**【启智ros机器人】**

**软件设计说明书**

**【SDD Team 205】**

**【v1.4】**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 肉弹葱鸡 | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 17373366 | 潘健行 | 完成文档第一部分和第七八部分 |
| 17373370 | 李祥宇 | 完成文档第二部分 |
| 17373397 | 冯昊 | 完成第五部分 |
| 17373245 | 刘名祖 | 完成第六部分 |
| 16005011 | 吕春辉 | 完成第四部分 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| V0.4 | 2020.4.17 | 潘健行 | 李祥宇 | 结构设计和第一部分 |
| V0.5 | 2020.4.18 | 李祥宇 | 吕春辉 | 第二部分与关键问题 |
| V0.6 | 2020.4.18 | 冯昊 | 刘名祖 | 第五部分 |
| V0.7 | 2020.4.18 | 刘名祖 | 潘健行 | 第六部分 |
| V0.8 | 2020.4.19 | 吕春辉 | 冯昊 | 第四部分 |
| V0.9 | 2020.4.19 | 潘健行 | 李祥宇 | 第七和第八部分 |
| V1.0 | 2019.4.20 | 吕春辉 | 冯昊 | 补充第四部分 |
| V1.1 | 2019.4.21 | 潘健行 | 李祥宇 | 补充了第三部分 |
| V1.2 | 2019.4.21 |  | 刘名祖 | 审查文档最终版本 |
| V1.3 | 2019.5.19 | 潘健行 | 李祥宇 | 根据使用gazebo模拟器的情况，对详细设计部分进行修改 |

# 1 范围

## 项目概述

ROS 的官方定义为：ROS是面向机器人的开源的元操作系统。它能够提供类似操作系统所提供的功能，如硬件抽象描述、 底层驱动程序管理、 共用功能的执行、 进程间消息传递、程序发行包管理等。

本项目最终想要完成一个简易机器人，选择C难度，实现可以进行目标检测和抓取的机器人。

可以通过电脑端的编译好的程序用终端输入指令或者通过前端直接点击相应的按钮实现机器人在指令下行进、自动进行建图、指定点自动避障巡航、为使用者拿回指定地点的物品等操作。

室内的环境并不是完全一成不变的，有时候会存在着家具位置的变化等等，因而我们的项目要实现对路径的动态规划。要能够做到自动避障寻路，物品抓取，和语音控制。为了机器人和其他人的安全，设计的行走及抓取的速度不应该太快。为了保证可靠性，起点到终点必须有一条有效路径，要控制机械臂抓取物品力度适中，不能把物品损坏，也不能让物品“脱手”。

该项目应用的场景主要是室内的智能家居机器人。在用户发出相应的指令之后，机器人根据指令来执行相应的任务。因为任务完成的地点可能不在机器人附近，所以机器人需要自己完成路径的规划前往指定的地点。因为任务的相对多样性，所以我们需要完成机器人对目标物体进行检测，以及机械臂的使用的相关工作，实现目标的抓取。最后，这一切指令应该使用前端或输入使用终端命令来完成。

## 文档概述

本文档为软件设计文档，包括需求概述、体系结构设计、接口设计、软件功能模块详细设计、运行与开发环境和需求可追踪性说明。该文档进一步完善了系统设计，说明简易机器人的详细功能设计以及性能指标，对后续的项目开发做出详细要求保证项目顺利完成。

## 术语和缩略词

简易机器人 Simple Robot 简称：SR

语音控制 Voice Control 简称：VC

室内导航 Indoor Navigation 简称：IN

物品抓取 Item Grab 简称：IG

激光传感器 Laser Sensor 简称：LS

即时定位建图 simultaneous localization and mapping 简称：SLAM

## 引用文档

1. 启智ROS版\_开发手册 版本20181109 2020年4月
2. SDD-软件设计说明 版本2020年 2020年4月

# 需求概述

## 概述

本项目为运行在室内的简易机器人，室内应满足运行环境，同时可设置多个障碍物与可被机器人抓取的目标物体。用户对机器人发出指令，机器人在接收指令后完成相应的功能，包括路径规划，避障，目标物体的识别与抓取。

同时，机器人的运行应保证效率且具有可靠性与安全性；开发代码应具有可扩展性；系统功能应保证完整性与健壮性。

## 业务需求

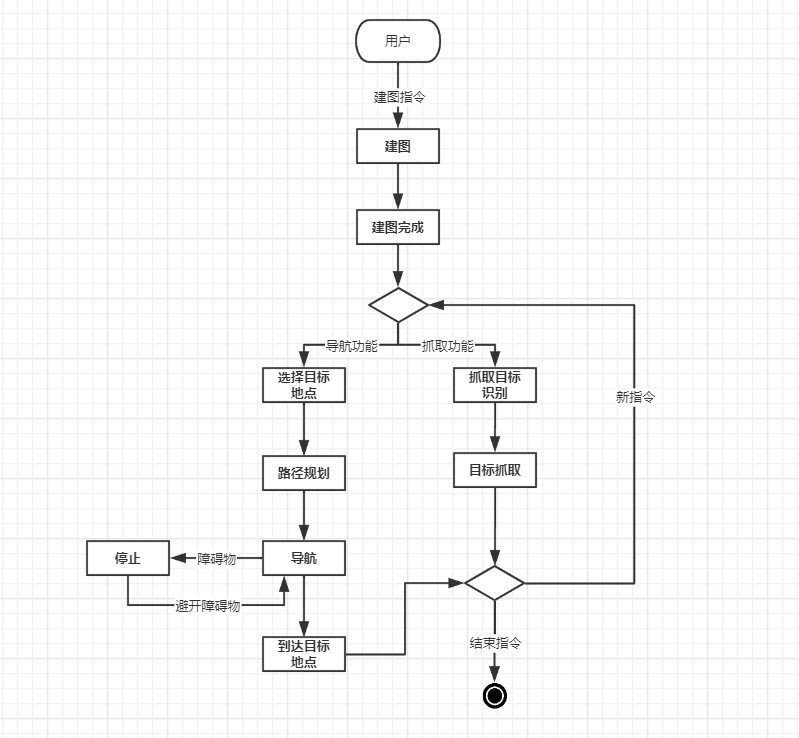


图 2.1 业务流程图

## 功能需求

### 启动机器人

**主要参与者：**用户

**目标：**启动机器人，使机器人处于可使用的状态；

**前置条件：**必须完整配置系统，且机器人电量充足；

**启动：**使用机器人之前；

**场景：**

1. 通过usb连接机器人与机载电脑
2. 按下机器人启动按钮
3. 机载电脑显示交互界面

**优先级：**高

**使用频率：**高

### 基本建图

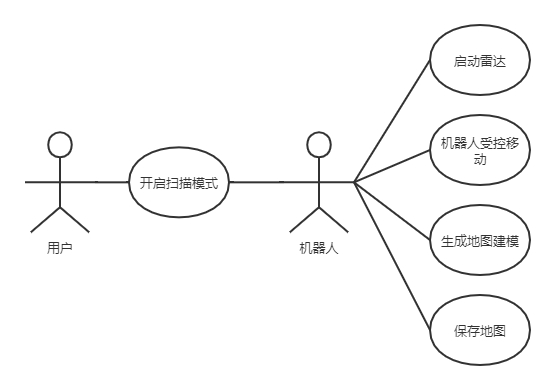


图 2.2 基本建图用例图

**主要参与者：**机器人，用户

**目标：**使机器人获取记录地图

**前置条件：**必须完整配置系统，保证激光雷达正常运行；

**启动：**第一次在新场地使用时；

**场景：**

1. 系统管理员将电脑调至扫描模式
2. 机器人启动雷达
3. 用户控制机器人在房间中移动
4. 机器人在移动过程中对房间进行建图
5. 机器人保存地图

**优先级：**高

**何时可用：**第一次在新场地使用时

**使用频率：**高

**次要参与者：**机器人所携带的传感器；

### 运动避障

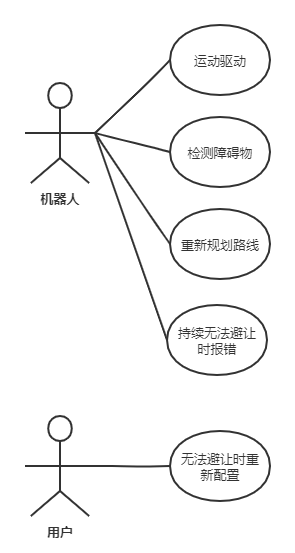


图2.3 运动避障用例图

**主要参与者：**机器人

**目标：**使机器人在运动中正确避障

**前置条件：**必须完整配置系统，保证激光雷达正常运行；

**启动：**所规划路径中出现障碍物

**场景：**

1. 机器人启动雷达
2. 机器人在所规划路线上移动
3. 在移动过程中发现所规划路径上出现障碍物
4. 重新规划路线
5. 在新规划路线上移动过

**异常情况：**

障碍物长时间存在且无法避让。处理：机器人在多次尝试重新规划路线， 发现无法避让后给出错误提示。

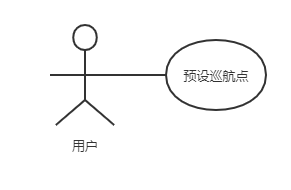
**优先级：**高

**何时可用：**第一次在新场地使用时

**使用频率：**高

**次要参与者：**机器人所携带的传感器；

### 定点巡航



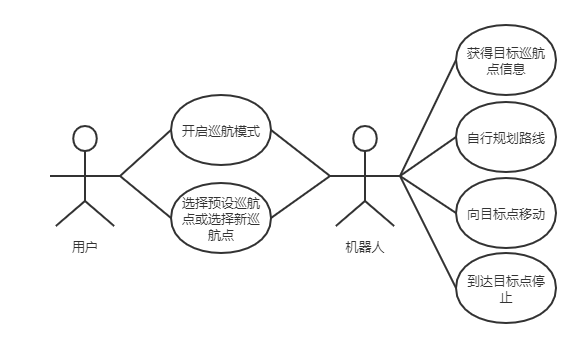


图 2.4 定点巡航用例图

**主要参与者：**机器人，用户；

**目标：**使机器人根据指令，避开障碍物顺利到达指定地点；

**前置条件：**必须完整配置系统，已经获得仓库地图；

**启动：**用户对机器人有操作需求时；

**场景：**

1. 用户选择预设目标点或指定新目标点；
2. 根据目标点名字搜索目标位置
3. 机器人根据目标点进行路径规划；
4. 在控制系统的管理下，机器人按照指定路线移动；
5. 机器人到达指定地点。

**异常状况：**

目标点设置错误：机器人无法正确解析目标点（该目标点不存在），或 目标点不可达。处理：机器人给出错误提示，不执行指令。

**优先级：**中

**何时可用：**用户对机器人有使用需求时

**使用频率：**中

**次要参与者：**机器人所携带的传感器；

### 目标检测及抓取

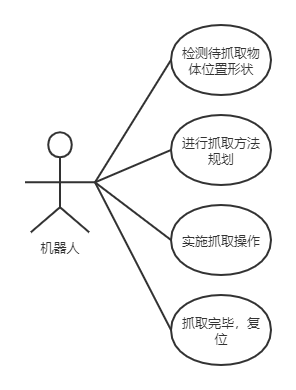


图 2.5 目标检测及抓取用例图

**主要参与者：**机器人；

**目标：**机器人能够自主完成指定物体的抓取；

**前置条件：**完整配置系统，机器人正常运行，已获得房间地图。

**启动：**机器人开始运动；

**场景：**

1. 机器人按规划路径移动到物体所在位置；
2. 机器人检测物体的位置细节以及物体形状；
3. 机器人进行物体的抓取规划；
4. 机器人对物体进行抓取；
5. 完成抓取命令；

**异常状况：**

抓取错误：机器人抓取目标物体失败。处理：用户重新下达指令，机器 人重新执行指令。

**优先级：**中

**何时可用：**用户对机器人有取物需求时

**使用频率：**中

**次要参与者：**机器人所携带的各类传感器；

## 非功能需求

### 性能指标

机器人应在一定的时间范围内完成对应的操作，包括路径规划、物体的检测与识别的算法运行时间；用户做出相应指令后的响应和启动时间。

### 质量属性

质量属性包括系统的安全性与可用性，运行速度不宜过快，与障碍物距离不宜过小，抓取所施加力应适中，不宜过大也不宜过小；系统的鲁棒性，在指令解析，路径规划，避障等过程中，如果发生异常情况应及时给出异常信息；代码的可扩展性和可维护性，对系统架构进行合理设计，类应低耦合高内聚，方法不宜过长，应只实现单一的功能。

# 数据库设计

在我们的机器人设计中会简单使用数据库。

设计方案概述如下

## 用户信息

|  |  |  |  |
| --- | --- | --- | --- |
| 用户名 | 密码 | 登录信息 | 用户权限 |

## 物品信息

|  |  |  |
| --- | --- | --- |
| 物品名 | 物品位置 | 是否被取走 |

## 关键词

|  |  |
| --- | --- |
| 关键词 | 关键词功能 |

## 日志

日志部分是简单地记录数据库中的修改日志，登录活动信息为了方便日后若有异常去查看。

# 体系结构设计

* 1. **总体结构**
     1. **软件体系结构**
        1. **软件总体结构**



图4.1 软件总体结构图

为实现简易机器人，将功能分为两大类：调度控制与用户交互。其中用户交互控制主要提供调度控制所需的数据（即参数）。

用户交互主要包含两大功能：为简易机器人指定目的位置；为简易机器人指定需要抓取的物体。包含两大关键模块，位置指定输入与指定物体输入。

调度控制主要分为两大调度运动调度与抓取调度。其中运动调度控制机器人的行进以及避障。行进功能包含三大实现模块：初始制图（即绘制房间的初始地图）；路径规划（即确定前往指定地点的路径）；运动（即驱动机器人前往指定位置）。避障功能在包含路径规划与运动的同时，还包含了障碍检测模块（即实时判断原规划路径上有无突然出现的障碍物）。

而抓取调度模块则主要负责物体定位与机械臂控制。物体定位包含两大基本模块：识别物体（即根据输入或默认，判断正确的抓取物体）；确定方位（即确定需要抓取的物体的具体位置）。机械臂控制包含控制机械臂抓取物体模块。

* + - 1. **软件体系结构类图**

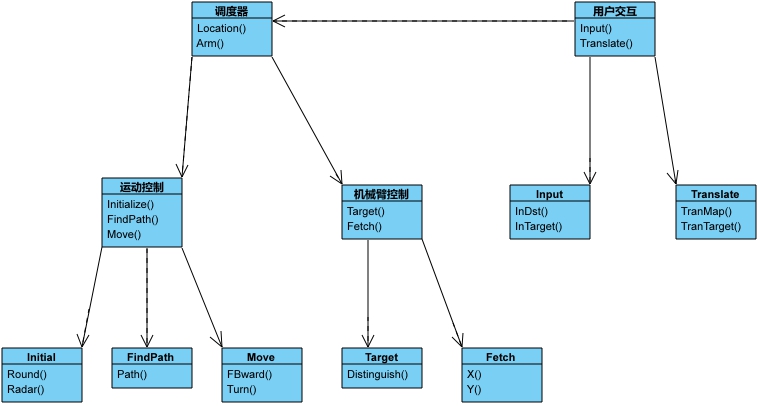
****

图4.2 软件整体类图

在软件总体结构的基础上给出总体类结构设计，将其上软件总体结构设计中的关键模块细化成相应的类。具体类中方法将在详细设计中指出。

* + - 1. **软件活动图**

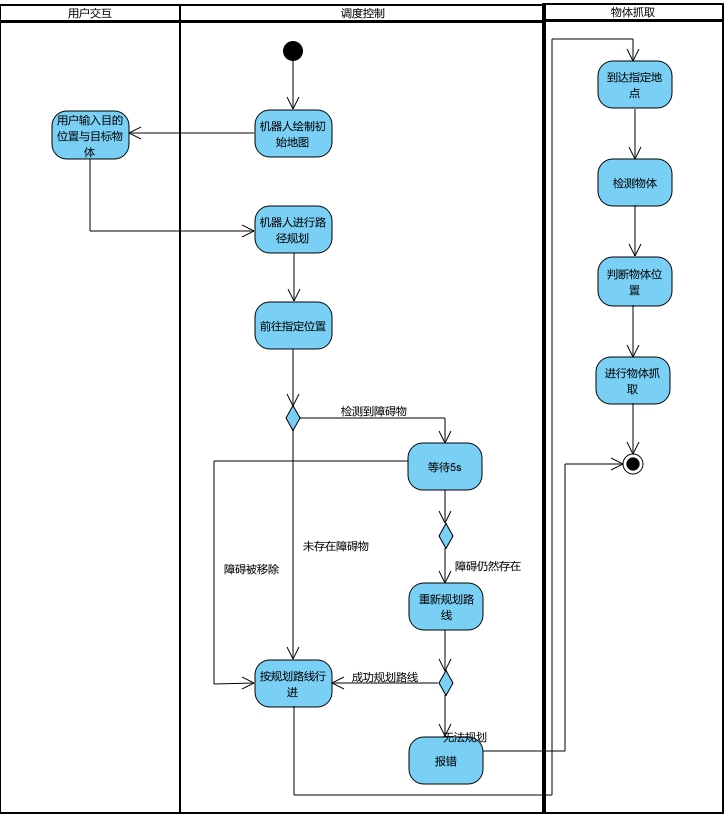
****

图4.3 软件活动图

依据整体设计给出关键模块为完成主要功能所进行的活动以及其之间的联系。具体过程为，初始化地图，然后读取用户输入并以此规划路线。路线规划完成后，机器人前往指定位置，期间实时监测障碍物。如遇障，等待、判断障碍是否被移除，移除则按照原路径行进。

如果顺利到达指定地点，则进一步根据用户输入进行物体识别，并抓取指定物体。任务结束。

* + 1. **硬件体系结构**

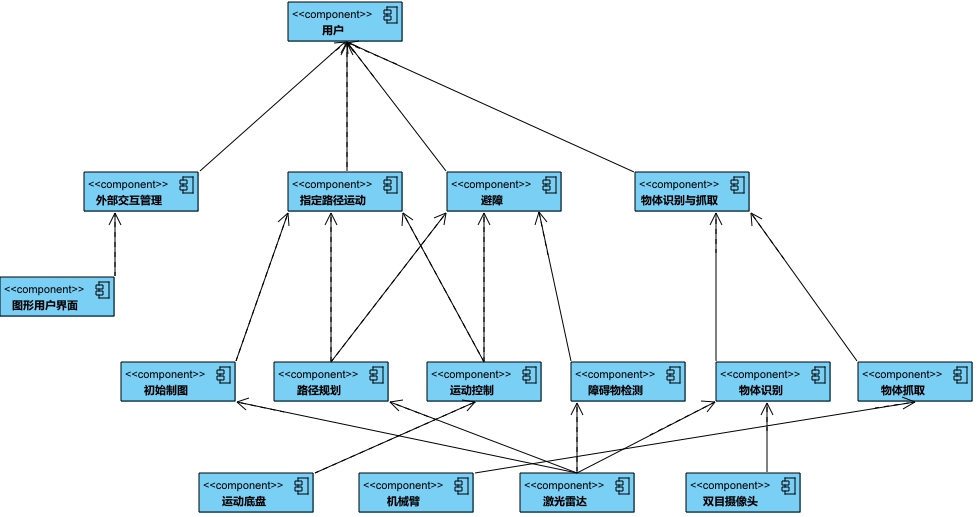
****

图4.4 体系构件图

构件结构给出了基本的功能以及完成该功能所使用的硬件。

* + 1. **技术体系结构**

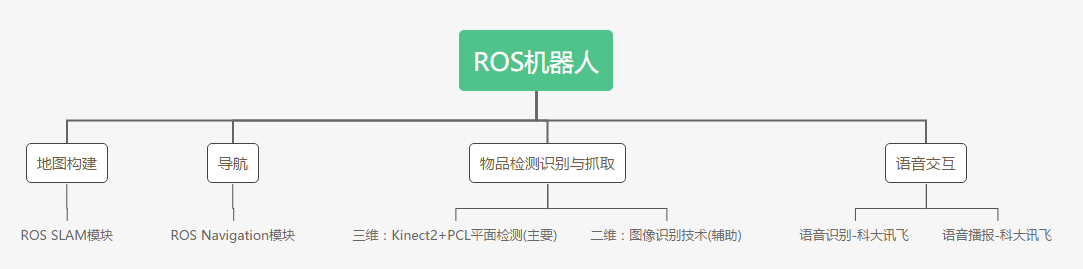


图 4.5 技术体系结构

本设计的实现是基于ROS的Publish-Subscribe通讯架构，重点使用其中的SLAM模块用于地图构建；使用Navigation模块用于导航；使用Kinect2模块和PCL的平面检测算法实现物品检测识别，并考虑引入图像识别技术作为物品检测识别的补充；使用科大讯飞的语音识别模块和语音播报模块用于语音交互。

* + 1. **部署和实施方案**

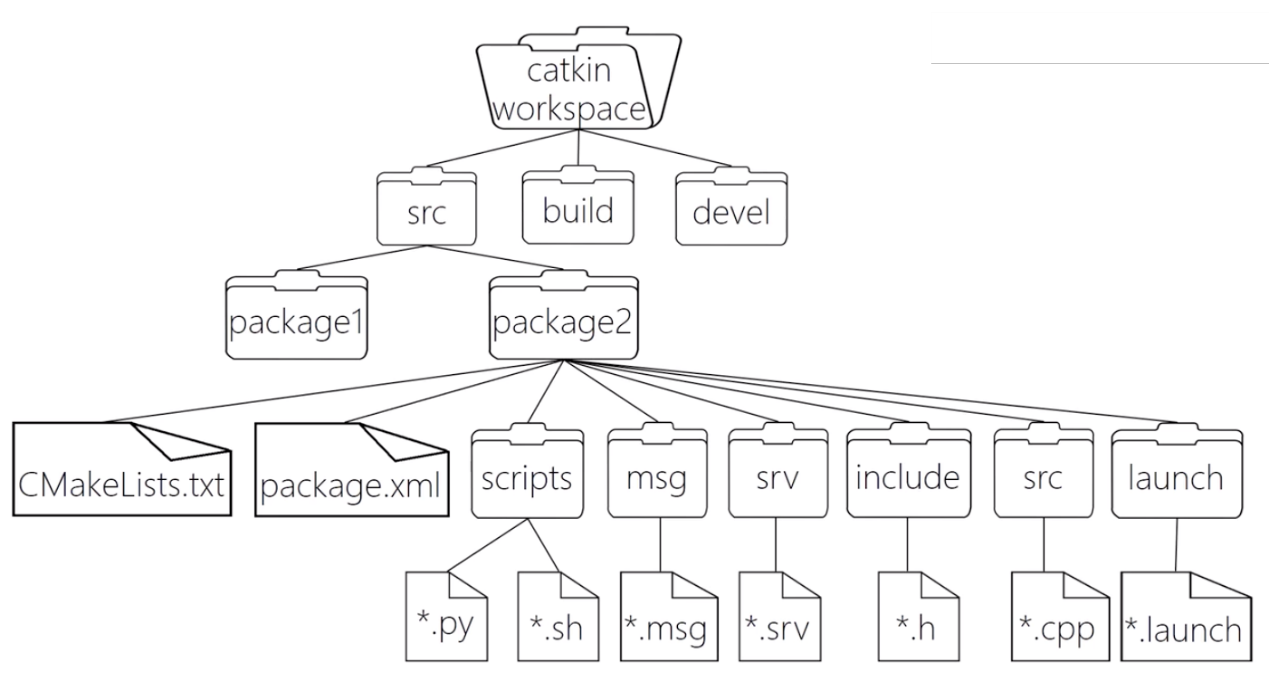


图4.6 系统部署图

系统主要是在ROS系统的基础上进行搭建，源代码主要存储在catkin工作空间的src目录下，分模块的编写每个独立的package，使用catkin编译，build目录下是cmake&catkin缓存和中间文件，devel中是目标文件。catkin是ROS定制的编译构建系统，是对CMake的扩展。package是ROS软件的基本组织形式，catkin的基本编译单元，一个package可以包含多个可执行文件（节点）。CMakeList.txt规定catkin的编译规则。package.xml定义了package的属性。\*.py,\*cpp是python,C++的源文件，\*.h是C++的头文件。\*.msg,\*.srv是相关的通信格式。\*.launch用于组织package下的多个可执行文件。各个进程之间使用topics或services进行通信，使用catkin编译整个项目。

## 关键问题及解决方案

### SLAM建图

建图是本机器人后续功能的基础之一，机器人的路径规划、导航以及目标物体的识别和抓取等功能都需要地图文件来作为基础。

我们使用SLAM的Hector SLAM算法，用手推动机器人进行移动，可以观察到机器人脚下则出现一片白色图案。这个图案，是由很多条线段叠加而成，这些线段是机器人本体中心地面投影和每一个激光雷达红色障碍点的连线，是测距激光飞行的轨迹，表示这条线段内部没有障碍物。将机器人推动绕场地一周后，可以将地图保存下来。

### 路径规划

本设计使用ROS的Navigation导航，其主要包含两个功能部分，一个是定位功能，一个是导航功能。其中定位功能使用amcl包，采用蒙特卡洛自适应定位算法，这是一种使用概率理论在已经地图中对机器人自定位置进行估计的方法；导航功能使用move\_base包，这个包将机器人导航需要用到的地图、坐标、路径和行为规划器连接到了一起。

需要注意的是，导航开始时我们需要修正机器人的默认位置，导航的过程中需要考虑机器人自身定位与实际位置之间的误差，必要时需要进行人为的监视并修正。路径规划由于误差的存在可能不能得到最优路径，但考虑对用户体验并不会造成影响同时不应对用户有很高的要求，因此不会着重考虑此误差。

### 物品检测识别和抓取

物品检测识别和抓取是指机器人在到达指定位置后，对指定位置的物品进行识别，然后调整机械臂，抓取物品，携带物品返回。

本项目的物品检测识别功能采用的主要是PCL的平面检测算法，启智 ROS 机器人头部 Kinect2 可以输出三维点云，PCL算法对三维点云进行分析处理。算法的主要思路是首先检测桌子水平面，然后遍历平面找到高度合适的标准平面。将标准平面上方一定距离的点云分离出来作为物品云集合，再使用Kd-Tree对云集和进行近邻搜索，然后分割平面上方的点云团簇，每个云团簇被认为是一个物品，并可以得到物品的三维坐标。但这样无法对物品进行分类，后续我们计划使用双目摄像头中的图像信息，使用机器学习预训练的模型来对物品进行分类。

物品的抓取需要通过物品的三维坐标和大小来控制机器人的机械臂和手爪的张开大小。但最终的效果目前不得而知，但考虑到该项目所应用的场景所抓取物品一般较为固定且重复，因此我们计划考虑配置抓取的相关参数，以达到更好的抓取效果。

# 接口设计

## 系统用户界面

系统用户界面，部署在PC端，主要用于用户对机器人进行控制操作，并获得机器人的反馈信息，实现人机交互。

实现形式上，系统用户界面采取图形用户界面设置，通过输入框、按钮、rviz界面的组合使用实现操作控制与信息采集，并返回机器人采集到的信息。

系统用户界面大致有以下四个页面

### 初始面

打开系统用户界面后的默认界面，界面询问用户是否需要建图，有如下按钮：

是（打开建图模式页面）；

否（打开主页面）；

退出（退出系统用户界面）。

### 主页面

打开系统用户界面后的默认界面，界面内有如下按钮：

巡航模式（打开巡航模式页面）；

抓取模式（打开抓取模式页面）；

退出（退出系统用户界面）。

### 建图模式页面

在主页面内点击建图模式，即切换到建图模式页面，页面内有以下输入框与按钮：

机器人巡航速度输入框（输入建图的巡航速度）；

保存建图（保存当前构建地图）；

退出（返回主界面）。

### 巡航模式页面

在主页面内点击导航模式，即切换到导航模式页面，页面内为rviz界面与以下按钮与输入框：

新建目的地（在所构建地图上设置目的地）；

新建目的地名字输入框（新建目的地的名字）；

保存目的地（保存所设置的目的地）；

若干已保存巡航点；（选择任一作为目的航点）

开始导航（根据输入的目的地，自行构造线路，开始导航）；

退出（返回主界面）。

### 抓取模式页面

在主页面内点击抓取模式，即切换到抓取模式页面，页面内有以下按钮：

抓取（机器人执行抓取操作，抓取物体）；

退出（返回主界面）。

## 系统软硬件外部接口

启智机器人配置了多种外设，我们主要使用的是控制手柄、激光雷达、姿态测量系统、立体相机。

### 控制手柄

将控制电脑安装到启智机器人上，连接相应的 USB 接口，将操控手柄连接到机器人底盘 面板的 USB 接口。

打开机器人底盘面板的 USB 电源开关。

打开机器人底盘面板上的红色急停开关，拨动手柄摇杆，控制机器人移动。

### 激光雷达

启智 ROS 机器人的底盘上安装了一枚红外激光雷达，该雷达的扫描角度为 360°，能够很高效的检测出周围的障碍物分布，并可以通过 SLAM 技术进行机器人的自身定位，为机器人的移动导航提供数据基础。

### 姿态测量系统

启智 ROS 机器人的底盘内置了一个三轴姿态测量系统，可以实时检测机体的朝向、翻滚及俯仰角度。为机器人的导航及行进提供重要数据，并让机器人在运动过程中发现倾倒风险，及时采取紧急措施。。

### 立体相机

启智 ROS 机器人的头部安装了最新一代的 RGB-D立体相机，可通过机械安装调节其视角，以便对准需要进行视觉识别的目标区域。立体相机可以输出 RGB 彩色视频流和 Depth 深度数据三维点云，借助 OpenCV 和 PCL 等开源图像库，可以对目标物进行准确识别和定位，以便进行后续的任务。

## 系统软硬件内部接口

由于我们采用了面向对象的设计思想，系统软硬件内部接口，主要就是各类之间的接口，以下，我将按照各类进行介绍。

### 地图模型

该模块主要实现的接口为mapGeneration(), mapCall(), mapSave()。mapGeneration建立地图模型的几口；mapCall则是为导航类等其他类提供的调用地图模型的接口；mapSave则很好理解，就是保存地图的接口。

### 导航

该模块主要实现的接口为naviPlan(res, dst, type)接口，naviPlan按照出发地、目的地以及用户需要的导航类型，返回规划路径，为用户实现导航功能。

### 目标识别

该模块主要实现的接口为learn(), match()接口，learn接口实现的是按照输入场景图片，训练匹配模型；match接口实现的则是根据匹配模型在当前场景进行匹配，匹配成功时返回目标坐标。

### 目标抓取

该模块主要实现的接口为fetch()接口，fetch接口实现的是，根据match返回的目标坐标，确定目标抓取所需要的参数值（高度信息，宽度信息等），并实现抓取。

### 主类

该模块主要实现的接口为系统用户界面需要调用的接口，主类调用以上类的接口，并将它们实现成GUI上对应的功能，例如通过move接口实现机器人巡航速度输入，这里不一一赘述。

# 详细设计

## 各模块总体介绍

总共分为以下四个模块：用户输入模块、建图模块、运动与路径规划模块、抓取模块。其中用户输入模块用于用户与机器人的交互，在机器人启动后负责对特征点进行输入标志点信息；建图模块进行地图初始化，建立并获取地图；运动与路径规划模块负责在地图中标定点的信息；抓取模块实现对目标地点的物品的检测和抓取。

## 控制模块

控制模块是系统的总控，用来接收用户命令与输入，并对机器人各个模块发送命令。

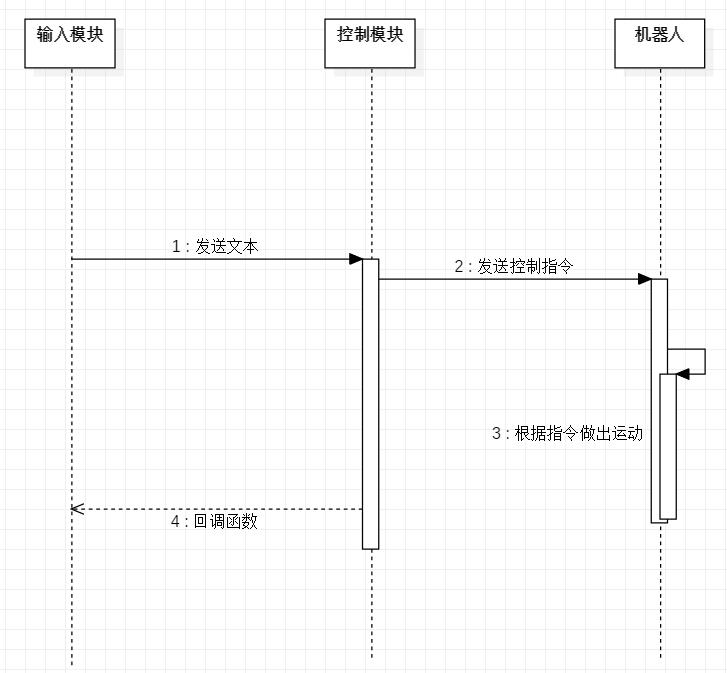


图 6.1 控制模块顺序图

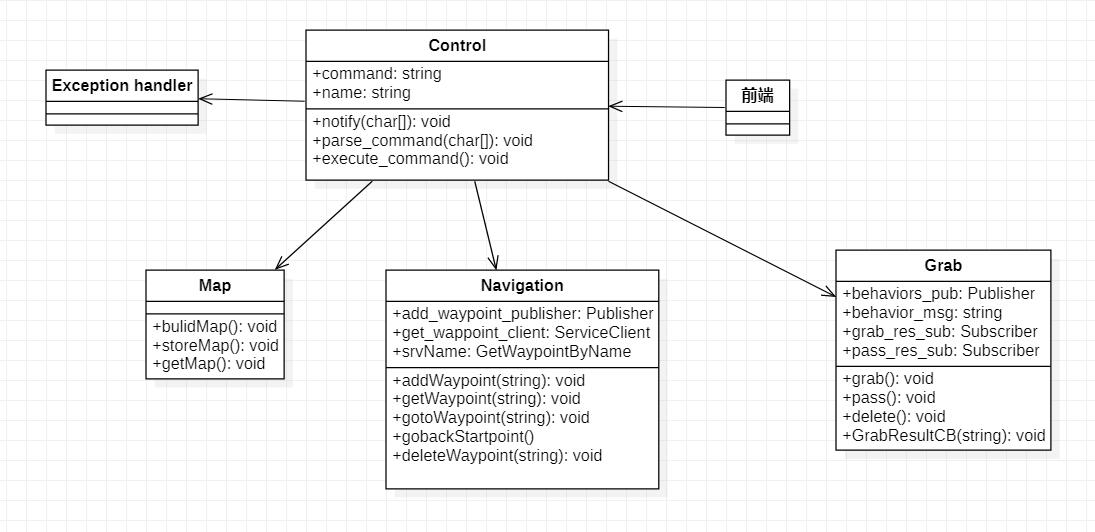


图 6.2 控制模块类图

控制模块用来负责用户和机器人之间的交互，用户通过前端发出各类命令与命令所需参数信息，前端获取信息后通过socket将信息发往后端，并调用控制类的notify方法。控制类则根据前端传来的信息使用parse\_command方法进行命令的解析，并通过execute\_command方法向后端各个模块发送相应的命令与命令的参数信息。命令主要包括建图相关命令，巡航相关命令，抓取相关命令。建图相关命令包括保存地图，建图完成命令；巡航相关命令包括添加新巡航点命令，向指定巡航点巡航命令，并需输入巡航点名称参数；抓取相关命令包括目标检测命令、目标抓取命令以及目标递送命令。

## 抓取模块

抓取模块实现的是物品抓取功能，包括抓和放两个方面，需要用到物品检测的结果以获悉抓取目标的形状、大小和空间位置。这里同样需要进行预先的适配操作。

物品抓取功能的实现流程如下：

（1）抓取准备

获悉抓取目标信息之后，机器人先调整自身状态，进入最佳抓取状态。

（2）抓取物品

机器人张开手爪向前，手爪并拢抓住物品，后退返回。这里的抓取套路是固定的，机器人的手爪并拢程度由物品形状和大小决定，机械臂高度由物品的空间位置决定。

（3）传递物品

机器人在抓取到物品之后，可带着物品到达指定地点，将物品交给相应的人员。

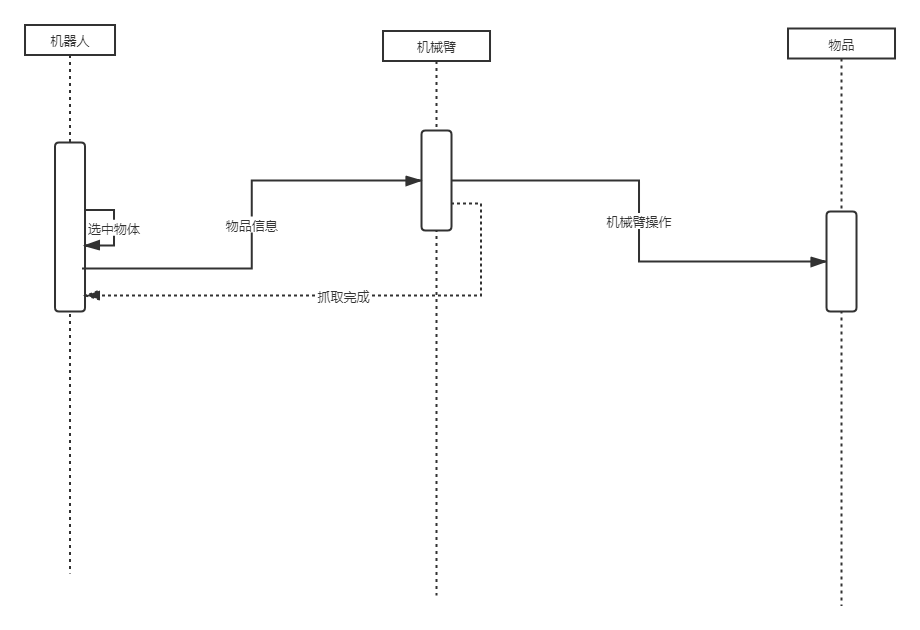


图6.3 抓取模块顺序图

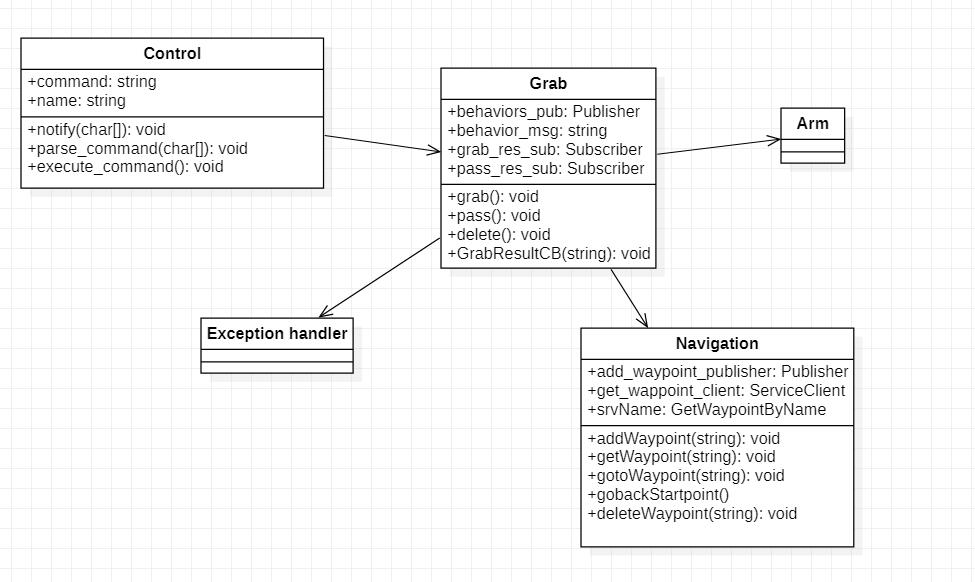


图 6.4抓取模块类图

抓取模块的主类是Grab类，用来负责机器人对物体的抓取。Behavior\_msg属性用来表示在该类的方法调用和运行过程中的状态的信息参数。Grab\_res\_sub和pass\_res\_sub两个参数是用来获取其他模块的topic发布的信息，即是否需要抓取和是否需要传递（递交抓取物品），behavior\_pub则是用来向其他的模块发布运行抓取模块的运行结果。

当需要进行抓取时，首先需要确定位置信息，如果不在物品旁，需要通过Navigation模块运动到指定位置。接着调用grab函数，设置behavior\_msg信息为“grab\_start”，并通过behavior\_pub向主题发布，此时会调用系统的机械臂模块进行抓取；当需要进行传递时类似，调用pass函数，设置behavior\_msg信息为“pass\_start”，并通过behavior\_pub向主题发布，此时也会调用机械臂模块。GrabResultCB函数为抓取和递送的回调函数，抓取递送后通过回调函数返回抓取递送结果。另外，有异常处理模块负责处理异常情况。

## 路径规划与运动模块

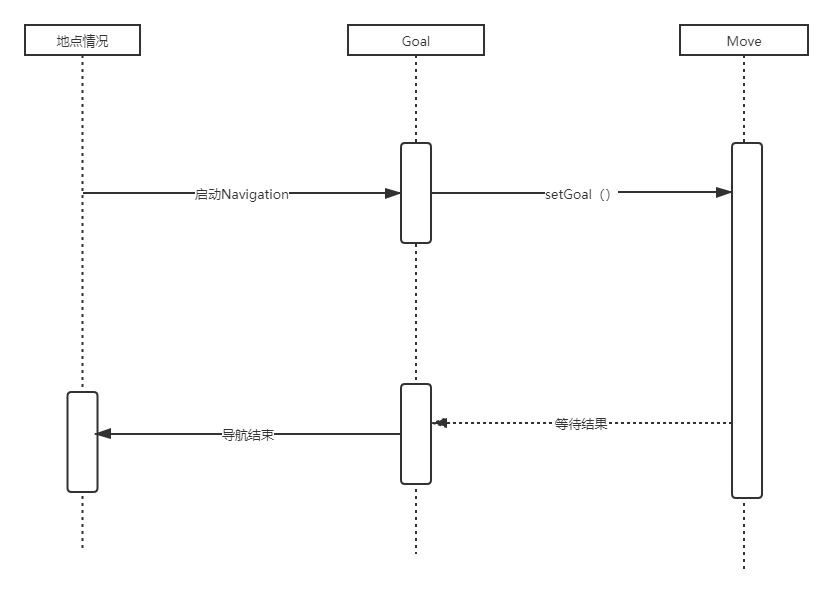
导航模块中，在机器人初次巡航时，导航模块应该记录下标定点的名称。而在记录完成后，则可让机器人到达各指定的标定点和出发的原点

图 6.5 路径规划与运动模块顺序图

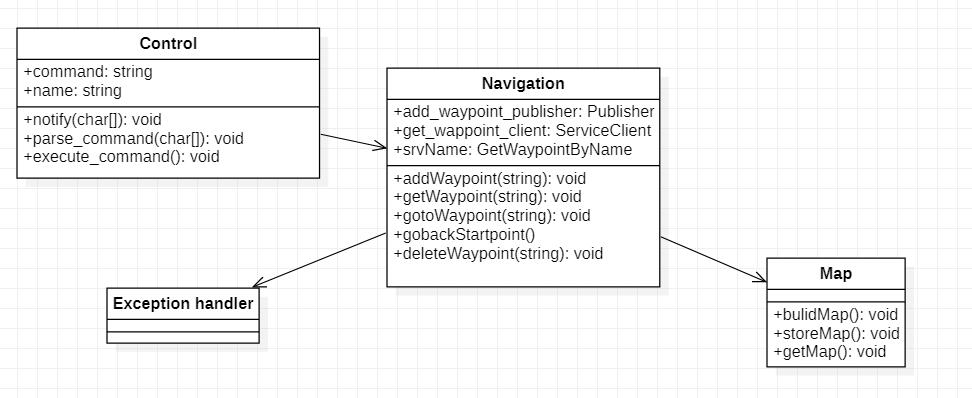


图 6.6 路径规划与运动模块类图

类图说明

运动与路径规划的主类是Navigation类，用来负责巡航点的确定和确定巡航点后，在控制模块发出指令后的具体指挥机器人运动。Add\_waypoint\_publisher属性用来在添加巡航点之后进行消息的发布。MoveBaseClient属性用来获取机器人的运动信息。

当调用这一模块时，使用addWayPoint添加巡航点，巡航点的名字由控制模块的信息提供。当想得知巡航点信息时，调用getWayPoint函数通过航点名称进行信息的获取。gotoWayPoint函数用来在建图已经完成后，控制机器人前往已经预设的指定的巡航点，首先通过航点名称，调用getWayPoint函数获取航点信息，接着发送运动向航点运动。gobackStartPoint用来在任何时候让机器人回到自己的出发点。上述两个模块需要在使用时调用建图模块部分以获取巡航点的具体位置。deleteWayPoint函数用来删除在建图后用户不想保存的巡航点，同时在用户的输入模块，也会显示对应的巡航点已经被删除。另外，有异常处理模块负责处理异常情况。

## 建图模块

建图实现即时定位与当前场景的地图构建。根据激光雷达的扫描结果，使用Hector SLAM算法构建地图并保存地图。

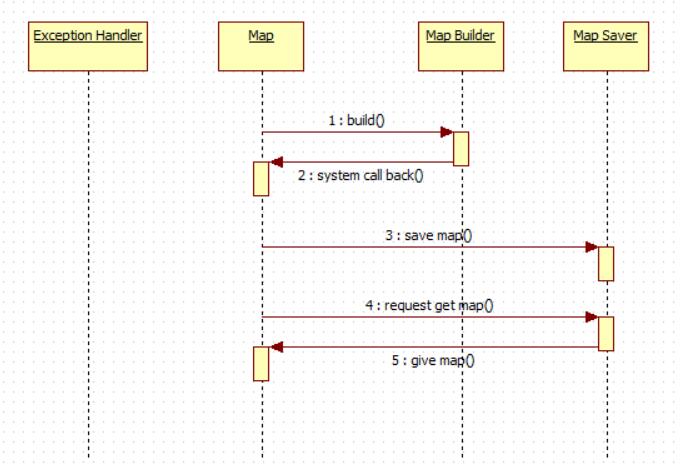


图6.7 建图模块顺序图

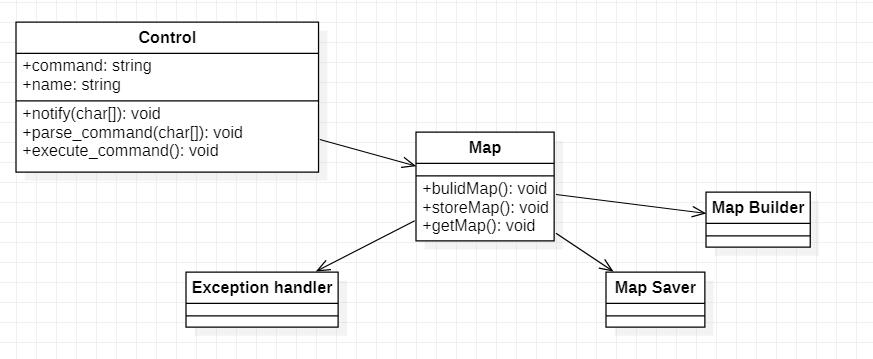


图 6.8 建图模块类图

类图说明

建图模块用于建图及相关的操作。建图模块的主类是Map类。Map类控制着建图的最终结果，map.pgm和map.yaml。这两个文件也是所建立场景的地图的全部信息。

当初次巡航时，需要调用buildMap函数进行建图操作，此操作在机器人第一次巡航时，由用户输入模块通知是否调用该函数进行建图操作。建图时，采用机器人自带的Radar，使用基于滤波SLAM框架的常用开源SLAM算法-Gmapping算法来构建地图，保证了建图的准确性。当建图完成后，需要调用storeMap函数将建图生成的文件进行保存，保存地址可以自行设定和修改。当机器人需要执行其它功能，如到达指定目的地点时，需要获取地图信息，就需要Map类的地图信息，此时使用getMap函数调取存储的地图信息。另外，有异常处理模块负责处理异常情况。

## 异常处理模块

异常处理模块用于检测和处理程序之中出现的异常情况。在机器人进行巡航，建图，以及抓取的过程中，机器人本身的设备可能出现各种各样的情况；同时，我们编写的代码也可能存在着bug未被发现，尽管将会进行单元测试和系统测试。这时，需要异常处理模块的介入，将故障模块强行终止，并恢复成初始完好的状态。

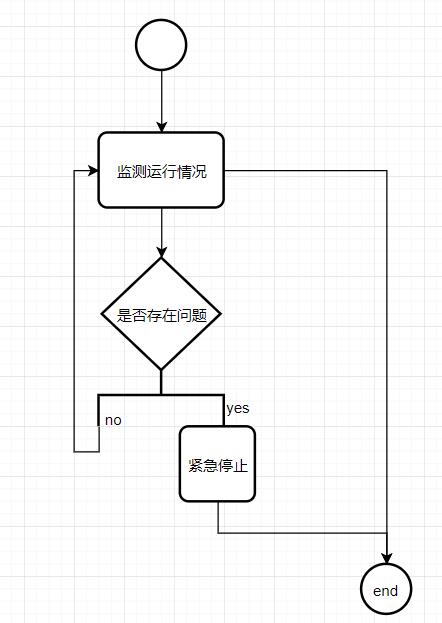


图6.10 异常处理模块的流程图

# 运行与开发环境

## 运行环境

内存4GB（指虚拟机的分配内存或者双系统电脑内存）以上

硬盘80GB以上的空间

一台启智ROS机器人

## 软件环境

操作系统Win10并装有Ubuntu14.04或16.04的双系统或虚拟机、还需要装有IDEA、PyCharm、Visual Studio等IDE以及各IDE运行所需要的环境。最后还要在双系统或者虚拟机中装好ROS操作系统和启智ROS自己的软件包。

# 需求可追踪性说明

## 功能需求说明

### 路径规划功能

对应于SRS中的基本建图部分。

功能模块：制图类Initial，路径规划类FindPath，运动类Move，用户输入类Input，用户指令交换类Translate。

设计：调用制图类中的制图方法，机器人建立运行地图后返回起点。利用交换类中的地图转换方法，把机器建图得到的图像在用户界面显示，点击图像可以设置目的地，调用用户输入类和转换类将用户输入信息转换成对机器人的指令。机器人调用路径规划类进行路径规划，规划完成后机器人开始按照规划路线运动，遇到不明障碍调用实时物体避障功能，直到到达指定目标点。

### 实时避障功能

对应于SRS中的定点巡航部分。

功能模块：障碍检测类Obstacle。

设计：机器人在按照原规划路径行进过程中，不断调用障碍检测模块进行障碍物检测，遇到障碍物时进行避障，或者在检测到障碍物消失后机器人开始继续运行。

### 用户操作功能

对应于SRS中的用户界面需求

功能模块：用户输入类Input，用户指令转换类Translate。

设计：用户可以请求获取地图请求，机器人接收到请求后制图并返回地图图像，用户根据图像选择目的地。机器人到达目的地后，传递给用户可以抓取的目标，用户选择物品后再次将信息传递给机器人。

### 物体抓取功能

对应于SRS中的目标检测及抓取部分

功能模块：识别物体类Target，抓取物体类Fetch，用户输入类Input，用户指令转换类Translate。

设计：机器人调用物体识别类，将生成的图像传输到用户交互界面，用户点击物体调用用户输入Input类及转换模块Translate类将用户操作转化为抓取物体的信息传输至机器人，机器人根据接收的信息调用抓取物体模块Fetch类抓取相应物品。

## 非功能性需求说明

### 运行性能

1. 在实验室环境下，确定起点与终点后，路径规划应当在300秒之内完成。
2. 在完成路径规划之后，机器人应该在1秒内完成响应和启动。
3. 机器人在路径上的运动速度应该保持在0.1m/s到0.2m/s之间，如果过快则应该紧急停止
4. 目标物体识别完成后，机器人应当在60s之内完成物体抓取
5. 机器人应该与所有障碍物和目标物体都保持至少0.2米的距离，一旦小于该距离，则立刻减速停止

### 可用性

1. 对于所有的正常情况，即起点和终点可达且没有障碍物，从起点到终点之间至少存在一条可达路径，那么机器人就可以正常的启动并且执行任务。如果不符合上述要求，则机器人在分析路径规划之后应该发出提示，拒绝开始执行任务。
2. 在机器人执行任务的时候，任何情况下不应该和其他障碍物包括人离得过近甚至发生碰撞，过近时需要紧急停止。
3. 在抓取物体过程中，机械臂对物体施加的力不应当过大而损坏物体，同时也不能过小而导致在机器人运行过程中物品掉落

### 易用性

用户界面应该简洁易用

### 安全性

1. 对于不能完成路径规划（无法从起点前往终点），应该给出信息。
2. 对于遇到的障碍物和人，应该主动避让，保持固定的距离
3. 抓取结束之后应该返回指定的地点，同时可以立刻进行下次的任务设定
4. 对于遇到突发情况停下时，应该重新把现在的位置设为起点，然后重新进行路径规划，继续完成任务
5. 机器人执行任务时可以随时中止任务，并让机器人恢复原状
6. 如果遇到了其他的情况，机器人应该立刻停止