天津中德应用技术大学

Tianjin Sino-German University of Applied Sciences

本科生毕业设计

基于 ROS 系统的智能超市服务机器人设计

Design of Intelligent Supermarket Service Robot Based on ROS System

学　　院　智能制造学院

专　　业　电气工程与智能控制

班　　级　19 电气控制（春）2 班

学　　号　19404080110

姓　　名　徐立飞

指导教师　刘春平

职　　称　教　授

完成时间　2023 年 06 月 02 日

天津中德应用技术大学
本科生毕业设计

基于 ROS 系统的智能超市服务机器人设计

Design of Intelligent Supermarket Service Robot Based on ROS System

学　　院 智能制造学院

专　　业 电气工程与智能控制

班　　级 19 电气控制（春）2 班

学　　号 19404080110

姓　　名 徐立飞

指导教师 刘春平

职　　称 教　授

完成时间 2023 年 06 月 02 日

天津中德应用技术大学

本科生毕业设计（论文）的声明

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

毕业设计（论文）作者签名：

年　　月　　日

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

毕业设计（论文）指导教师签名：

年　　月　　日

摘　　要

为了满足人们对商品日益增长的需求，超市已成为满足人们生活中的必需品必不可少的地方。因此，超市的数量不断增加，规模不断扩大，商品的种类翻倍递增。但是，随着超市规模的扩大也带来了许多负面影响。比如，以大型超市来说，超市货架众多，商品也是种类繁多，特别是对于时间不够富裕的人以及第一次光临此超市的人，想找到自己想买的商品还需寻找一段时间，不仅效率低下而且容易影响顾客的购买心情。因此，智能超市服务机器人的出现便能有效地解决此问题。

近年来，随着技术的快速发展，机器人也越来越智能。另外，由于我国老龄化的加剧，使之劳动力减少，况且人们也特别希望从繁重的、枯燥乏味的工作中解脱出来。所以，智能超市服务机器人的广泛普及与使用将成为今后整个社会发展的重要方向。一方面，可以增加顾客购物体验，便捷快速地获得自己想要寻找的商品；另一方面，减少人力，并将人们从繁琐枯燥的工作中解放出来。

服务机器人在我国的发展时间较短，技术还不够成熟，设计实践是求进步的重要途径。本文将试图通过与服务机器人相关的设计原理和方法的研究，为超市服务机器人的设计和实践提供一些理论指导。

本文设计了基于 ROS 系统的智能超市服务机器人，主要从外观造型、结构、功能、编程逻辑及设计原则进行探讨，运用实时建图（SLAM）实现自主的移动与路径的规划，机械臂实现对商品的抓取。最终希望完成一种功能完善、外观合理、交互方便的超市服务机器人，帮助超市有效解决服务工作，给顾客带来更多的购物乐趣和便利。

关键词：ROS；SLAM；服务机器人

ABSTRACT

In order to meet people's growing demand for goods, supermarkets have become an indispensable place to meet people's daily necessities. Therefore, the number and scale of supermarkets have been increasing, and the variety of commodities has doubled and increased. However, with the expansion of supermarket scale, it has also brought many negative effects. For example, in a large supermarket, there are many shelves and a wide variety of goods. Especially for those who are not rich enough in time and those who visit the supermarket for the first time, it takes a while to find the goods they want to buy, which is not only inefficient but also easy to affect customers' purchasing mood. Therefore, the emergence of intelligent supermarket service robot can effectively solve this problem.

In recent years, with the rapid development of technology, robots have become more and more intelligent. In addition, due to the intensification of aging in China, the labor force is reduced, and people especially want to be free from heavy and boring work. Therefore, the widespread popularization and use of intelligent supermarket service robots will become an important direction for the development of the whole society in the future. On the one hand, it can increase the shopping experience of customers and quickly obtain the goods they want to find; On the other hand, reduce manpower and free people from tedious and boring work.

Service robots have a relatively short development time in China, and the technology is not mature enough. Design practice is an important way to make progress. This paper will try to provide some theoretical guidance for the design and practice of supermarket service robots through the research of design principles and methods related to service robots.

This paper designs an intelligent supermarket service robot based on ROS system. It mainly discusses the appearance, structure, function, programming logic and design principles. It uses real-time mapping (SLAM) to realize independent movement and path planning, and the robot arm to grasp the goods. Finally, I hope to complete a supermarket service robot with perfect functions, reasonable appearance and convenient interaction, help the supermarket effectively solve the service work, and bring more shopping fun and convenience to customers.

Key Words: ROS; SLAM; Service Robot

目　录

第 1 章 绪论

1.1 研究背景

随着机器人与视觉传感器和语音识别等技术交融与快速进步，服务机器人也在不断崛起。智能服务机器人是一种具备对周围环境的感知、决策自身行为与控制自己行为等能力的系统[1]。智能机器人通过传感器（比如激光雷达、红外线等）获取外界信息，从而根据动作要求完成特定任务。机器人技术自 20 世纪问世以来，计算机技术与人工智能技术不断发展。今天，人类正处于一个科技兴盛的时代，人们的生产生活也越来越离不开机器人的协助。在我们的生活中，使用较多的家用机器人与工业上的机器人较为成熟和完善，比如家用的小型的扫地机器人，在我们的生活里是非常常见的，也确确实实为我们带来了便利。比如，在工业中，该领域的生产、装配以及运送等效率高、精度高的工作都需要机器人来帮持。在服务业，比如商场、银行、景区等，都需要机器人提供服务。在农业领域，机器人可以实现高效率的播种和采摘，从而节省人力劳动。甚至在太空探索中，如火星探测车“勇气”号和月球车“玉兔”号，它们都是人类在探索外太空神秘领域中，发现新事物迈出的关键一步。

如今，在不断发展的社会中，科学技术水平也是不断的在创新中，极大的改善了我们的生活方式。由于工业机器人的发展时间早于服务机器人，所以对于工业机器人的技术更加的成熟。因此，我国正在努力向服务机器人领域发展。

现如今，各个国家都对机器人技术的发展及其重视，外加政府的支持，不少传统行业也作了转型，政府纷纷的将各类机器人的发展纳入重要的发展战略方针。科研院校也在对机器人的研究上进行了大量的资金投入，多样的机器人种类，不同的功能作用，也正在源源不断的打开了不一样的应用环境，随着机器人更加智能化，这对社会生产力和经济的发展也产生了深远的影响。

另外，由于影视剧、短视频等文化的推动，人们更加希望在生活上的各种问题，机器人能够解决。机器人要想理解人们所要表达的想法，这无形中增加了机器人的智能化，正因如此，不断地推动着机器人行业的发展。

1.2 国内外发展现状

1.2.1 服务机器人发展现状

近年来，由于软件和硬件技术的飞速发展，产品一直更新换代，应用的场景和服务类型也不断扩展，在全球范围里服务机器人展现出了日益增长的市场需求。我国智能服务机器人的市场规模不断增加，行业的发展存在巨大潜力。

积极发展机器人行业已经成为许多国家的发展战略。近年来，具备清洁、家教、餐饮服务、护理等功能的机器人正在走进人们的生活中。随着各种机器人在各行各业的广发使用，使得全球机器人规模快速的增长。按照销售的收入计算，2021 年全球的机器人市场达到了 221 亿美元，预计到 2023 年将增加到 337 亿美元。

日本为推动机器人技术的大力发展，对机器人行业加大了支持，政府不断投入资金进行支持。另外，日本企业也做出了积极的反响，并表现出了对服务机器人浓厚的兴趣。与此同时，韩国也不甘落后，对机器人行业同样给出了相关支持政策和资金的帮持，并把服务机器人技术的水平作为国家的经济发展指标之一。

在 21 世纪 20 年代，美国提出了机器人行业将在今后的 5 年、10 年，甚至是 15 年中，作为经济促进的关键作用，重点是在制造领域、医疗领域和服务领域中；并总结了当前社会的主要机遇、需要解决的挑战与采取的解决的办法，其中包括两个方面，分别是对技术进行创新和采取的解决办法，从而保证美国在机器人领域中的领先地位。

通过坚持不懈的努力发展和对先进技术的运用，我国的服务机器人领域正在飞速发展中，机器人投入的应用环境更加多种多样。在 2021 年里，中国机器人的市场规模预计 839 亿元；2016 年到 2023 年，平均增长率为 18.3%。其工业机器人领域达到 445.7 亿，服务机器人领域达到 302.6 亿，特种机器人领域达到 90.7 亿[2]。经过数据的显示，在 2022 年的前半年中，我国仅增加的新注册服务机器人企业就已高达了 7.9 万家，由此可见，服务机器人行业的热度。2022 年举办的 CIFTIS 上，具备消毒、配送、引导、调酒等功能的服务机器人大放光彩，吸引了众多人们的关注，极好的表现了服务机器人领域的活力生机，大力推动了服务机器人的发展[3]。

1.2.2 SLAM 技术发展现状

SLAM 是 Simultaneous Localization And Mapping 的缩写，中文叫即时定位与地图构建[4]。它是由 Smith 等人最早提出来的，它能够在未知的场景下进行构建增量式的地图，与此同时，也能够在构建地图期间对机器人自身位置进行精确的定位。这对超市服务机器人的后续工作非常重要，因此，这正是 SLAM 技术会一直成为研究机器人自主移动的热点问题。从获取环境信息的传感器的不同，可以将其分为激光 SLAM(Lidar-SLAM)和视觉 SLAM(VSLAM)。视觉 SLAM 的方法计算量大，易受到光照的影响，在大场景下使用容易受到限制，而由于激光雷达测距精度高，计算过程简单，可靠性高，因此目前被广泛应用。

最早被提出来用于解决 SLAM 问题的方法是扩展卡尔曼滤波（EKF，全称为 Extended Kalman Filter）[5]。这种方法在处理不确定的信息时有它的独到之处，由于 EKF SLAM 算法使用的是最大似然数据，后将 EKF 应用于在线 SLAM，因此，对于一些近似和限制性的假设，EKF 就会受到限制。与此同时，EKF SLAM 算法需获取每一个时刻信息，把这

些信息分解成概率分布，导致计算的代价相对昂贵。此后，很多学者把无迹变换（UT，Unscented Transform）加入 EKF 的框架当中，并提出了基于 UKF（Unscented Kalman Filter）的 SLAM 算法，这种方法是利用无迹变换来处理均值和方差的非线性的传递问题，提高了 SLAM 算法的稳定性，并取得了较高的估计精度[6]。

Hector SLAM 算法就是将激光点和已有的场景地图“对齐”，也就是扫描匹配[7]。扫描匹配是使用当前帧和已有的场景地图数据建立误差函数，使用高斯牛顿法算出最优解与偏差量。实现激光点到栅格地图的转换。

目前应用最广泛的 2D SLAM 方法是 Gmapping 算法[8]。该算法主要是在 2D 激光雷达基础上应用 Rao-Blackwellized Particle Filters（RBPF）粒子滤波算法实现二维的栅格地图的构建，每一个粒子都携带着一张地图，把定位和建图的过程分开，先定位后建图。Gmapping 有效利用了里程计信息，其建图的稳定性较高。王依人等利用高精度的激光雷达数据并整理了用于里程计读数的建议分布函数。白崇岳等使用多种传感器的融合方式，如激光雷达、IMU（惯性测量单元）、摄像头等。Liu Z H 等调整了算法参数。总而言之，为了提高粒子多样性，国内外的学者们纷纷开始调整重采样，进而提高构建地图的效率和精度。

综上所述，对于实时建图技术的研究还会有很大的空间，目前海内外的学者们大多是从构建地图效率、构建地图精度与自身定位精度等方面进行研究，但偏向理论研究，应用在真正的场景环境中比较少，理论联系实际还需要时间。

1.2.3 自主导航（路径规划）技术发展现状

自主导航技术，也就是常说的路径规划，是在实时建图技术后的又一项非常重要的技术[9]。它是在构建好的场景地图模型、机器人自身的位置与目标点位置的基础上计算出相应路线。目前，根据需要规划路径的环境知晓程度，可将全局与局部路径规划相结合。前者的环境是地图上已知的不再发生变化的环境，后者的环境是原有地图上未有过的物品而组成的新环境，然而未知的环境又包括移动的物品和固定的物品。相较于全局路径规划，局部路径规划要面对的环境更复杂多变，如何在复杂的场景中保证自主导航的稳定性与实时性，是现在主要的研究问题。

之前，路径规划的研究多为静态环境中，但随着越来越多的人们在不断探索研究过程中，目前的路径规划技术通过机器人搭载的各种传感器可获取更多实时场景信息，就算是未知的场景，也可实现机器人的自主导航功能。

通过阅读文献得出，在已知的场景环境下，全局路径规划的研究目前是集中于对算法搜索速度的提高和生成路径效果的改变。而对于局部路径规划效果不良的问题，主要是包括障碍物的形状大小、障碍物本身和机器人的相对速度和相对距离的不同，或者是算法自身对未知环境的限制。

1.3 研究意义

在当前生活的社会中，人们的生活已经离不开超市，在购买生活必需品时，大多数商品都是去我们周边的超市进行购买和采集。由于大量顾客会经常到超市去采购商品，随之而来的是消费者更高的要求，例如超市对顾客的服务的效率和服务的质量等等。比如当前经常发生的消费者不能立刻找到所需物品的具体位置，虽然超市都配备有导购员，可是数量有限，不能对顾客及时服务，如果顾客很多的情况下，也只是会给顾客指一下所需商品的大概位置，还是需要顾客自己慢慢寻找。由于消费者次次都需要在琳琅满目的商品中找到那个自己想要的一个，加之购物的人群基数较大，严重影响购物时间。此时此刻，如果能有一款服务机器人在超市中对顾客进行服务，便能为顾客带来不一样的购物体验，增加购物效率。

在科技水平不断发展和机器人技术持续提高的今天，机器人的智能技术手段已经逐步的在超市等服务行业进行了广泛的应用。若将超市服务机器人引入到超市市场中，这将会为超市行业带来更好的服务品质和效率。一方面，它可以耐心地对消费者进行帮助，使顾客轻松愉悦地完成购物，节省寻找商品的无效时间；另一方面，它也能将超市工作人员从枯燥无味的、重复的工作中得到摆脱，既减少了员工的人力成本又提高了消费者的购买效率，极大地提升了超市的服务质量。

1.4 研究内容

服务机器人正在一步一步地融入我们的生活当中，所以对机器人的各个方面都要有更加严格的要求。因而，机器人设计必须要考虑到用户的使用体验和心理感受，最大化的使智能服务机器人造福人类社会。

研究内容如下：

（1）设计本次课题的机器人，并在机器人功能的设计、对机器人的设计原则、机器人软件部分、机器人硬件部分等方面进行分析。

（2）理论联系实际，通过对超市服务机器人的外观设计、功能实现、硬件选型的搭建，使超市服务机器人最终以实物具体展现。

（3）使用 ROS 系统完成对服务机器人的编程，使功能符合人们日常购物习惯。

（4）实时建图与自主导航。研究 SLAM 时机器人定位问题和算法，以及建图以后快速实现路径规划。

（5）对商品的抓取。运用机械臂实现对物品的抓取，配合视觉传感器进一步检测物品的具体位置，提高抓取的准确度。

第 2 章 功能介绍及设计原则

服务机器人是这样定义的，它是指具备感知、识别、决策、执行等能力，可以与人们进行智能交互，帮助人们解决实际问题的。

本课题研究的超市服务机器人需要具备跟随、定位、导航、语音交互、抓取、物体的探测等功能。首先引导机器人在超市构建地图，其次构建地图同时进行定位和导航，最后机器人根据购物需求自主完成取货任务。因此，本课题设计的机器人的主要功能是实时建图、自主导航、语音交互以及抓取功能等等。

2.1 功能介绍

1. 实时建图

当第一次进入陌生的超市环境中，人们无法知晓超市的详细环境以及具体的物品位置，更何况机器人。因此，我们要想让机器人完全熟悉超市的环境信息，我们需要构建地图。实时建图完成后，机器人通过地图信息，就能够获取超市的物品位置，熟悉超市的环境，为接下来的自主导航提供有利条件。

2. 自主导航

也就是通常所说的路径规划，该功能是完全自主独立完成的，即机器人可以找到一条能够使机器人安全通过的路径，并且这条路径是从机器人目前的定位位置到目标终点位置的一条连续的、可一直移动的路径。机器人便可以根据顾客所要的目标物品在构建的地图的环境中自行规划路线，到达目标位置。

3. 语音交互

语音交互功能就是让机器人能够“能听、会说”，比如小爱同学。机器人可以根据识别我们的语音指令完成相应动作，另外，机器人本身也需要根据当前作业状态给我们一种回应。总的来说，语音交互让我们与机器人的沟通可以更加自然、便捷的方式进行。

4. 物品抓取

作为一款超市服务器人，不能像导购机器人那样，只是带领顾客去其想去的地方。本次课题的服务目标为顾客在等待区等待，机器人可以自行前往目标物品所在地，利用所搭载的机械臂进行抓取，并将所需物品返回等待区直至交与顾客手中。

本节只是对机器人功能进行简单介绍，详细算法与具体实施详见第四章。

2.2 设计原则

在设计方面，服务机器人也和工业机器人类似，也有着相对应的设计原则来指导。这一节里，将根据工业机器人的设计相关理论，分别从安全、功能、低成本、审美、可拓展等方面来设计适合于服务机器人的设计原则。

2.2.1 安全原则

在人类居住的环境中，安全则是一项重要内容。任何一台设备都须安全操作以及配备明显的急停按钮。在任何情况下，超市服务机器人都不能对人和环境造成危害。例如，在行进时，速度的控制是否合适；在外观造型上，圆滑的外观最为适宜，整个机器人的重心也应放在下盘较合理。另外，机器人要有良好的避障能力和提示音功能，避免撞到物体和人类。

2.2.2 满足功能需求原则

超市服务机器人作为商店的小助手，在功能设计方面，要实现其互动性、准确性和快速性，达到好看实用的效果。机器人要有对语言识别的功能，便于用语言对机器人达到控制行为的效果，还应具备回复功能，这样就可以和机器人对话。并能准确的识别顾客说出所要商品，快速的反应出商品所在位置将商品交到顾客手上。

设计任何一台设备时，核心是它的功能，对于没有任何功能的产品，它就没有立足之地。对于超市服务机器人，满足以下方面即可：

（1）对于繁重、枯燥、机械的工作，服务机器人可以把人解放出来。基于此，在设计超市服务机器人时，依据在某一领域上可以取代人类以及帮助人类的设计原理。比如，扫地机器人，可以帮助并替代人类从扫地劳动中解放。

（2）目前，老龄化社会已经到来，来自人力成本的压力也在日益增加。人工智能的出现，以及现代教育所提倡的创新思维和动手能力的模式，推动了服务机器人的发展。

（3）核心功能是：初始时刻，它在等候区待命，顾客来临时，说出想要商品，通过之前对超市的建图，机器人通过路径规划到达所需商品位置，利用机械臂和机械爪将商品送到顾客手中。

2.2.3 低成本原则

所有产品的设计都需要考虑的原则是成本，超市服务机器人也是如此，依据前面所说的移动方式，因此选择经济型的 STM32 芯片的底盘控制系统（主控器），移动方式采用三轮全向轮结构，由步进电机提供动力，做到结构简单且效果较好。

2.2.4 审美原则

这是一个“看脸”的时代。对于机器人的功能设计而言，不仅仅是要考虑其实用性，也要考虑到我们所设计出来的机器人给人们带来的心理感受。相对于超市服务机器人的外观造型设计方面，既要考虑机器人的功能达到人们的要求，还要在外观上展现美感和品质。超市服务机器人的设计要遵循并且应用各种各样的美学规律，搭配出独特的造型，并充分考虑外观造型给人们的感受，设计出一款符合大众审美的机器人。

2.2.5 可拓展原则

可拓展原则是指在拓展范围里不用重新设计，新的功能就能够加入原有系统中。在设计前期，要充分考虑到电子器件以及功能的升级换代问题，需要运用模块化的设计。这样的话，才能便于更新，也能让机器人更好的按照人们购物需求去改变，与此同时，也能让机器人在市场中占据一定的地位。

2.3 本章小结

本章首先简单介绍了超市服务机器人所具备的主要功能，根据功能选择出适合机器人的设计方法和算法。其次参考工业机器人的设计原则，提出适合超市服务机器人的设计原则，使机器人功能更强大、作业更可靠。

第 3 章 机器人软硬件系统设计

3.1 操作系统选择

现如今，常见的服务机器人操作系统有 Android 系统和 ROS 系统[10]。ROS 和 Android 一样都是开源的操作系统，在功能方面的差距也是寥寥无几，其独特之处便在于能够支持多种语言，如 C++、Python、Octave 以及 LISP，甚至还支持多种语言的混合使用，能够简化开发者工作。因为它是基于 Linux 的系统，其可靠性也会更高，体积可以做到更小，适合嵌入式设备。另外，ROS 是一种分布式处理框架，开发者可以单独设计可执行文件，ROS 还提供了设备驱动、可视化界面等，方便优化及扩展。基于上述原因，本课题的机器人选择 ROS 操作系统。

3.1.1 ROS 的定义

ROS（Robot Operating System），翻译过来就是机器人操作系统。ROS 系统具有高度灵活性的软件架构，可以用来编写机器人的软件程序[11]。2007 年，标志 ROS 系统诞生的事件是 STAIR 项目与个人机器人项目之间的合作。经过研究，在 2010 年，属于机器人的操作系统 ROS 正式问世，因其拥有的点对点设计和对编程语言不过分依赖的优点等等，很快就掀开了学习使用 ROS 的序幕。ROS 是开源的操作系统。它可以提供与操作系统相似的功能，其中包含了对硬件的抽象描述、管理底层驱动程序、执行共用功能、传递程序之间信息和管理程序发行包，也可以提供一种工具程序与函数库，用在多机整合程序中，进行获取、构建、编写以及运行。在机器人研发过程中，ROS 系统主要目标是对程序的代码能够进行重复的使用，使程序执行的过程中，能够进行独立设计，把实时地、分散地合并起来。

3.1.2 ROS 的组成

根据系统的实现角度，ROS 的组成主要有两个，分别是文件系统级与计算图级。

1. 文件系统级

文件系统级指可以通过电脑的硬盘找到并能够进行查看的源代码[12]。并且不同的功能文件放在不同文件夹里，具体如下：

（1）功能包（Package）：ROS 软件中的基本单元，包含 ROS 节点、库、工具以及可执行的代码等。

（2）功能包清单（Manifest）：每一个功能包都有一个 package.xml 的清单，记录功能包的基本信息。

（3）元功能包（Meta Package）：把多个同一目标的功能包组织起来。

（4）元功能包清单：与功能包清单相类似，会额外包含着在运行中所依赖的功能包。

（5）消息（Message）：在节点之间进行发送和接受通信信息。

（6）服务（Service）类型：定义了通信过程中的请求与回复的数据类型。

（7）代码（Code）：用于存放源代码的一个文件夹。

2. 计算图级

计算图级指的是在 ROS 系统中，进行节点之间的通信时所包含的基本概念[13]。包含以下内容：

（1）节点（Node）：相当于“软件模块”，是运算任务执行的进程，通常多个节点构成单个系统。

（2）消息：节点与节点之间的通信机制。

（3）话题：通过发布/订阅的形式进行传递，从而获取数据。

（4）服务：一种同步的传输模式，包含了请求和应答的通信数据类型。

（5）节点管理器：用于对以上概念进行统筹管理，使节点井井有条的执行。

3.1.3 ROS 的特点

（1）点对点设计。ROS 系统通过节点管理器与服务等机制，进行分散激光雷达、视觉传感器等功能的实时运算带来的压力。

（2）可运行多种程序语言。ROS 可以编译和运行 C++、Python、Lisp 等语言。目前，也已经实现 Octave 和 Java 的测试。

（3）简易化和集成化。编程的模块化设计可以使对代码的使用和集成更加方便。

（4）方便用于测试。ROS 拥有自己的测试平台和可视化工具，对功能和算法进行调试和验证时更加的便利。

（5）开源操作系统。当前已有大量的 ROS 系统开发者，并在交流社区贡献了大量代码。

（6）不过分依赖操作系统。比如 Federa、Gentoo、Arch、Debian、OpenSUSE、OS X、Slackware、Ubuntu 等等。

3.2 机器人硬件设计

3.2.1 模型设计

基于上一章的功能介绍和设计原则，对超市服务机器人的模型设计如图 3-1 所示，具体包含以下单元：

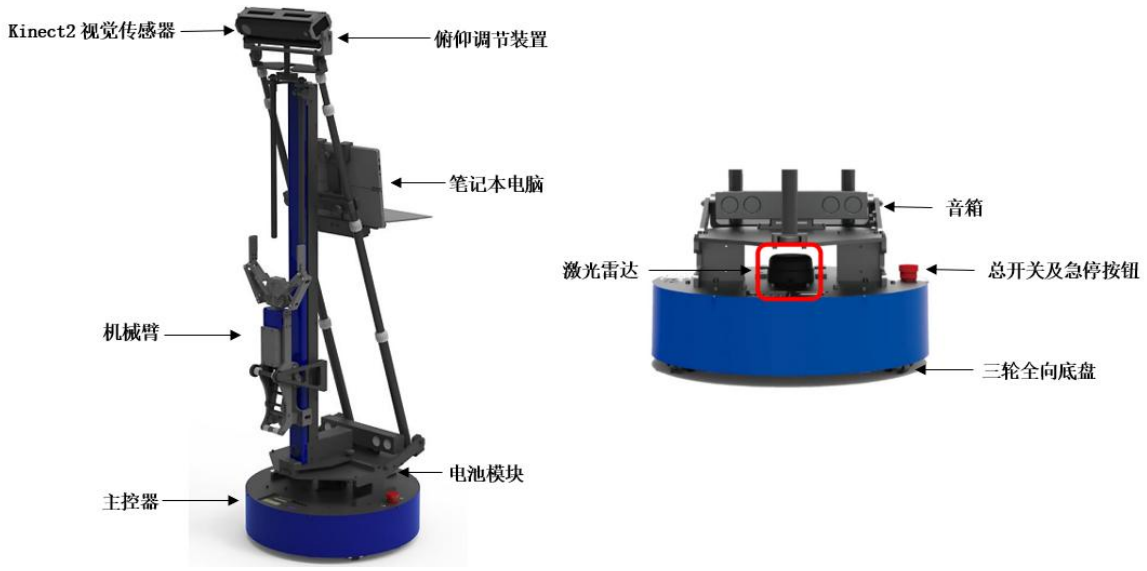

图 3-1 超市服务机器人模型图

（1）Kinect2 视觉传感器。超市服务机器人的上方具备一个视觉传感器，包含了彩色相机、红外相机、红外发射器和电源指示灯，如图 3-2 所示，视觉传感器使用的是 Kinect2 RGB-D 相机，它是复合型传感器，能够对彩色图像信息进行获取。其中“RGB”分别代表着“红绿蓝”，可显示彩色图像并能够对物体颜色进行感知和识别；“D”是“Depth”的缩写，翻译过来的意思是深度，因此它又能够测量到物体距离。通过一幅一副的彩色图像和包含着距离信息的点阵（三维点云）来获取数据。Kinect2 红外相机的使用是配合着红外发射器的，其发射出红外激光，当照到障碍物上时会将激光反射，从而让红外相机进行接收。

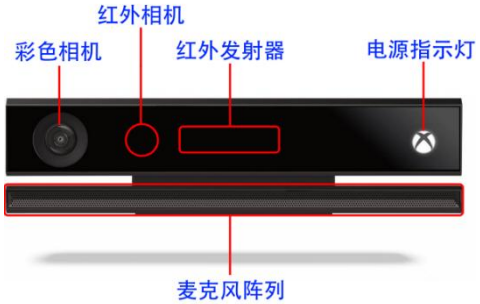

图 3-2 Kinect2 RGB-D 相机

（2）俯仰调节装置。主要用途是对机器人的视角进行调节和固定。

（3）笔记本电脑。安装 ROS 系统的笔记本电脑，利用应用软件来编写并运行程序，也是可视化界面的工具。

（4）音箱。即扬声器，是实现机器人语音输出的重要元件。

（5）激光雷达。机器人搭载的是思岚（SLAMTEC）RPLidar A2 激光雷达，它是在移动机器人领域中非常常用的传感器之一。采集信息时，激光雷达高速旋转 360°，高达 10rad/s，每秒可以获得 3600 个距离值，测距范围为 0.15~12 米。

（6）总开关。用来闭合或切断机器人的总电源。

（7）急停开关。底盘左侧的“急停开关”，这个开关为一个红色按钮，默认为按下状态，此时为断电信号；若要运行机器人，则需要通电信号，对急停开关进行逆时针旋转即可通电。如遇到紧急情况，用力拍下可停止其行为。

（8）电池模块。超市服务机器人的动力，来源于电池模块的供电，它是由 7 颗电池串联连接组成，每颗锂离子电池的容量为 3500mA/h，内部配有电池保护板，对电池起到保护作用。

（9）主控器。控制底盘电机和机械臂的作用。通过与计算机之间的通讯，控制机器人的底盘电机和机械臂的动作。

（10）三轮全向底盘。多个小滚轮构成一个全向轮，3 个全向轮之间相差 120°，由此便组成了一个圆，使其结构更加简单且运动效果更良好。从而使机器人在平面移动时使其可以在任意方向移动。其核心部件为电机，电机参数如表 3-1：

表 3-1 电机参数表

名称	具体参数
工作电压	24V
额定功率	17W
持续工作电流	1.4A
空载转速	15000RPM
减速比	64：1
编码器线数	12 线

3.2.2 电气控制系统

超市服务机器人电气系统如图 3-3 所示，包括 USB-HUB、电池模块、电源控制板、控制器和伺服电机。搭载 ROS 系统的计算机则是经过 USB 与底盘面板上的 USB-HUB 进行连接，并用电脑固定器和键盘底托固定于设备的躯干部分。笔记本电脑的 USB 接口可通过 USB-HUB 将其扩展为多路。经过扩展后，便可将主控器、USB 转接口、激光雷达、面板接口分别连接到 USB 接口上。

底盘内部的电源管理板控制开关机、执行机构的急停，并为各控制器、传感器提供电源。

如图所示，红色导线为电源供电部分，蓝色导线传输的是控制信号，黑色导线是专用线缆，用于将供电导线合并于总线。

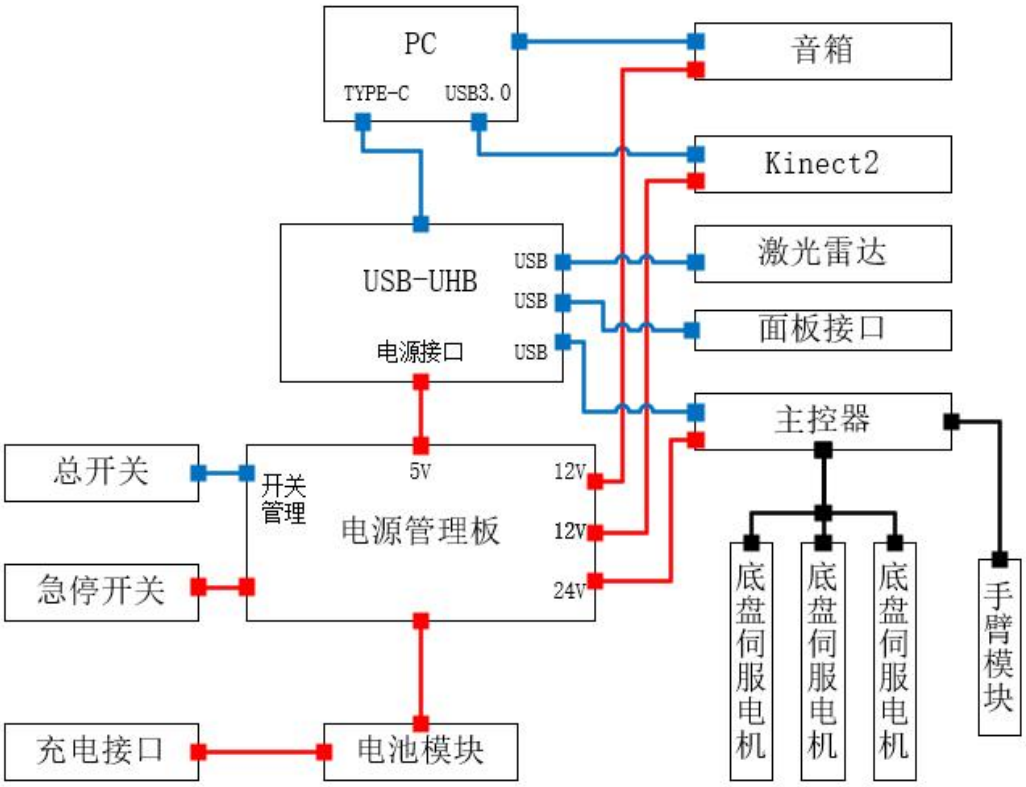

图 3-3　电气控制系统图

3.3 软件系统

3.3.1 通信设计

如图 3-4 的通信链路图，通过 RS485 总线实现控制器与伺服电机模块的通信，发送速度、位置等信息的周期为 50ms，与此同时对电机反馈的机器人实时位置和电机绕组上的电流信息进行接收。重新封装反馈数据后，以周期为 50ms 向计算机发送。计算机运用 USB 接口与控制器进行连接，把 USB 接口通过控制器内部的 FTDI 接口转换为 UART 串行接口，将这个串行接口以 115200 波特率与主控器的控制核心 STM32 进行通信。

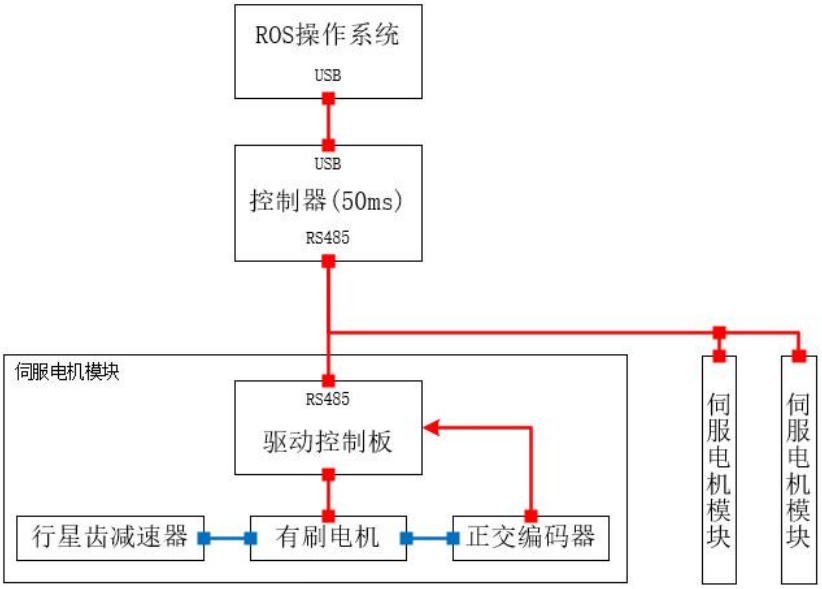

图 4-11 通信链路图

伺服电机模块内部包括四大部分，分别是有刷电机、行星齿轮减速器、正交编码器、驱动控制板。有刷电机尾轴部分直接连接到正交编码器，输出轴连接到行星齿轮减速器，经过减速后再输出。TI 的 TMS32F28062 数值信号控制器作为驱动控制板的核心，实现控制位置环、速度环、电流环的作用。

3.3.2 应用软件介绍

若想完成程序的编写与调试，就需要 IDE 进行辅助，能够开发 ROS 的 IDE 有很多，比如 Visual Studio Code（VScode）、Eclipse、QT、PyCharm 等等。与此同时，选择合适的可视化工具来反映程序的结果也是至关重要的。因此，在本次课题中，选择微软的 Visual Studio Code（VScode）进行程序代码的编写，使用 Terminal 运行代码，通过可视化工具 Rviz 对机器人系统中各种传感器数据、机器人模型和环境等信息进行可视化。

1. Visual Studio Code（VScode）

VScode 是一款跨平台的免费代码编辑器。它是由微软公司开发并且维护的，基于 Electron 框架构建。支持 Windows、Mac 和 Linux，并且支持多种编程语言，包括 JavaScript、TypeScript、Python、C++等，并提供了丰富的插件和调试工具，可用于开发 Web 应用程序、桌面应用程序、移动应用程序等不同类型项目。

2. Terminal

翻译过来就是终端的意思。什么是终端？需要先了解一下操作系统。操作系统可以分成两个部分组成，一个是内核，另一个是用户交互界面。内核部分是主要负责系统逻辑操作，是由大量的命令组成，是保证系统运行的重要命脉，与用户不直接接触；交互界面就是我们在开机之后看到的东西，如窗口画面、应用软件、程序等。但如果我们需要对系统内核的某个逻辑进行修改时，就需要这个终端来完成，终端就是接通系统内核与交互界面之间的桥梁，允许用户对系统的内核进行间接的控制，类似系统的大脑，用户可以在界面上打开应用程序“Terminal”，随后输入命令，系统可以快速反馈。

3. Rviz（三维可视化工具）

就是可视化工具的意思，它是一个 ROS 系统的 3D 可视化工具。运用 Rviz 能够实现对类似于环境、传感器等信息通过图形来展示，也可以发布信息，从而检测机器人数据，并进一步监测机器人运动的效果[14]。可视化工具为开发者提供了多方位的观察功能，包括机器人视角、全景地图等。

3.4 本章小结

本章是机器人的软硬件系统设计，首先是对机器人操作系统的选择，说明了选择 ROS 系统的原因，其次围绕 ROS 系统主要介绍了它的基本概念、主要组成部分以及其特点。然后是模型设计，分别介绍了各个单元。最后介绍了本次课题中所用的主要应用软件，通过软件进行编程调试和可视化机器人数据。

第 4 章 超市服务机器人功能实现

在超市环境中，机器人应该具备独立移动和语音交互的功能，并且通过语音识别到商品后，可以自行移动到目标商品，对所需商品进行抓取，并且将商品运送到顾客手中。

因此，对超市服务机器人的自主导航技术的研究主要在实时建图、机器人的定位与路径规划这三个部分，语音交互主要是包括语音识别和语音输出这两方面，对商品的抓取在于机械臂控制和物品探测两个方面。现如今，已经将实时建图与机器人定位合并成 SLAM。本章将研究机器人的自主导航系统，其实现的原理是应用了 SLAM 算法，对比 Hector SLAM 算法和 Gmapping 算法，进一步提高机器人自主导航系统建图的效率和精确度。应用机械臂与视觉传感器相结合，提升抓取准确性。

4.1 实时建图

随着科学技术的发展，SLAM 技术也越来越完善，其技术被广泛应用在无人驾驶、无人机与定位导航系统等。实时建图技术也被公认为实现机器人自主移动的关键技术。在本节中，将分析两种构建地图算法，并在最终选择适合于超市服务机器人的算法。通过在 ROS 系统测试激光雷达获取的数据，证明算法的可行性。

首先，实时建图功能的实现需要对激光雷达数据的应用，激光雷达是移动机器人常用的一种传感器，其工作原理是用一个高速旋转的激光测距探头将周围 360°的障碍物分布状况测量出来，形成障碍物轮廓的俯视二维点阵输入 ROS 系统里[15]。激光雷达的测距探头位于旋转部分，在其旋转时带着探头高速旋转，转动 360° 后，可得到一幅二维点阵俯视图，这个点阵图上绘制有障碍物轮廓。

在确认激光雷达正常驱动之后，启动激光雷达，运行指令如图 4-1 所示：

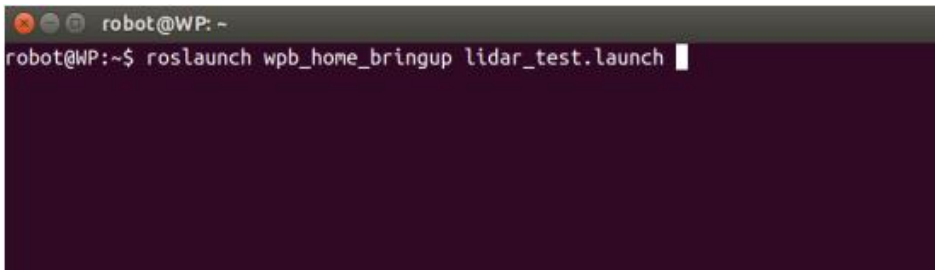

图 4-1 雷达测试指令图

按下回车，可视化界面被打开，机器人的 3D 模型将会在可视化界面上显示出来，与此同时，从界面中可以看到机器人周围会显示红色的三维点，这就是被雷达识别到的障碍物，如图 4-2 所示。

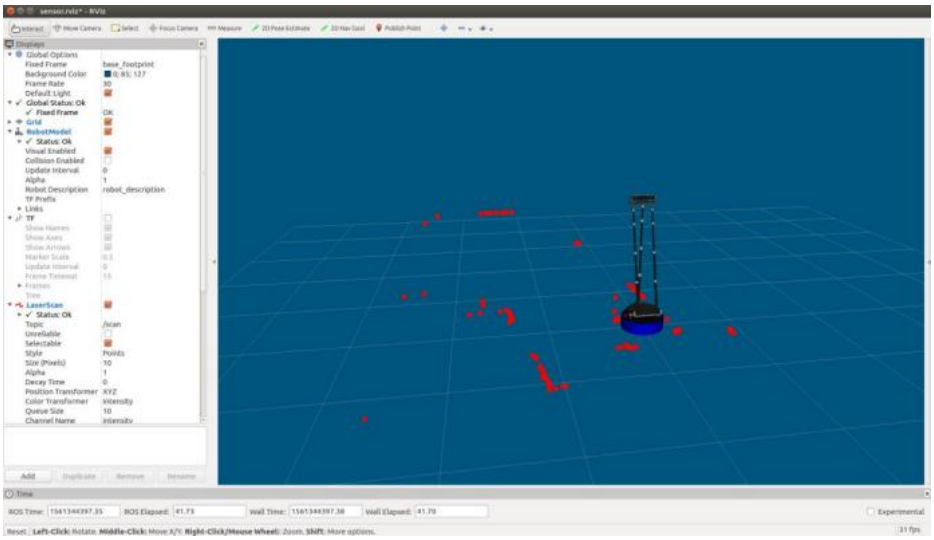

图 4-2 激光雷达运行图

保持此界面别关闭，再新打开一个终端程序，输入如下指令：

rosrun lidar_pkg lidar_data_node

这条指令会从激光雷达的“/scan”主题里不断获取激光雷达数据包，并把测距数值显示在终端程序里，如图 4-3 所示。所以我们会看到从 Point[0]到 Point[359]的 360 个距离值不断的刷新，这个距离值是一个浮点数，单位是米。比如终端里显示“Point[180] = 1.051000”表示激光雷达在 180°的这个角度测量到的障碍物距离值是 1.051 米。

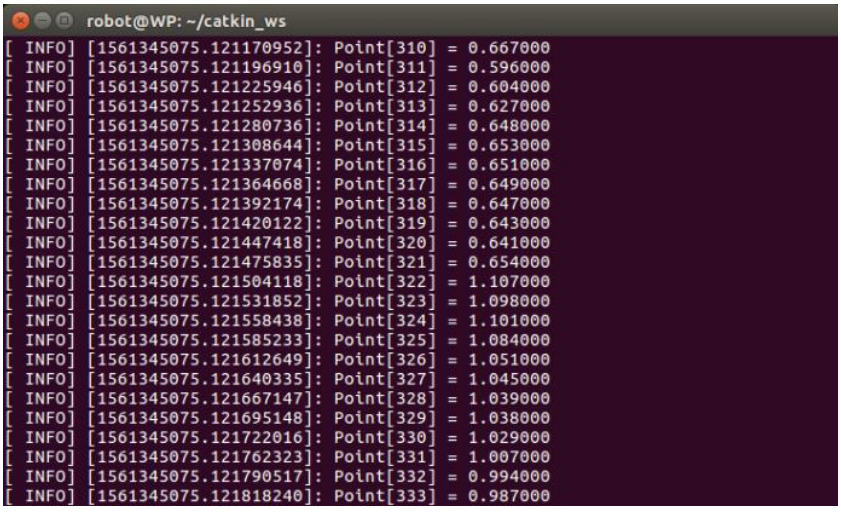

图 4-3 激光雷达数据刷新图

其次，激光雷达进行不断扫描时，所得到的数据是一个切面图，在雷达的测距范围里反映的是能够被雷达扫描到障碍物的大致轮廓，如图 4-4 所示。

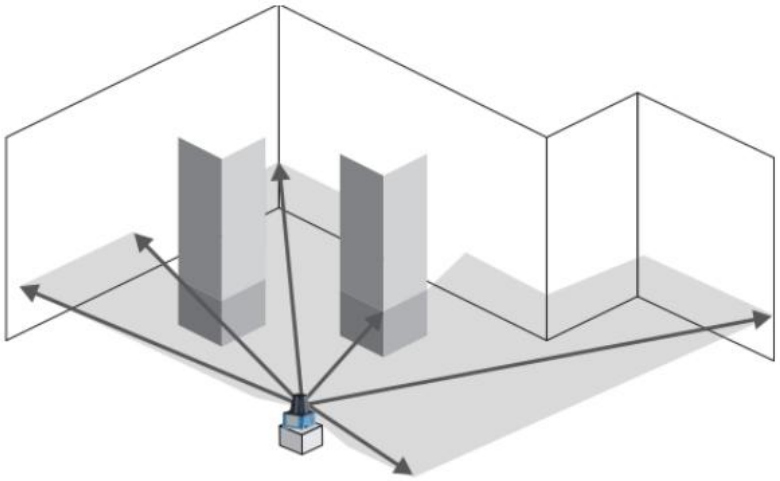

图 4-4 雷达扫描原理图

因此，机器人移动时，激光雷达并不是全部的障碍物都能扫描出来，只能在一定的范围内识别到障碍物的边缘轮廓的大致形状，并且实时更新计算出机器人本身在本次环境场景中的定位置，在界面中显示。如图 4-5 所示，机器人在相邻的 3 个位置 A、B、C 上分别进行扫描。

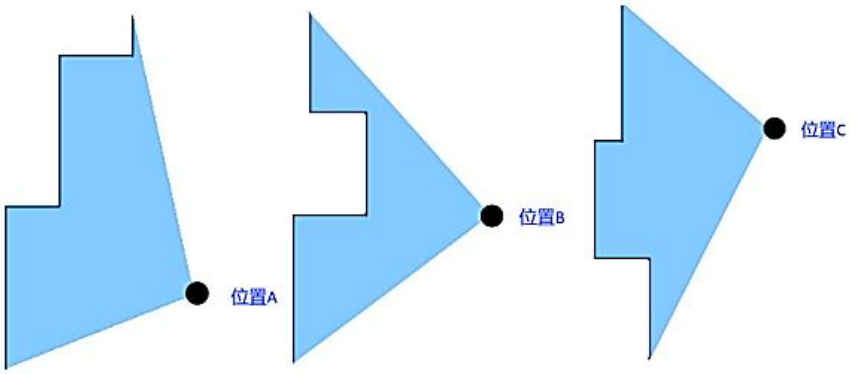

图 4-5 障碍物轮廓扫描图

其中，虽然在 3 个位置上有不同的扫描结果，但细细观察，3 个不同障碍轮廓可以进行重合的。因此假设将相似部分的轮廓合成同一障碍的边缘轮廓。如图 4-6 所示，将 A、B 两个位置进行相似轮廓合并：

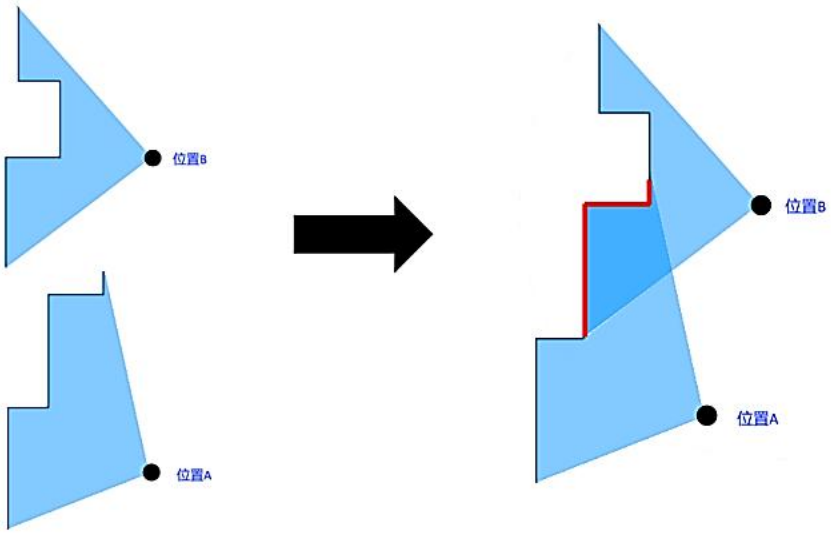

图 4-6 位置 A 与 B 扫描叠加图

再比如，图 4-7 所示的 B、C 进行合并：

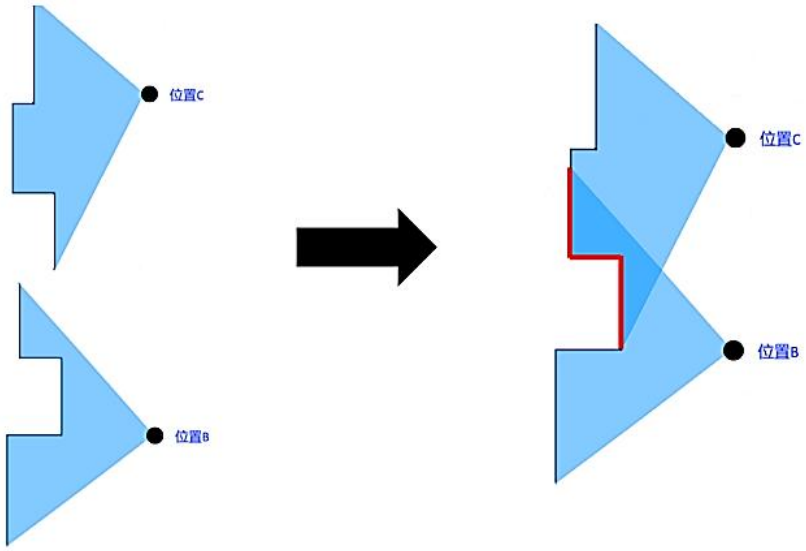

图 4-7 位置 B 与 C 扫描叠加图

由上所述，我们就可以得到一个比较大且相对完整的地图。虽然只是二维的地图，但也能完全的显示出障碍物在环境中的分布。在建图中，还可以利用上述的重合障碍物的概念，对障碍物之间的相互关系和机器人的实时位置进行推导，因此 SLAM 具有构建地图和实时定位的功能。

SLAM 算法多种多样，接下来对两种比较主流的算法分别研究并比较。

4.1.1 Hector SLAM 算法

Hector SLAM 算法是一种拥有较好鲁棒性二维 SLAM 算法，该算法不需里程计，但是却对激光雷达的要求标准高，激光雷达需刷新频率快且噪声小。

概率栅格地图为激光雷达对真实的场景进行描述的一种方式，经过验证，可以对任意的场景进行地图的构建。因为激光雷达的数据具有离散性的特点，不能够直接用在构建栅格地图和扫描匹配，因此需要先由插值获取连续的概率栅格地图。

获得了栅格地图之后，需要扫描匹配后续到达的激光雷达数据，扫描匹配过程就是把激光雷达所扫描到的实时数据和已经构建的地图与之对齐的一个过程，原系统在这一步的过程中参考借鉴了图像对齐的方式构建误差函数，使用高斯牛顿法（Gauss-Newton method）获得最佳的地图[16]。此方法在激光雷达的数据更新过后，在前一时刻的状态附近迭代出目前时刻的最优匹配，并计算出位姿增量。Hector SLAM 算法所建立的 Gauss-Newton 方程为：

$$\xi^* = \arg\min_{\xi} \sum_{i=1}^{n} \left[1 - M\left(s_i(\xi)\right)\right]^2$$

$$\xi = \left(p_x, p_y, \psi\right)^T$$

$$s_i(\xi) = \begin{pmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{j,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}$$

$$\sum_{i=1}^{n} \left[1 - M\left(s_i(\xi + \Delta\xi)\right)\right]^2 \to 0$$

$$\Delta\xi = H^{-1} \sum_{i=1}^{n} \left[\nabla M\left(s_i(\xi) \frac{\partial s_i(\xi)}{\partial \xi}\right)\right]$$

上式中，ξ为栅格地图的旋转矩阵，p_x，p_y，ψ分别代表着x，y坐标轴的偏移量与旋转的角度，$s_i(\xi)$则代表在世界坐标轴下激光雷达位置，M 函数代表坐标点的障碍物概率值。H 是 Hessian 矩阵，所示如下：

$$H = \left[\nabla M\left(s_i(\xi)\right) \frac{\partial s_i(\xi)}{\partial \xi}\right]^T \left[\nabla M\left(s_i(\xi)\right) \frac{\partial s_i(\xi)}{\partial \xi}\right]$$

4.1.2 Gmapping 算法

这是一种基于 RBPF 的能够有效解决实时定位与构建地图的算法[17]。SLAM 要解的问题是控制数据的$u_{1:t}$与观察数据的$z_{1:t}$进行求解位姿和场景地图的联合分布。利用条件概率将上述问题分解开来求解位姿，就是先进行定位后进行建图。公式如下：

$$p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(m|x_{1:t}, z_{1:t}) \cdot p(x_{1:t}|z_{1:t}, u_{1:t-1})$$

公式中，x为机器人运动轨迹；m为通过z观测到的特征地图；u为里程计的测量值。

被里程计提供的位姿信息存在着较大的不确定性，与里程计的分布相对比，激光的分布更靠近真实值，因此说明将激光雷达的信息引入到提议分布中时，到时候提议分布与目标分布将更加接近。与此同时，由于粒子需要将里程计状态的整个空间进行全方位覆盖，其中仅有少数粒子符合真实的目标分布，所以在计算权重时，粒子的权重变化会很大。如需舍弃权重较小的粒子，复制权重大的粒子使其收敛至真实状态附近。这种操作需进行频繁重采样，将会造成 RBPF 另一个弊端，就是会发生粒子退化。这也正解释了 RBPF 要有大量粒子并去执行频繁重采样。

此时应用最近一次的观测进行改进提议分布，所以提议分布如下：

$$p\left(x_t \left| m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}\right.\right) = \frac{p\left(z_t \left| m_{t-1}^{(i)}, x_t\right.\right) p\left(x_t \left| x_{t-1}^{(i)}, u_{t-1}\right.\right)}{p\left(z_t \left| m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}\right.\right)}$$

此时，提议分布同时使用了激光雷达和里程计里的信息，相比较于公式 4-7 中的提议分布减少了粒子退化的状况，且更加容易到达函数峰值，对地图的构建精度也将更准确。

4.1.3 建图效果对比

通过前面两节对 SLAM 算法的研究，这一节将会对 SLAM 两种不同算法来进行建图测试仿真，验证算法建图的效果。

测试场地为实验室，模拟超市场景，软件操作平台为 Ubuntu16.04 版本，装有 ROS（Kinetic）系统的笔记本电脑。根据 ROS 系统官网进行安装 Hector SLAM 算法和 Gmapping 算法，分别运行两种建图算法，观察其效果。

使用 USB Type-C 连接线将机器和 PC 机连接好，并保证通讯正常，按下总开关并检查急停开关，在电脑上桌面上双击 Terminal 应用程序，终端程序打开后。分别输入并运行两种算法指令，如图 4-8 和 4-9 所示：

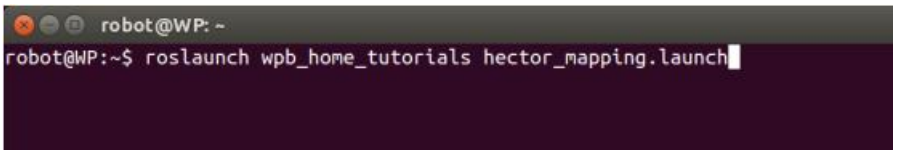

图 4-8 Hector 运行指令

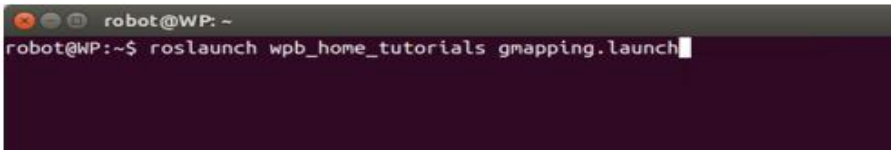

图 4-9 Gmapping 运行指令

按下 Enter 健，运行 Rviz。两种算法的建图效果如图 4-10 和 4-11 所示。

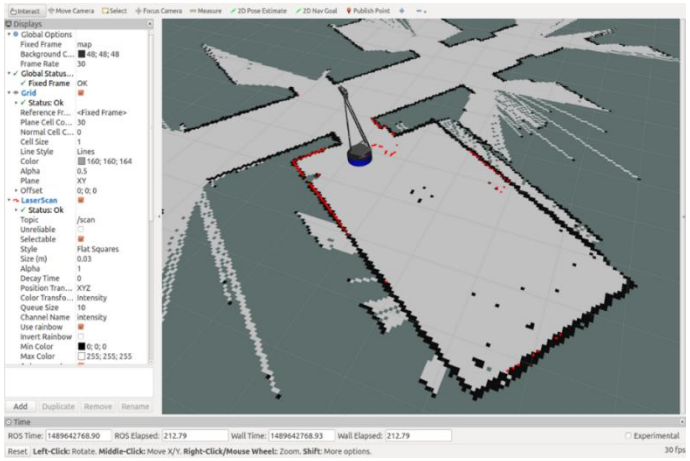

图 4-10 Hector SLAM 算法建图效果图

两种算法分别把构建后的地图用不同的颜色在可视化界面里显示出来，地图的各个颜色所代表的意义如表 4-1 所示。

表 4-1 地图颜色含义表

地图颜色	含义

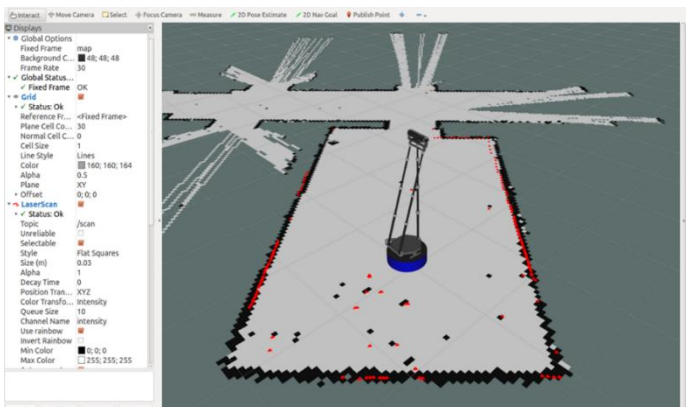

图 4-11 Gmapping 算法建图效果

红色	已知障碍点
灰色	未知区域尚未探索
白色	无障碍物的安全可通行区域
黑色	静止不动的障碍物

最后，通过建图效果对比可以看出，Gmapping 算法构建出来的场景地图的边缘更加清晰且完整，而 Hector SLAM 算法构建的地图在其边缘位置明显可以看出较为凌乱，因此 Gmapping 算法的建图效果要优于 Hector SLAM 算法。另外，Hector SLAM 算法仅依靠激光雷达工作，因此它对激光雷达的要求更高；而 Gmapping 算法则是在 Hector SLAM 算法上，还引入了其他的信息，比如电机码盘里程计等，使建图效果更佳。因此，Gmapping 的稳定性要更好。综上，本次课题的选择为 Gmapping 算法进行实时建图。

4.2 自主导航

通过上一节对建图技术的研究，能够得到一张相对准确的地图并能够保存到指定位置，又可以根据构建的地图返回实时的位置。本节将继续研究导航系统的另一个关键系统——自主导航技术（路径规划技术）。

在上一节中，通过激光雷达的扫描探测，所构建出来的地图格式是机器人非常常用的栅格地图，它可以将障碍物分布在平面上进行描述并划分出多个栅格。如图 4-12 所示，在 Rviz 里的显示的是实时建图所建立的平面地图，将地图放大后，如图 4-13 所示，可以看到许多的小栅格。白色是可安全通过，此处无障碍物；黑色是不可通过，此处有障碍物；而灰色是无法判断能否通过，是由于该部分还没有探索。

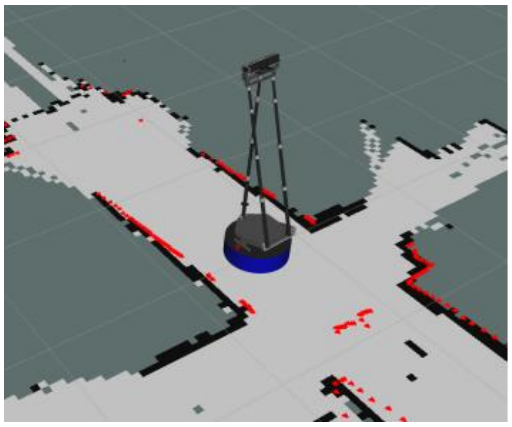

图 4-12 二维平面地图

由此可见，在栅格地图中，机器人自主导航问题就是寻找一条可以安全通行的路径，使其按照我们的控制要求在两点间自主移动。因此，我们需要保证以下两点：

（1）实时定位问题。机器人要每时每刻准确无误的知道自己的当前位置，才能确保找到目标点并且不让自己迷路。

（2）自主导航问题。规划出一条可以连续通行的安全路径是自主移动机器人的关键技术，如图 4-14 所示。但不能单单考虑只要是白色区域就是安全通行路径，还要考虑到

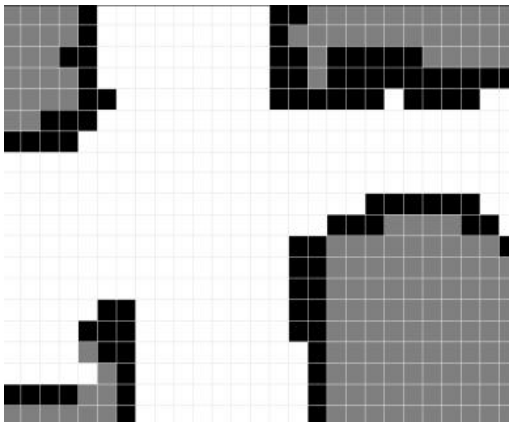

图 4-13 栅格地图

机器人自身的体积范围，保证白色区域的空间可以使机器人完全通过。所以会引入安全边界的概念，边界宽度等同于机器人底盘的直径，移动到安全边界里时会提示机器人会碰撞。

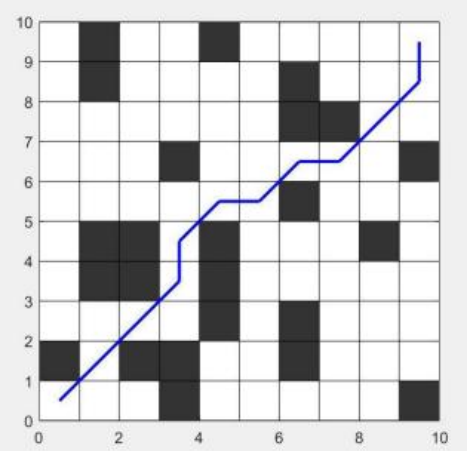

图 4-14 可通行栅格图

在 Rviz 中，自主导航运行之后，机器人在白色安全路径中寻找最短的一条并避开所有障碍，如图 4-15 所示，这条紫色的线就是机器人规划的路径。

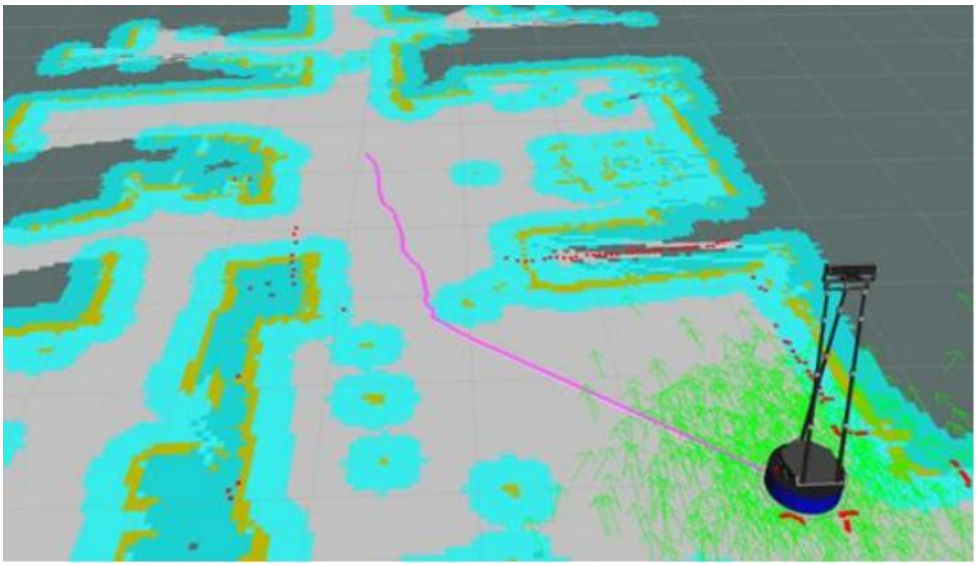

图 4-15 导航路径规划图

ROS 的自主导航技术包括两部分：

首先是 AMCL，全拼为 Adaptive Monte Carlo Localization，翻译过来是蒙特卡洛自适应定位算法[18]。在已知的场景地图中，该算法使用的是概率理论的方法，从而估计机器人本身的自定位置。其次，有了定位数据后，再应用机器人自主导航的中心枢纽——

Move_Base 进行路径规划，它是通过激光雷达、地图、AMCL 定位等数据，进行全局和局部的路径规划。

1. AMCL（蒙特卡洛自适应定位算法）

机器人在二维平面移动时，该算法可以在地图上定位机器人的位置。属于一种概率定位，先是估计机器人可能出现的多个位置，然后逐一筛选，只留下可靠的定位位置。比如在图 4-15 中，在机器人脚下周围的那些分散箭头就是估计出的位置，移动过程中，这些箭头从分散到逐渐收敛，直至达到可靠的位置。

2. Move_Base

在 ROS 系统的框架下，不仅有许多建图定位的算法，而且还有导航系统功能包提供的导航服务，其中非常重要的便是具有核心角色的 Move_Base 包。如图 4-16，在该框架中，自主导航系统的运行以及交互接口都由 Move_Base 功能包来提供。

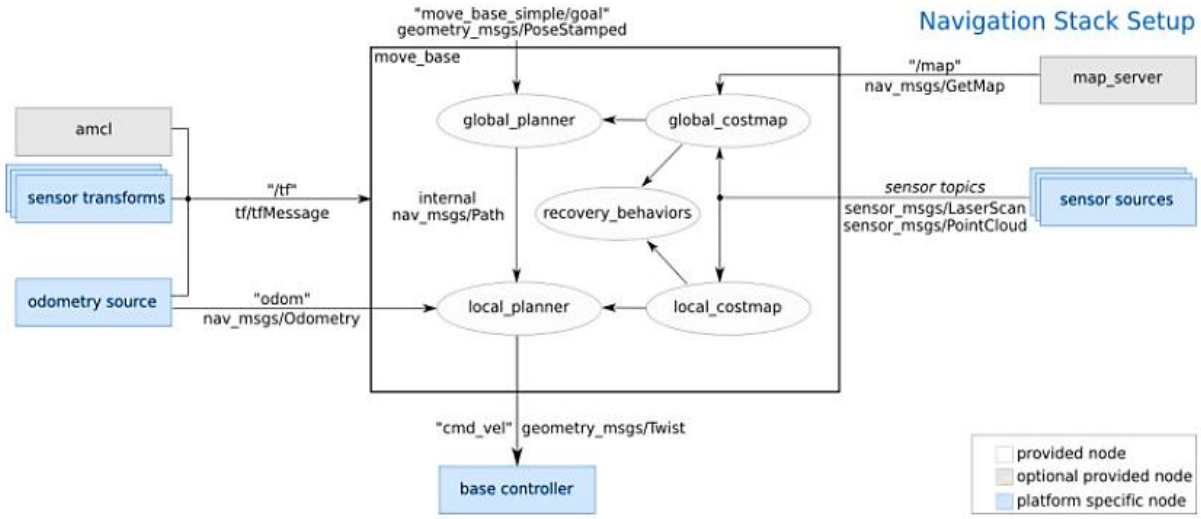

图 4-16 导航框架图

为了实现机器人全局最优路径规划与实时避障路径规划，Move_Base 需要订阅机器人发布的深度传感器信息（sensor_msgs/LaserScan 或 sensor_msgs/PointCloud）和里程计信息（nav_msgs/Odometry），同时完整的 tf 坐标变换也是实现路径规划的重要基础。导航框架最终的输出是控制机器人的速度指令（geometry_msgs/Twist）。

最终，Move_Base 功能包实现机器人导航中的最优路径规划，AMCL 实现二维地图中的机器人定位。

4.3 语音交互

在 Kinect2 视觉传感器的下方，是 4 个排布均匀的麦克风，可用于采集正前方的声音信息，可以用来接收语音指令，用于对外界声音信息的获取。ROS 系统的配套源码里提供了一个可以使用科大讯飞云服务进行语音指令识别的 Package 包，能够配合 Kinect2 的麦克风阵列完成语音识别功能。由于语音识别功能使用的是科大讯飞的云服务，所以需要将机器人电脑连接到互联网上，以便与讯飞的云服务器建立连接。

确认机器人电脑已经连接上网后，启动 xfyun_waterplus。输入指令：

roslaunch xfyun_waterplus sr_tts_en.launch

此时，科大讯飞的语音识别引擎启动，便可以听到机器人有节奏的发出“嘟~” 的提示音，这是语音识别开始的信号。我们需要在两次“嘟”的提示音之间将要识别的话说给机器人听，它才能够正确的识别。比如，我们在“嘟”的一声之后对机器人说“Hello”，机器人识别完毕后，会将识别结果显示在终端程序里，如图 4-17 所示。

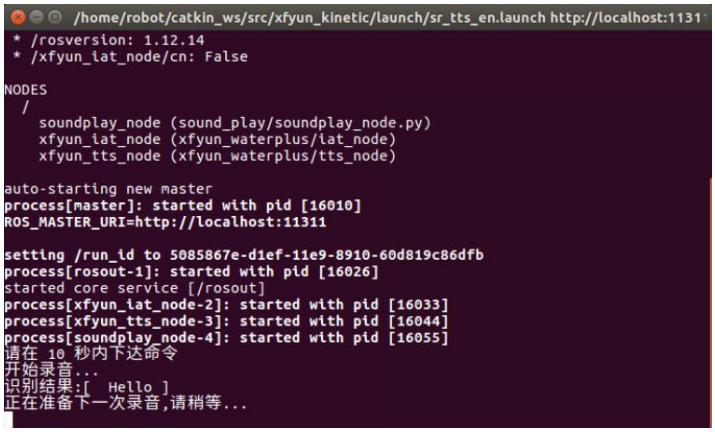

图 4-17 语音识别结果图

而语音输出功能是利用 ROS 系统的 sound_play 包，它可以将消息从字符串格式转化成语音输出。

具体做法是定义了一个 ros::NodeHandle 节点句柄 n，并使用这个句柄生成一个名为 tts_pub 的消息发布对象，该对象会向主题“/robotsound”发送消息，消息格式为 sound_play::SoundRequest。sound_play 语音合成包会从这个主题获取消息，把所获取的消息中的字符串转换成语音数据，再将语音通过音箱播放出来，因此能够使机器人进行说话。

4.4 物品抓取

对于物品的抓取，利用机器人的机械臂来完成，机械臂共有两种状态，一是折叠状态，二是展开状态，类似于人的手臂进行伸展。若想准确无误的抓取到物品，提高抓取的准确度，仅靠地图中构建的物品位置是不够的，还需要利用视觉传感器对物品的探测，才能更加精准的抓到所要物品。运用视觉传感器输出的三维点云，调用 PCL 算法，实现对物品的探测。

4.4.1 机械臂控制

1. 机械臂

其作用是用于抓取商品。初始时，默认该机械臂折叠状态，若需抓取商品时，再将机械臂升高并自然展开，随后，再依据商品的高度调整手臂的高度。如图 4-18 所示，具体阶段如下：

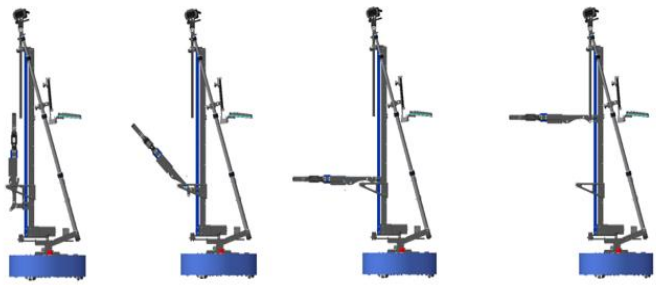

图 4-18 机械臂展开图

（1）展开阶段：将机械臂缓慢的升高，升到一定高度的时候，机械臂会摆脱折叠支架的束缚，重力的作用下向前伸展。完全伸展后，机器人的机械臂会处于水平状态。

（2）调节阶段：由于商品的不同高度，就需要基座滑块带动机械臂升至不同的高度，这样，才能在不同的高度上抓取物品。

（3）折叠阶段：机械臂折叠阶段是不论当前的机械臂高度，直接下降到最低端，直至收进折叠支架里，并保持竖直状态。

2. 机械手爪

如图 4-19 所示，手爪是抓取物品必不可少的一部分，超市服务机器人的机械手爪具有两个状态，一是张开状态，二是闭合状态。在进行编写程序时可以对手爪张开的间距进行设置，以更好的抓取不同大小的物体。只需要将需要抓取的目标物品坐标传递给抓取节点，机器人就能自主完成抓取行为。

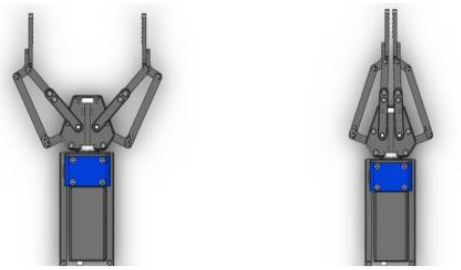

图 4-19 机械爪张开、闭合图

机械臂抓取动作以节点的形式安排在 wpb_home_behaviors 包中，这是自带的机器人的行为服务基础软件包。

4.4.2 物品探测

进行抓取动作时，首先要确保机器人搭载的视觉传感器已经探测到目标物品。因此，利用机器人头部传感器输出的三维点云，对物品进行探测。运用 PCL 平面检测算法，能够将目标物品的坐标标注出来，并计算其三维坐标。对于物品探测的功能，应用的是对 PCL 点云库功能的调用，在程序中，void GrabResultCallback 为获取抓取物品坐标的回调函数，回调函数的参数 geometry_msgs::Pose 携带了抓取目标物品的三维坐标。

在进行物品探测前，需要对超市服务机器人的头部 Kinect2 的姿态进行调节和校准。如图 4-20 所示，对于物品抓取来说，这个俯仰角可以有一定的浮动空间，建议选择在 25°到 30°之间。俯仰角过大会导致远处视野受限，当机器人和物品距离过远时无法探测到物品；俯仰角过小会导致近处的点云缺失，在机器人测量自己和桌面（或货架）距离的步骤时会出现误判。

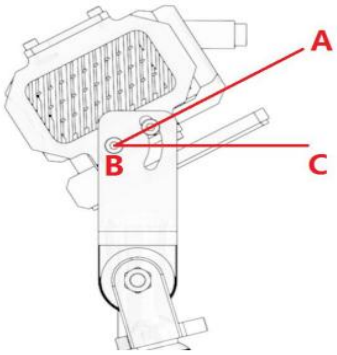

图 4-20 俯仰调节图

最后，手动调节完成角度后，还需要仔细调节 wpb_home.yaml 中的 Kinect2 高度（kinect_height）和俯仰数值（kinect_pitch），让 Rviz 里的地面点云尽可能贴合地面基准栅格。另外，还有对于抓取时补偿偏移量的调试，如图 4-21 的数值调节图。

图 4-21 数值调节图

4.5 本章小结

本章对机器人的各个功能进行详细讲解，包括实时建图、自主导航、语音交互以及物品抓取，并对每个功能实现所应用的算法和方法详细介绍。每个功能的实现思路做出简单陈述，具体实现详见下一章和附录。

第 5 章 程序设计及测试

通过前面的软硬件系统设计以及对机器人各个功能的实现，机器人已经具备在超市中进行服务的条件，若想完全自主可靠的进行作业，需要按照先前设计的系统流程图进行编程，使各个功能联系起来，并且通过编程使其按照我们设计的作业逻辑进行对顾客的服务。下面将对机器人的程序进行编程设计，并且运行具体功能进行一一测试，进一步研究机器人作业的效果。

5.1 程序设计

本节开始对机器人进行编程设计，如图 5-1 所示的系统流程图。

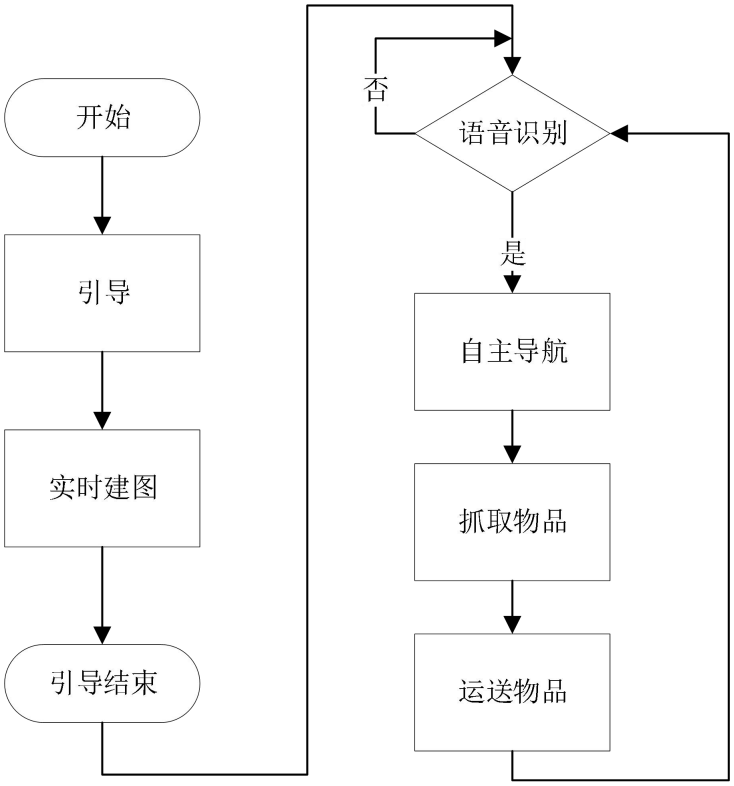

图 5-1 系统流程图

程序的实现思路如下：

（1）程序里实现了一个有限状态机，通过变量 nState 的赋值变化来控制状态的相互跳转，如表 5-1 所示的状态值含义表。

表 5-1 状态值含义表

状态值	含义
STATE_READY	等待引导
STATE_FOLLOW	机器人跟随引导移动，并使用激光雷达对环境建图
STATE_ASK	机器人到达等待区后，等待识别所需物品
STATE_GOTO	机器人从等待区出发，去往物品位置处
STATE_GRAB	机器人在物品位置处执行抓取动作
STATE_COMEBACK	机器人在抓取物品后返回等待区
STATE_PASS	机器人回到等待区后将物品交给面前的顾客

（2）void InitKeyword()是对语音识别关键词合集的初始化函数。桌子（或货架）的位置用物品的名字来命名，在编写程序时将可能会出现的物品名称都写到 arKeyword 中，在语音识别的时候会在需要识别的句子中识别是否有物品关键词的出现。

（3）string FindKeyword(string inSentence)是用于从语音识别的句子中寻找关键词的函数。

（4）void AddNewWaypoint(string inStr)函数能将机器人在地图中的当前位置保存为航点，参数为要保存的航点名称。

（5）void Speak(string inStr)为语音输出函数，参数为语音输出的句子。

（6）void KeywordCB(const std_msgs::String::ConstPtr & msg)函数为语音识别的回调函数，语音识别系统每识别到一个句子就会调用一次这个函数，同时将识别结果以参数形式传递到这个函数中。

（7）在主函数 int main()中，开始部分是对变量的初始化和各种主题订阅和发布操作，在后面的 while 循环里，是有限状态机的代码实现部分，在每种状态里根据条件完成状态跳转。

5.2 测试效果

（1）确保搭载 ROS 系统的笔记本电脑已连接互联网，保证语音识别和语音合成功能可以正常使用。

（2）将机器人移动到建图的起点，打开 Terminal 终端程序，运行服务指令，如图 5-2 所示：

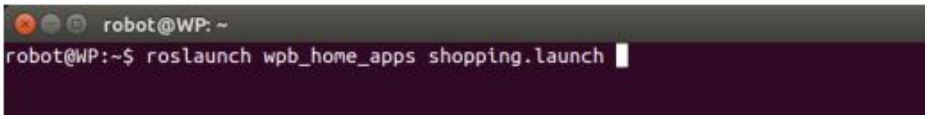

图 5-2 服务指令图

执行该指令，会弹出 Rviz 可视化界面，此时，建图功能已开启，并且在机器人的当前位置生成起点“start”，如图 5-3 所示。

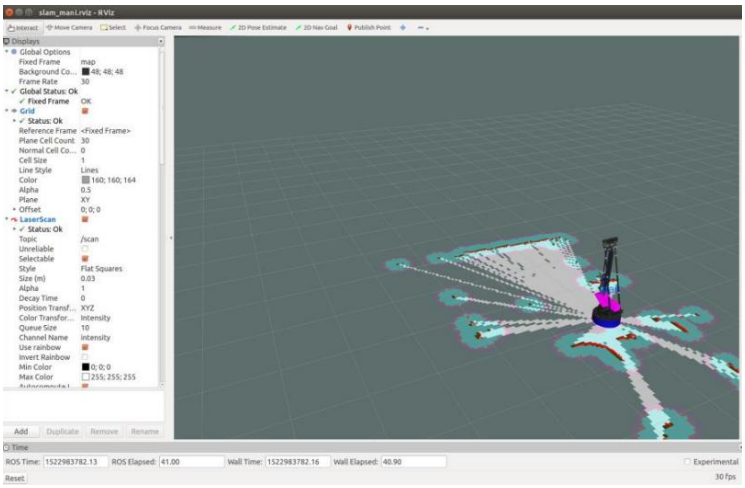

图 5-3 可视化界面图

（3）此时，我们站在距离机器人正前方 0.7 米处，机器人开始自动跟随并进一步完善地图的构建。

（4）将机器人引导到桌子（或货架）前，站在机器人和桌子（或货架）之间，并且让机器人的正面面向物品，如图 5-4 所示。

图 5-4 引导细节图

在“噔噔噔”的语音识别提示音后，说出物品名称，比如“water”，机器人在成功记录下物品位置后，做出回应来提示已经成功记录物品位置。此时在 Rviz 里就可以看到机器人将自己在地图里的当前位置和姿态记录为“water”的地点。

（5）物品的位置记录完成后，将机器人带到等待区，在听到“噔噔噔”的语音识别的提示音后，对机器人说“Stop following”，机器人停止跟随。同时 Rviz 里能观察到机器人把当前位置记录为“master”，如图所示。

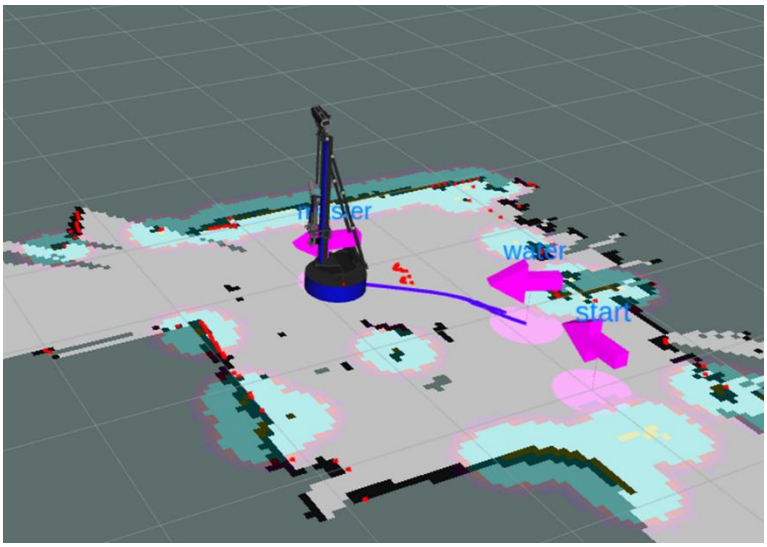

图 5-5 建图完成图

（6）在语音识别的提示音提示后说出一句带有物品关键词的句子，比如“Please give me water”，机器人成功识别到后，机器人便向“water”的位置移动。

（7）机器人到达“water”的位置后，抓取行为便启动，机械臂展开进入抓取状态，同时，视觉传感器对物品进行探测，如图 5-6 所示。机器人左右微调机器人位置，进一步对准并抓取物品。

图 5-6 机器人抓取图

（8）完成物品的抓取后，会自主导航回“master”位置，如图 5-7 所示。这个“master”就是刚才发出“Stop following”指令时的机器人位置。

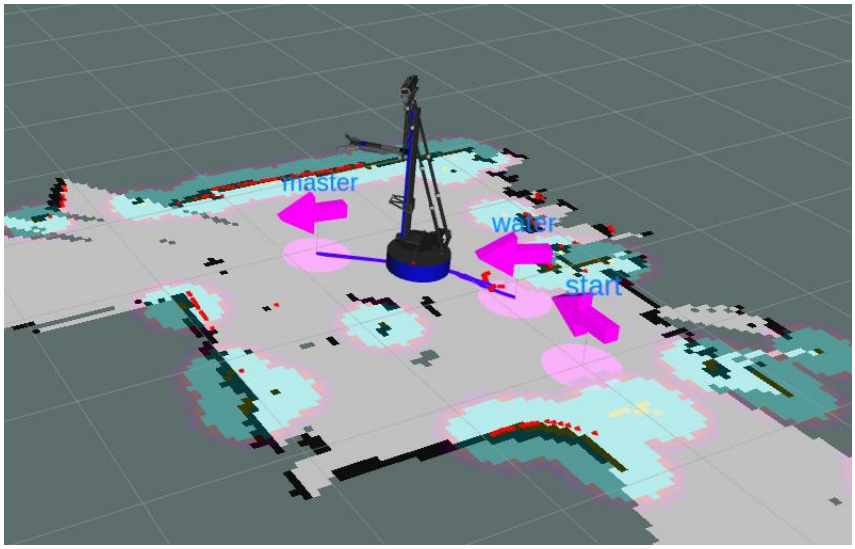

图 5-7 抓取返回图

（9）机器人回到“master”后，机械手爪松开，将物品交给面前的顾客。随后，会将机械手爪折叠收起。继续等待下一条的语音指令。

5.3 效果分析

通过上一小节的运行测试，总体来说符合功能需求，要想使机器人的各项功能更加完善，还需进一步调试。

在引导环节，由于是机器人跟随着人进行移动，因此，机器人前面的引导员非常重要，其跟随效果的好坏与引导员距离机器人的远近和引导员的步伐速度有很大关系。通过多次测试发现，在机器人前面引导过程中，尽量将距离控制在 0.5-0.7 之间，单位米，如果过远，将会影响与机器人的语音交互。另外，引导时，需要缓慢的往前移动，观察机器人是否跟随。顺利的话，机器人会在引导员到达其正前方 0.5 米的位置时开始跟随。为了获得理想的跟随效果，最好身着浅色服装，因为黑色衣物会吸收传感器发出的探测射线，没有回射信号会导致传感器无法检测到人体，导致跟随失败，从而影响实时建图的准确性。

5.4 本章小结

在本章中，对机器人的程序进行编程设计，将各个功能模块联系起来，使机器人流畅的运行工作，并测试机器人的完成效果，通过机器人的运行表现，符合本次课题的设计目标。

结　论

智能服务机器人的出现确实带动了社会发展和生活水平，但服务机器人目前现有技术还在上升时期，完全自主的服务机器人必将成为未来的潮流。本文以智能超市服务机器人作为研究对象，利用激光雷达读取超市环境，然后针对机器人的自主导航问题、避障问题以及商品抓取等问题进行深入研究。在建构地图过程中考虑机器人移动速度以及建构效果，在自主导航过程中考虑实时避障等问题。在物品夹取中考虑夹取精度等问题。最后通过编程调试，确保功能的完整性和稳定性。

本论文的主要研究成果和创新型有：

（1）本课题属于应用型的研究，在现实生活当中仔细观察，探索实际购物流程，运用大学期间所学知识设计一款实用的机器人，满足功能需求。对服务机器人的发展具有深远意义。

（2）在理论层面，以人工智能技术为核心，在仿生学、计算机科学等诸多学科前沿的理论的基础上，设计出了智能超市服务机器人；在方法层面，运用诸多高效的、精确的算法、运动控制、机械臂控制等技术，将技术和技术相结合。

（3）利用 ROS 系统的模块化编程特点，后期可以根据目标超市的不同规模、不同需求进行硬件和软件设计的量身定制，对模块中的代码，单独编译与组合。

随着近年来技术的发展，技术之间的更新换代极其迅速，机器人的更加智能化是必然趋势。在进行机器人的设计过程中，该服务机器人涉猎的知识点非常之广，本人的知识面也有限，在设计过程中遇到了不少的难题，接下来，我将就目前的设计成果与智能科技的发展，做出以下展望：

（1）针对智能超市服务机器人的实时建图问题，通过增加视觉传感器，与激光雷达相配合，建构更加准确真实的超市环境。同时，考虑在不同情况的地面以及不同水平面的地面等特殊环境下的实时建图问题。

（2）目前超市服务机器人的机械臂的工作空间有限，抓取范围小，对物品高度的要求非常严格，后续将继续研究机器人的机械臂，推进六轴机械臂的智能超市服务机器人。

（3）运用先进的人工智能技术，如人脸检测、物品识别等技术，让超市服务机器人自主的识别物品以及识别人脸，不再单一的记录顾客位置，当顾客位置移动时，机器人通过扫描四周找到当前服务顾客，并自主移动顾客身边。

参考文献

[1] 林为梁. 基于触视觉融合的机器人目标识别和滑动检测研究[D].燕山大学,2022.

[2] 陶永,刘海涛,等.我国服务机器人技术研究进展与产业化发展趋势.机械工程学报:1-19[2022-10-31].

[3] 服务机器人驶入应用快车道[J].机器人产业,2022(05):36-37.

[4] Xin Liu, Zhong Wang, et al. Simulation experiment of CCM-SLAM based on ROS[C]//.第 34 届中国控制与决策会议论文集（9）.2022:135-139.

[5] Yuan Debao,Zhang Jian,et al. Robustly Adaptive EKF PDR/UWB Integrated Navigation Based on Additional Heading Constraint.[J]. Sensors (Basel, Switzerland),2021,21(13).

[6] Ming Tang,Zhe Chen,et al. Robot Tracking in SLAM with Masreliez-Martin Unscented Kalman Filter[J]. International Journal of Control, Automation and Systems,2020.

[7] 王铎. 复杂环境下智能小车 SLAM 自适应导航研究[D].天津职业技术师范大学,2023.

[8] 许文杰. 基于激光雷达的栅格-拓扑地图构建算法研究与实现[D].电子科技大学,2019.

[9] 王彬. 室内全向自主移动机器人路径规划研究[D].中原工学院,2022.

[10] 李梦玮,赵晓飞,巩潇.基于故障树的服务机器人信息安全测评系统模型[J].工业技术创新,2020,07(03):24-29.

[11] 张英,廖如超,等.基于激光测距及 ROS 系统的无人机导线跟随方法研究[J].电子器件,2021,44(04):935-940.

[12] 孙琪. 自主导航机器人的定位与路径规划技术研究[D].长春理工大学,2020.

[13] 杨茸. 基于 DM 地标与激光雷达融合的物流机器人组合导航方法研究[D].重庆邮电大学,2021.

[14] 夏育泓,邓三鹏,等.基于多源信息的移动机器人建图方法研究[J].装备制造技术,2022(11):5-8+30.

[15] 赵连强,柳国良,等.一种自动规划路径 AGV 机器人的设计[J].科技风,2021(32):7-9.

[16] 夏天,任彧.基于加权 SDF 地图的二维激光 SLAM 方法[J].计算机应用与软件,2022,39(04):143-148.

[17] 宋鑫鹏,赵倩.基于 ROS 和 SLAM 的无人消杀机器人系统设计[J].自动化仪表,2023,44(01):61-65+71.

[18] 丁瑞敏. 深度学习型车间巡检机器人目标检测算法研究[D].西安工业大学,2022.

致　　谢

青葱四年，落笔为终；岁月不居，时节如流。我的四年大学时光即将结束，始于 2019 年金秋，终于 2023 年盛夏。光阴似箭，没来得及品味就已飞快消逝，目之所及，皆是回忆。在校园中，留下的是我们充满活力的青春，带走的是我们沉甸甸的收获。在此，感谢学校对我的栽培。纵有千般万般的不舍，但仍然心怀感激。

涓涓师恩，铭记于心。我要特别感谢我的指导老师——刘春平老师，从本次的毕业设计与最终的定稿，每一个环节都离不开刘春平老师的指导与帮助。大学四年，也正是各位老师的谆谆教导，让我学到了不少东西，掌握了许多技能。课上，老师耐心为我们传授知识，为我们答疑解惑、分享经验；课下，又是我们无话不谈的好朋友，帮助我们解决课余生活中的问题。谢谢您们，您们辛苦了。

不忘初心，以梦为马，不负韶华，只争朝夕。感谢努力的自己，坚持的自己。喷泉的漂亮，是因为压力；瀑布的壮观，是因为没有退路；人生同样如此，即使未看到胜利的曙光，但我仍会在心里保存一份光亮，勇毅前行。

言有穷而情不可终。愿你我都有归期。言辞有尽，感谢一切！

山水有来路，早晚复相逢。天津中德应用技术大学，我们后会有期。

附　　录

附录 1：程序源代码

```
#include <ros/ros.h>
#include <std_msgs/String.h>
#include "wpb_home_tutorials/Follow.h"
#include <geometry_msgs/Twist.h>
#include "xfyun_waterplus/IATSwitch.h"
#include <sound_play/SoundRequest.h>
#include <move_base_msgs/MoveBaseAction.h>
#include <actionlib/client/simple_action_client.h>
#include <waterplus_map_tools/Waypoint.h>
#include <waterplus_map_tools/GetWaypointByName.h>
#include <tf/transform_listener.h>
#include <geometry_msgs/PoseStamped.h>

using namespace std;

#define STATE_READY         0
#define STATE_FOLLOW        1
#define STATE_ASK           2
#define STATE_GOTO          3
#define STATE_GRAB          4
#define STATE_COMEBACK      5
#define STATE_PASS          6

typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction> MoveBaseClient;
static string strGoto;
static sound_play::SoundRequest spk_msg;
static ros::Publisher spk_pub;
static ros::Publisher vel_pub;
static string strToSpeak = "";
static string strKeyWord = "";
```

```
static ros::ServiceClient clientIAT;
static xfyun_waterplus::IATSwitch srvIAT;
static ros::ServiceClient cliGetWPName;
static waterplus_map_tools::GetWaypointByName srvName;
static ros::Publisher add_waypoint_pub;
static ros::ServiceClient follow_start;
static ros::ServiceClient follow_stop;
static ros::ServiceClient follow_resume;
static wpb_home_tutorials::Follow srvFlw;
static ros::Publisher behaviors_pub;
static std_msgs::String behavior_msg;

static ros::Subscriber grab_result_sub;
static ros::Subscriber pass_result_sub;
static bool bGrabDone;
static bool bPassDone;

static int nState = STATE_READY;
static int nDelay = 0;

static vector<string> arKeyword;

// 添加航点关键词
void InitKeyword()
{
    arKeyword.push_back("start");     //机器人开始启动的地点,最后要回去
    arKeyword.push_back("water");
    arKeyword.push_back("oreo");
    arKeyword.push_back("biscuit");
    arKeyword.push_back("lays");
    arKeyword.push_back("cookie");
    arKeyword.push_back("sprite");
    arKeyword.push_back("cola");
    arKeyword.push_back("orange juice");
    arKeyword.push_back("chip");
    arKeyword.push_back("hand wash");
```

```
        arKeyword.push_back("dish soap");
        arKeyword.push_back("shampoo");
        arKeyword.push_back("bread");
    }

    // 从句子里找 arKeyword 里存在的关键词
    static string FindKeyword(string inSentence)
    {
        string res = "";
        int nSize = arKeyword.size();
        for(int i=0;i<nSize;i++)
        {
            int nFindIndex = inSentence.find(arKeyword[i]);
            if(nFindIndex >= 0)
            {
                res = arKeyword[i];
                break;
            }
        }
        return res;
    }

    // 将机器人当前位置保存为新航点
    void AddNewWaypoint(string inStr)
    {
        tf::TransformListener listener;
        tf::StampedTransform transform;
        try
        {
            listener.waitForTransform("/map","/base_footprint", ros::Time(0),
ros::Duration(10.0) );
            listener.lookupTransform("/map","/base_footprint", ros::Time(0), transform);
        }
        catch (tf::TransformException &ex)
        {
            ROS_ERROR("[lookupTransform] %s",ex.what());
```

```
            return;
        }

        float tx = transform.getOrigin().x();
        float ty = transform.getOrigin().y();
        tf::Stamped<tf::Pose> p = tf::Stamped<tf::Pose>(tf::Pose(transform.getRotation() ,
tf::Point(tx, ty, 0.0)), ros::Time::now(), "map");
        geometry_msgs::PoseStamped new_pos;
        tf::poseStampedTFToMsg(p, new_pos);

        waterplus_map_tools::Waypoint new_waypoint;
        new_waypoint.name = inStr;
        new_waypoint.pose = new_pos.pose;
        add_waypoint_pub.publish(new_waypoint);

        ROS_WARN("[New Waypoint] %s ( %.2f , %.2f )" , new_waypoint.name.c_str(), tx, ty);
    }

    // 语音说话
    void Speak(string inStr)
    {
        spk_msg.arg = inStr;
        spk_msg.volume = 1.0f;
        spk_pub.publish(spk_msg);
    }

    // 跟随模式开关
    static void FollowSwitch(bool inActive, float inDist)
    {
        if(inActive == true)
        {
            srvFlw.request.thredhold = inDist;
            if (!follow_start.call(srvFlw))
            {
                ROS_WARN("[CActionManager] - follow start failed...");
            }
```

```
        }
        else
        {
            if (!follow_stop.call(srvFlw))
            {
                ROS_WARN("[CActionManager] - failed to stop following...");
            }
        }
}

// 物品抓取模式开关
static void GrabSwitch(bool inActive)
{
        if(inActive == true)
        {
            behavior_msg.data = "grab start";
            behaviors_pub.publish(behavior_msg);
        }
        else
        {
            behavior_msg.data = "grab stop";
            behaviors_pub.publish(behavior_msg);
        }
}

// 物品递给开关
static void PassSwitch(bool inActive)
{
        if(inActive == true)
        {
            behavior_msg.data = "pass start";
            behaviors_pub.publish(behavior_msg);
        }
        else
        {
            behavior_msg.data = "pass stop";
```

```
            behaviors_pub.publish(behavior_msg);
        }
}

// 语音识别结果处理函数
void KeywordCB(const std_msgs::String::ConstPtr & msg)
{
    ROS_WARN("------ Keyword = %s ------",msg->data.c_str());
    if(nState == STATE_FOLLOW)
    {
        // 从识别结果句子中查找物品（航点）关键词
        string strKeyword = FindKeyword(msg->data);
        int nLenOfKW = strlen(strKeyword.c_str());
        if(nLenOfKW > 0)
        {
            // 发现物品（航点）关键词
            AddNewWaypoint(strKeyword);
            string strSpeak = strKeyword + " . OK. I have memoried. Next one , please";
            Speak(strSpeak);
        }

        // 停止跟随的指令
        int nFindIndex = msg->data.find("top follow");
        if(nFindIndex >= 0)
        {
            FollowSwitch(false, 0);
            AddNewWaypoint("master");
            nState = STATE_ASK;
            nDelay = 0;
            Speak("ok");

        }
    }

    if(nState == STATE_ASK)
    {
```

```
            // 从识别结果句子中查找物品（航点）关键词
            string strKeyword = FindKeyword(msg->data);
            int nLenOfKW = strlen(strKeyword.c_str());
            if(nLenOfKW > 0)
            {
                // 发现物品（航点）关键词
                strGoto = strKeyword;
                string strSpeak = strKeyword + " . OK. I will go to get it for you.";
                Speak(strSpeak);
                nState = STATE_GOTO;
            }
        }
}

// 物品抓取状态
void GrabResultCallback(const std_msgs::String::ConstPtr& res)
{
        int nFindIndex = 0;
        nFindIndex = res->data.find("done");
        if( nFindIndex >= 0 )
        {
            bGrabDone = true;
        }
}




// 物品递给状态
void PassResultCallback(const std_msgs::String::ConstPtr& res)
{
        int nFindIndex = 0;
        nFindIndex = res->data.find("done");
        if( nFindIndex >= 0 )
        {
            bPassDone = true;
```

```
        }
    }


    int main(int argc, char** argv)
    {
        ros::init(argc, argv, "wpb_home_shopping");

        ros::NodeHandle n;
        ros::Subscriber sub_sr = n.subscribe("/xfyun/iat", 10, KeywordCB);
        follow_start = n.serviceClient<wpb_home_tutorials::Follow>("wpb_home_follow/start");
        follow_stop = n.serviceClient<wpb_home_tutorials::Follow>("wpb_home_follow/stop");
        follow_resume                                                                    =
n.serviceClient<wpb_home_tutorials::Follow>("wpb_home_follow/resume");
        cliGetWPName                                                                     =
n.serviceClient<waterplus_map_tools::GetWaypointByName>("/waterplus/get_waypoint_name");
        add_waypoint_pub                                                                 =
n.advertise<waterplus_map_tools::Waypoint>( "/waterplus/add_waypoint", 1);
        spk_pub = n.advertise<sound_play::SoundRequest>("/robotsound", 20);
        spk_msg.sound = sound_play::SoundRequest::SAY;
        spk_msg.command = sound_play::SoundRequest::PLAY_ONCE;
        vel_pub = n.advertise<geometry_msgs::Twist>("/cmd_vel", 10);
        clientIAT                                                                        =
n.serviceClient<xfyun_waterplus::IATSwitch>("xfyun_waterplus/IATSwitch");
        behaviors_pub = n.advertise<std_msgs::String>("/wpb_home/behaviors", 30);
        grab_result_sub                                                                  =
n.subscribe<std_msgs::String>("/wpb_home/grab_result",30,&GrabResultCallback);
        pass_result_sub                                                                  =
n.subscribe<std_msgs::String>("/wpb_home/pass_result",30,&PassResultCallback);

        InitKeyword();

        ROS_WARN("[main] wpb_home_shopping");
        ros::Rate r(30);
        while(ros::ok())
        {
```

```
// 1、刚启动，准备
if(nState == STATE_READY)
{
    //  启动后延迟一段时间然后开始跟随
    nDelay ++;
    // ROS_WARN("[STATE_READY] - nDelay = %d", nDelay);
    if(nDelay > 100)
    {
        AddNewWaypoint("start");
        nDelay = 0;
        nState = STATE_FOLLOW;

    }
}

// 2、跟随阶段
if(nState == STATE_FOLLOW)
{
    if(nDelay == 0)
    {
       FollowSwitch(true, 0.65);
    }
    nDelay ++;
}

// 3、询问要去哪个航点
if(nState == STATE_ASK)
{


}

// 4、导航去指定航点
if(nState == STATE_GOTO)
{
    srvName.request.name = strGoto;
```

```
                    if (cliGetWPName.call(srvName))
                    {
                        std::string name = srvName.response.name;
                        float x = srvName.response.pose.position.x;
                        float y = srvName.response.pose.position.y;
                        ROS_INFO("[STATE_GOTO] Get_wp_name = %s (%.2f,%.2f)",
strGoto.c_str(),x,y);

                        MoveBaseClient ac("move_base", true);
                        if(!ac.waitForServer(ros::Duration(5.0)))
                        {
                            ROS_INFO("The move_base action server is no running. action
abort...");
                        }
                        else
                        {
                            move_base_msgs::MoveBaseGoal goal;
                            goal.target_pose.header.frame_id = "map";
                            goal.target_pose.header.stamp = ros::Time::now();
                            goal.target_pose.pose = srvName.response.pose;
                            ac.sendGoal(goal);
                            ac.waitForResult();
                            if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
                            {
                                ROS_INFO("Arrived at %s!",strGoto.c_str());
                                Speak("OK. I am taking it.");
                                nState = STATE_GRAB;
                                nDelay = 0;
                            }
                            else
                            {
                                ROS_INFO("Failed to get to %s ...",strGoto.c_str() );
                                Speak("Failed to go to the waypoint.");
                                nState = STATE_ASK;
                            }
                        }
```

```
        }
        else
        {
            ROS_ERROR("Failed to call service GetWaypointByName");
            Speak("There is no this waypoint.");
            nState = STATE_ASK;
        }
    }

    // 5、抓取物品
    if(nState == STATE_GRAB)
    {
        if(nDelay == 0)
        {
            bGrabDone = false;
            GrabSwitch(true);
        }
        nDelay ++;
        if(bGrabDone == true)
        {
            GrabSwitch(false);
            Speak("I got it. I am coming back.");
            nState = STATE_COMEBACK;
        }
    }

    // 6、抓完物品回来
    if(nState == STATE_COMEBACK)
    {
        srvName.request.name = "master";
        if (cliGetWPName.call(srvName))
        {
            std::string name = srvName.response.name;
            float x = srvName.response.pose.position.x;
            float y = srvName.response.pose.position.y;
```

```
                    ROS_INFO("[STATE_COMEBACK] Get_wp_name = %s (%.2f,%.2f)",
strGoto.c_str(),x,y);

                    MoveBaseClient ac("move_base", true);
                    if(!ac.waitForServer(ros::Duration(5.0)))
                    {
                        ROS_INFO("The move_base action server is no running. action
abort...");
                    }
                    else
                    {
                        move_base_msgs::MoveBaseGoal goal;
                        goal.target_pose.header.frame_id = "map";
                        goal.target_pose.header.stamp = ros::Time::now();
                        goal.target_pose.pose = srvName.response.pose;
                        ac.sendGoal(goal);
                        ac.waitForResult();
                        if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
                        {
                            ROS_INFO("Arrived at %s!",strGoto.c_str());
                            Speak("Hi,master. This is what you wanted.");
                            nState = STATE_PASS;
                            nDelay = 0;
                        }
                        else
                        {
                            ROS_INFO("Failed to get to %s ...",strGoto.c_str() );
                            Speak("Failed to go to the master.");
                            nState = STATE_ASK;
                        }
                    }

                }
                else
                {
                    ROS_ERROR("Failed to call service GetWaypointByName");
```

```
                    Speak("There is no waypoint named master.");
                    nState = STATE_ASK;
                }
            }

            // 7、将物品给主人
            if(nState == STATE_PASS)
            {
                if(nDelay == 0)
                {
                    bPassDone = false;
                    PassSwitch(true);
                }
                nDelay ++;
                if(bPassDone == true)
                {
                    PassSwitch(false);
                    Speak("OK. What do you want next?");
                    nState = STATE_ASK;
                }
            }

            ros::spinOnce();
            r.sleep();
        }

        return 0;
}
```