物品检测与抓取机器人

设计说明书

SDD102

V4.0

分工说明

小组名称	rushbot	
学号	姓名	本文档中主要承担的工作内容
17373060	杨开元	完成详细设计中 4、5、6 模块；合作完成体系结构设计；完成接口设计
17373517	王正达	完成详细设计中 1、2 模块；合作完成体系结构设计
15241035	彭文鼎	完成详细设计中 3、8 模块；完成范围与需求概述
17373288	尹俊成	完成详细设计中 7 模块；完成需求可追踪性说明

版本变更历史

版本	提交日期	主要编制人	审核人	版本说明
V1.0.0	2020.4.20	王正达	杨开元	本小组软件设计说明初稿
V1.1.0	2020.4.22	王正达	杨开元	部分格式修改
V2.0.0	2020.4.23	王正达	杨开元	完成 issue#4 对应修改
V3.0.0	2020.5.19	王正达	杨开元	修改了详细设计部分
V4.0.0	2020.6.7	王正达	杨开元	本小组软件设计说明最终稿

目录

1. 范围 1

1.1 项目概述 1

1.1.1 开发背景 1

1.1.2 主要功能和需求 1

1.1.3 应用场景 1

1.2 文档概述 2

1.3 术语和缩略词 2

1.4 引用文档 2

2. 需求概述 3

2.1 总体描述 3

2.2 用例图 5

2.3 用例模型 5

2.3.1 用例一：启动机器人 5

2.3.2 用例二：关闭机器人 6

2.3.3 用例三：初始化地图场景建模 6

2.3.4 用例四：货架位置标记 7

2.3.5 用例五：指定商品获取 8

2.4 功能需求 9

2.4.1 功能一(用例三) 9

2.4.2 功能二(用例四) 10

2.4.3 功能三(用例五) 10

2.5 非功能需求 11

2.5.1 系统性能需求 11

2.5.2 系统质量需求 11

（1）可维护性与可扩展性 11

（2）可靠性 11

（3）易用性 11

2.6 用户界面需求 11

3. 数据库设计 12
3.1 实体概述 12
3.1.1 用户 12
3.1.2 机器人运动参数 12
3.1.3 地图 12
3.1.4 关键词库 13
3.1.5 机械臂运动参数 13
3.1.6 日志 13
3.2 ER 图 13
4. 体系结构设计 14
4.1 总体结构 14
4.1.1 软件体系结构 14
4.1.2 硬件体系结构 17
5. 接口设计 18
6. 详细设计 19
6.1 运动模块 20
6.2 SLAM 建图模块 20
6.3 导航模块 22
6.4 检测模块 24
6.5 抓取模块 28
6.6 语音识别模块 30
7. 运行与开发环境 31
7.1 硬件环境 31
7.1.1 机器人的结构组成 31
7.1.2 主要硬件支持 32
7.1.3 工作环境 32
7.2 软件环境 32
7.2.1 Ubuntu16.04 32
7.2.2 ROS 33

7.2.3 基础软件包.....33
7.2.4 扩展软件包.....33
8. 需求可追踪性说明.....34
8.1 功能性需求可追踪性说明.....34
8.1.1 启动和关闭机器人.....34
8.1.2 初始化地图场景建模.....34
8.1.3 货架地点记忆.....34
8.1.4 指定商品获取.....34
8.1.5 异常处理.....35
8.2 非功能需求可追踪性说明.....35
8.2.1 系统可扩展性需求.....35
8.2.2 系统易用性需求.....35
8.2.3 可靠性需求.....35

1. 范围

1.1 项目概述

1.1.1 开发背景

随着计算机技术的发展，网络的普及，硬件和制造业的进步，我们在越来越多的领域都能看到机器人的身影。人类希望将那些枯燥重复、有流程约定的任务交到价格低廉、可以全天候工作的机器人手中去，进而提高效率，减少成本。

近些年来，机器人领域不断在取得进步，机器人开发也离普通人越来越近。ROS 系统就是一个适用于机器人开发的开源操作系统，提供了操作系统应有的服务，如抽象、消息传递、包管理等等。ROS 还是一个分布式的节点框架，进程被封装在易于分享和发布的程序包和功能包中，工程也可以被 ROS 的基础工具方便地整合在一起。

与此同时，越来越多的新技术也正在 ROS 系统中发挥作用，如开源的 OpenCV、PCL 图像库、语音识别，结合这些技术机器人也有了更广泛的作用领域，更好的人机交互体验等等。

1.1.2 主要功能和需求

我们项目的最终目标是实现一个可以自主移动、主动避障以及对目标物进行检测和抓取的简易机器人。它拥有自主分析地图信息以及在此基础上预先进行路径规划的功能。为改善人机交互体验，我们也会尝试着加入语音识别与控制系统。

在功能性需求之外，我们还需要保证机器人能够稳定地完成高质量的工作，即系统的可靠性。此外，系统的可扩展性需求也很重要。机器人系统应能够适应一定程度内系统容量的扩大和管理内容的增加。最后，为了改善用户的体验以及扩大机器人的适用人群，机器人系统使用起来应该足够简单。

1.1.3 应用场景

该机器人简单易用，功能较为全面的特点使得其有很多值得展望的应用场景。

该机器人可以在无人职守的商店里 24 小时值班并销售物品，面对提前规格化的货架和机器人自己建立的地图，只需要让其记录商品所在货架的位置，加上语音识别模块，就能很好地和人类交互，让这种机器人在夜间或全天服务再好不过。

通过改进机械臂的构造，该机器人可以实现物品的分捡，可以被安排在物流公司这种急需劳动力、垃圾分类点这种环境较差的场景中。

此外，该机器人也可以在家中充当保姆的角色，将其安置在行动不便老人的家中，通过老人和机器人的互动，机器人可以学习用户身边的物品，并在用户发出指令之后将物品取来或归位。

1.2 文档概述

本文档对系统的需求与业务逻辑进行进一步分析设计，对需求分析阶段出现的问题进行详细复盘，建立设计与需求之间的关系，给出完整的软件开发设计说明书。

1.3 术语和缩略词

表 1 术语和缩略词列表

术语和缩略词	简要解释
ROS	Robot Operating System 编写机器人软件的高度灵活性框架
OpenCV	Open Source Computer Vision Library 轻量高效的开源计算机视觉库
PCL	Point Cloud Library 实现 3D 视图下处理计算的点云库

1.4 引用文档

【1】王鹏飞. 基于 ROS 的多传感器信息融合自主导航控制系统设计与实现[D]. 南京邮电大学, 2019.

【2】北京六部工坊科技有限公司. 启智 ROS 机器人开发手册 V1.1.0

【3】 SRS102 需求规格说明书

【4】 SDP102 项目开发计划

2. 需求概述

2.1 总体描述

机器人主要使用场景为在空间有限、货架有限的超市内。主要任务是协助用户购物，无人售货等。

在顾客使用之前，该机器人需要超市管理员对机器人进行相关配置。首先机器人将在管理员操控下进行地图建模。然后，超市管理员将打开建立好的地图文件并根据实际情况标记货架位置。以上工作完成之后机器人可以在超市入口或特定位置等待顾客。

顾客进入超市时，对机器人发出语音指令，说明待机器人抓取的物品，机器人将当前位置记录为顾客位置，之后机器人从当前位置自动规划路径前往包含该物品的货架，进行物品的识别和抓取。最后，机器人导航回到初始位置，发出相应提示。在整个业务流程中，机器人实现自主避障。完成一个业务循环。

超市管理员可以通过机器人机载电脑或者远程控制的方式启动、关闭机器人；通过 UI 界面查看地图及机器人所处位置；机器人将对成功获取到的指令及时做出相应，在到达取物地点和抓取物品成功后发出提示，并对错误和异常情况做出判断、处理和提示，以提供良好的可调试性。

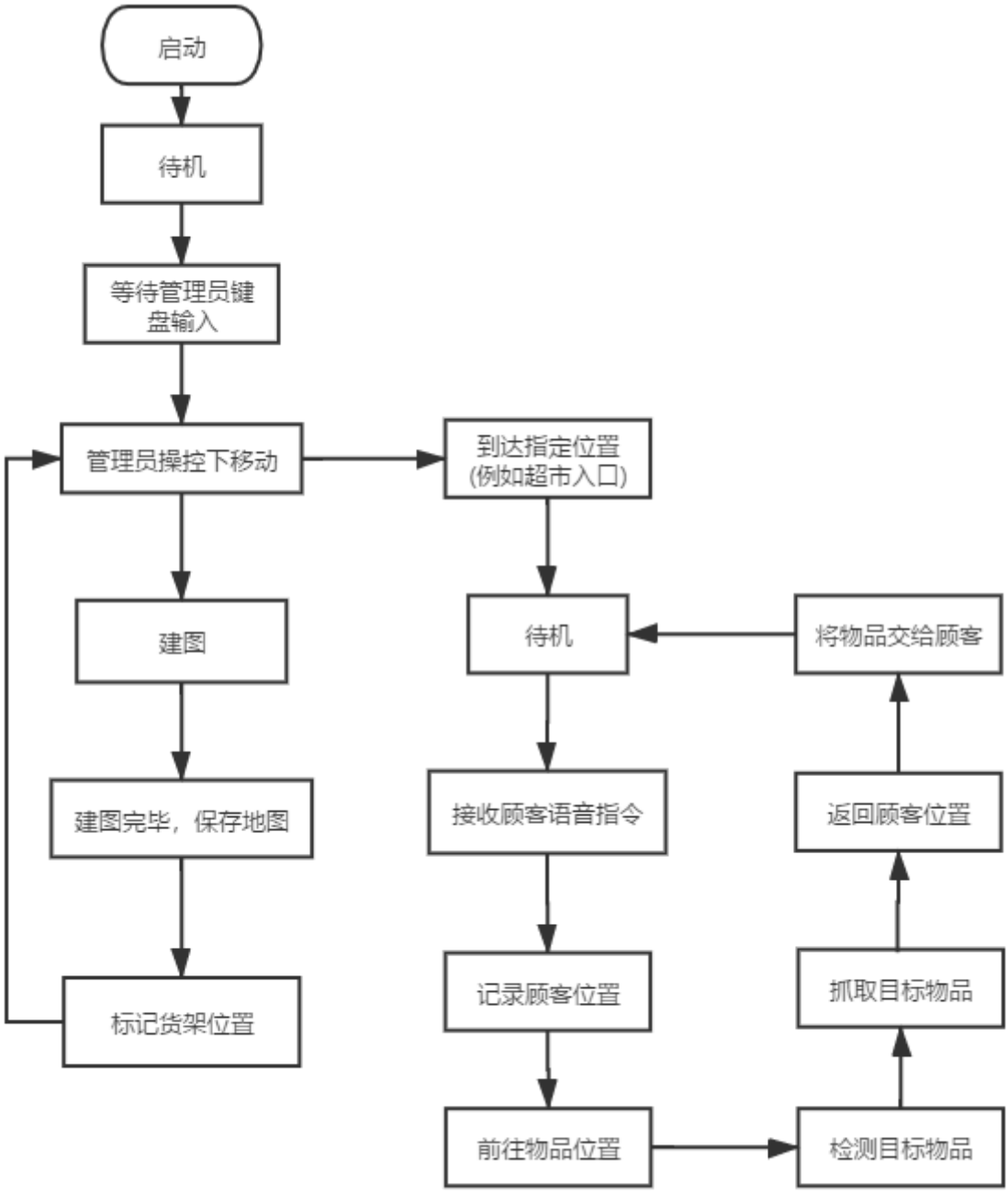
启动
待机
等待管理员键盘输入
管理员操控下移动
建图
建图完毕，保存地图
标记货架位置
到达指定位置(例如超市入口)
待机
接收顾客语音指令
记录顾客位置
前往物品位置
检测目标物品
抓取目标物品
返回顾客位置
将物品交给顾客

图 1 业务流程图

2.2 用例图

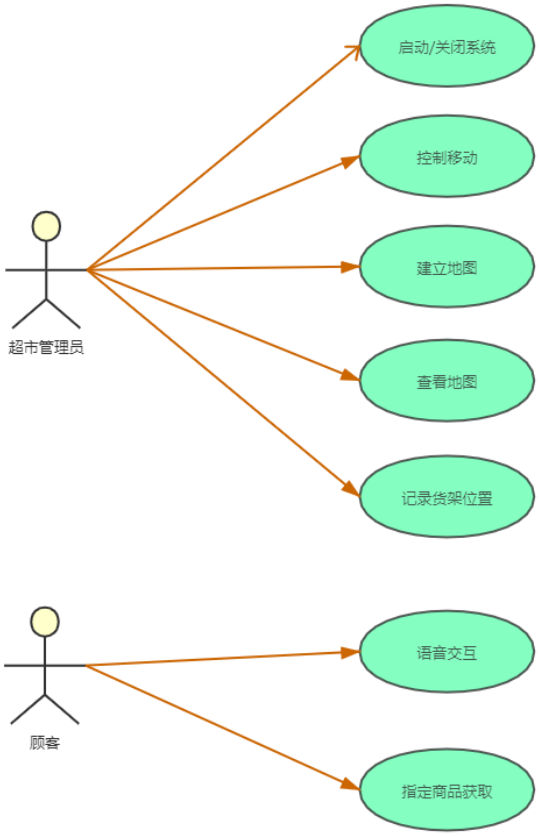

图 2 用例图

2.3 用例模型

2.3.1 用例一：启动机器人

主要参与者：超市工作人员

目标：使机器人系统各模块进入工作状态，打开人机交互界面

前置条件：机器人各模块配置完毕；超市工作人员了解规范启动步骤

启动：超市工作人员希望使用机器人

场景：

1. 接通机器人电源；
2. 通过 USB 接口连接机载电脑和机器人运动控件；
3. 启动机载电脑并打开机器人系统；

4. 机载电脑显示人机交互界面；

优先级：高

何时可用：第一个增量

使用频率：低

次要参与者：机器人运动控件、机载电脑

2.3.2 用例二：关闭机器人

主要参与者：超市工作人员

目标：关闭机器人，确保下次启动时各模块状态正常

前置条件：机器人各模块配置完毕；超市工作人员了解规范关闭步骤；机器人未处于工作状态

启动：超市暂时不再需要机器人

场景：

1. 关闭机器人系统；
2. 解除机载电脑和机器人运动控件的 USB 连接；
3. 关闭机载电脑；

优先级：高

何时可用：第一个增量

使用频率：低

次要参与者：机器人运动控件、机载电脑

2.3.3 用例三：初始化地图场景建模

主要参与者：超市工作人员

目标：确保机器人系统正确记录超市地图

前置条件：机器人系统已正常启动；超市工作人员熟悉超市地形

启动：超市工作人员点击“开始建图”按钮

场景：

1. 机器人放置于超市入口处并提示启动成功；
2. 超市工作人员点击“开始建图”按钮；

3. 超市工作人员键盘操控机器人在超市内穿梭移动；
4. 机载电脑人机界面实时显示地图建模情况；
5. 超市工作人员操控机器人回到超市入口；
6. 超市工作人员通过机载电脑界面查看场景建模情况；
7. 确保建模无误后，超市工作人员点击“停止建图”按钮；
8. 机器人收到指令，回到待命状态；

异常情况：

1. 场景建模不完善 出现点：6

解决方案：

1. 工作人员点击“停止建图”按钮，操作机器人回到超市入口，点击“开始建图”按钮，重新开始地图建模；

优先级：高

何时可用：第一个增量

使用频率：中

次要参与者：机器人运动控件、机载电脑

2.3.4 用例四：货架位置标记

主要参与者：超市工作人员

目标：确保机器人系统正确记录货架位置

前置条件：机器人系统已正常启动；超市工作人员熟悉超市地形；所有商品名称已录入系统数据库；地图建模已经完成；

启动：超市工作人员点击“记忆货架”按钮

场景：

1. 机器人放置于超市入口处并提示启动成功；
2. 超市工作人员点击“记忆货架”按钮；
3. 机器人 UI 界面弹出建好的地图。
4. 超市工作人员在地图上对所有货架的位置进行一一标记与命名；
5. 标记完毕，工作人员关闭地图。
6. 机器人进入待命状态；

异常情况：

无

解决方案：

无

优先级：中

何时可用：第二个增量

使用频率：中

次要参与者：机器人运动控件、机载电脑

2.3.5 用例五：指定商品获取

主要参与者：顾客

目标：命令机器人到相应货架取回指定商品

前置条件：超市地图初始化已完成，指定商品位置已正确记录；机器人处于待命状态；

启动：顾客点击“获取商品”按钮并选择目标商品；

场景：

1. 机器人放置于超市入口处并处于待命状态；
2. 顾客在机器人交互界面上点击“获取商品”按钮；
3. 顾客说出自己的需求，比如：请给我一瓶水。
4. 机器人收到指令，记录顾客所在位置，提示“马上出发，请您在原地耐心等待”；
5. 机器人规划前往指定商品所在货架的路径；
6. 机器人移动，路上进行自主避障；
7. 机器人到达目标货架；
8. 机器人摄像头检测商品位置；
9. 机械臂启动，开始抓取目标商品；
10. 目标商品抓取成功；
11. 机器人规划返回路径；
12. 机器人移动，路上进行自主避障；

13. 机器人返回顾客所在位置；

14. 机器人松开机械臂，将商品移交给顾客；

15. 机器人进入待命状态；

异常情况：

1. 机器人在移动过程中遇到无法绕过的障碍物
2. 机器人返回顾客位置时顾客已经离开

解决方案：

1. 机器人停在原地，发出警报声，并显示“请帮我移开障碍物”提醒周围的超市管理人员前来处理，管理人员移开障碍物之后，机器人继续移动。
2. 机器人停在原地，发出警报声，并显示“我找不到顾客了”，提醒周围的超市管理人员前来处理，管理人员前来取走商品放回货架，并停止机器人的报警状态。

2.4 功能需求

2.4.1 功能一(用例三)

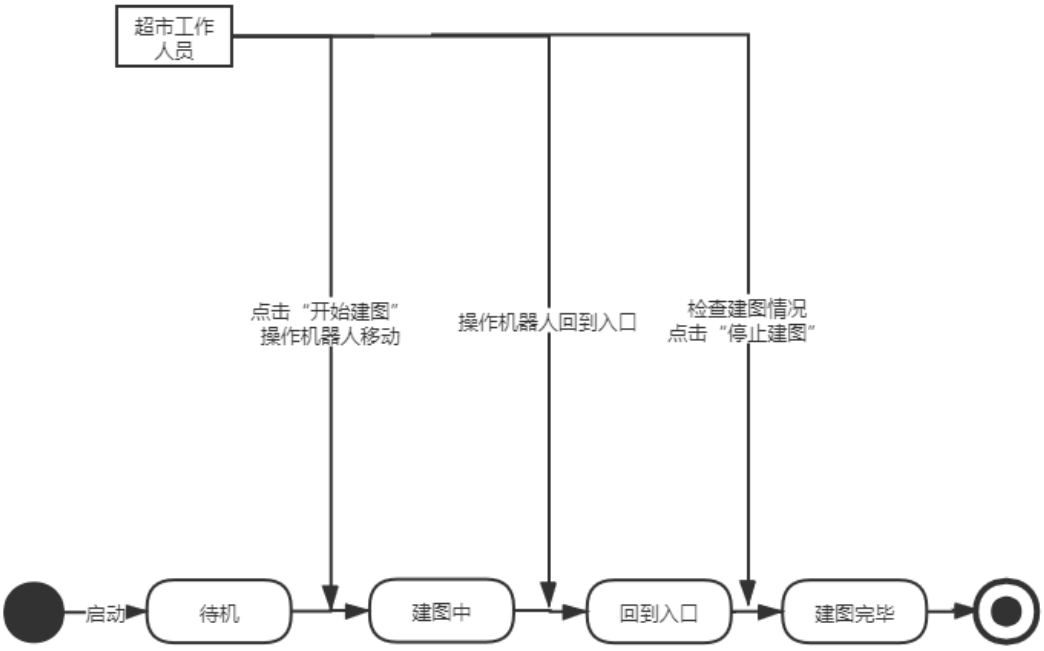

图 3 用例三示意图

2.4.2 功能二(用例四)

图 4 用例四示意图

2.4.3 功能三(用例五)

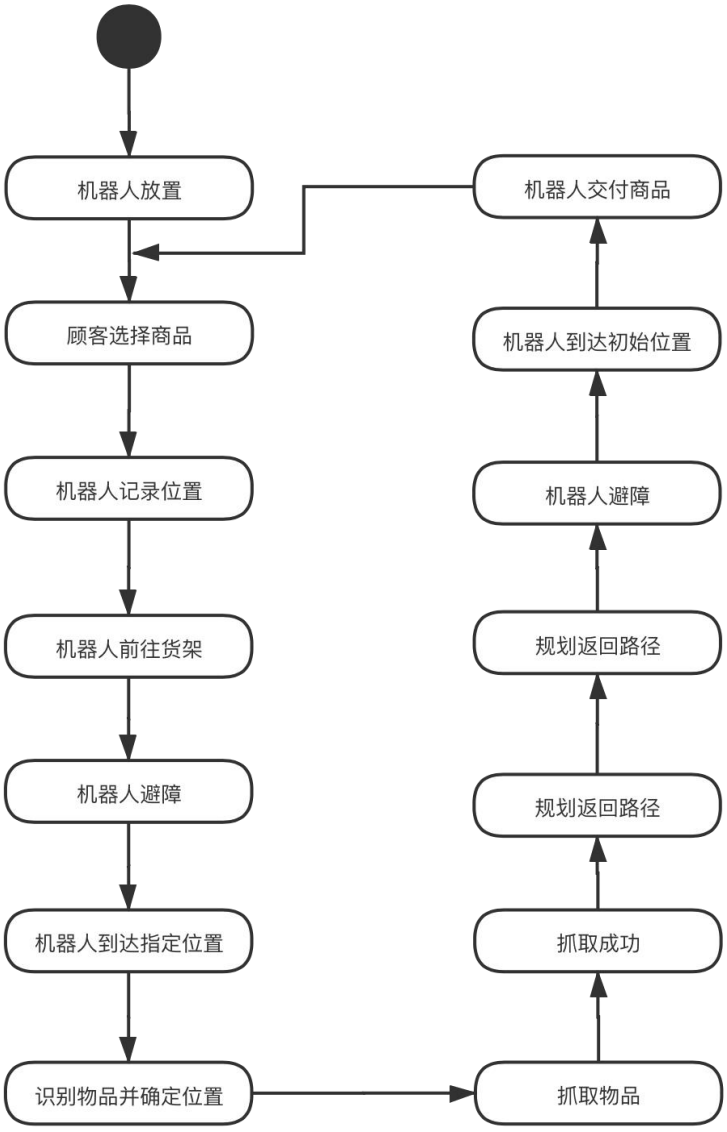

图 6 用例五示意图

2.5 非功能需求

2.5.1 系统性能需求

响应时间：机器人系统的各个功能环节都要求较短的响应时间，尤其是取货时需要较高效率，否则会大大降低用户体验和业务表现。同时客户端界面跳转需要实时，比如不超过 3s.

资源利用率：指系统投入服务器、数据库、硬件设备等资源，所发挥的资源利用百分比。希望投入的资源最大化的利用，而不被闲置。系统不必要且消耗资源大的数据或功能需要定期清理或优化。

2.5.2 系统质量需求

（1）可维护性与可扩展性

机器人设备应具备良好的可维护性和可扩展性，能够适应系统内容的扩大和管理内容的增加，包括软硬件平台、系统结构、功能设计。随着功能数量增加和复杂化，要求系统具有能够便于持续维护和灵活扩展。

（2）可靠性

系统要求能够稳定可靠地运行，在大量的测试与实际使用中提供正确的服务。

（3）易用性

系统对用户而言需要易于学习和使用，设计的重点在于让产品的设计能够符合使用者的习惯与需求。具体需求包括易学习性、易操作性、用户错误防御机制、用户界面美观等。

2.6 用户界面需求

顾客通过语音发送指令控制机器人交互，用户界面则是系统的控制面板。

用户界面提供的信息有：一是向用户反馈机器人响应结果，二是为用户提供指令输入模块，三是可以查看机器人状态以及任务完成情况等信息。

超市管理员可以通过机器人机载电脑或者远程控制的方式启动、关闭机器人；通过图形界面查看地图及机器人所处位置；查看机器人对物品特征的学习情况；查看完成业务次数等等调试控制信息。

3. 数据库设计

数据库是软件开发的重要组成部分，其使用完整性约束保证数据的格式。数据库保存了整个系统的状态，并成为多个系统之间传递信息的纽带，在软件开发中有非常重要的地位。

本项目中使用的数据库，是由若干个数据实体的内部数据和实体之间的关系构成的，我们通过对类模型的分析和类之间关系的分析，得到如下的数据库总体设计。

3.1 实体概述

3.1.1 用户

机器人有多个使用者，如运维人员、超市工作人员和顾客，对不同的使用者，拥有不同的系统权限，机器人需要保存相关的用户配置来完成用户区分、用户识别、用户授权。

3.1.2 机器人运动参数

用户使用交互界面来启动、控制机器人系统的 UI，用户可以设置机器人相关数据，调节机器人的参数。数据库需要对这些用户定制数据进行存储。

3.1.3 地图

机器人在建图之后，需要将地图数据保存到数据库中，为后续的导航、路径规划功能提供支持。

3.1.4 关键词库

机器人使用语音识别模块进行部分操控，对于语音识别功能，机器人系统需要建立关键词库以达到更准确的语音识别和语义理解。

3.1.5 机械臂运动参数

机器人机械臂受机器人系统控制，实现对物体的抓取，工作人员需要预先测试、调节机械臂的上升高度、闭合宽度等等，实现对物体的有效抓取。

3.1.6 日志

对于异常情况，如机器人出现侧翻、目标点云识别失败等等情况，机器人需要生成对应的日志以供后续调试。在正常情况下，也应生成对应的事务日志，以便运维人员关注系统的运行情况。

3.2 ER 图

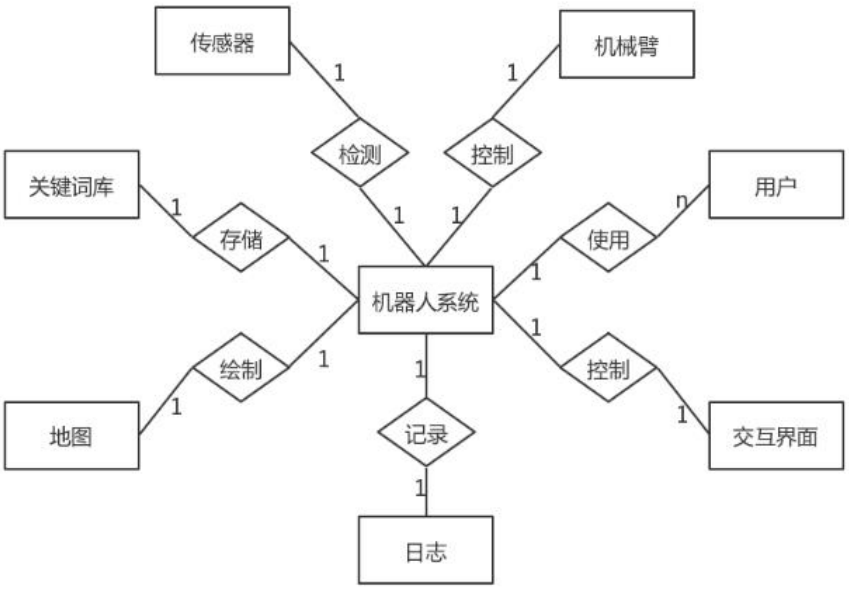

图 7 ER 图

4. 体系结构设计

4.1 总体结构

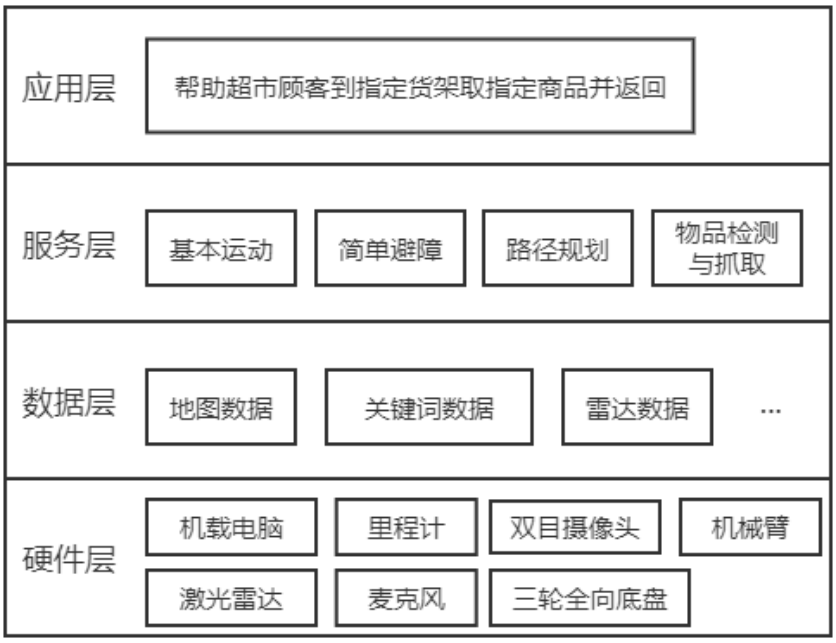

图 8 总体结构图

4.1.1 软件体系结构

...

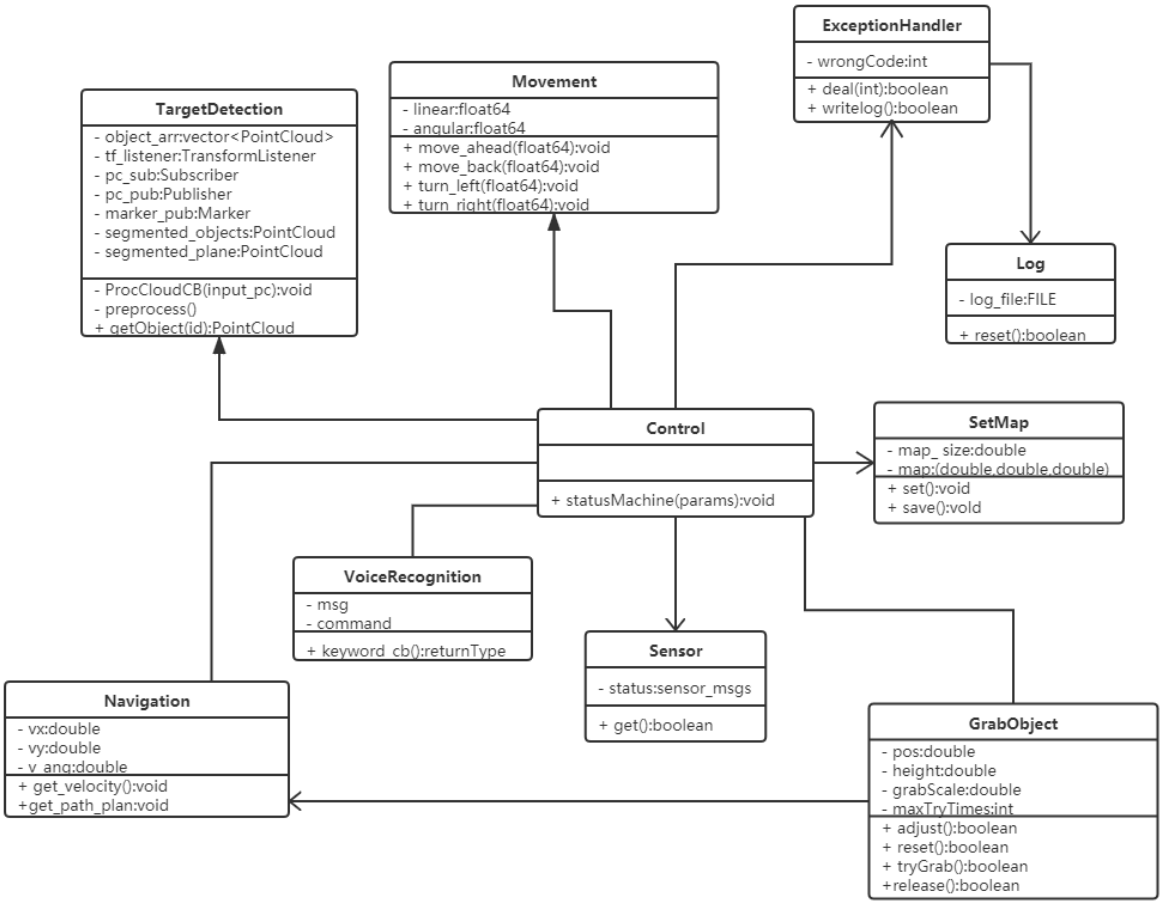

图 9 类图

系统类图如上所示，Control 类为控制类，也是整个系统的主类。

语音识别模块 VoiceRecognition 在系统运行过程中不断监听麦克风输入，如果捕捉到信息，识别后将传入 Control 类状态机，根据当前状态判断命令有效性之后，决定执行或丢弃。

传感器模块 Sensor 根据需求在 Control 类控制下捕捉系统周边环境信息或物品特征信息，之后交付 Control 类以便下一步处理。

初始时，超市工作人员需要对场景进行建图与存储，在这个过程中会实例化 Movement 类以满足机器人运动需求，实例化 SetMap 类，完成建图需求。

之后用户可以语音控制机器人进行取物。在取物过程中，TargetDetection 模块负责物品的定位和三维数据收集，之后将调用 GrabObject 类不断尝试抓取物体。在物体检测、抓取与释放之前都会调用 Navigation 以对准物体。

ExceptionHandler 在整个过程中由 Control 类调用，负责异常的处理，于此同时，它会调用 Log 类进行日志的记录。

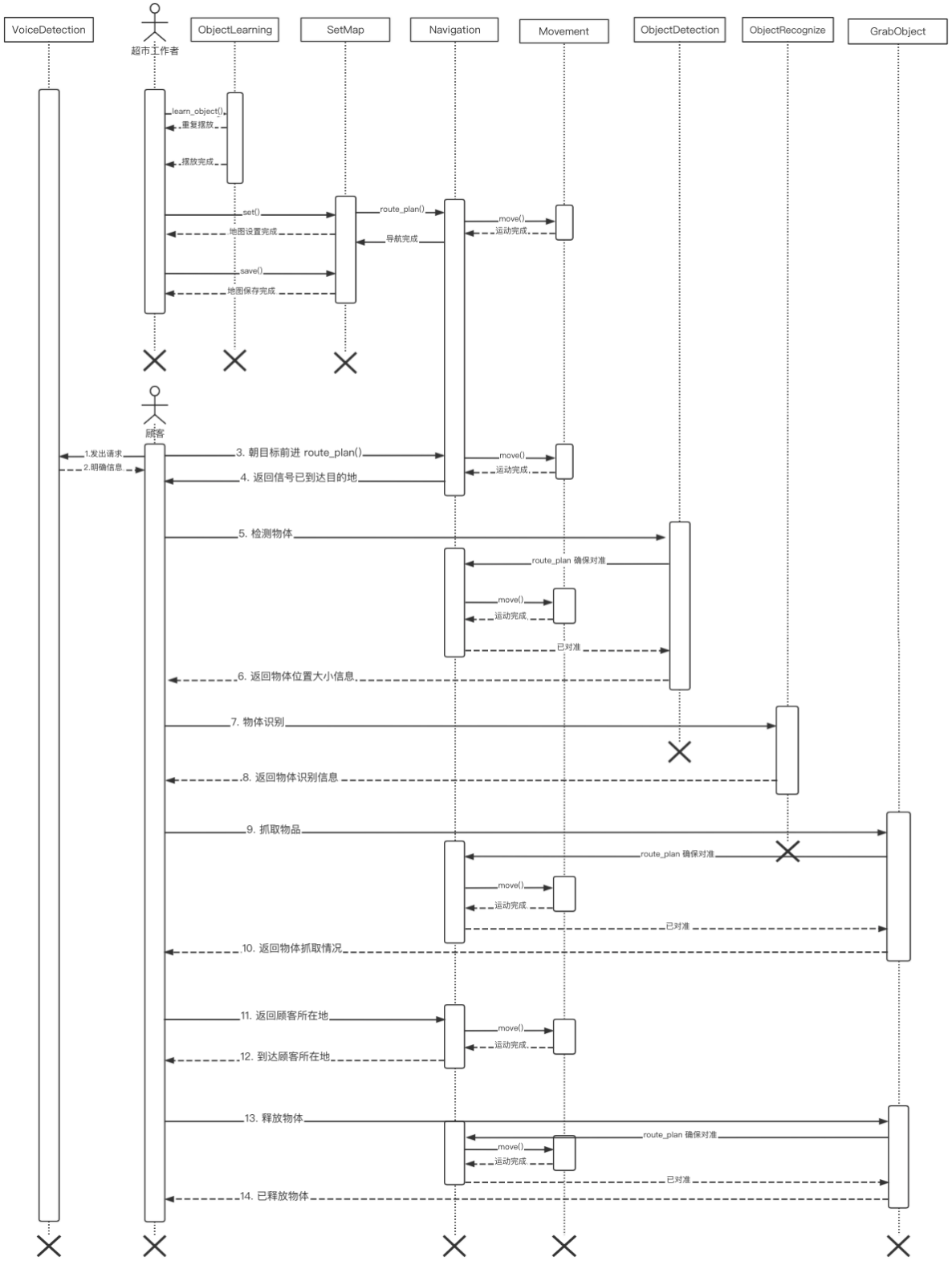
VoiceDetection
超市工作者
ObjectLearning
SetMap
Navigation
Movement
ObjectDetection
ObjectRecognize
GrabObject
learn_object()
重复摆放
摆放完成
set()
route_plan()
move()
运动完成
地图设置完成
导航完成
save()
地图保存完成
顾客
1.发出请求
2.明确信息
3. 朝目标前进 route_plan()
4. 返回信号已到达目的地
5. 检测物体
route_plan 确保对准
已对准
6. 返回物体位置大小信息
7. 物体识别
8. 返回物体识别信息
9. 抓取物品
10. 返回物体抓取情况
11. 返回顾客所在地
12. 到达顾客所在地
13. 释放物体
14. 已释放物体

图 10 时序图

4.1.2 硬件体系结构

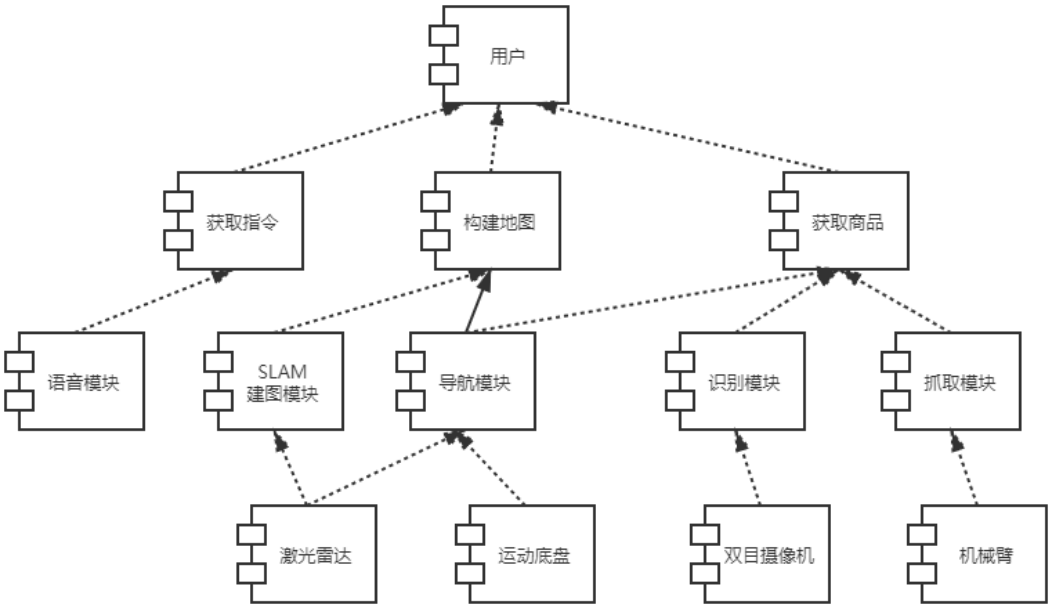

图 11 系统构件图

5. 接口设计

运动模块为三项全向底盘提供数据，抓取模块为机械臂提供数据。

对于 SLAM 建图模块，由激光雷达、里程计、双目传感器提供数据，该模块向记载电脑提供地图信息。

检测模块由双目传感器提供数据，之后向机载电脑传递物品数据。

语音识别模块由麦克风提供数据。

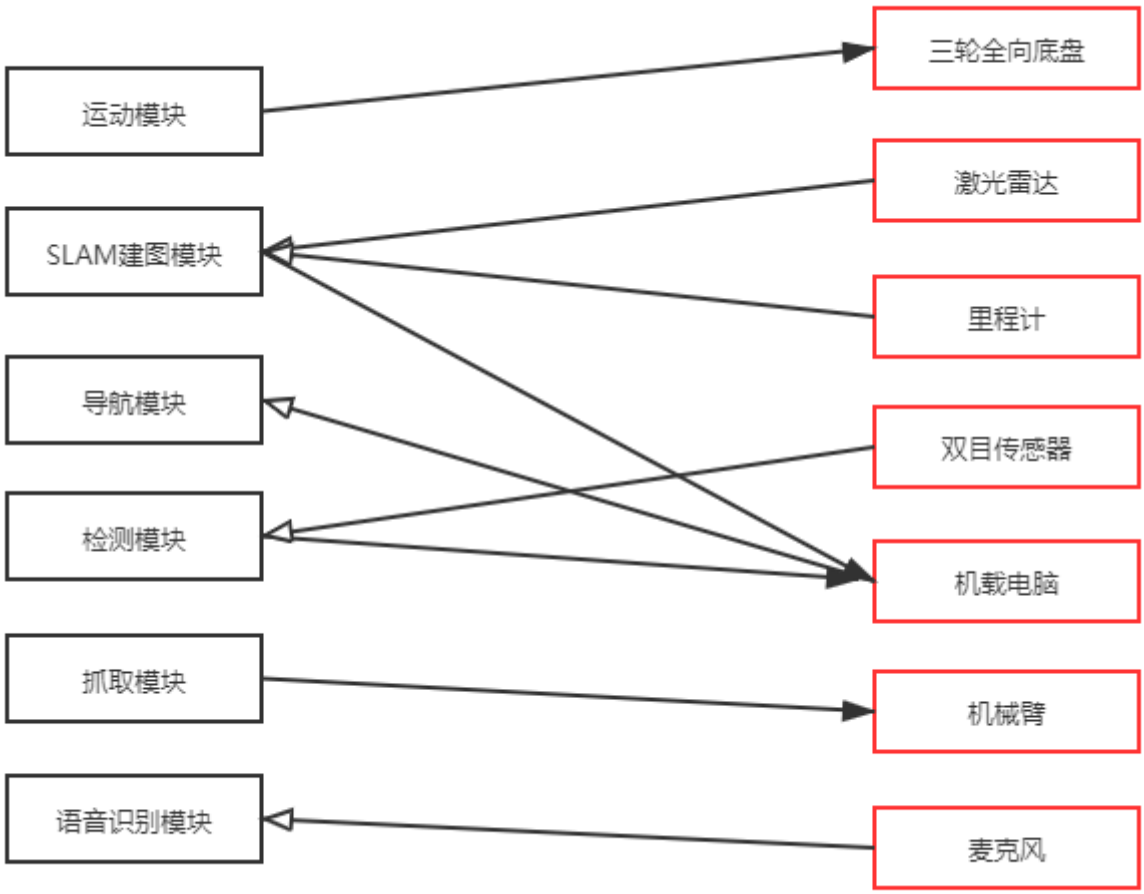

图 12 系统接口设计

6. 详细设计

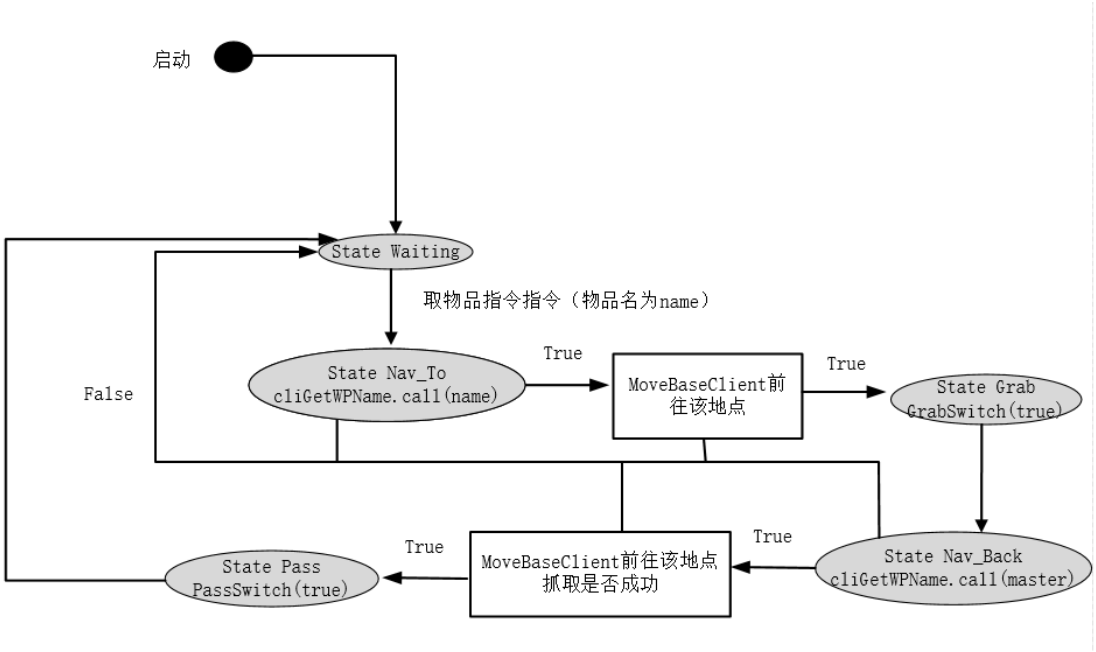

图 13 主程序状态转移图

如上图所示，主程序一共有 Waiting、Nav_to、Grab、Nav_Back、Pass 五个状态。按下机器人的启动键之后，机器人便开始工作。Ndelay 是留给用户准备的时间（默认为 10 秒钟）。准备时间过后，机器人便执行 AddNewWaypoint 方法，将当前机器人所在的位置作为新的航点加入，同时进入 Waiting 状态。

在 Waiting 状态中，机器人等待用户的取商品指令（商品名为 name）。机器人接收到用户的指令之后进入 Nav_to 状态，并且调用 cliGetWPName.call(name)方法，查询 name 商品对应的航点。若查找失败，机器人返回 Waiting 状态；否则机器人继续调用 MoveBaseClient 的相关方法前往该航点的位置，此过程出错则机器人同样返回到 Waiting 状态，成功则进入 Grab 状态。

机器人进入 Grab 状态后，立即调用 GrabSwitch(true)方法尝试抓取物品。然后机器人进入 Nav_Back 状态。

在 Nav_Back 状态中，机器人立即调用 cliGetWPName.call(master)方法。若调用失败则返回 Waiting 状态；调用成功则继续调用 MoveBaseClient 的相关方法返回用户所在位置。只有当相关所有方法都成功返回且之前的抓取操作成功，机器人才会进入 Pass 状态，否则进入 Waiting 状态。

在 Pass 状态中，机器人调用 PassSwitch()方法将物品交与用户。然后返回 Waiting 状态。

在所有的方法调用中，若方法调用出错，机器人会将相关的错误写入日志。若错误严重，机器人会崩溃关机。

下面是程序按功能划分的模块讲解。

6.1 运动模块

【功能】

运动模块主要负责：

控制机器人实现基本的平移运动和旋转运动。

机器人通过 /cmd_vel 这一 topic 和底层运动模块进行交互，该 topic 传递 geometry_msgs::Twist 类型消息，该类型消息中包含两个三维向量(Vector 3)，分别表示机器人三轴的角速度和线速度。

当我们需要控制机器人运动时，向 /cmd_vel 发布消息，交由机器人底层模块订阅 /cmd_vel 并处理。

【输入】

角速度和线速度三维向量。

【通信流程】

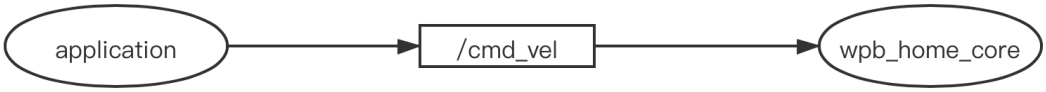

图 14 运动模块通信示意图

6.2 SLAM 建图模块

【功能】

机器人在超市管理员操控下绕超市一周，过程中即时通过 SLAM 模块完成对超市地图场景的建模。

【输入】

机器人调用 amcl 获取的自定位置信息和使用 tf 发布的有关坐标框架之间关系的信息；

传感器通过 ROS 发布的 sensor_msgs/LaserScan 或 sensor_msgs/PointCloud 信息；

使用 tf 和 nav_msgs/Odometry 发布的测距信息；

【输出】

建好的超市地图场景(map.pgm 和 map.yaml)

【设计】

建图部分主要是通过 waterplus_map_tools 自动化进行，逻辑关系大抵如下图所示。

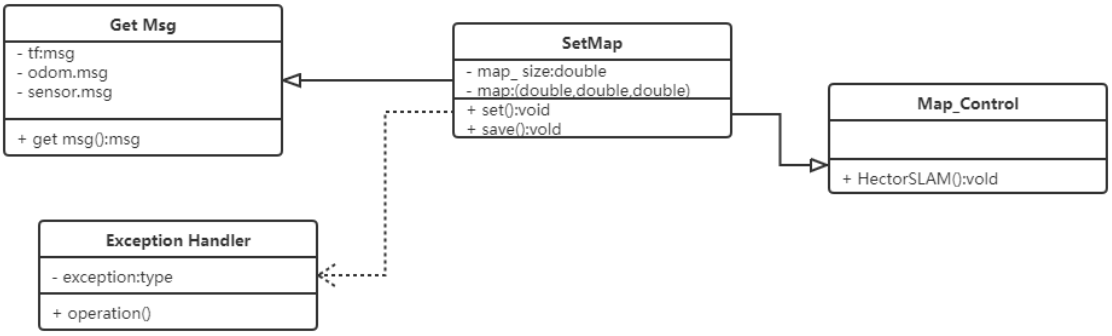

图 15 SLAM 建图模块类图

Get_Msg 类为软硬件交互类，通过 get_msg 方法实时获取 amcl 的自定位信息，测距信息及传感器信息等硬件反馈的信息。SetMap 为总控类，它接受 msg 信息，使用 set 方法完成对地图的建模，并通过 save 方法保存地图。Map_Contrl 类控制整个建图过程，主要是实现 HectorSLAM 算法建图。设置一个异常处理类，当建图过程中发生异常，调用异常处理类进行异常处理。

具体到我们的程序中，我们通过 keyboard_ctrl 节点实现键盘控制机器人运动，在机器人绕场一周过程中进行实时建图。

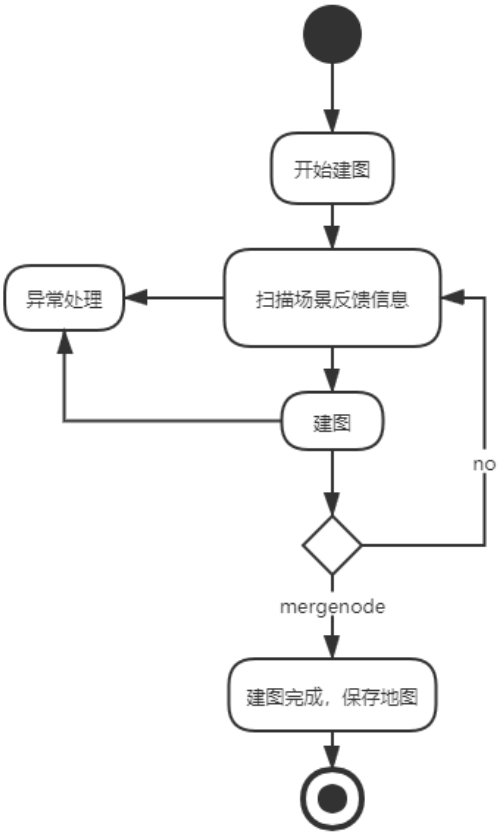

图 16 SLAM 建图模块流程图

6.3 导航模块

【功能】

根据感知模块获取的所需状态数据，实现机器人的路径行为规划。包含定位和导航的各个模块。

【输入】

通过调用 amcl 获取的位置信息和使用 tf 发布的有关坐标框架之间关系的信息；

传感器通过 ROS 发布的 sensor_msgs/LaserScan 或 sensor_msgs/PointCloud 信息；

使用 tf 和 nav_msgs / Odometry 发布的测距信息；

SLAM 建图后的 map 信息；

机器人当前需要到达的目的地信息；

【输出】

- 机器人的 x 速度，y 速度， θ 速度
- 整体路径规划得到的路径

【设计】

导航模块同样依赖 waterplus_map_tools 进行，大抵逻辑关系如下图。

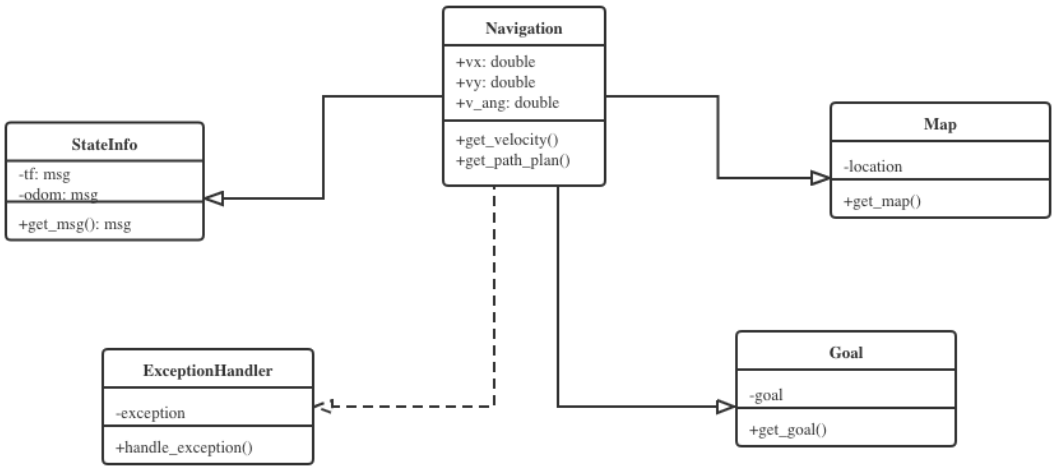

图 17 导航模块类图

具体到 rushbot_main.cpp 中执行流程如下：导航在 STATE_NAV_TO 状态下进行，首先将之前语音识别中得到的航点名称 strGoto 传递给 srvName，进而确定目标的 pose 信息，然后新建一个 MoveBaseGoal 类型的 goal，并将之前确定的目标 pose 信息附加上时间戳交给 goal；新建一个 MoveBaseClient 类型的 ac，通过 ac.sendGoal(goal)将目标信息传递给真正执行导航的 waterplus_map_tools 部分代码，然后就只需要等待结果即可。

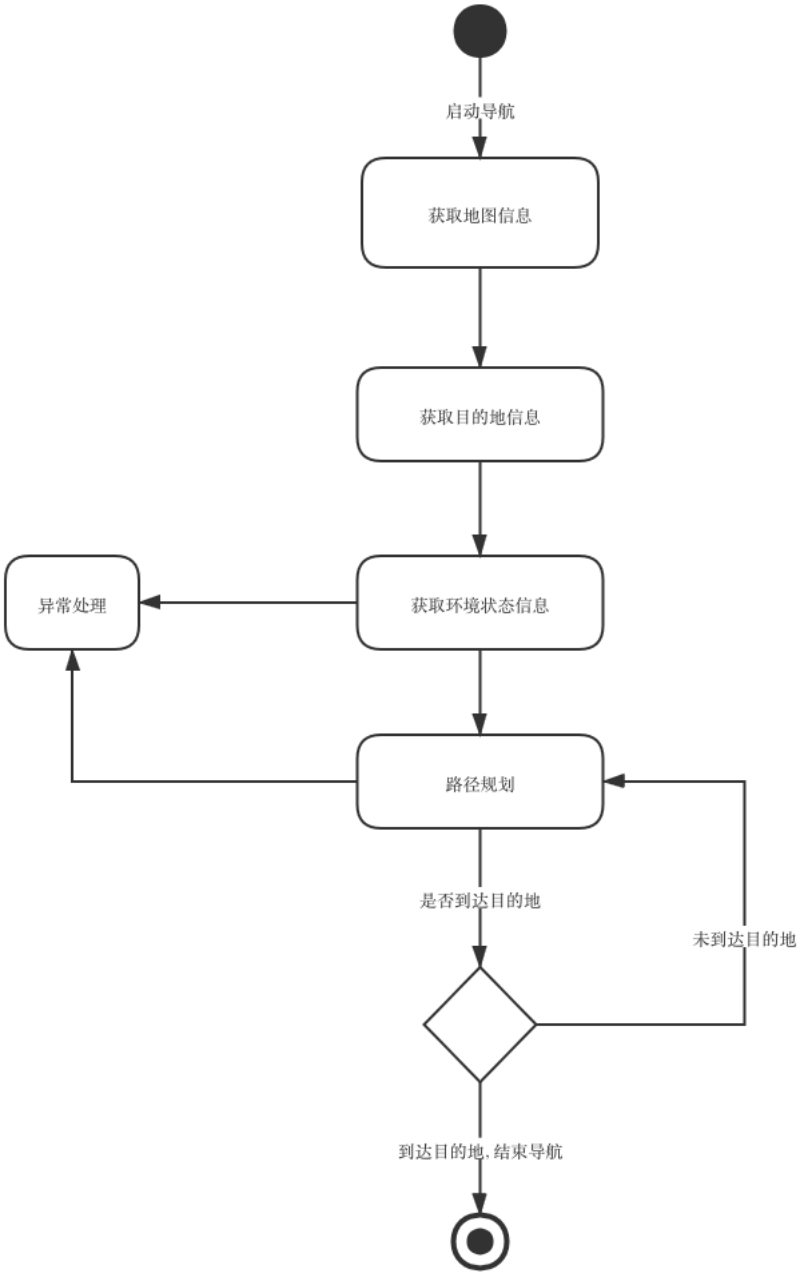

图 18 导航模块流程图

6.4 检测模块

【功能】

检测模块用于对物品位置进行检测，生成物品的点云信息，为之后物品抓取提供信息。该功能通过 ROS 机器人头部的 Kinect2 实现相关功能。

【设计】

1. 检测初始化

在开始检测之前，需要进行一些准备工作，完成相关数据的订阅以及为数据的传递发布相关主题。

为后续点云三维坐标转换，需要创建 tf_listener。

为标注物品的空间位置，发布名为 “obj_marker” 主题。为显示平面点云，向外发布 “segmented_plane” 主题。为显示检测出的物品的点云集合，发布名为 “segmented_objects” 主题。

之后，在函数中启动检测，通过相关回调函数处理点云数据。

2. 检测数据预处理

对于每一帧生成的点云信息，首先通过之前定义的 tf_listener 完成点云坐标到机器人坐标系的转换。之后将 ROS 格式数据转换为 PCL 格式数据。

3. 检测数据提取

在经过预处理之后，数据可以使用 PCL 相关算法进行分析。

首先，使用 PCL 的分割对象将原点云中的水平平面检测出来，将其平面标识进行存储。

接下来，在所有的水平平面中，寻找符合实际情况（实际平面高度）的平面。

在挑选出的平面上空，提取符合物品高度的相关点云数据，对其进行分析。

4. 检测数据分析

对于提取出来的点云数据，使用 KD-Tree 进行空间分割和近邻的查找，将整个点云数据分离为多个点云团簇。对于每一个点云团簇，认为其是一个物品。

5. 检测结果发布

对于每一个系统认定的物品，通过检测初始化阶段定义的 obj_marker 向外发布物品位置信息。

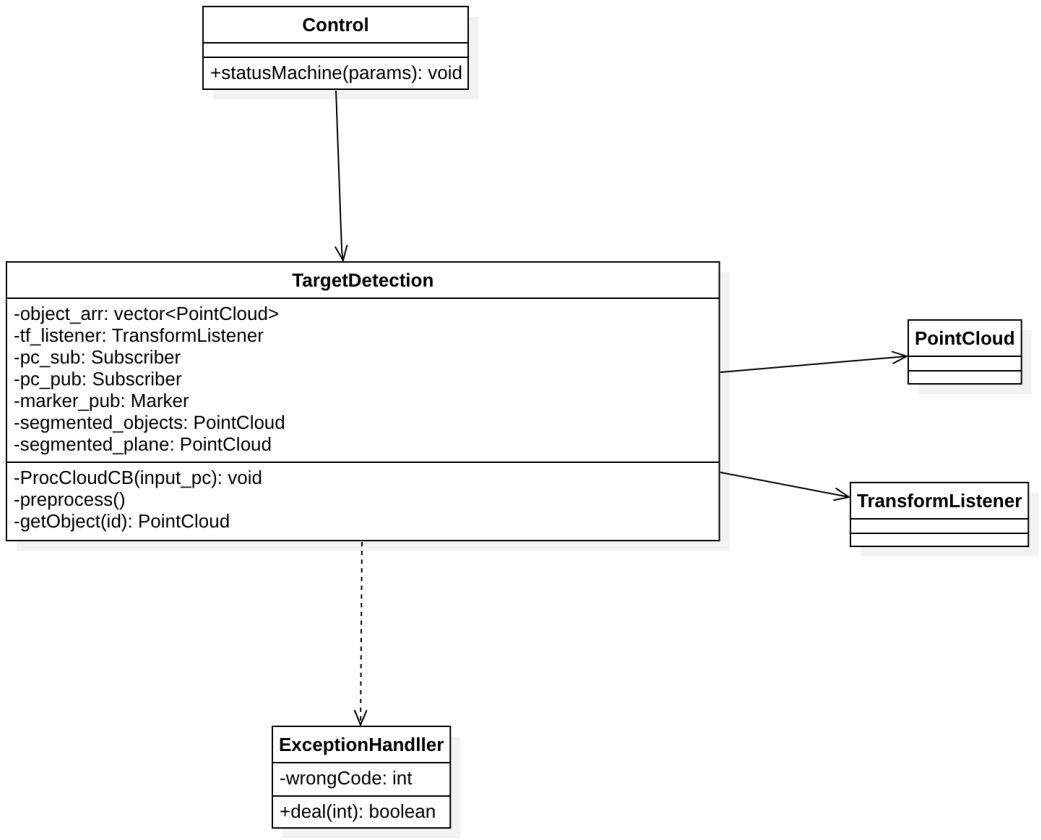

图 19 检测模块类图

检测模块的主类是 TargetDetection 类，控制类通过 Target Detection 类实现物体的检测和位置信息的获取。主类的 preprocess 方法初始化 tf_listener，pc_sub，pc_pub，marker_pub，segmented_objects 和 segmented_plane。主类通过 ProcCloudCB 回调函数处理生成的每一帧点云信息，在函数内找到符合要求的平面。将平面上方特定距离内部的点云使用 KD-Tree 进行点云切割，将分割出的每一个点云作为一个物体存储在 object_arr 中。

一个主类中，有一个 TransformListerner 实例，有多个 PointCloud 的实例。

在运行过程中，ExceptionHandler 对所有出错情况进行异常处理。

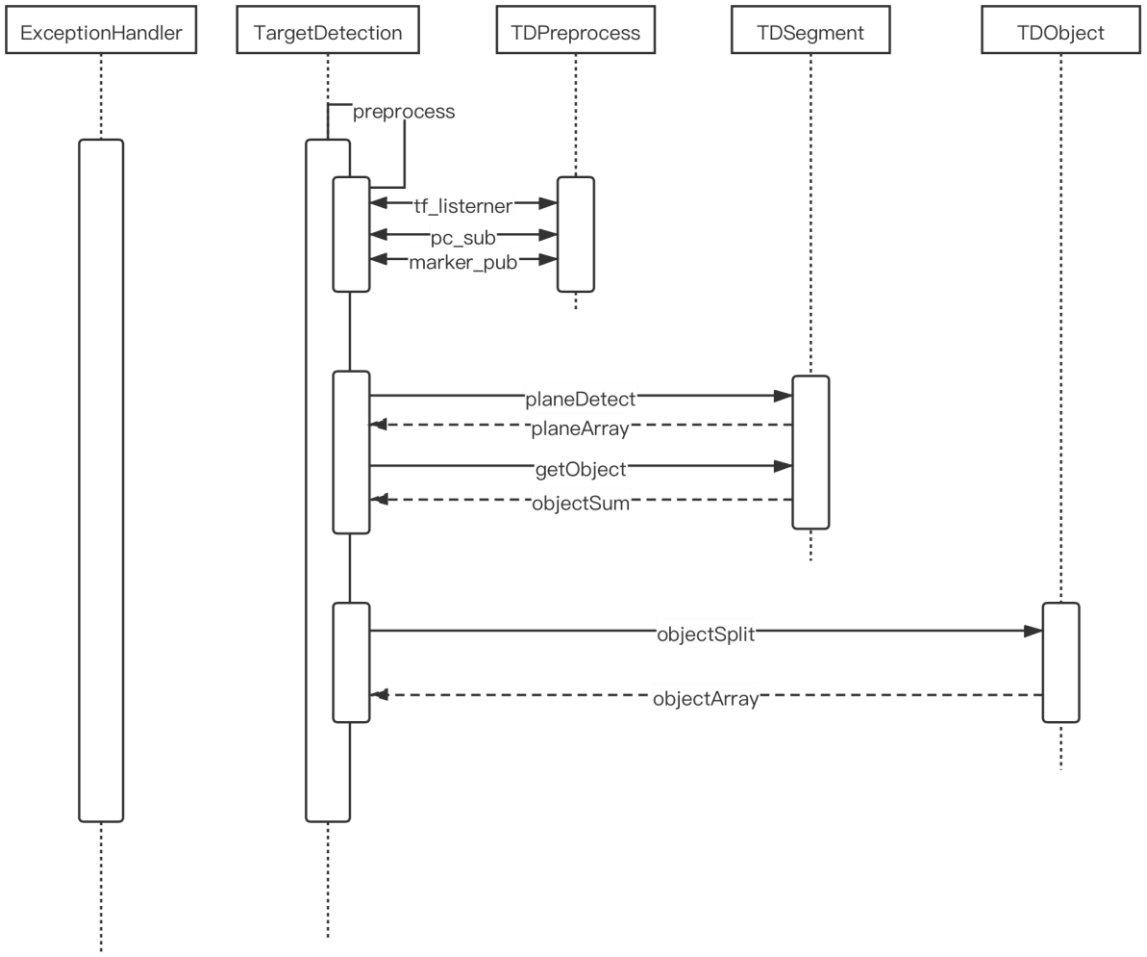

图 20 检测模块时序图

【通信流程】

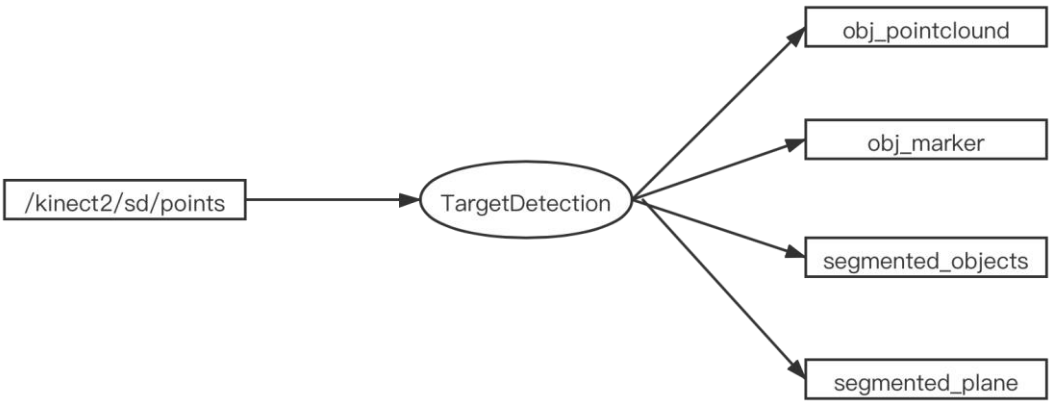

图 21 检测模块通信示意图

6.5 抓取模块

抓取模块实现的是有关物品抓取的功能。

在抓取物品之前，需要给机器人指定目标抓取物品。之后机器人需要进行物体检测并寻找到该物品，然后获悉物品的形状、大小和空间位置等相关信息。这里需要进行预先的适配操作，详细操作流程可参考机器人开发手册。

物品抓取功能的实现流程如下所述：

1. 抓取之前的准备工作

得到抓取目标信息之后，机器人需要根据信息调整自身的状态，靠近抓取平面。

2. 抓取物品

机器人左右平移对准物品之后，前进靠近物品，手爪逐渐合并尝试抓取物品，然后后退返回。如果抓取失败，则在最大尝试抓取次数内重复尝试抓取该物品。在此过程中，物品的体积大小和空间位置高低等因素会影响机器人能否抓取成功。

3. 抓取结束

机器人带着物品后退，完成物品抓取。

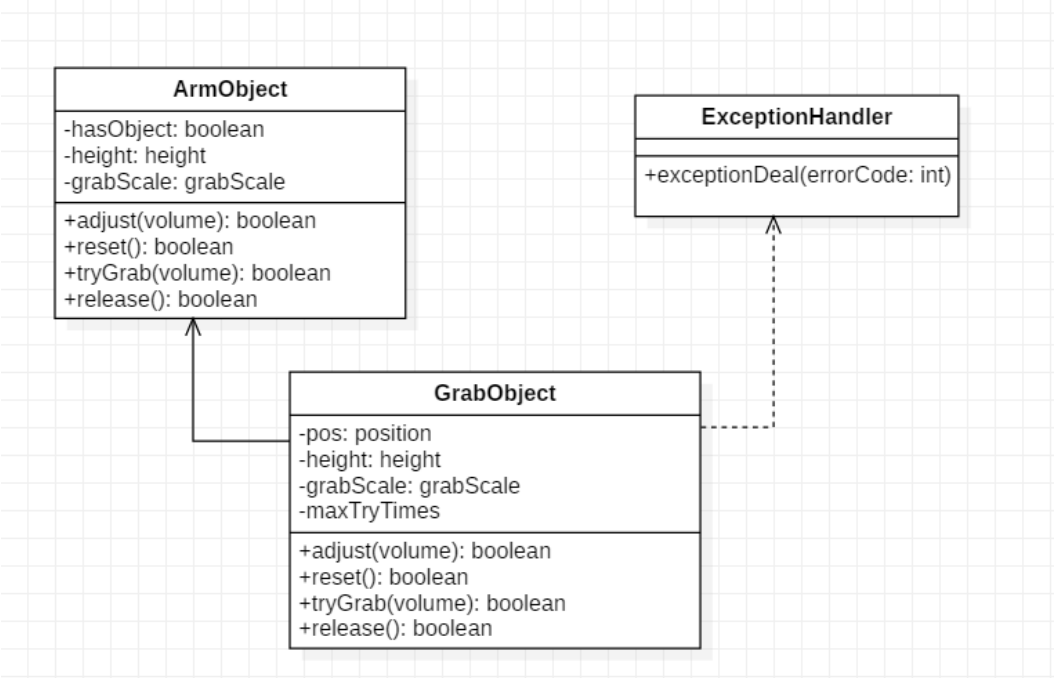

图 24 抓取模块类图

抓取模块由 GrabObject、ArmObject 和 ExceptionHandler 三个类组成。其中 GrabObject 为主类，调用 ArmObject 中的同名方法来完成具体的动作。adjust 方法来完成调整机械臂的高度等抓取参数的功能，tryGrab 方法在 maxTryTimes 次数内尝试对物品进行抓取，release 方法用于放开抓取到的物品，reset 方法用于回到抓取前的初始状态。ExceptionHandler 类用于处理可能出现的异常情况。

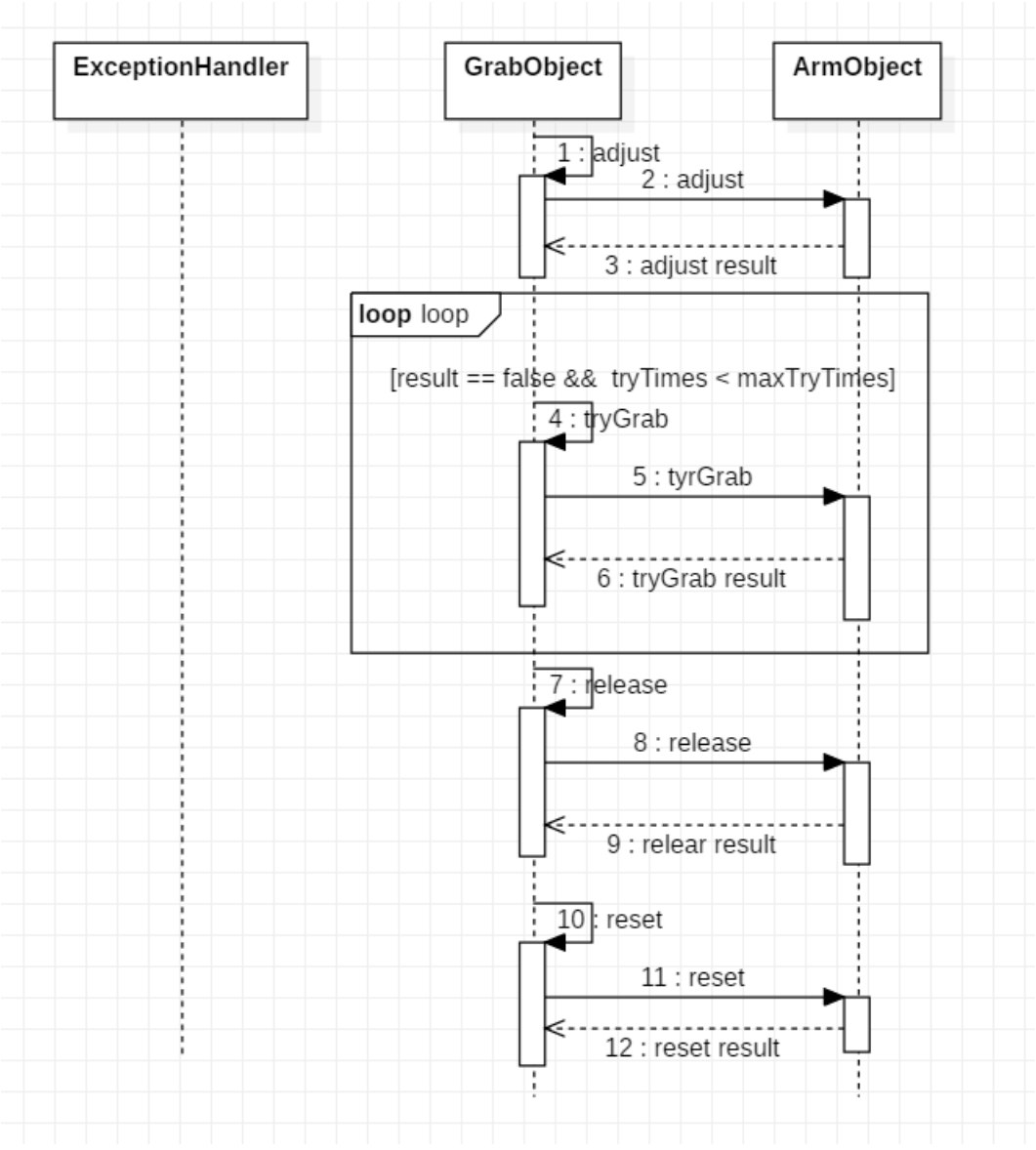

图 25 抓取模块时序图

【通信流程】

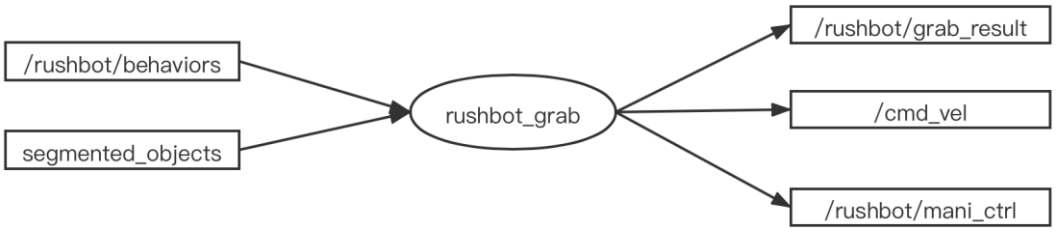

图 26 抓取模块通信示意图

6.6 语音识别模块

【功能】

语音识别部分主要由两部分代码构成：

第一部分是科大讯飞公司的语音识别引擎 xfyun_waterplus，这一部分主要完成语法文件与关键词库的建立与维护、用户语音的录制与识别，并将最终识别到的指令以字符串的形式传递给我们的 rushbot_main.cpp 文件。

第二部分即是我们在 rushbot_main.cpp 文件中对于语音识别结果的处理。首先：FindKeyword 函数可以从识别字符串中提取出我们所需要的关键字，进而在 KeywordCB 函数中基于机器人所处的不同状态与识别到的关键字进行进一步处理，具体处理流程见图 27。

【输入】

模块输入端通过用户的语音指令直接完成数据获取。

【输出】

根据识别到的命令执行相应的操作。

【设计】

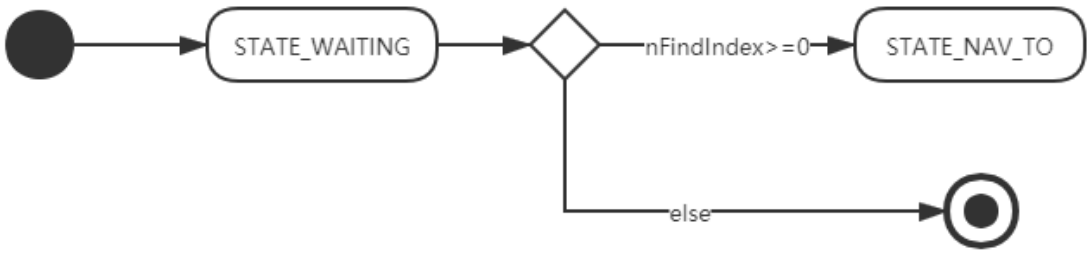

图 27 KeywordCB 函数处理流程

7. 运行与开发环境

7.1 硬件环境

7.1.1 机器人的结构组成

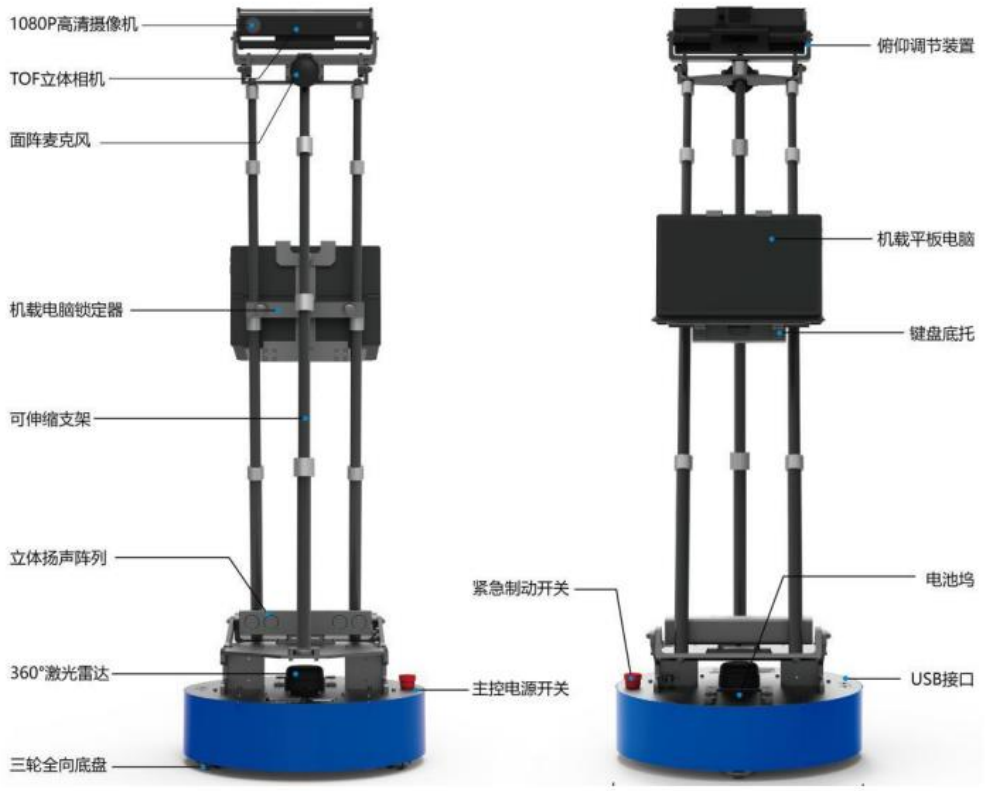

图 30 启智机器人结构组成图

7.1.2 主要硬件支持

名称	数量	参数
开发环境	1	ROS
主控器	1	Intel I3 处理器、4G 内存、128GSSD、触摸屏、键盘
激光雷达	1	360° 无死角、最大距离 8 米
视觉传感器	1	Kinect 2
伺服电机模块	3	20W 伺服电机、内置驱动
轮子	3	3 个全向轮
电池	1	24V3.5AH 锂离子动力电池

图 31 启智机器人主要硬件支持

7.1.3 工作环境

启智机器人重量约为 30kg（包含抓取组件），承载能力 10kg。启智机器人是室内机器人，在此环境之外运行可能会损坏机器人。工作平面需要能够承载不小于 40kg 的重量。如果表面太软，则机器人可能卡住，运动受阻。建议使用商用地毯、瓷砖等材质。启智机器人原则上在水平平面上工作，坡道坡度不大于 15 度，坡道倾斜程度过大可能导致倾覆。此外，机器人不具备防水的功能，需要保证运行环境的干燥。机器人设计工作温度为 15℃-35℃之间，使用中需要远离明火和其它热源。

7.2 软件环境

7.2.1 Ubuntu16.04

Ubuntu 是一个以桌面应用为主的开源 GNU/Linux 操作系统，Ubuntu 是基于 DebianGNU/Linux，支持 x86、amd64（即 x64）和 ppc 架构，由全球化的专业开发团队（Canonical Ltd）打造。

本项目的开发和运行，都是基于 Ubuntu16.04 操作系统环境下开展的。

7.2.2 ROS

ROS 是一个适用于机器人的开源的元操作系统。它提供了操作系统应有的服务，包括硬件抽象，底层设备控制，常用函数的实现，进程间消息传递，以及包管理。它也提供用于获取、编译、编写、和跨计算机运行代码所需的工具和库函数。

为了与 Ubuntu16.04 相互配合,我们在项目中使用的是 Kinetic 版本的 ROS。

Roboware Studio

我们的 IDE 采用的是 Roboware Studio。这是济南汤尼机器人科技有限公司基于 Visual StudioCode 开发的 ROS 专用 IDE。该软件有中文版本，还提供全中文的使用文档，十分适合中国的机器人开发者。

7.2.3 基础软件包

表 2 基础软件包

Package 名称	内容
wpb_home_bringup	启智 ROS 机器人的基础功能
wpb_home_behaviors	启智 ROS 机器人的行为服务
wpb_home_tutorials	启智 ROS 机器人的应用例程
wpbh_local_planner	启智 ROS 机器人的导航局部规划器

启智机器人出厂标配的电脑里已经安装好 ROS、IDE 和启智机器人的源码包，可以直接使用。同时该软件包会在开源网站 Github 上持续进行维护更新。

7.2.4 扩展软件包

启智机器人除了基础功能的软件包以外，还提供了一个扩展软件包，该软件包里包含了大量复合任务的实现例程。因为该软件包需要依赖 xfyun_waterplus 和 waterplus_map_tools 等第三方软件包，所以和基础软件包相互独立，以方便持续维护更新。

8. 需求可追踪性说明

8.1 功能性需求可追踪性说明

8.1.1 启动和关闭机器人

本系统硬件部分包括机载电脑和机器人主体部分，由具有一定先验知识的超市工作人员将两部分通过 usb 接口正确连接，在保证机载电脑正常开机、机器人放置得当等前置条件的情况下，打开机器人硬件总开关并双击系统图标，即可开启系统。

系统开启后，主控制类 Control 状态机启动，之后状态机进入等待状态。

使用结束后，用户通过点击机载电脑系统界面的关闭按钮就可关闭系统，随即关闭机载电脑和机器人总开关，断开 usb 连接。

8.1.2 初始化地图场景建模

在成功启动机器人之后，超市管理人员可以启动键盘操控运动节点，该节点一直在监听用户键盘输入，当其成功捕获到有效的指令之后，就产生相应的运动。

于此同时，管理人员启动地图建模模块 SetMap。SetMap 中的 set 方法将在机器人运动的过程中对地图进行 SLAM 建图。

建图工作完成之后，用户可调用 SetMap 的 save 方法保存建好的地图文件并结束进程。

8.1.3 货架地点记忆

在 8.1.2 的初始化建图过程中，用户可以通过“记录+商品种类”命令来使系统记录当前位置和该商品种类，位置和商品种类是相互对应的。

8.1.4 指定商品获取

系统完成上述工作之后，便满足了物品抓取的前置条件。此时，当 voiceRecognition 类捕获到用户抓取指令中的关键词“目标物”之后便传给

Control 相应的目标抓取指令。之后系统进入移动状态，调度 Navigation 类和 Movement 类前往目的地。到达目的地之后，系统进入目标抓取状态，调度目标检测类 TargetDetection 获取物品位置和三维点云，之后调度 GrabObject 类完成目标物的抓取任务，在此过程中 CameraCollect 类和 ArmObject 类会参与协助。抓取成功后，系统进入移动状态，调度 Navigation 类和 Movement 类回到出发地，进入移交物品状态，调度 GrabObject 类中的 release 方法将所抓取的物品放下。系统进入等待状态。

8.1.5 异常处理

为了提高整个系统工作的稳定性，Control 类可以处理在系统实际运行过程中可能出现的异常情况，并且调用 ExceptionHandler 抛出并且处理不同的异常。

8.2 非功能需求可追踪性说明

8.2.1 系统可扩展性需求

项目使用分层体系结构，不同层次内高内聚，不同层次之间低耦合，因此，针对某个系统层次的扩展就显得十分方便。

8.2.2 系统易用性需求

为了提高系统的易用性，我们在开发设计时尽量简化用户的操作，并且加入语音识别控制来改善用户的使用体验。

8.2.3 可靠性需求

我们在开发设计时对实际应用场景中的所有情况，包括合法输入和非法输入进行全面的考虑与处理，并通过代码的合理组织保证系统逻辑层面的稳固，以确保系统在实际运行时的可靠性。