端的朝向和目标之间的角度差距。机器人航向夹角示意图如图 3.20 所示。采取 180-θ的方法来评价，使θ与评价得分成负相关。$velocity(v,w)$用来评价当前轨迹的速度大小。

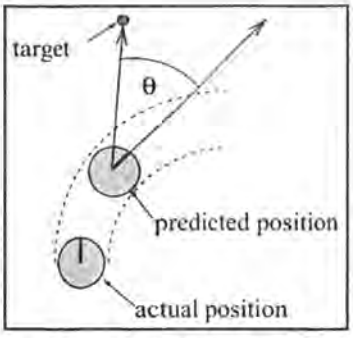


图 3.20 机器人航向夹角示意图

Fig 3.20 Schematic diagram of the angle of the robot course

② 平滑处理

平滑处理也叫做归一化处理。以上三个部分计算出来以后进行归一化再相加。有如下公式：

$$\text{normal\_head(i)} = \frac{head(}{\sum_{i=1}^{n} head(i)} \tag{3.49}$$

$$\text{normal}_{\text{dist(i)}} = \frac{dist(i)}{\sum_{i=1}^{n} dist(i)} \tag{3.50}$$

$$\text{normal}_{\text{velocity(i)}} = \frac{velocity(i)}{\sum_{i=1}^{n} velocity(i)} \tag{3.51}$$

其中，n 为采用的所以轨迹，i 为待评价的当前轨迹。

3.4.2 实验结果与分析

在 ROS 平台下，利用 ArbotiX Simulator 仿真环境，模拟局部避障的动态窗口算法实验如图 3.22 所示，其中黄色的矩形块为障碍物，外面的浅绿色为膨胀区域，黄色箭头指向的方向表示生成的多条路径，绿色实线代表评价后最终选择的路径，左边绿色箭头表示起始点，右侧绿色箭头表示终点。由仿真实验图可以看出，机器人可以在众多路径中评价出一条最佳轨迹，局部避障的动态窗口算法可以有效地的避开障碍物，到达目的地。

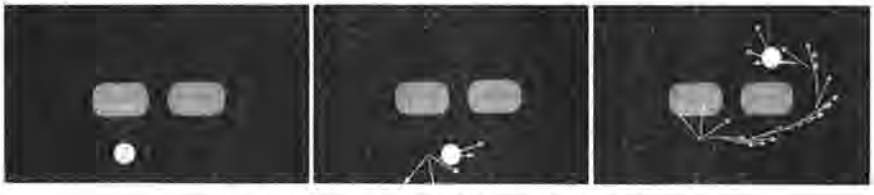


图 3.21 局部避障的动态窗口算法仿真实验

Fig 3.21 Simulation experiment of dynamic window algorithm for local obstacle avoidance

3.5 本章小结

本章主要阐述了移动机器人自主导航的基本算法。首先建立了动态机器人模型，然后对导航地图做了具体分析，为了在2D路径规划下避开3D障碍物，本文提出融合的2.5D导航地图模型，将障碍物映射到2D栅格地图上，并做了实际实验。然后讲解了基于粒子滤波的蒙特卡罗定位算法，做了仿真实验。接着阐述路径规划基本算法，并改进了基于A*算法的路径规划算法，做了仿真实验。最后，研究了局部避障的动态窗口算法，并做了仿真实验进行验证。

第4章 基于分布式架构的云计算系统

传统的 SLAM 多采用集中式的架构，具有执行效率低、硬件要求高、资源难以有效利用等缺点。分布式计算[52]是使用多个处理器单元或存储单元处理任务的并发计算过程，可以较好地执行计算密集任务。因此，在前人研究的基础上，依托云计算的优势，结合 ROS 平台和 Scoket 通信机制，本文提出了一种基于分布式架构的云计算 SLAM 系统，通过在云端部署基于 Hdoop 的分布式框架对 SLAM 算法进行处理，将复杂运算移至云端执行，极大地加快了 SLAM 的运行效率，减轻了本地机器人对硬件的依赖。

4.1 云计算

4.1.1 基本概念与分类

云计算[3,53]是一种依靠互联网的计算方法，基于这种方法共享的软硬件资源根据不同的需要将服务提供给各种终端，这些资源例如服务器、存储、应用软件和服务等，云计算同时也是分布式计算、网络存储、虚拟化等传统计算与网络技术融合发展的最终产物。

云计算有三个层次的服务：基础设施即服务（IaaS），平台即服务（PaaS）及软件即服务（SaaS）。架构图如图 4.1 所示。

图 4.1 云计算架构图

Fig 4.1 The architecture of cloud computing

第一个层次:基础设施即服务（IaaS）是使硬件基础设施可用作为一个服务，消费者通过互联网可以从完善的计算机基础设施获得服务。例如亚马逊 EC3/S3 等。

第二个层次:平台即服务（PaaS）是指将软件开发的平台作为一种服务，使用户得到一个平台在硬件基础设施。因此，PaaS 也是 SaaS 模式的一种应用。例如 Google Application Engine 等。

第三个层次:软件即服务（SaaS）是通过互联网提供软件的模式，用户不必购买软件，而是向提供商租用基于 Web 的软件。例如， Google Docs 是这样一个服务的例子。

4.1.2 云计算的优势

云计算的优势有：

(1) 虚拟化

云计算可以支持用户在任意地点、利用各式各样的终端设备访问应用服务。所获取的资源全部来自虚拟化服务器，而不是固有的物理实体。

(2) 可靠性高

云使用了数据多副本容错、节点同构可交换等技术确保服务的高可靠性，这些机制令云计算与本地计算相比拥有更可靠的性能。

(3) 可扩展性高

云能够动态伸缩，完全能够满足各种应用和用户规模增长的需求。

(4) 价格低廉

因为云可以采用极其廉价的节点来构成云，而且云的通用性使资源的利用率比传统系统显著提高，所以用户可以任意享受云的低成本带来的优势，用户常常仅仅花销几千元就能够实现之前需要数十万元才能实现的任务。

4.2 基于分布式架构的云计算系统

机器人要执行如定位、避障、路径规划、视觉处理和环境重构等主要任务，这些任务如视觉处理和环境重构的计算量是巨大的，虽然当前处理器的速度越来越快，这些任务可以实时的在基于 X86 架构的平台上完成，但是这些平台体积相对较大、计算能力有限且不易扩展、成本较高而且需要较高的电能消耗，需要外接昂贵并且体积较大的电池，不仅限制了机器人的外观设计，增加成本而且不利于做长时间的在线处理。此外，当一个新的机器人进入同样的环境，它将再次复制所有动作再次探索和构建自己的地图，这使得系统非常低效。本文利用云计算的优势，将部分计算分布到云计算平台上，云计算平台又是一个分布式运算系统，这样不仅解

决了计算能力扩展、提高运算效率、节约了成本而且实现了资源共享。

4.2.1 基于分布式架构的云计算系统架构设计

本文提出的基于分布式架构的云计算系统架构，包括了分布式 ROS[54]架构、Hadoop 分布式文件系统（HDFS[55]）和 Hadoop Map/Reduce[56]框架。ASW(Amazon Web Services)是 Amazon 提供的完整的云计算服务，提供了可靠性的基础设施，可用来实现高可扩展性的网络服务，减少维护及管理成本，本文的云计算平台基于亚马逊的 EC2（亚马逊弹性计算网云）。运行本地 ROS 节点收集传感器数据，提取关键帧，利用 ROS 发送消息传递机制，将关键帧数据包经过 Socket 网络传输机制输入云端，实现云端与本地的通讯。云端接到数据，再次利用 ROS 消息机制，将数据发布给后端 HDFS 文件系统，通过在云计算平台上自行构建实验所需要的环境，部署所需要的算法，在 Hadoop 集群通过 Map/Reduce 完成算法处理。系统架构图如图 4.2 所示。

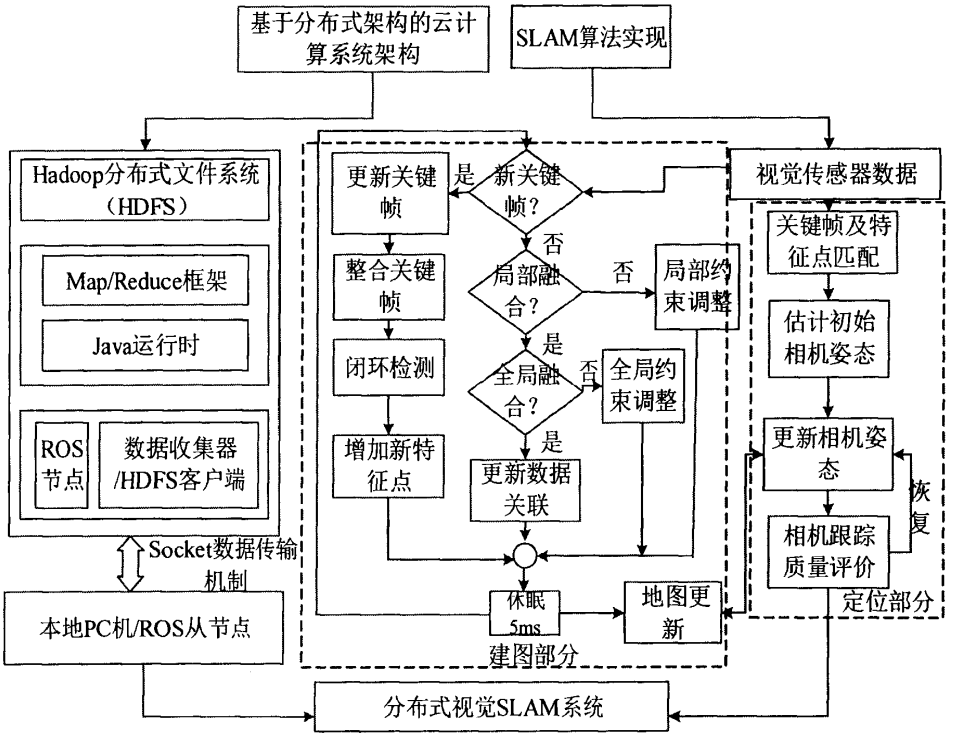


图 4.2 基于分布式架构的云计算系统架构

Fig 4.2 Cloud computing system architecture based on Distributed architecture

4.2.2 ROS 操作系统及消息机制

ROS[54](Robot Operating System)是一个开源的机器人操作系统，能够为异质计算机集群表达近似操作系统的功能，即是一个机器人软件平台。ROS 提供了大量的常用于机器人系统的硬件设备驱动、库、可视化程序及数据通信程序，有力地提高了机器人系统开发的效率，方便了科研人员对机器人系统的研究。ROS 也是一个松散耦合的分布式平台，为用户提供了一个灵活的模块化通讯机制。在基于 ROS 的开发中，一个大型项目常常会被分为几个小项目独立的进行开发，每一个小项目均以包（Package）的形式存在，而每个包运行的进程被称之为节点（Node），通过话题（Topic）和其它节点的消息相互通信，交换节点间消息，消息传递系统是基于松散耦合的。

ROS 通过核（Core）的内核管理这些节点和话题。当一个节点启动后，它会向核（Core）注册自己的名称，并将自己需要发布和接收的话题提交给 Core，后者则负责维护节点和话题之间的关联列表。每当有节点启动或结束，Core 均会更新列表以确保关联的准确性。基于这样的机制，一个节点可以发布和订阅多个话题，而一个话题也可以被多个发布者发布和订阅，并且发布者和订阅者之间并不需要知道对方的存在。因此，不同节点负责不同的功能，共同完成任务，如在导航中，不同节点负责收集传感器数据，控制电机和运行定位算法等，共同完成导航任务。这个机制的目的就是把信息的产生和消费之间进行解耦。在逻辑层面上，话题可以被当作一个传递消息的总线。消息通信机制示意图如图 4.3 所示。

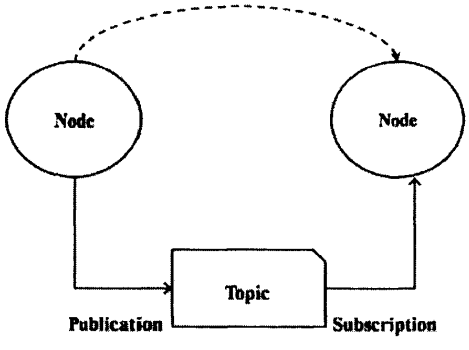


图 4.3 ROS 消息通信机制

Fig 4.3 Message communication mechanism based on ROS

4.2.3 HDFS 集群

HDFS[55](Hadoop Distributed File System）是一个主从结构，是由一个

NameNode(名字节点)的主服务器负责管理文件命名空间和客户端访问文件。另有一些数据节点，进行管理对应节点的存储。除此之外，还提供了一个可靠的，可扩展的分布式计算平台。Hadoop 是一个基于 java 的支持在大型计算机集群上运行的数据密集型分布式应用程序的框架。

HDFS 内部操作是将一个文件分割成一个或多个的块的，数据节点存放着这些块，名字节点用来对文件命名空间的文件或目录进行操作，同时确定块与数据节点的映射关系。HDFS 的架构如图 4.4 所示。

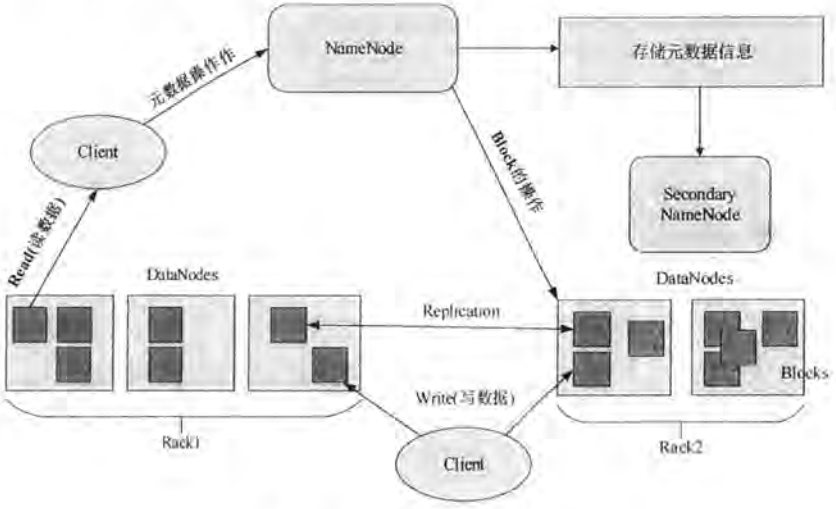


图 4.4 HDFS 架构

Fig 4.4 The architecture of HDFS

NameNode 是 Master 节点，负责管理数据块映射，配置副本策略，管理 HDFS 的名称空间，处理客户端的读写请求等。Secondary NameNode 负责分担 NameNode 的工作量，归并 fsimage（元数据镜像文件）和 fsedits（元数据的操作日志）然后再发给 NameNode。DataNode 是 Slave 节点，负责存储 client 发来的数据块 block，执行数据块的读写操作。

实验平台使用 Hadoop 分布式文件系统（HDFS）存储关键帧数据，集群包含计算节点和存储。

4.2.4 MAP/Reduce 框架

Map/Reduce[56]adoop 的核心组件之一，是一种编程模型，也就是说可以通过 Map/Reduce 很容易在 Hadoop 平台上进行分布式的计算编程。Map/Reduce 用于执行多个计算任务的机制，在多个节点上的并行运算减少了计算密集型的处理或执

行时间。图 4.5 显示了在 Hadoop 执行 Map/Reduce 的基本原理。Map 任务是以键/值对形式处理一个输入列表。Reduce 任务是合并 Map 任务的结果。这些任务可以并行运行在一个集群。每一个 Map/Reduce 任务可以运行在群集上的单个节点上运行。

实验中，Map 任务负责读取全部关键帧，执行定位算法，输出键/值 map<帧 ID，变换矩阵>, Reduce 任务是对进行点云拼接，最终输出拼接好的点云地图。

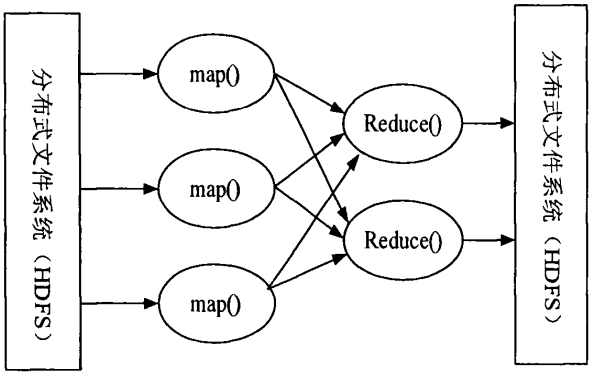


图 4.5 Hadoop Map/Reduce 框架
Fig 4.5 The sketch of Hadoop Map/Reduce

4.3 关键帧选取策略

关键帧选取的必要性是在本地采集图像数据时，把每一帧都上传到云端是不明智的，不但传输耗时耗资源而且在地图拼接时因为帧与帧之间距离很近，将会造成地图的不断更新，导致浪费时间及空间。因此，在本地将数据进行关键帧处理后，最终只需把关键帧传到云端进行地图更新即可。

4.3.1 关键帧选取过程

初始化第一帧为关键帧，将当前帧与前一关键帧进行配准，并计算与上一关键帧的运动，估计该运动的大小 e。配准算法第二章已具体阐述，这里不再赘述。配准得到旋转矩阵 r 和平移矩阵 t，然后将旋转与平移的范数加和，度量运动大小值的计算如下：

$$e = \min(2\pi - \|r\|, \|r\|) + \|t\| \tag{4.1}$$

设定匹配阈值 $goodmatch$，度量运动大小阈值 E_{error} 和 E_{key}。

(1) 若匹配的特征点数少于 $goodmatch$，表示该帧图像质量不高，则放弃该帧；

(2) 若 $e > E_{error}$，表示运动太大，则放弃该帧；

(3) 若 $e < E_{key}$，表示距离前一个关键帧太近,则放弃该帧。

最后，只有当特征匹配良好，运动估计准确，而且又距前一个关键帧有一定距离，才将该帧作为新的关键帧，传送到云端。

4.3.2 实验结果与分析

在保证云端建图质量的情况下，传输的关键帧越少，对网络的压力越小，地图拼接更新的频率就会越小，就会越节约时间与空间。针对矿电楼208场景，通过大量的实验，反复调整关键帧的三个阈值的大小，最终选取的数值大小如表3所示，最终获取的关键帧数量如表4所示。

表3 关键帧阈值选择

Table 3 Threshold selection of key frames

阈值类别	$goodmatch$	E_{error}	E_{key}
大小	4	0.3	0.06

表4 关键帧提取数量

Table 4 Number of key frame extraction

图像类型	采集的帧数量	关键帧数量
彩色图	782	161
深度图	782	161

4.4 基于 Socket 的网络传输机制

网络上两个程序依据一个双向的通信连接完成数据的交换，这个连接的一端称为一个 Socket。Socket 是一种特殊的文件，一些 Socket 函数就是对如读/写 IO、打开、关闭等的操作。

4.4.1 OSI 七层网络通信模型

OSI[57]（Open System Interconnection，开放系统互连）网络模型，称为开放式系统互联参考模型。该模型为网络功能结构提供了一种普适框架，该模型在逻辑上共分成了七层，其最本质的功能是帮助不同类型的主机实现数据传输。其示意图如图

4.6 所示。

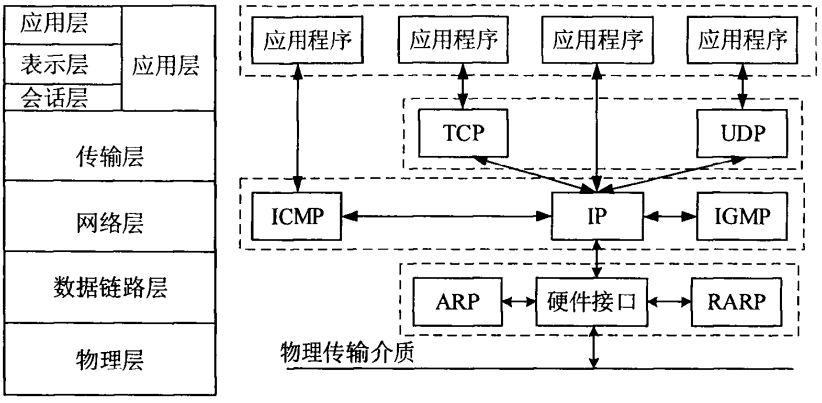


图 4.6 OSI 七层网络通信模型 OSI

Fig 4.6 Seven layer network communication model of OSI

第一层：物理层

规定通讯设备机械、电气等的特性，目的是建立、维护和拆除物理链路连接。主要功能是将信号以二进制数据形式在物理介质上传输数据，协议如 ISO2110、IEEE802、IEEE802.2 等。

第二层：数据链路层

负责物理层面上互连及节点之间的通信传送。主要功能是保证有地址的帧能有效的传输以及错误检测，协议如 HDLC、ARP、RARP 等。

第三层：网络层

将数据传输到目标地址。主要功能是为数据包寻址及选择路由，协议如 ICMP、IGMP、IP 等。

第四层：传输层

通过错误校正和流控制方式提供可靠且有序或向无连接的的传送，管理网络中端到端的信息传送。主要功能是提供端到端的接口，协议如 TCP、UDP 等。

第五层：会话层

为应用之间提供建立和维护通信的机制，包括访问验证和会话管理。主要功能是通过向两个实体的表示层提供建立和使用连接的方式来管理数据交换。

第六层：表示层

解析来自应用层的信息，给所有信息赋予相应的含义及转换成适于网络传送的格式。主要功能是数据格式化、数据加密、代码转换等。

第七层：应用层

提供网络应用程序访问的网络服务接口及标准化应用程序中通信细节。主要功

能是直接对用户提供服务，协议如 Telnet、FTP、HTTP 等。

4.4.2 TCP 数据传输机制

TCP[58](Transmission Control Protocol)属于传输层协议，提供诸如数据流传送、可靠性、有效流控、全双工操作和多路复用等的服务。依靠面向连接、端到端的可靠的数据包发送。

在数据传送时，服务器端率先初始化 Socket，接着将其与端口号进行绑定，然后监听端口，接下来使用 Accept()函数阻塞，等候客户端链接。如果这时有一客户端初始化一个 Socket，最后接着连接服务器，在连接成功之后就建立了客户端与服务器端连接。这时客户端发起数据请求，同时服务器端接受并处理请求，然后把回应数据传送客户端，此时客户端读取数据，数据传输结束后关闭连接，一次交互完结。原理示意图如图 4.7 所示。

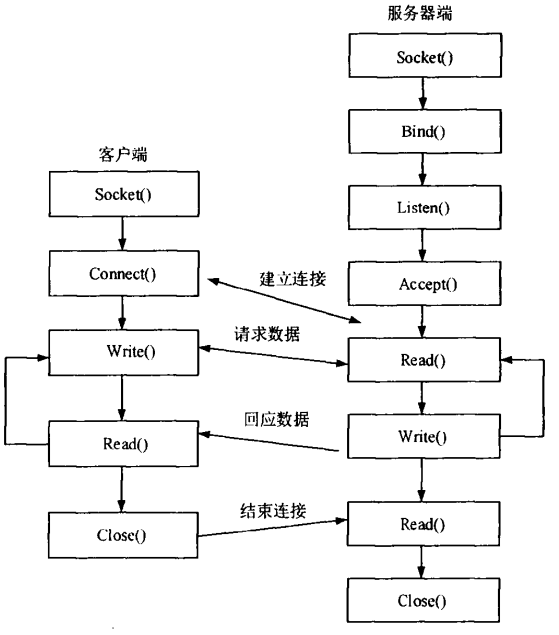


图 4.7 TCP 数据传输机制

Fig 4.7 Data transmission mechanism of TCP

TCP 连接的三次握手，其示意图如图 4.8 所示：

(1) 假设客户端向服务器端发送建立连接请求；

(2) 服务器端回馈连接请求并向客户端发送连接请求；

(3) 客户端确认已经连接。

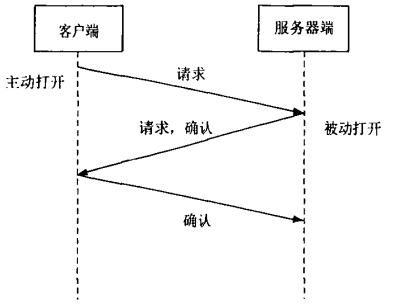


图 4.8 三次握手示意图

Fig 4.8 Schematic diagram of the three handshake

由于 TCP 是全双工工作机制，所以每个方向都需要独自进行关闭。

TCP 断开连接的四次握手，其示意图如图 4.9 所示：

(1) 假设客户端向服务器端发起断开请求；

(2) 服务器端接收请求，并确认，断开客户端方向的连接；

(3) 服务器端向客户端发送断开请求；

(4) 客户端接受请求，并确认，断开服务器端方向的连接。

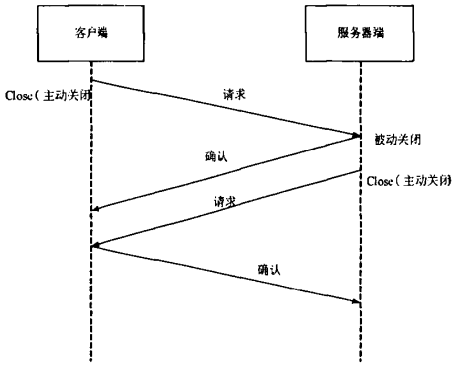


图 4.9 四次握手示意图

Fig 4.9 Schematic diagram of the four handshake

4.5 本章小结

本章讨论了基于分布式计算的云计算系统架构系统设计。首先阐述了分布式计算与云计算的基本概念，对云计算进行了深入探讨。接着介绍了系统环境的搭建原理。一方面，为了解决建图的时间与空间问题，另一方面也为了更快的传输，数据传输采取了关键帧策略，并对实验数据做了具体分析。最后阐述了网络传输的机制。

第5章 移动机器人系统实验及分析

本章设计了基于ROS的移动机器人系统平台，通过整合前面章节的SLAM算法、机器人定位、避障与路径规划等算法，实现移动机器人的自主3D建图与避开3D障碍物，使得移动机器人能够自主探索未知区域，以及实现局域网分布式实时SLAM和在云端实现3D构图。

5.1 基于ROS的移动机器人平台

5.1.1 移动机器人软件平台

移动机器人的软件平台基于ROS平台，ROS包含多个版本，而且更新频率较高。为确保稳定性和兼容性，本文采用了比较经典的Indigo版本，操作系统也选用了稳定的Ubuntu 14.04 LTS版本。在基于ROS的项目开发中，一个大型项目常常被划分为几个小项目独立开发，每个小项目均以包（Package）的形式存在，基于这样的机制，一个项目便能被设计成用几个模块化程度很高的子项目组成。

本课题设计的机器人系统便遵循模块化的项目组织结构，将整个系统分成若干个模块，每个模块独立编程、独立调试。ROS的模块化组织结构示意图如图5.1所示。

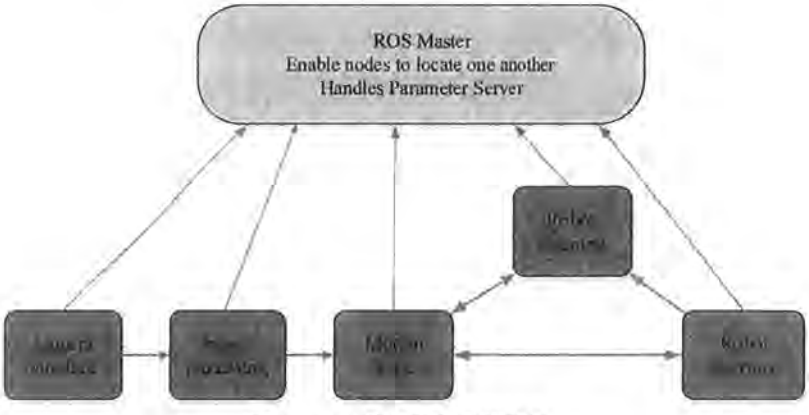


图5.1 ROS的模块化组织结构

Fig 5.1 Modular organizational structure of ROS

5.1.2 移动机器人硬件平台

移动机器人的硬件平台使用了美国iRobot公司所生产的iRobot Create机器人

平台。Create 是基于 iRobot 公司的扫地机器人 iRobot Roomba 制造的，保留了 Roomba 的运动控制器和电源管理系统。其最大的特点是支持开发，根据官方提供的通信协议，可以控制 Create 底盘的运动和码盘数据的读取。该机器人底盘具有运行稳定、续航时间长等特点，非常适合用于移动机器人算法的研究。机载计算机采用 Intel 的第五代 NUC，拥有一个固有机身设计，内置第五代智能英特尔酷睿™ i5-6260U 双核处理器，锐炬™ 540 显卡，支持系统扩展，尺寸大小为 *115mm*111mm*48mm*,既小巧又功能强大，是理想的机载运算处理控制器。数据采集传感器采用微软的 Kinect 深度相机。整个移动机器人平台如图 5.2 所示。

图 5.2 移动机器人平台

Fig 5.2 Mobile robot platform

5.1.3 移动机器人系统总体架构

结合本课题所要进行的 3D 自主地位与建图、导航系统等方面的研究，遵循模块化的项目组织结构，所构建的移动机器人系统由四个模块组成，包括信息采集模块、数据处理模块、运动控制模块和远程监控模块，移动机器人实验平台系统架构如图 5.3 所示。

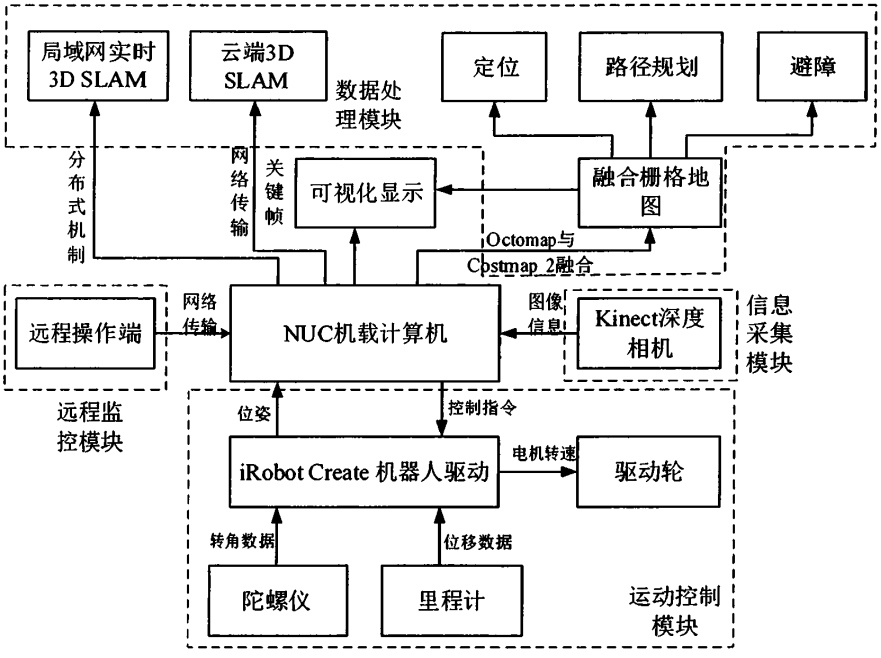


图 5.3 移动机器人系统架构

Fig 5.3 The Architecture of Mobile Robot System

在图 5.3 所示的移动机器人系统中，数据采集模块主要为 Kinect 深度相机。Kinect 将环境信息实时发送给移动机器人的控制核心(NUC 机载计算机)，在运动控制模块中，使用光电编码器记录系统轮子所转过的角度，从而推算机器人的位姿。编码器具有严重的累积误差，因此该系统中加入 ADXRS613 高精度单轴陀螺仪，通过扩展卡尔曼滤波多传感器数据融合算法，降低编码器的积累误差。同时，机器人向电机发送电机转速控制指令。远程控制模块主要由远程操作员端构成，通过互联网与移动机器人平台的机载计算机相连，进行控制指令的发送和机器人数据的显示，实现远程与机器人进行交互。该系统的核心为数据处理模块，其中包括导航地图融合、定位、避障、路径规划、局域网分布式 SLAM 和通过将关键帧数据通过网络传输到云端进行环境建模等功能，这些功能模块由前面章节所述算法进行开发。

5.2 基于 ROS 移动机器人导航系统

同步定位与建图与导航结合是移动机器人实现完全自主移动的关键，系统采用 ROS 的导航栈(Navigation Stack)来实现导航系统的基本架构，通过设置 TF 坐标变换系统、栅格地图等即可完成导航栈的基本配置。

5.2.1 系统坐标变换

ROS 导航栈需要使用 TF（transforms frames）坐标变换系统来发布机器人的坐标变换树（系统中不同坐标系之间的偏移），本质上 TF 的作用是完成系统中任一个点在全部坐标系之间的坐标转换，所谓坐标转换就是把一个点在某个坐标系的描述，变换成在另外一个坐标系下的描述。通俗讲，如果给定某坐标系下的一个点的坐标，就能得到在其他坐标系下的相应的的坐标。

如图 5.4 所示，移动机器人平台拥有移动的底盘和位于底盘上方的 Kinect 传感器。一个坐标系的原点位于机器人的底盘中心，另一个坐标系位于 Kinect 的中心。将位于底盘的坐标系定义为 base_link，将位于 Kinect 的坐标系定义为 base_kinect。因为 Kinect 采集的数据是基于 Kinect 中心来表示的，所以不能直接进行 SLAM 过程。数据的坐标变换后才能利用这些数据。如图 5.4 所示的简单模型中，已知 Kinect 测得位于在其正前方 0.3m 处有障碍物，而 Kinect 安装于底盘前方 0.1m，上方 0.2m 处。从而可以获取从 base_link 到 base_kinect 的坐标变换关系，base_link 坐标系平移(x: 0.1m, y: 0.0m, z: 0.2m)。相应的，可以得到障碍物到机器人底盘的距离，ROS 通过 TF 变换系统帮助用户完成这些变换的工作。需要事先将相应坐标添加到坐标变换树中，用户才能使用 TF 定义和存储这些变化关系，并且树的结构特点决定了 TF 没有回路，以确保两个坐标之间的变换是单向的。而且 TF 假设所有的变换都是从父节点到子节点的。假设 base_link 为父节点，因为所有传感器都是基于底盘安装的。因此，base_link 和 base_kinect 之间的变换矩阵为(x: 0.1m, y: 0.0m, z: 0.2m)。

设置完 TF 系统后，需要创建用于发布变换信息的节点，然后创建收听变换信息的节点，这些在 ROS 的 TF 系统都可以很方便地进行管理。

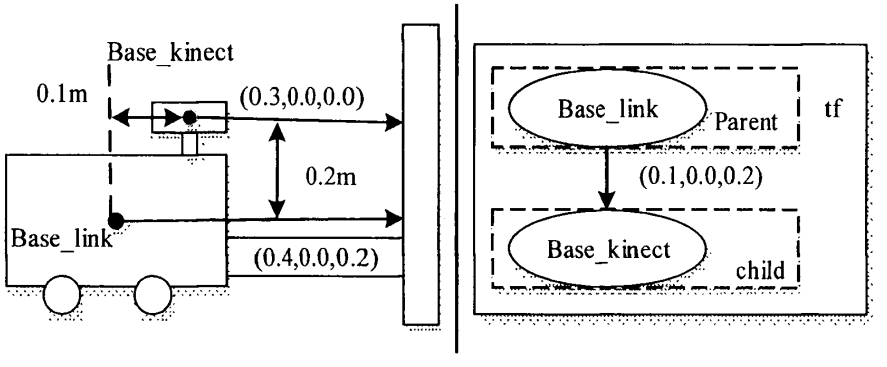


图 5.4 移动机器人平台 TF 模型示意图

Fig 5.4 Schematic diagram of TF model of mobile robot platform

5.2.2 Navigation 栈构架

ROS 的导航栈(Navigation Stack)的基本思想为：得到传感器数据和结合机器人码盘推算出的 Odometry（里程计），同时输出前进速度和转向速度命令给机器人底盘。移动机器人 ROS 导航栈配置，如图 5.5 所示，主要包括 TF 发送变换、发布里程计信息、发布传感器数据和基本的导航配置等。

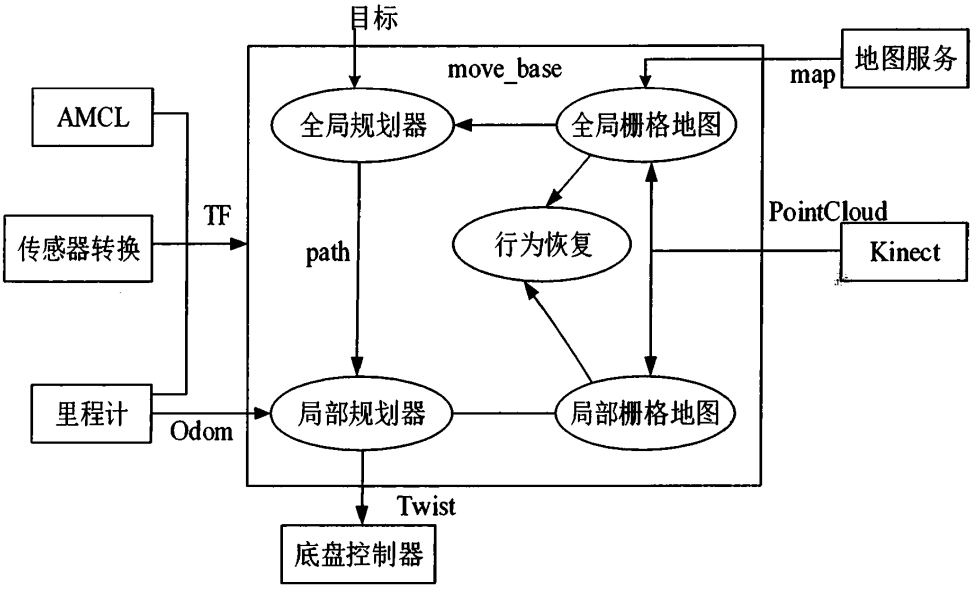


图 5.5 导航栈系统架构

Fig 5.5 The Framework of ROS Navigation Stack

导航栈需要必要的 TF 变换信息，通过 TF 系统，建立传感器、地图、机器人等坐标系关系。传感器数据获取通过订阅 sensor_msgs/PointCloud 消息进行，通过 nav_msgs/Odometry 消息来发布里程计信息。在获得环境地图之后，根据传感器数据和里程计数据利用蒙特卡罗定位（AMCL）自动定位机器人，实施路径规划，在避开 3D 障碍物的同时，以最优速度到达目标位置。对于底盘控制，导航栈利用 geometry_msgs/Twist 消息发送速度指令，其中包括线速度和角速度。该消息位于机器人坐标系的 cmd_vel 话题。 iRobot Create 底盘的驱动节点订阅 cmd_vel 话题，并将该消息转换为底盘通信协议，控制机器人的移动。底盘移动控制(move_base)是导航栈的核心组件。它的主要任务是接收用户指定的目标点，并试图到达。底盘移动控制节点通过将全局和局部路径规划相结合，不断反馈，完成导航任务。其中，全局路径规划用于对静态障碍物的规划，而局部路径规划用于对动态障碍物的避

障。

5.2.3 实验结果及分析

在实际实验中，系统中运行的节点与这些节点通过的主题实现的发布者到订阅者的连接关系，如图 5.6 所示。可以看出，系统完整的导航栈与 TF 关系及节点通过主题相连关系。

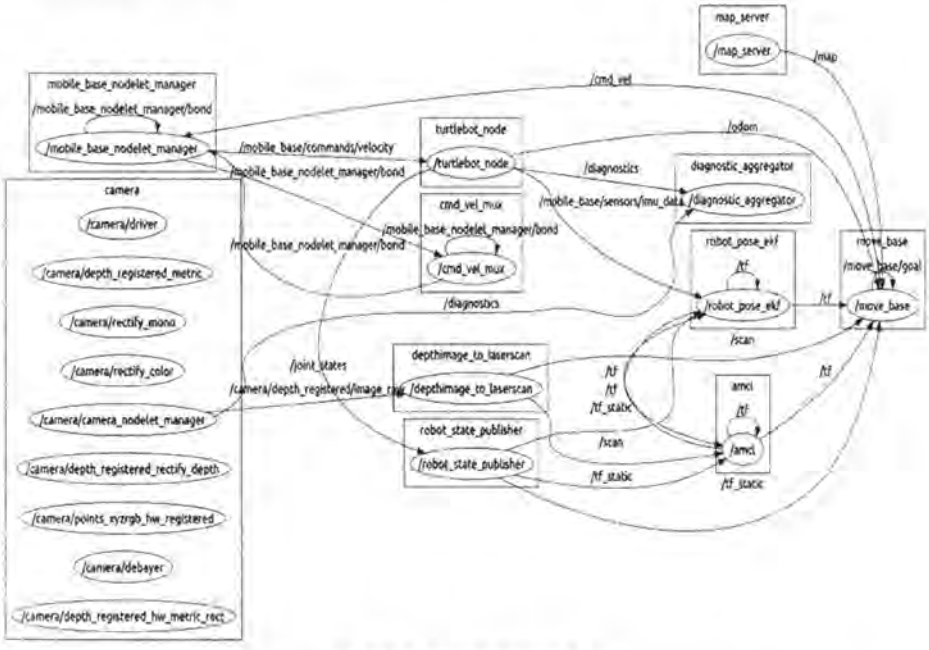


图 5.6 系统的节点间关系

Fig 5.6 The relationship between node and node in practical system

5.3 室内机器人导航实际实验

在移动机器人导航系统的配合下，以 Kinect 作为传感器，在建立的差分移动机器人模型的基础上，基于蒙特卡罗定位算法，在提出的 2.5D 导航地图上实施避障和路径规划算法，可以完成室内环境下移动机器人自主环境建模与 3D 避障实验。

5.3.1 移动机器人自主三维建图实验

在局部导航框架下，初始化机器人，不断在更新的 2.5D 地图上指定目标点，移动机器人可以绕过障碍物，自主探索室内未知区域，并进行三维环境地图的构建。如图 5.7 所示，是某机器人公司某会议室实际环境，及机器人正在自主构图的实际

场景。如图5.8所示，是对某机器人公司某会议室进行的自主三维点云地图构建效果图。

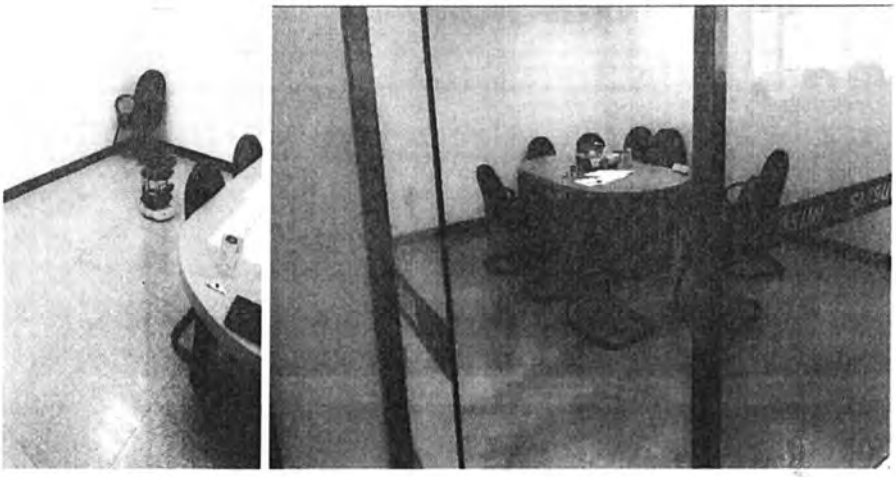


图5.7 某机器人公司某会议室实际环境

Fig 5.7 A meeting room of the actual environment of robotic company

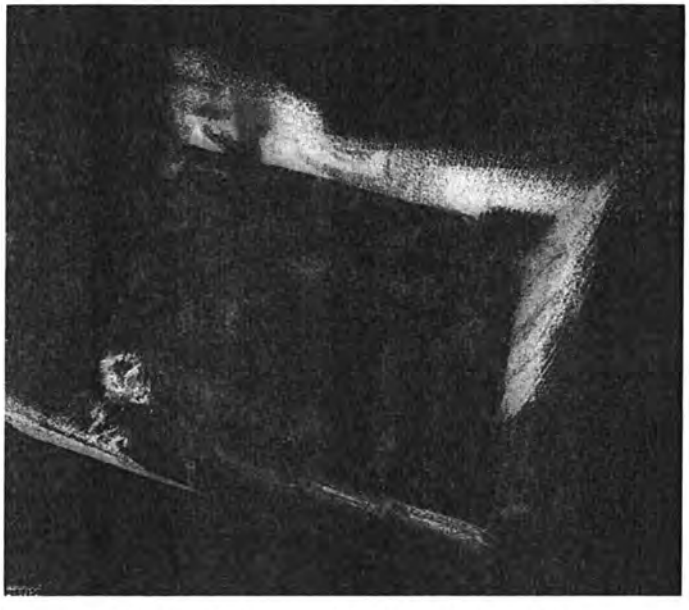


图5.8 某机器人公司某会议室三维点云地图

Fig 5.8 The 3D point cloud map of robotic company a conference room

5.3.2 移动机器人的 3D 避障实验

在全局导航和局部导航框架下，初始化机器人，机器人自主探索未知环境，并建立 2.5D 地图，在 2.5D 地图上指定机器人所要达到的目的地，移动机器人可以避开 3D 障碍物，朝着目标驶向目的地。如图 5.9 所示，在某楼 208, 移动机器人避开中空的椅子的过程，实际场景如图中（a)、(b)、(c)、(d)、(e)、(f) 所示。相应的过程在地图上的具体展示如图 5.所示。

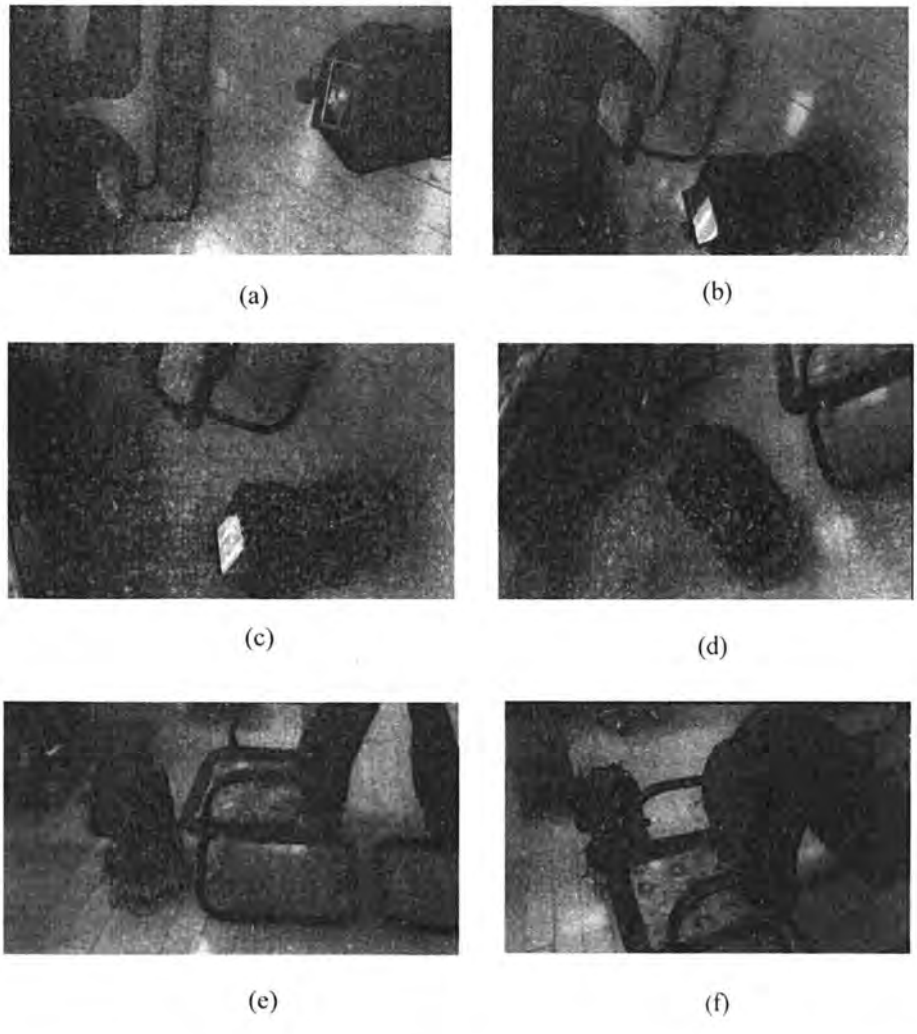


图 5.9 移动机器人避开 3D 障碍物的实际场景
Fig 5.9 The actual scene of the mobile robot to avoid 3D obstacles

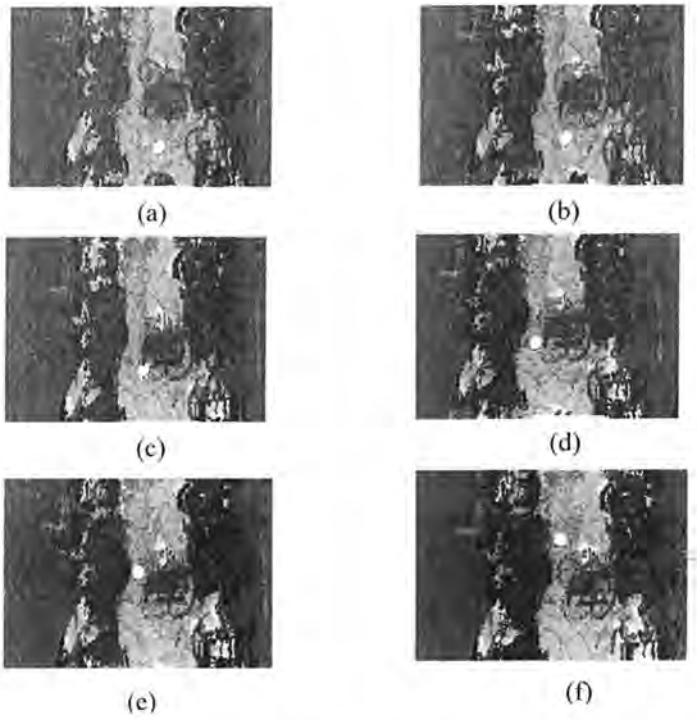


图 5.10 移动机器人避开 3D 障碍物过程在地图上的展示
Fig 5.10 Mobile robots to avoid the 3D obstacles in the process of map display

5.4 基于分布式架构的视觉 SLAM 实验

基于分布式架构的实验形式主要可以分为局域网互联和云-本地两种，为验证所提出的系统架构，本文在局域网环境里，基于 ROS 的分布式机制，做了实时视觉 3D SLAM 实验；接着，在本地提取关键帧之后传送到云端，通过部署 Hdoop 分布式框架对 SLAM 算法进行处理并生成点云数据。

5.4.1 基于分布式机制的局域网实时视觉 SLAM 实验

在局域网环境里，在本地服务器（台式电脑）运行 master 节点管理器，master 提供 topic 寻址、参数服务器等功能。客户端（机器人上机载计算机）节点通过环境变量 ROS_MASTER_URI 配置 IP 地址指向同一个 master，客户端获取数据的节点在节点管理器注册之后，这样，机器人端实时采集的数据通过 TCP/IP 通信将数据传到本地服务器进行处理。为了提高系统的实时性能，减少占用的网络带宽，降低延迟，将机器人上 Kinect 传感器节点的消息发布设定为 5HZ。并且对传输数据进行了 1/4 降采样处理，通上诉过处理，要求 25fps，实际可以达到大约 21 fps，可

以满足实时 3D 视觉 SLAM 要求。可以实现如图 5.11 所示，展示了服务器端处理的过程与实际效果，其中红框所圈的内容是绑定的 IP 地址。图 5.12 展示了带宽情况。

图 5.11 局域网实时 3D SLAM 实验

Fig 5.11 Local area network real time SLAM 3D experiment

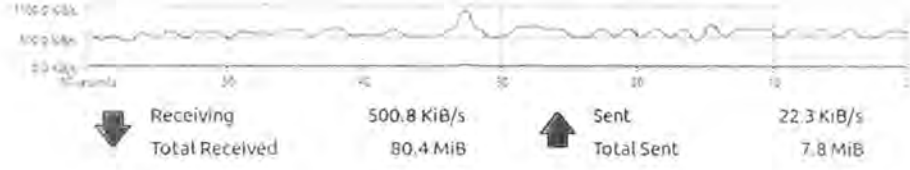


图 5.12 带宽用量

Fig 5.12 The bandwidth useage

5.4.2 基于分布式架构的云计算视觉 SLAM 实验

因为云端没有显卡，所以在云端看不到构图效果，只能将最后生成的地图的 PCD 格式的文件传到本地看。

机器人运行本地 ROS Kinect 节点，采集大约 782 帧，每帧 640*480，经过提取大约 161 帧关键帧，利用 ROS 发送消息传递机制，将关键帧数据包经过 Socket 网络传输机制输入云端。云端接到数据，再次利用 ROS 消息机制，将数据发布给后端 HDFS 文件系统，实验设置了 4 节点群集，每个节点是一个英特尔酷睿双核 4GB 的 RAM 服务器。HDFS 文件系统运行在这些节点上，MAP/Reduce 框架执行构图算法的任务，首先 MAP 过程读取全部关键帧，执行定位算法，输出 map<帧 ID，变换矩阵>,key 值帧 ID 从 1 开始顺序编号，对应的 value 为相应旋转矩阵。然后输入 Reduce 过程进行点云拼接，Reduce 的机制是反复合并相同 key 值直到没有相同的 key 值将输出，每对 key 值 ID Reduce 后，取最大 ID 加 1，与下一个 ID 形成新 key，这样就能依次进行点云拼接，而且每五帧生成一个共同的新 key 值，value 值对应点拼接的点云图，进行点云稀疏滤波处理，最终 Reduce 输出包含点云数

1006383 个的点云地图。对某楼 208 进行的云端建图效果如图 5.13 所示，其真实场景如图 2.21 所示。本地与云端不同节点数处理 SLAM 算法所需要的时间如图 5.14 所示，该算法通过在本地与在云端使用单一节点，两个节点，4 个节点的 Hadoop 集群，可以看出，在云端处理相同数量的帧所用时间比本地少，云端节点越多，处理相同数量的帧所用时间越少，并且数据量越大，越能体现出分布式计算的优势。

图 5.13 云端建图效果

Fig 5.13 The effect of building map on cloud

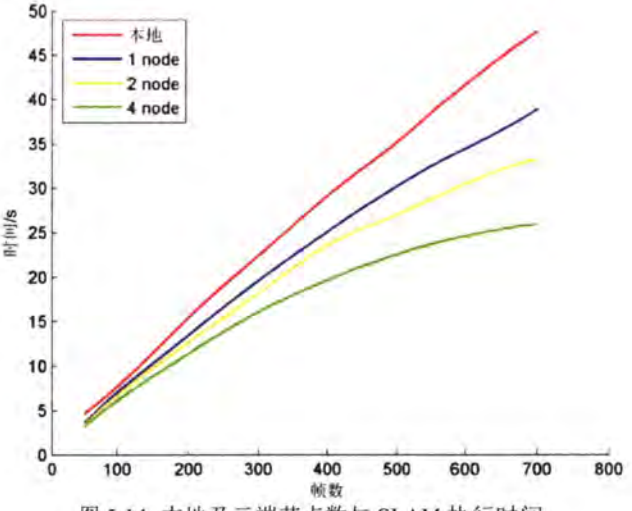


图 5.14 本地及云端节点数与 SLAM 执行时间

Fig 5.14 The execution time of number of local and cloud nodes and SLAM

5.5 本章小结

本章设计了基于 ROS 的移动机器人导航系统，通过软硬件平台的搭建，实现了移动机器人自主探索未知区域的自主 3D 建图与避开 3D 障碍物，并分别在不同

场景进行了实际实验与分析。在局域网完成了分布式实时视觉 SLAM 实验及云端实现分布式 3D 构图实验。

第6章 总结与展望

6.1 总结

本文研究了基于深度相机的视觉 3D SLAM 及机器人室内自主导航，并将 3D SLAM 算法分布到局域网及云计算平台上，以便实现所建地图便于与其他机器人共享。系统涉 3D SLAM 算法、机器人定位算法、路径规划算法、避障算法、云计算、Kinect 深度相机、ROS 软件平台、iRobot Creat 机器人硬件平台等。

现总结如下：

(1) 阐述了室内环境下 3D SLAM 算法的基本原理，重点研究了基于深度相机的稀疏特征点法的 3D SLAM 算法基本流程，如特征点检测与描述子生成、特征匹配、运动估计、闭环检测、全局姿态优化等。其中改进了误匹配算法，利用词袋模型解决了闭环检测问题，并做了实际实验验证算法的效果。

(2) 在蒙特卡罗定位算法的框架下，对机器人进行建模，深入研究室内机器人精确自主导航的方法，包括导航地图表达、全局路径规划算法与局部避障路径规算法的实现，提出 2D 与 3D 地图融合的 2.5D 导航地图模型，并改进了基于 A*的路径规划算法，使机器人可以快速地寻找出一条最优路径，作了仿真与实际实验验证算法的可行性。

(3) 在分布式计算框架下，建立了通信模型，阐述了云计算的优势、数据由本地传上云端的网络传输机制、关键帧选取策略等，在此基础上构建基于分布式计算的云计算系统，将系统的部分算法分布到云平台上计算。

(4) 在基于 ROS 软件系统平台下进行自主移动机器人的总体设计与实现，开展实际实验，验证了未知环境自主 3D 构图算法效果；避开 3D 障碍物并到达目的地效果；局域网实时 3D SLAM 与云端分布式架构的 3D 构图。

由此可见，系统实现了未知环境视觉 3D SLAM 与未知环境移动机器人自主构图、避开 3D 障碍物并到达目的地以及展现了分布式计算的 3D SLAM 的优势。

6.2 展望

本文设计的基于分布式计算的视觉 SLAM 系统和移动机器人自主导航系统，在实现云端构图、机器人定位、路径规划、避开 3D 障碍物及自主环境建模等方面取得了较为满意的结果。但在实际设计和实验过程中发现该系统也存在一些不足，需要进一步完善，主要体现在：

（1）在现有带宽条件下，由于数据传输过程网络延迟较为严重，不能保证SLAM的实时性，只能将实时采集的数据存入云端数据库，然后再进行SLAM算法的实施，而且云端没有显卡，导致不能实时看到建图效果。因此，可以对传输数据做压缩处理，选择更为有效地数据传输算法，保证SLAM算法的实时性。通过搭建Web显示平台，将所构建地图显示在网页上，可以解决云端无法显示图像的问题。

（2）在移动机器人自主3D SLAM过程中，只单纯依赖Kinect深度相机这一种传感器，由于Kinect深度相机自身的局限性，不能用于光照过于强烈的环境，而且采用基于特征值的SLAM算法，导致遇到纹理特征不明显的地方，比如白墙等，导致算法失效，使整个系统适应环境有限。因此，可以采取多传感器融合进行解决，比如IMU、廉价激光雷达等，根据不同传感器的特点，用信息矩阵进行管理，针对不同的环境对不同传感器数据设置不同权重，这样可以有效解决移动机器人的环境适应能力。

参考文献

[1] Dissanayake M W M G, Newman P, Clark S, et al. A solution to the simultaneous localisation and map building (SLAM) problem[J]. IEEE Transactions on Robotics & Automation, 2001, 17(3):229--241.

[2] Kerl C, Sturm J, Cremers D. Dense visual SLAM for RGB-D cameras[J]. 2013, 8215(2):2100-2106.

[3] Mell P, Grance T. The NIST definition of cloud computing[J]. Communications of the Acm, 2011, 53(6):50-50.

[4] Guivant J, Nebot E, Baiker S. Autonomous navigation and map building using laser range sensors in outdoor applications[J]. Journal of robotic systems, 2000, 17(10): 565-583.

[5] Wendel J, Meister O, Schlaile C, et al. An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter[J]. Aerospace Science and Technology, 2006, 10(6): 527-533.

[6] Royer E, Lhuillier M, Dhome M, et al. Monocular vision for mobile robot localization and autonomous navigation[J]. International Journal of Computer Vision, 2007, 74(3): 237-260.

[7] TARÇIN S, ÖZÜTEMİZ K B, KOKU A B, et al. Comparison of Kinect and Bumblebee2 in Indoor Environments[J]. REM, 2011: 2.

[8] Kehoe B, Patil S, Abbeel P, et al. A survey of research on cloud robotics and automation[J]. IEEE Transactions on Automation Science and Engineering, 2015, 12(2): 398-409.

[9] Sankoff G. The grammaticalization of tense and aspect in Tok Pisin and Sranan[J]. Language Variation and Change, 1990, 2(03): 295-312.

[10] Kuffner J J, LaValle S M. Space-filling trees: A new perspective on incremental search for motion planning[C]//2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011: 2199-2206.

[11] Waibel M, Beetz M, Civera J, et al. A world wide web for robots[J]. IEEE Robotics & Automation Magazine, 2011, 18(2): 69-82.

[12] Arumugam R, Enti V R, Bingbing L, et al. DAvinCi: A cloud computing framework for service robots[C]//Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010: 3084-3089.

[13] Grisetti G, Grzonka S, Stachniss C, et al. Efficient estimation of accurate maximum likelihood maps in 3d[C]//2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2007: 3472-3478.

[14] Grisetti G, Stachniss C, Burgard W. Nonlinear constraint network optimization for efficient map learning[J]. IEEE Transactions on Intelligent Transportation Systems, 2009, 10(3): 428-439.

[15] Rusinkiewicz S, Levoy M. Efficient variants of the ICP algorithm[C]//3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on. IEEE, 2001: 145-152.

[16] Engelhard N, Endres F, Hess J, et al. Real-time 3D visual SLAM with a hand-held RGB-D camera[C]//Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden. 2011, 180.
[17] Ni K,Dellaert. Multi-level submap based slam using nested dissection[C]//Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, 2010: 2558-2565.
[18] Kümmerle R, Grisetti G, Strasdat H, et al. g 2 o: A general framework for graph optimization[C]//Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011: 3607-3613.
[19] Engelhard N, Endres F, Hess J, et al. Real-time 3D visual SLAM with a hand-held RGB-D camera[C]//Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden. 2011, 180.
[20] Derpanis K G. Overview of the RANSAC Algorithm[J]. Image Rochester NY, 2010, 4(1): 2-3.
[21] Kainz B, Hauswiesner S, Reitmayr G, et al. OmniKinect: real-time dense volumetric data acquisition and applications[C]//Proceedings of the 18th ACM symposium on Virtual reality software and technology. ACM, 2012: 25-32.
[22] Newcombe R A, Izadi S, Hilliges O, et al. KinectFusion: Real-time dense surface mapping and tracking[C]//Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on. IEEE, 2011: 127-136.
[23] Chen J, Bautembach D,Izadi S. Scalable real-time volumetric surface reconstruction [J]. ACM Transactions on Graphics (TOG), 2013, 32(4): 113.
[24] Endres F, Hess J, Sturm J, et al. 3-D mapping with an RGB-D camera[J]. IEEE Transactions on Robotics, 2014, 30(1): 177-187.
[25] 王全玉, 贾金苗, 赵连翔,等. 机器人云操作平台的人机交互技术研究[J]. 华中科技大学学报(自然科学版), 2012(S1):254-257.
[26] 陈宏兴, 周风余, 田天,等. 服务机器人云计算平台 SOA 接口层模型设计[J]. 山东大学学报(工学版), 2015, 45(4):31-39.
[27] 张展宇. 基于扩展卡尔曼滤波器的单目视觉 SLAM 研究[D]. 南开大学, 2008.
[28] Bay H, Ess A, Tuytelaars T, et al. Speeded-up robust features (SURF)[J]. Computer vision and image understanding, 2008, 110(3): 346-359.
[29] 陈晓明, 蒋乐天, 应忍冬,等. 基于 Kinect 深度信息的实时三维重建和滤波算法研究[J]. 计算机应用研究, 2013, 30(4):1216-1218.
[30] PrimeSense[EB/OL].http://www.primesense.com/solutions/technology/.
[31] Lowe, David. Distinctive Image Features from Scale-Invariant Keypoints[J], International Journal of Computer Vision, 2004, 60(2): 91-110.
[32] Burt P, Adelson E. The Laplacian pyramid as a compact image code[J]. IEEE Transactions on communications, 1983, 31(4): 532-540.
[33] Beis J S, Lowe D G. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces[C]//Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on. IEEE, 1997: 1000-1006.
[34] Bentley J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM, 1975, 18(9): 509-517.
[35] Andrieu C, De Freitas N, Doucet A, et al. An introduction to MCMC for machine

learning[J]. Machine learning, 2003, 50(1-2): 5-43.
[36] Quan L, Lan Z. Linear n-point camera pose determination[J]. IEEE Transactions on pattern analysis and machine intelligence, 1999, 21(8): 774-780.
[37] Yang Y, Newsam S. Bag-of-visual-words and spatial extensions for land-use classification[C]//Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems. ACM, 2010: 270-279.
[38] Peng X, Wang L, Wang X, et al. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice[J]. Computer Vision and Image Understanding, 2016.
[39] Zhang Y, Jin R, Zhou Z H. Understanding bag-of-words model: a statistical framework[J]. International Journal of Machine Learning and Cybernetics, 2010, 1(1-4): 43-52.
[40] Norouzi M, Fleet D J, Salakhutdinov R R. Hamming distance metric learning[C]//Advances in neural information processing systems. 2012: 1061-1069.
[41] Marquardt D W. An algorithm for least-squares estimation of nonlinear parameters[J]. Journal of the society for Industrial and Applied Mathematics, 1963, 11(2): 431-441.
[42] Moré J J.The Levenberg-Marquardt algorithm: implementation and theory [M]//Numerical analysis. Springer Berlin Heidelberg, 1978: 105-116.
[43] Thrun S. Learning occupancy grid maps with forward sensor models[J]. Autonomous robots, 2003, 15(2): 111-127.
[44] Wurm K M, Hornung A, Bennewitz M, et al. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems[C]//Proc.of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation. 2010, 2.
[45] Hornung A, Wurm K M, Bennewitz M, et al. OctoMap: An efficient probabilistic 3D mapping framework based on octrees[J]. Autonomous Robots, 2013, 34(3): 189-206.
[46] Meagher D. Geometric modeling using octree encoding[J]. Computer graphics and image processing, 1982, 19(2): 129-147.
[47] Thrun Sebastian, Fox Dieter, Burgard Wolfram, Dellaert Frank. Robust Monte Carlo localization for mobile robots[J], Artificial Intelligence, 2001, 128(1): 99-141.
[48] Misa T J, Frana P L. An interview with Edsger W. Dijkstra[J]. Communications of the Acm, 2010, 53(8):41-47.
[49] Yao J, Lin C, Xie X, et al. Path planning for virtual human motion using improved A* star algorithm[C]//Information Technology: New Generations (ITNG), 2010 Seventh International Conference on. IEEE, 2010: 1154-1158.
[50] 宋雪梅, 李兵. 蚁群算法及其应用[M]. 哈尔滨工业大学出版社, 2006.
[51] Seder M, Petrovic I. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles[C]//Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007: 1986-1991.
[52] Peleg D. Distributed computing[J]. SIAM Monographs on discrete mathematics and applications, 2000, 5.
[53] 张恒, 刘艳丽, 刘大勇. 云机器人的研究进展[J]. 计算机应用研究, 2014, 31(9):2567-2575.
[54] Quigley M, Conley K, Gerkey B, et al. ROS: an open-source Robot Operating System[C]//ICRA workshop on open source software. 2009, 3(3.2): 5.

[55] Borthakur D. The hadoop distributed file system: Architecture and design[J]. Hadoop Project Website, 2007, 11(2007): 21.
[56] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
[57] Nakai S. Hierarchically distributed network management system using open system interconnection (OSI) protocols: U.S. Patent 6,105,061[P]. 2000-8-15.
[58] Adelman K A, Kashtan D L, Palter W L, et al. Method and apparatus for a TCP/IP load balancing and failover process in an internet protocol (IP) network clustering system: U.S. Patent 6,078,957[P]. 2000-6-20.

致谢

时光飞逝，两年半的研究生求学生活即将结束，在论文完成之际，谨向我的导师、同学和亲朋好友致以诚挚的感谢。

这期间要感谢我的导师吴成东教授与张云洲教授。在本论文的选题、结构安排、研究方法及提纲的拟定等各个方面，给予了我最无私的指导和帮助，使得本文最终能够定稿。吴老师与张老师博大精深的学识、锐意创新的意识、严谨求学的态度，都给学生留下了深刻的印象，成为我们学习的榜样。感谢吴老师、张老师在项目的设计过程中给予了大量的指导和帮助下，帮助我解决了许多难题，是他们手把手将我带入了机器人研究领域。同时吴老师、张老师认真负责的态度，严谨治学的精神一直感染着我们，使我们能够在老师的严格要求下取得不断的进步，在困难面前永不低头。吴老师、张老师奋发进取、忘我工作的精神，成为学生学习的楷模。在此，谨向尊敬的导师致以崇高的敬意和衷心的感谢！

在攻读硕士期间，得到了学长商博、胡禹超、杨建宇、杨奉帅、曹远宁、夏崇坤及同届同学吴博、段强、申勇、赵鹏的无私帮助，在此衷心的感谢你们！也感谢研究所里的各位老师和同学在两年时间里给予我的极大的指导和帮助！在这两年的时间里，大家互相帮助、互相鼓励，给我留下了美好的回忆。

感谢云机器机小组成员商博、夏崇坤、吴博、段强、申勇、赵鹏、胡航、吴运幸、李忠坪、于洪伟、秦操、刘艳娇、胡美玉、许可、夏瑞、高成强，正是在你们的不懈努力下，项目才得以顺利进行，谢谢你们！

衷心感谢百忙中抽出宝贵时间评审本文的各位专家、教授，在此致以诚挚的敬意！

衷心感谢所有帮助和支持我的老师、同学和朋友们！

衷心感谢一直陪伴我的强啸啸，感谢多年来的陪伴、支持、鼓励与包容！

衷心感谢我的父母！