**简易机器人**

**软件设计说明书**

**SDD102**

**V0.1.0**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | NaN | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 16061062 | 乔艺璇 | 编写第1、3.1、5、6.1部分 |
| 15071060 | 宋晓曦 | 编写第3.1、4.2、6.4、7部分 |
| 16061065 | 林亭妤 | 编写第2、4.1部分 |
| 16061123 | 祁令姿 | 编写第3.1、6.5、6.6部分，文档整合 |
| 16061054 | 梁夏宇 | 编写第3.2、6.2、6.3、8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 |
| 0.1.0 | 2019.04.21 | 全体 | 祁令姿 | 初版设计文档，待完善 |
| 0.2.0 | 2019.06.07 | 乔艺璇 |  | 增加类图中的函数与细节说明、系统结构图以及时序图。TODO:需要补充类和模块的细节描述。 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 范围

## 项目概述

本项目是北京航空航天大学计算机学院2019年春季学期所开设的《软件工程》课程要求开发的项目。本项目在课程所提供的软硬件资源的基础上，开发一款使用于如家中或办公室等室内场景，根据传感器数据自动或手动进行地图建立，并在此基础之上实现路径规划、运动与更新地图数据，以及根据图像信息利用机械臂完成物体抓取的机器人。本项目可以作为尝试室内智能机器人的基本功能的原型机，并可以借此机会，学习合作进行工程化开发和学习嵌入式系统的相关知识。

## 文档概述

本文是ROS机器人开发的设计说明书。

本文档包含以下八个章节：

1. 范围 概括性描述本项目的内容与本文档的用途，解释相关术语并列举引用文档。
2. 需求概述 从用户视角介绍系统的业务环境、用户、承载的主要业务流程。详细描述系统完整的功能与相关应用场景用例，并给出用例图进行描述。
3. 体系结构设计 描述总体结构和关键问题及解决方案。描述软件体系结构、硬件体系结构、技术体系结构、支撑体系（部署和实施方案）结构等各个方面的设计细节，并使用一系列图表进行说明。
4. 接口设计 说明系统用户界面、系统的软硬件外部接口和内部接口的设计。
5. 数据库说明 以ER图进行描述。
6. 运行与开发环境 明确系统开发与运行的硬件环境与软件环境，并介绍用户界面的设计方案。
7. 详细设计 对于系统的每个关键模块逐个给出一个模块的详细设计方案。包括该模块的类图及其顺序图等内容。
8. 需求可追踪行说明 说明SRS文档中功能同本文档总体结构部分的对应关系，说明设计如何满足需求各项功能和非功能性需求。

## 术语和缩略词

**表 1.3-1 术语与缩略词**

|  |  |
| --- | --- |
| ***术语或缩略词*** | ***解释或全称*** |
| ***ROS*** | 全称是Robot Operating System，ROS是一个机器人软件平台，它能为异质[计算机集群](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E9%9B%86%E7%BE%A4/1728905" \t "_blank)提供类似操作系统的功能。 |
| ***ER图*** | E-R图全称为实体-联系图(Entity Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的[概念模型](https://baike.baidu.com/item/%E6%A6%82%E5%BF%B5%E6%A8%A1%E5%9E%8B/3187025" \t "_blank)。 |
| ***SRS*** | SRS全称为软件需求说明书，即Software Requirements Specification, 软件需求说明书的编制是为了使用户和软件开发者双方对该软件的初始规定有一个共同的理解， 使之成为整个开发工作的基础。 |
| ***SDD*** | SDD全称为软件设计文档，即Software Design Document。它是根据项目需求而设计的软件文档。 |
| ***Ubuntu*** | Ubuntu（是一个以[桌面](https://baike.baidu.com/item/%E6%A1%8C%E9%9D%A2" \t "_blank)应用为主的开源GNU/Linux操作系统 |
| ***Kinect2*** | Kinect2 视觉传感器装备在ROS 机器人的头部，可以在 Rviz 中查看其数据形态。 |
| ***SLAM*** | SLAM (simultaneous localization and mapping), 意为即时定位与地图构建。 |

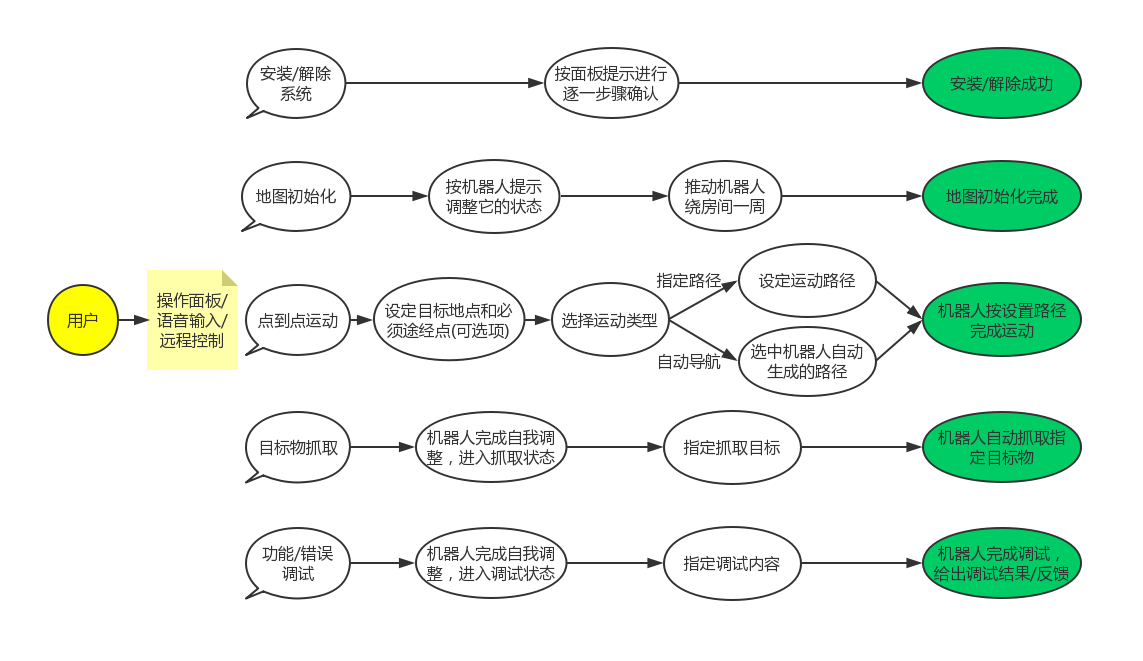
## 引用文档

**表 1.4-1 引用文档**

|  |  |  |  |
| --- | --- | --- | --- |
| ***编号*** | ***标题*** | ***版本*** | ***发行日期*** |
| ***1*** | 启智ROS机器人开发手册 | V1.1.0 | -- |
| ***2*** | SRS-102组需求规格说明书 | V0.2.0 | -- |
| ***3*** | SRS-102组开发计划 | V0.1.0 |  |

# 需求概述

## 业务需求



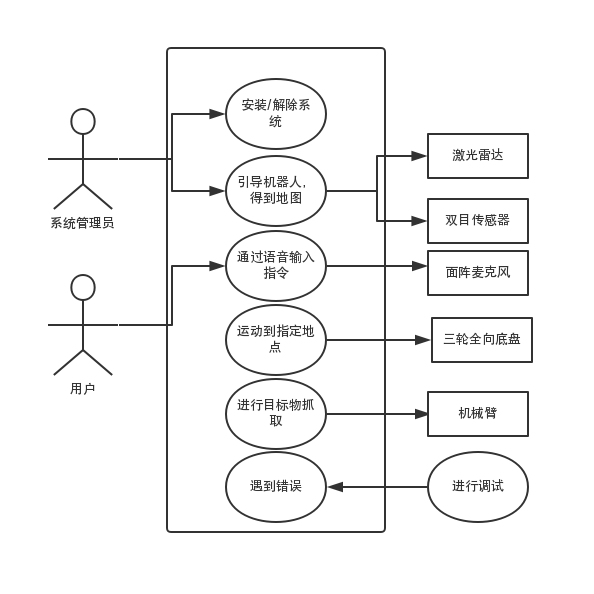
**图 2.1-1 业务流程示意图**

注：（1）地图初始化无法通过远程操控完成，必须要有现场人工辅助；（2）语音输入功能支持效果可能较差（或者完全不支持）。

## 用户描述

第一次使用时，用户需安装系统并初始化地图，地图初始化方法为手动推行机器人绕地图一圈。每次通过操作面板输入指令时，需要保证指令格式正确，以便机器人识别。在使用过程中，需要保证机器人运行环境干燥，以免硬件受到损伤。对于抓取目标物的指令，用户需保证目标物重量在机器人承重范围（10kg）内。

## 用例模型



**图 2.3-1系统用例模型**

## 启动机器人用例

主要参与者：用户/系统管理员

目标：启动机器人，使机器人处于可被使用的状态

前置条件：完整配置系统，机器人电量充足

启动：用户准备使用机器人

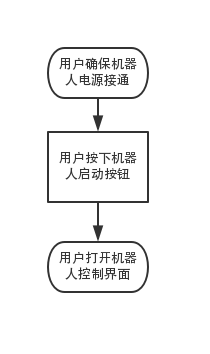
场景：

1. 用户确保机器人电源接通
2. 用户按下机器人启动按钮
3. 用户打开机器人控制界面

优先级：高

何时可用：机器人关闭时

使用频率：低



**图 2.4-1启动机器人用例**

## 地图扫描用例

主要参与者：系统管理员

目标：使机器人获取仓库地图

前置条件：完整配置系统，激光雷达正常运行

启动：在新场地中第一次使用机器人

场景：

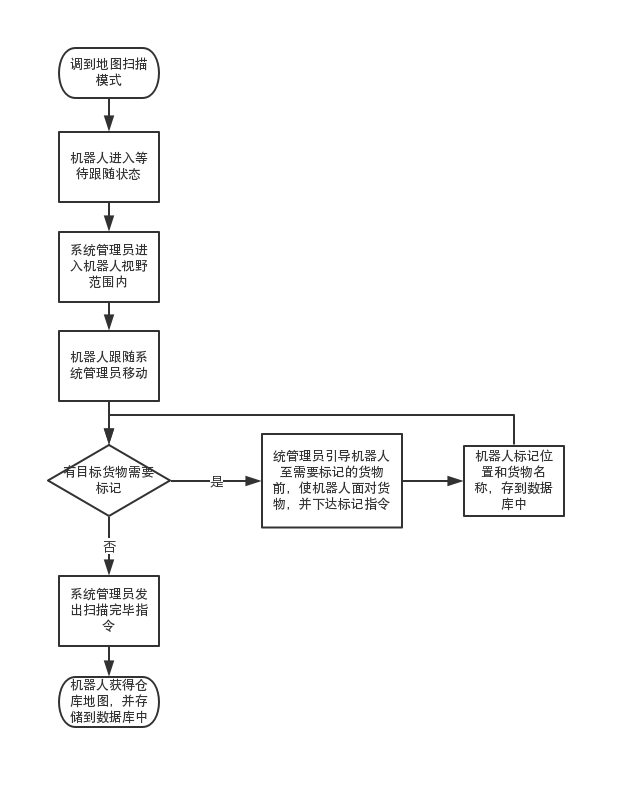
1. 系统管理员将机载电脑调到地图扫描模式
2. 机器人进入等待跟随状态
3. 系统管理员进入机器人视野范围内
4. 机器人跟随系统管理员移动
5. 系统管理员引导机器人至需要标记的货物前，使机器人面对货物，并下达标记指令
6. 机器人标记位置和货物名称，存到数据库中
7. 系统管理员引导机器人在地图上移动完一圈后，发出扫描完毕指令
8. 机器人通过激光雷达获得仓库地图，并存储到数据库中

优先级：高

何时可用：第一次在新场地使用时

使用频率：低

次要参与者：激光雷达



**图 2.5-1地图扫描用例**

## 路径规划及避障用例

主要参与者：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：完整配置系统，已获得仓库地图

启动：用户对机器人有派遣需求时

场景：

1. 用户将机器人控制系统调至派遣模式
2. 用户通过语音控制下达指令内容‘Go to xxx’
3. 机器人检索数据库中内容，获取‘xxx’位置
4. 系统计算路径
5. 机器人沿上一条获得的路径运动
6. 机器人到达指定地点

异常状况：

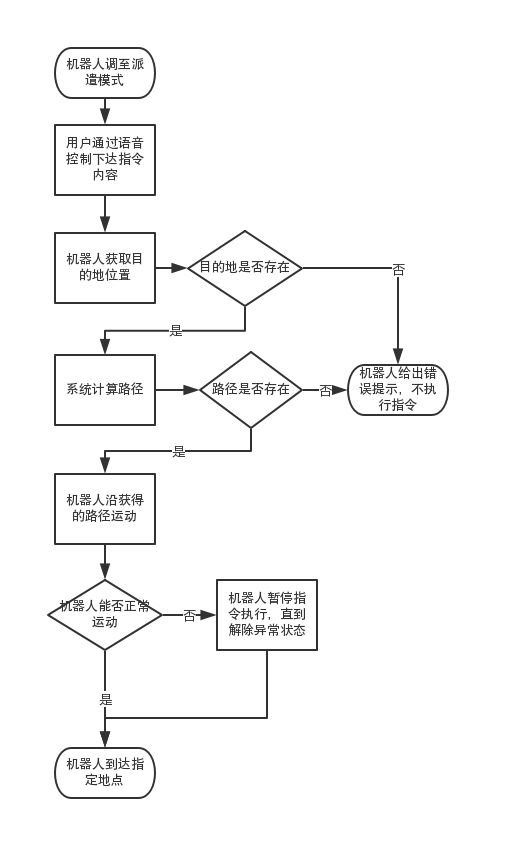
1. 指令内容错误：目的地不在数据库内——出现点：场景3，处理：机器人给出错误提示，不执行指令。
2. 路径计算错误：目的地无法到达——出现点：场景5，处理：机器人给出错误提示，终止指令执行。
3. 运动错误：机器人陷入无法移动的情况，如倾倒、悬空——出现点：场景6，处理：机器人给出错误提示，暂停指令执行，用户解除机器人异常状态。

优先级：中

何时可用：用户对机器人有派遣需求时

使用频率：高

次要参与者：机器人控制系统



**图 2.6-1路径规划及避障用例**

## 取物用例

主要参与者：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，并抓取指定货物

前置条件：完整配置系统，机器人正常运行，已获得房间地图，有符合要求得目标物在指定位置

启动：用户对机器人有取物需求时

场景：

1. 用户切换系统到取物模式，机器人载入地图
2. 用户下达指令：‘Bring me xxx’（xxx为货物名称）
3. 机器人识别关键词‘xxx’，在数据库中检索‘xxx’的对应位置
4. 机器人系统计算初始位置到货物物位置的路径
5. 机器人沿上一条获得的路径运动
6. 机器人到达指定地点
7. 机器人检测目标物，调整自身位置
8. 机器人抬起机械臂，抓取货物
9. 机器人沿来时路径运动，返回起点

异常状况：

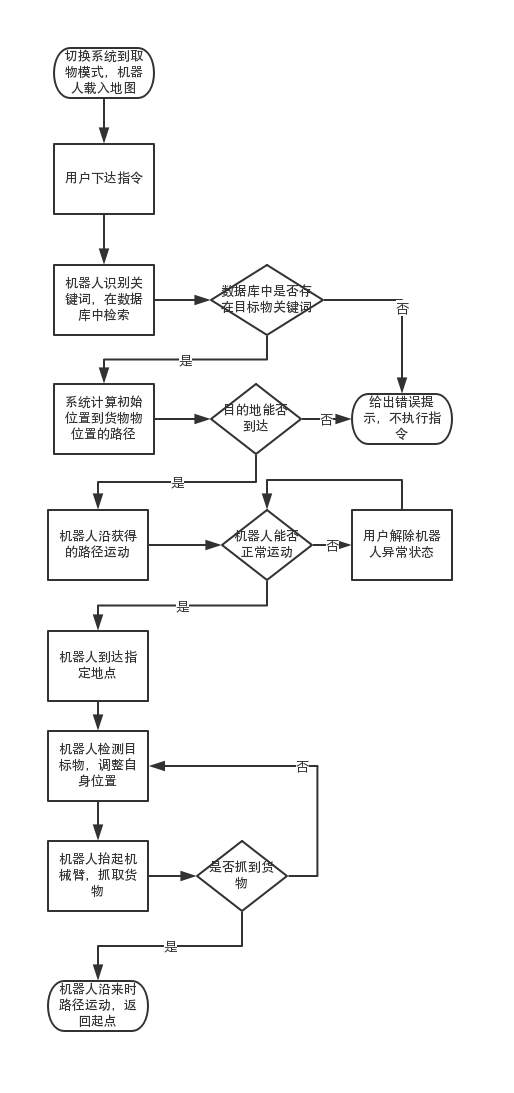
1. 指令内容错误：数据库中不存在关键词——出现点：场景3，处理：机器人给出错误提示，不执行指令。
2. 路径计算错误：目的地无法到达——出现点：场景5，处理：机器人给出错误提示，终止指令执行。
3. 运动错误：机器人陷入无法移动的情况，如倾倒、悬空——出现点：场景6、9，处理：机器人给出错误提示，终止指令执行，用户解除机器人异常状态。
4. 识别错误：抓取目标识别失败——出现点：场景8，处理：机器人再次识别。
5. 抓取错误：机器人抓取失败——出现点：场景9，处理：用户重新下达指令，机器人重新执行指令。

优先级：中

何时可用：用户对机器人有取物需求时

使用频率：高

次要参与者：机器人控制系统、双目传感器、机械臂、目标货物



**图 2.7-1取物用例**

# 体系结构设计

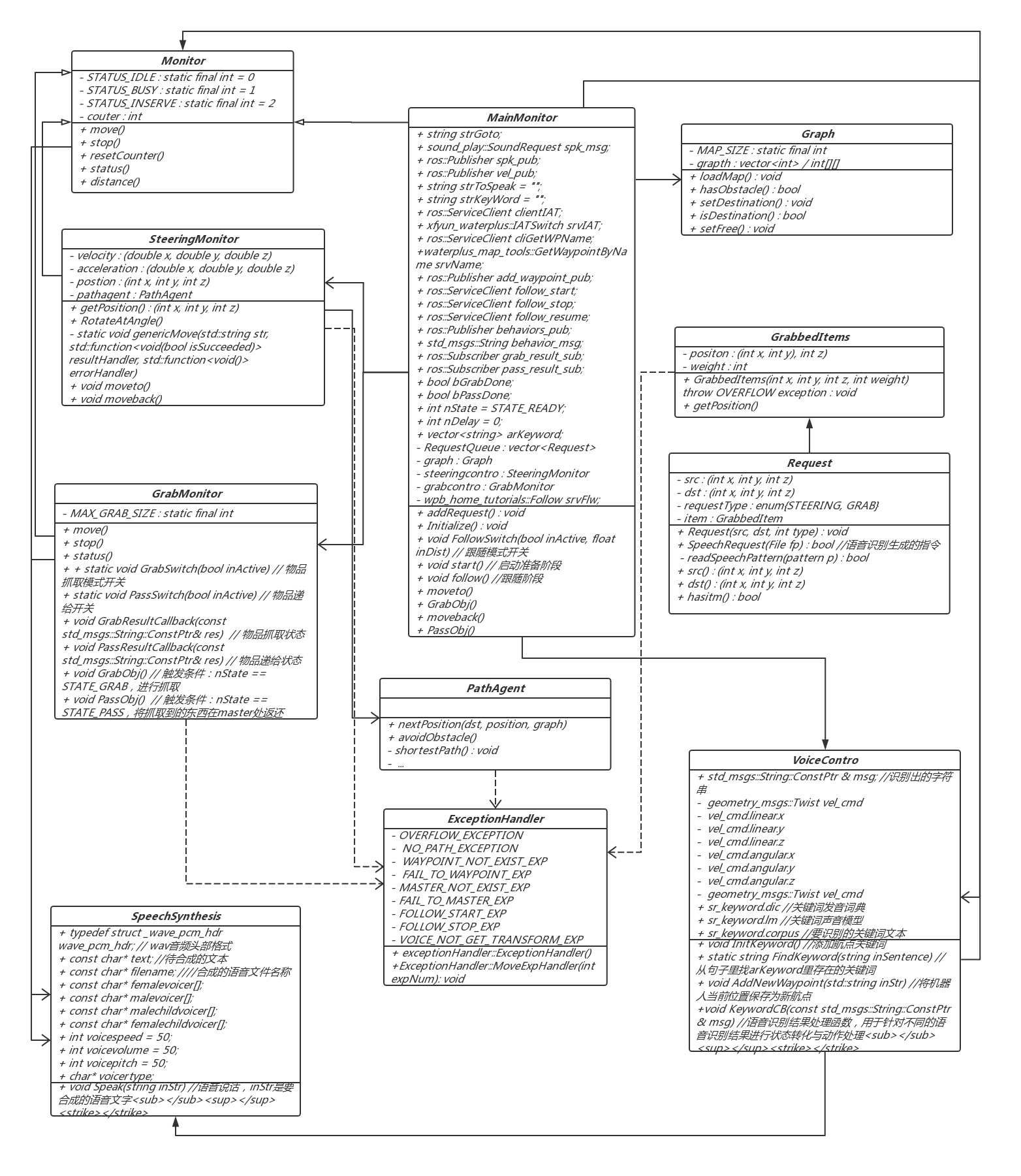
## 总体结构

### 系统体系结构

**图 3.1-1 系统体系结构图**

### 软件体系结构

#### 软件体系结构类图

1. UML类图

**图 3.1-2 软件体系结构类图**

1. 类层次结构与关系说明

1.Monitor类：所有控制类的父类，包含了基本控制函数，如：

move()：控制机器人行走

stop()：控制机器人停下

resetCounter()：重置系统计时器。Counter是系统计时器。

status()：查询机器人现在的状态。BUSY为正在执行指令，IDLE为空闲挂起状态，INSERVE表示正在服务但是没有指令的状态。

distance()：机器人和目的地址的距离。

2.SteeringMonitor类：行走控制类，继承Monitor类，包含了行走控制函数，并实例化了路径规划类。通过向/cml\_vel这一topic发布速度更新消息控制机器人的运动速度与运动方向

velocity : (double x, double y, double z) 表示机器人当前速度

acceleration : (double x, double y, double z) 表示机器人当前加速度

postion : (int x, int y, int z) 表示机器人当前所处位置

pathagent : PathAgent 表示路径规划智能体

getPosition() : (int x, int y, int z) 用于得到机器人的下一步行走位置并进行状态和位置的更新

RotateAtAngle()：使机器人以某一个角度进行旋转，以达到左右转，向后转等等功能。

3. SpeechSynthesis类：语音合成播报类，对文本进行语音合成。从而达到与用户交互的目的。

void Speak(string inStr)：语音合成函数，并使机器人播报信息，inStr是要合成的语音文字。

4.GrabMonitor类：机器人的抓取控制器类，用于控制机器人抓取的动作。

MAX\_GRAB\_SIZE：机器人最大的抓取重量

5.ExceptionHandler类：异常处理类，用于处理系统可能产生的各种异常。

OVERFLOW\_EXCEPTION：溢出异常，如要抓取的物体过重。

TURN\_SIDE\_EXCEPITION：机器人侧翻异常。

NO\_PATH\_EXCEPTION：路径规划中无可以行走路径的异常。

exception(int type)：产生异常函数，根据异常type的内容执行不同的异常处理程序。

6.Request类：

src : (int x, int y, int z) 请求的起始地点

dst : (int x, int y, int z) 请求的目的地点

requestType : enum{STEERING, GRAB} 请求的种类，STEERING表示避障路径规划型，GRAB表示避障路径规划抓取型。

item : GrabbedItem 请求对应的抓取物

Request(src, dst, int type) : void请求的构造函数

SpeechRequest(File fp) : bool语音识别生成指令，此函数会调用科大讯飞的语音识别SDK，生成语音识别文本，然后将文本转化为指令并录入。

readSpeechPattern(pattern p) : bool 对语音识别文本进行格式匹配，只有固定格式的请求才会被接受，返回值是是否匹配成功。

src() : (int x, int y, int z) 获取请求的源地址

dst() : (int x, int y, int z) 获取请求的目的地址

hasitm() : bool 判断是否为抓取类请求

7.GrabbedItems类：抓取物品类，被抓取类请求实例化。

positon : (int x, int y), int z) 要抓物品的所在地点

weight : int 要抓取物品的重量

GrabbedItems(int x, int y, int z, int weight) throw OVERFLOW exception : void+ status() 抓取物品的构造函数，可能会产生抓取物品过重的异常。

getPosition() 用于获取抓取物的地点

8.PathAgent类：智能路径规划类，用于路径规划。实现了机器人的基本避障功能。此类会调用传感器类，查看周围的障碍物信息，并且会查询地图类，得到可能的路径，并实时进行下一步路径规划。

avoidObstacle()：调用传感器，计算出地图上的不可行走点，并将可行路径传回。

nextPosition(dst, position, graph)：用于查询机器人下一步行走位置

shortestPath() : void 最短路径算法函数，用于规划最短路径

9.MainMonitor类：主控制类，继承Monitor类，用于机器人总控与调度。

RequestQueue : vector<Request> 请求队列，用于存放用户在UI界面输入的请求。

graph : Graph 房间地图

steeringcontro : SteeringMonitor 行走控制器

grabcontro : GrabMonitor 抓取控制器

addRequest() : void 用于向队列里添加请求

Initialize() : void 初始化主控制器的函数

10.Graph类：地图类

MAP\_SIZE : static final int 房间地图大小

grapth : vector<int> / int[][] 地图实体

loadMap() : void 加载地图函数

hasObstacle() : bool 判断地图上某点是否有障碍物

setDestination() : void 设置目的地

isDestination() : bool 判断是否到达目的地

setFree() : void 完成请求，解除目的地

11. VoiceContro类：语音控制类，可以进行语音识别，并进行控制命令的关键词匹配，然后根据关键词对机器人下达命令。

speechRecognition()：语音识别函数，调用科大讯飞的语音识别功能。

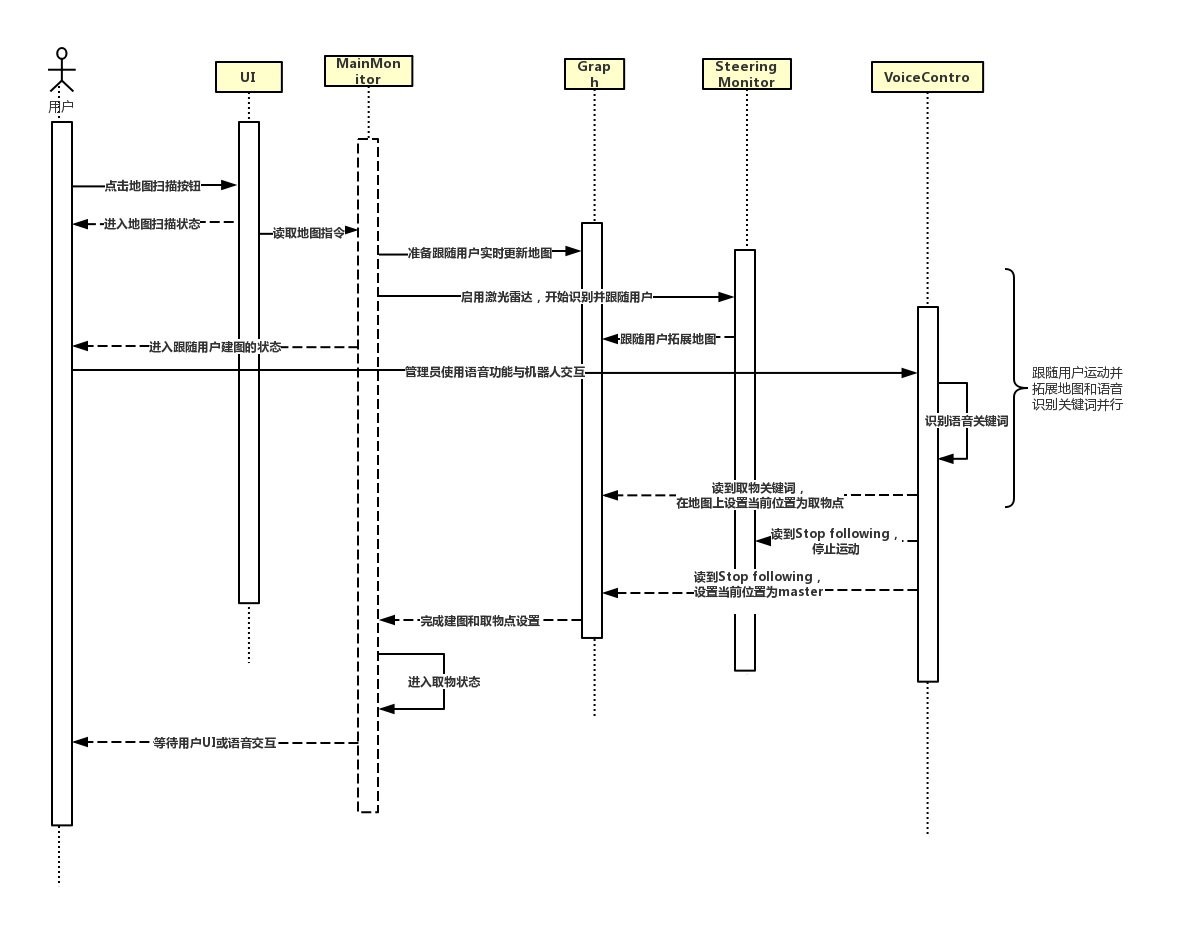
void InitKeyword()：添加航点关键词

static string FindKeyword(string inSentence)：从句子里找arKeyword里存在的关键词

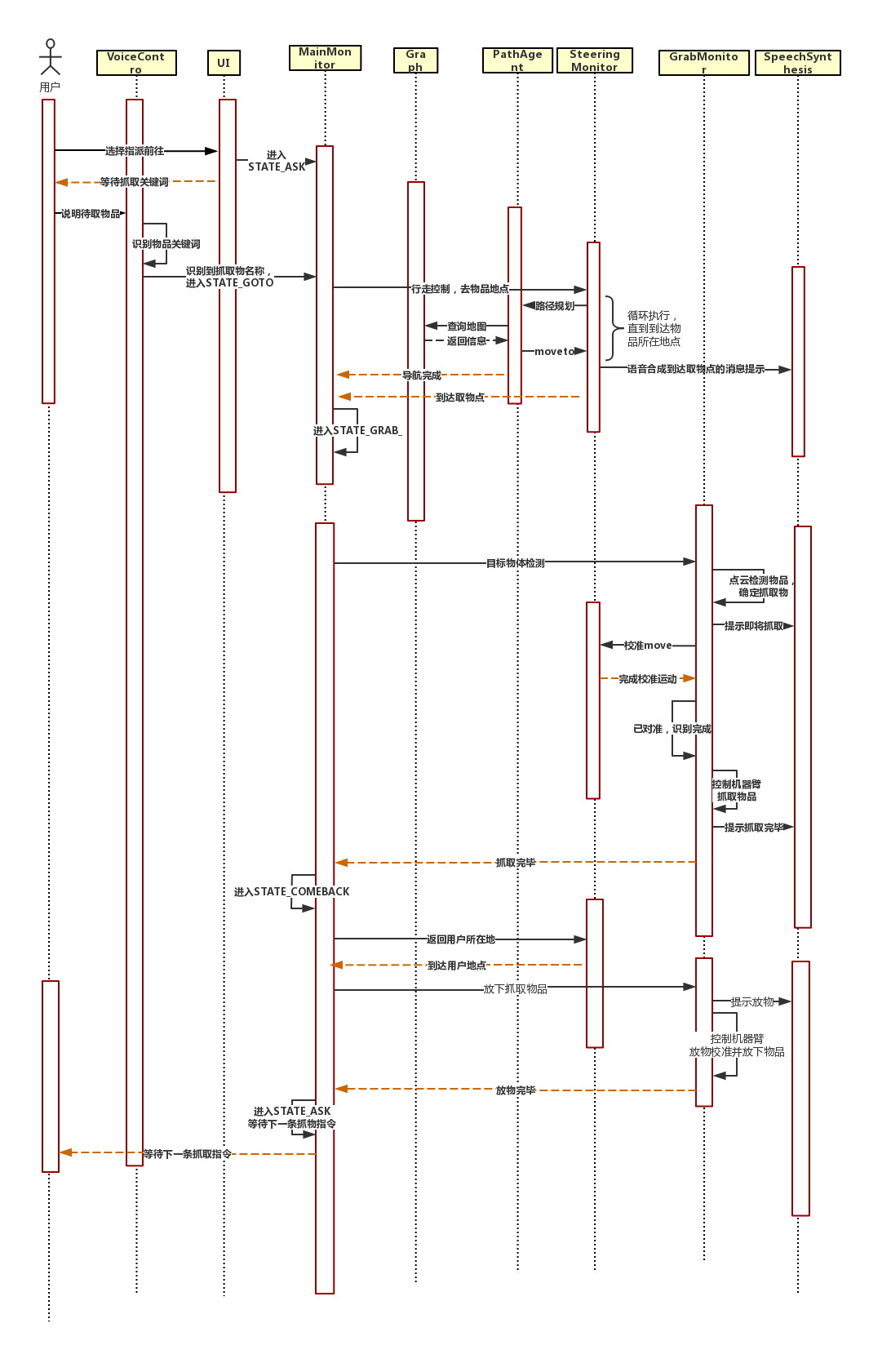
void AddNewWaypoint(std::string inStr)：将机器人当前位置保存为新航点

void KeywordCB(const std\_msgs::String::ConstPtr & msg)：语音识别结果处理函数，用于针对不同的语音识别结果进行状态转化与动作处理

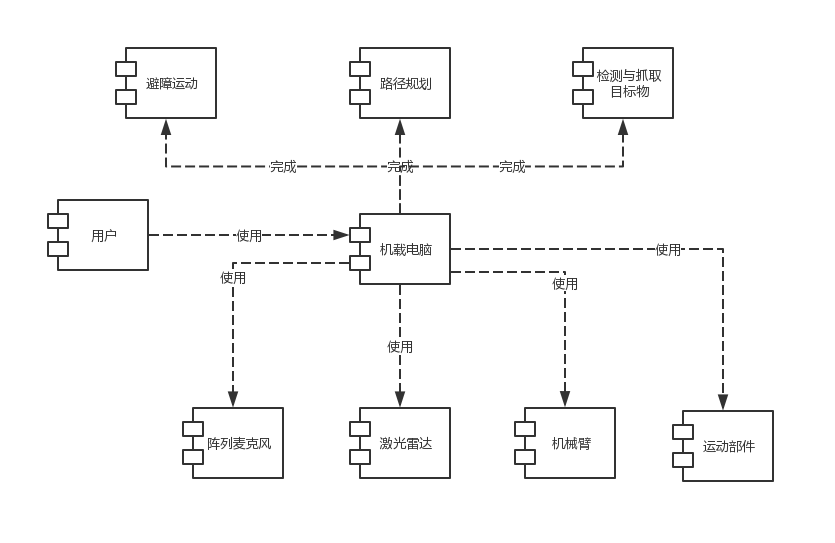
#### 软件体系结构时序图

本部分以时序图的形式展现出软件的体系结构，为了避免时序图过于冗长，我们将时序图在逻辑上分为两部分，第一部分是地图扫描与取物点设置，是取物的前期准备阶段；第二部分是指派前往与抓取物品，是正式的取物阶段。

**图 3.1-3 地图扫描与取物点设置时序图**

**图 3.1-3 指派前往与抓取物品时序图**

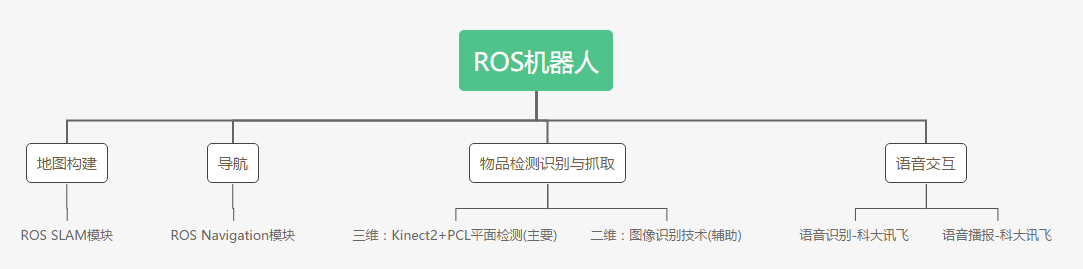
### 硬件体系结构

****

**图 3.1-2 硬件体系结构**

机载电脑控制阵列麦克风获取用户指令；使用激光雷达、运动部件完成路径规划和运动；使用激光雷达、运动部件和机械臂实现目标物检测与抓取。

### 技术体系结构



**图 3.1-3 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。

### 部署和实施方案



**图 3.1-4 部署和实施方案**

系统的开发与运行都建立在ROS系统之上，源码储存与目标文件都符合ROS的文件管理规则。代码以C++为主，各种脚本辅助编译。在软件发行后，目录结构会去掉源码相关目录，仅保留目标可执行文件。

## 关键问题及解决方案

### 语音识别

本设计中，语音交互是机器人与用户的重要交互方式，语音识别是实现重点之一。为满足交互需求，本设计采用的解决方案是：基于科大讯飞的语音识别功能进行二次开发，定制个性化语音识别关键词。

科大讯飞的个性化语音识别关键词的定制流程如下：修改本地语音识别文件；将修改后的文件上传到官网，在官网完成模型训练；训练完成后，下载经过训练之后的模型的文件；用新模型文件替换旧模型文件。

需要注意的是这里的语音识别的基本实现方案是关键词匹配而不是语义分析，并且科大讯飞的语音识别功能只支持英文，还会过滤一句话的首字母、只识别其后的小写字母，因此对关键词的定制和使用都有要求。以上在开发的时候需要注意，也需要在交互之前对用户做出提示。

### 路径规划

路径规划是机器人的重点功能之一，该功能的实现涉及定位和导航两大方面。本设计在定位上采用蒙特卡洛自适应定位算法，使用概率理论在已知地图中对机器人进行位置估计，直接以估计结果作为定位结果；在导航上采用ROS的move\_base包，将导航用到的地图、坐标和行为规划器连接。

需要注意的是，由于算法精度不足、地图校准不精确和机器人初始位置校对不准确等问题，机器人的位置可能会与实际位置存在一定误差，进而可能会出现机器人“远远躲避”障碍物、规划路径离实际障碍物较远，不能得到最优路径的情况。

但由于场景设定，考虑到这里的场景空间足够大、对机器人的命令执行速度没有很高要求，以及为了实现友好的使用体验不应对用户做太高要求，本设计将不会着重考虑前述误差。

### 物品检测识别

本设计中，对物品的检测识别和抓取是机器人的重点功能之一，其中物品抓取功能较为机械化、程式化，主要包括进入抓取状态、调整机械臂、手爪前伸合拢抓住物品、带物品返回等步骤，这里不再一一详述。下面重点讲解机器人的物品检测识别功能的解决方案。

机器人的物品检测识别功能采用的主要是PCL的平面检测算法，该算法对三维点云进行分析处理。算法思路是先将水平平面检测出来；再遍历所有平面，找出高度符合要求的平面作为标准平面；然后剔除标准平面内的点云，将标准平面上方一定距离之内的点云分离出来，作为物品点云集合；最后用Kd-Tree对物品点云集合进行近邻搜索查找，将互相分离的点云团簇分割出来。每个团簇认为是一个物品。

通过上面的分析可以看到，机器人的物品检测识别结果信息实际上是物品的形状、大小和空间位置，本质上是将物品视为一块独立的特殊空间。换言之，在机器人眼中，是没有苹果和梨的区别的，只有空间位置A和B的区别。而这种等同于位置区分/划分的识别，在很多情况下在二维条件就能通过图像识别来实现，如果这时物品的空间位置信息可以通过其他方式得到补全，那么其实没必要采用上述的更复杂的三维识别方案。基于此，本设计考虑引入图像识别方案作为上述方案的补充。

# 接口设计

## 系统用户界面

**图 4.1-1 系统用户界面示意图**

机器人与用户交互的接口分为两大部分，即UI和语音。

系统用户UI界面承担了用户与系统进行交互的一部分功能，这三个按钮分别可以将系统调至“地图扫描模式”、“派遣模式”、“紧急停机模式”。

除了系统用户界面之外，用户也可以在使用中通过语音下达指令，进行地图的目标物标记、下达派遣指令和取物指令等功能。

## 软硬件接口

+MapAgent::getMap(): map 载入已保存的地图文件。

+MapAgent::genMap(): map 生成新的地图文件并保存。

+Mapsearch::slam(radar message):slam\_info传入雷达数据，进行slam地图构建。

+Mapsaver::saveMap(slam\_info): map 保存slam地图信息。

+PathAgent::getPath():Velocity 根据地图信息，计算路径。

+PathCal::move(double, double):Velocity根据路径信息，移动机器人。

+GrabObject::adjust(volume):boolean 调整机械臂的高度与抓取参数。

+GrabObject::grabAttend(volume):boolean 调整机械臂的专区，直至抓取成功，或达到最大抓取数量。

+CameraCollect::getImage():img 获取带有三维信息的景深图像

+TargetDetection::preprocess(ros\_pcl, TransformListener):pcl

+TDPreprocess::kinect2base(ros\_pcl, TransformListener):ros\_pcl 进行坐标变换。

-TDCld2Object::volumeStatic(pclArray):volumeArray:从点云簇到物体的转换。

-Motion::set\_target(location, direction):void 设置运动的目标地点与方位信息。

-Motion::act():void 执行一次运动。

-Motion::has\_obstruction():bool 检测是否有障碍物。

# 数据库设计

**表 5-1 数据表**

|  |  |  |
| --- | --- | --- |
| **数据表名** | **属性名** | **属性类型** |
| **用户（包含管理员）** | 姓名 | char(32) |
| 账号 | char(32) KEY |
| 密码 | char(64) |
| **红外激光传感器** | 序号 | int(8) KEY |
| 信息 | int(16) |
| **碰撞传感器** | 序号 | int(8) KEY |
| 信息 | int(16) |
| **双目传感器** | 序号 | int(8) KEY |
| 信息 | int(16) |
| **物品地点记录** | 序号 | int(16) KEY |
| 时间 | char(16) |
| 地点 | char(16),char(16),char(16) |
| **地图** | 序号 | int(16) KEY |
| 时间 | char(16) |
| 内容 | map |
| **指令关键词** | 序号 | int(16) KEY |
| 关键词 | char(16) |
| **日志** | 时间 | char(16) |
| 内容 | string |
| **机械臂控制参数** | 序号 | int(16) KEY |
| 时间 | char(16) |
| 机械臂的上升高度 | double |
| 机械臂的闭合宽度 | double |

# 详细设计

## 主控逻辑

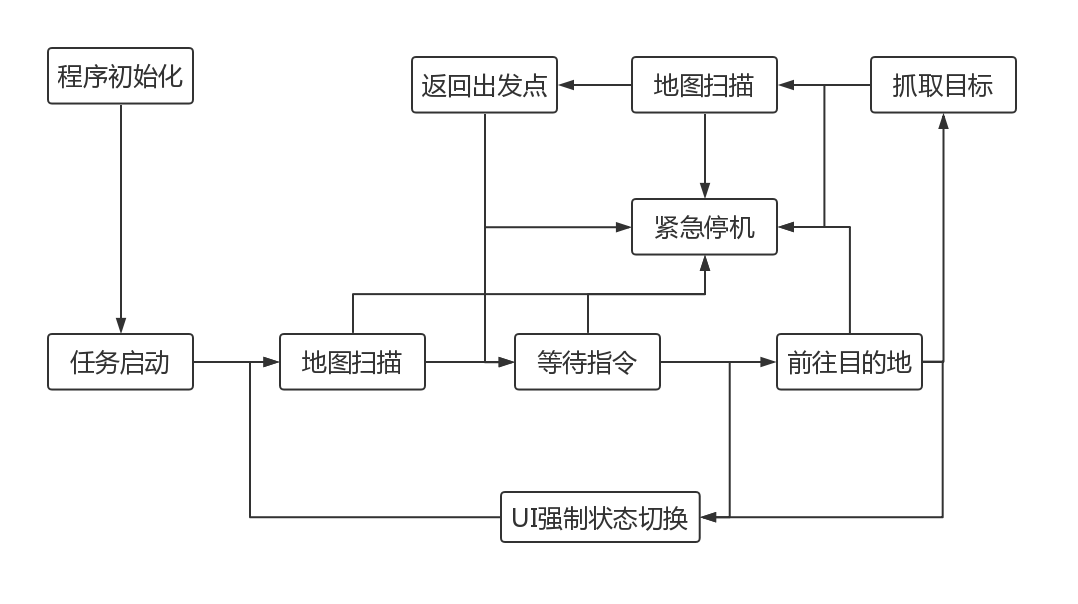
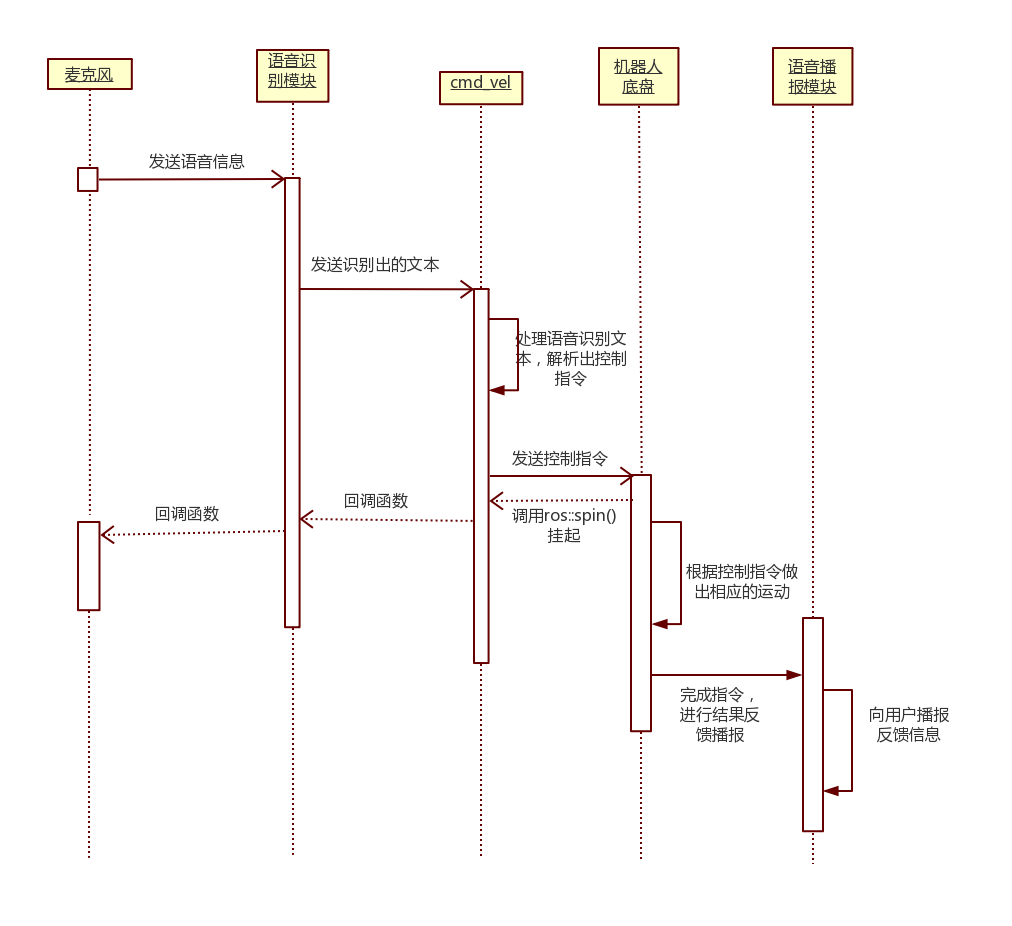


图 6.1‑1 主控单元逻辑

程序主要运行流程如图，机器人的一次正常执行流程为：程序初始化、任务启动、地图扫描、等待指令、前往目的地、抓取目标、地图扫描、返回出发点。每一个流程都可以被用户强制停止，此外一些状态可以通过UI进行切换。

## 语音模块

1. 硬件：阵列麦克风，条状，位于ROS机器人头部Kinect2下方
2. 顺序图与交互模型

图表 6.2‑1 **语音模块顺序图**

1. 类图与类结构说明

语音识别类：

对麦克风接收到的语音信息进行识别，并将识别的内容返回到一个文本中。

cmd\_vel类：

根据语音识别的结果，下达控制指令。具体发语音识别控制关键词如下

|  |  |
| --- | --- |
| ***语音识别关键词*** | ***控制指令*** |
| ***Forward*** | 以0.1米/秒的速度向前移动 |
| ***Backward*** | 以0.1米/秒的速度向后移动 |
| ***Left*** | 以0.1弧度/秒的速度向左旋转 |
| ***Right*** | 以0.1弧度/秒的速度向右旋转 |
| ***Stop***  ***Start***  ***water***  ***tea***  ***cola***  ***milk***  ***coffee***  ***juice***  ***wine***  ***whiskey***  ***stop following***  ***master*** | 停止运动  机器人开始启动的地点  物品关键词  物品关键词  物品关键词  物品关键词  物品关键词  物品关键词  物品关键词  物品关键词  停止建图跟随  用户所在位置，用于标定物品返回点 |

语音播报类：

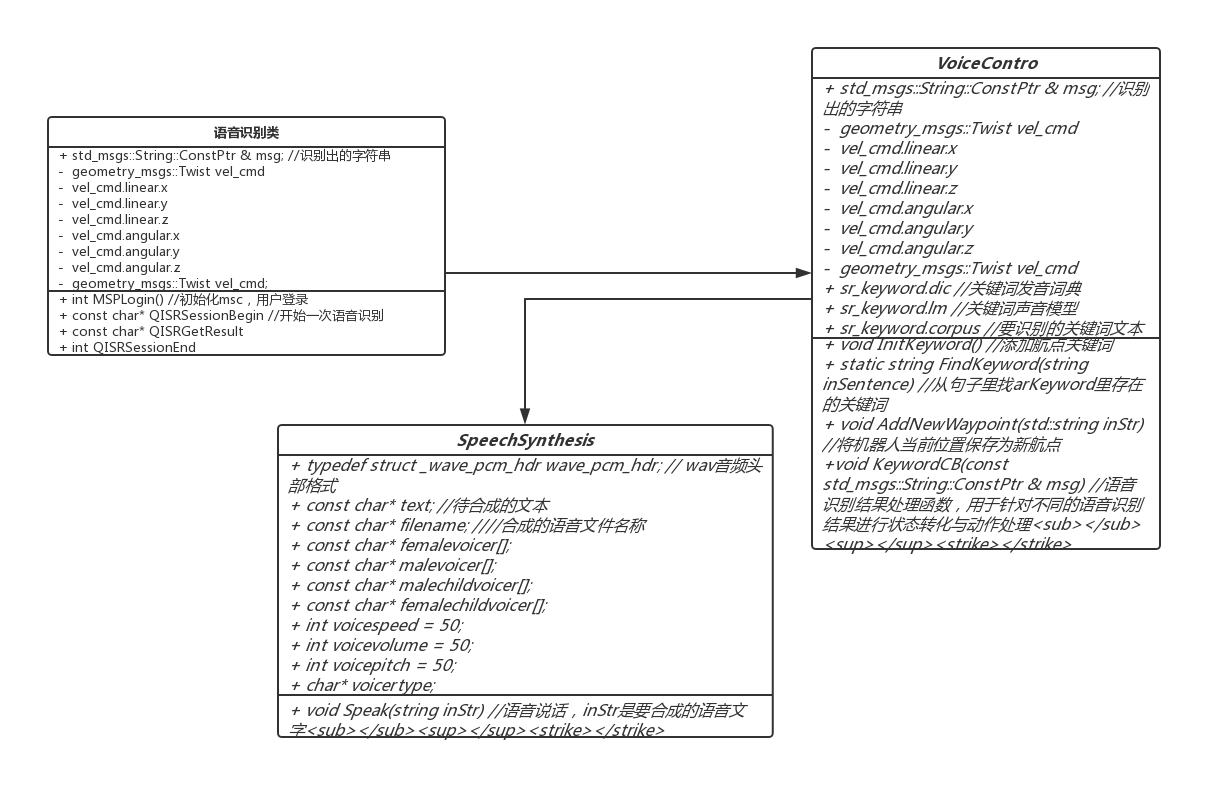
用于机器人完成控制指令后的信息反馈，此模块主要是为了更好的交互性能，用语音合成技术达到与用户的交互目的。

图 6.2‑2 **语音模块类图**

## 检测模块

检测模块实现的是物品检测功能，需要用到Kinect2视觉传感器，需要预先对Kinect2视觉传感器进行适配和校准（详细流程可参考机器人开发手册）。

物品检测功能的实现流程如下：

（1）检测预准备

检测的预准备工作包括：创建用于坐标转换的tf\_listener，订阅Kinect2的点云数据，创建和发布用于展示检测结果的主题。

检测预准备工作完成后，模块进入检测等待状态，随时可以对回传的点云数据进行分析处理。

（2）预先处理

预先处理的工作包括点云坐标转换和数据格式转换。点云坐标转换的任务是将点云坐标值从基于Kinect2传感器转换成基于机器人地面投影中心，数据格式转换的任务是将ROS格式数据转换成PCL格式数据。

经过预先处理之后，就能直接使用PCL的平面检测算法进行分析检测。

（3）点云分析检测

这里使用PCL的平面检测算法进行点云的分析检测。

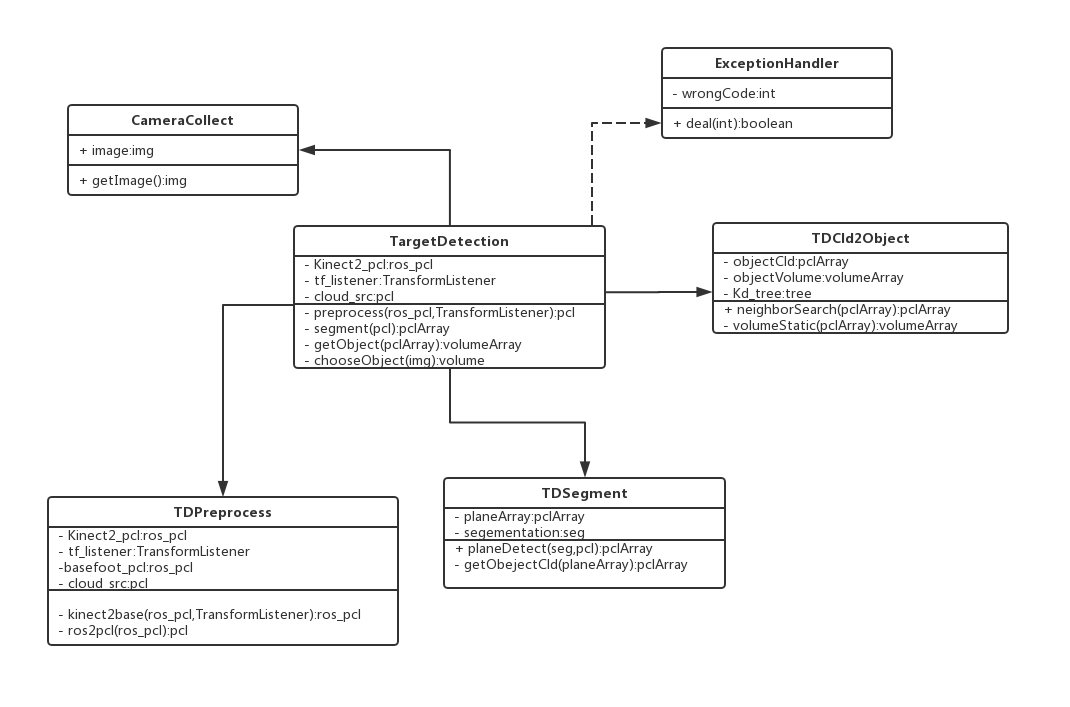
PCL的平面检测算法的思路是：先使用分割对象segmentation将水平平面检测出来；再遍历所有平面，找出高度符合要求的平面作为标准平面；然后剔除标准平面内的点云，将标准平面上方一定距离之内的点云分离出来，作为物品点云集合；最后用Kd-Tree对物品点云集合进行近邻搜索查找，将互相分离的点云团簇分割出来。每个团簇认为是一个物品。

（4）生成检测结果

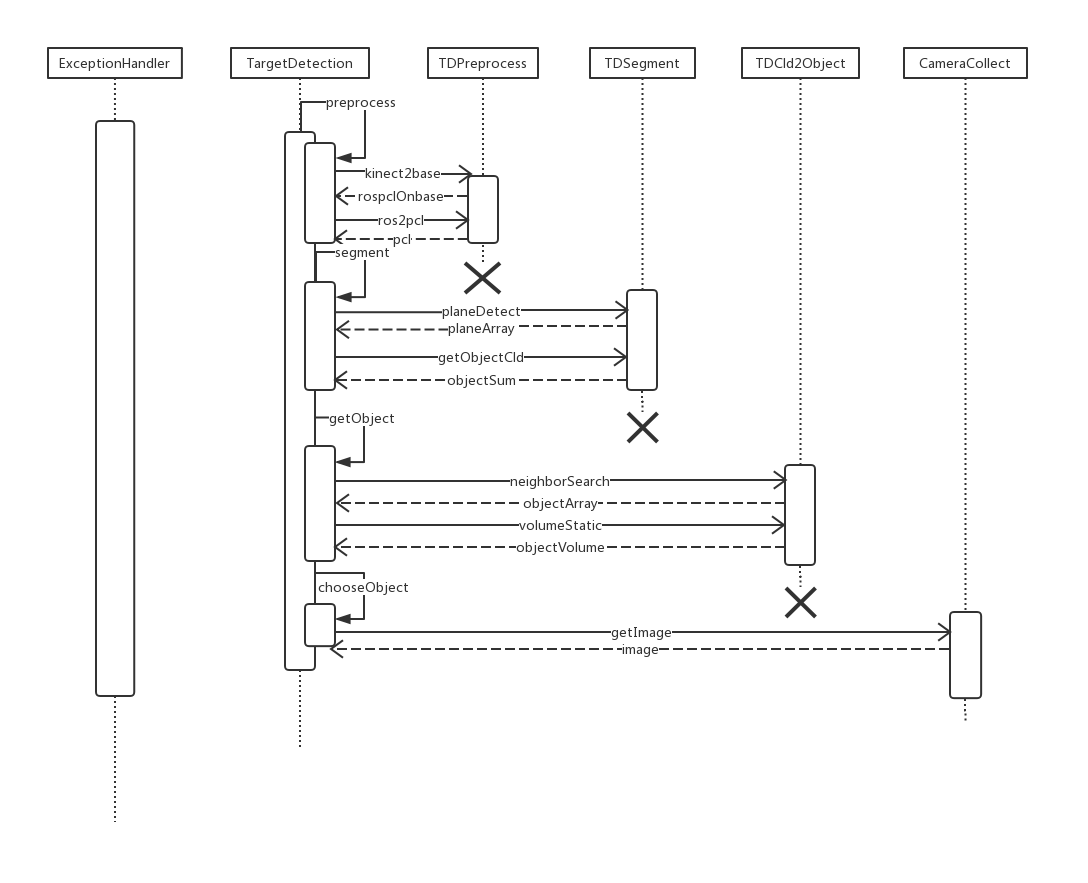
这里的工作是完成从点云团簇到物品的转换，得到物品的形状、体积，并对物品进行标号。

（5）发布检测结果

这里的工作是完成检测结果的发布，主要通过步骤（1）中创建的用于展示检测结果的主题来完成。检测结果发布之后，其他模块就能通过相应的主题获取需要的结果信息。



图表 6.3‑1 **检测模块类图**

检测模块的主类是TargetDetection，控制类通过TargetDetection实现物体检测与识别。主类的preprocess方法将初始出TDPreprocess类，通过kinect2base方法进行坐标转换，通过ros2pcl方法进行数据格式转换。主类segment方法将利用TDSegment中planeDetect方法得到平面点云，之后利用getObjectCld筛去平面点云得到其上方物体点云集合。主类的getObject方法调用TDCld2Object类中neighborSearch方法对物品点云集合进行近邻搜索查找，将互相分离的点云团簇分割出来。之后调用同类的volumeStatic方法完成从点云团簇到物品的转换，得到物品的形状、体积。最后主类通过调用chooseObject方法结合CameraCollect中返回的信息完成识别特定物体的工作。ExceptionHandler对所有出错情况进行异常处理。

图表 6.3‑2 检测模块时序图

## 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，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：

（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——通过平移对准物品，停止运动，抬起机械臂到适当高度。

（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；如果抓取失败，再次尝试抓取，或者超过最大尝试次数，报告失败。

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。物品过大、空间位置过高将直接导致抓取失败。

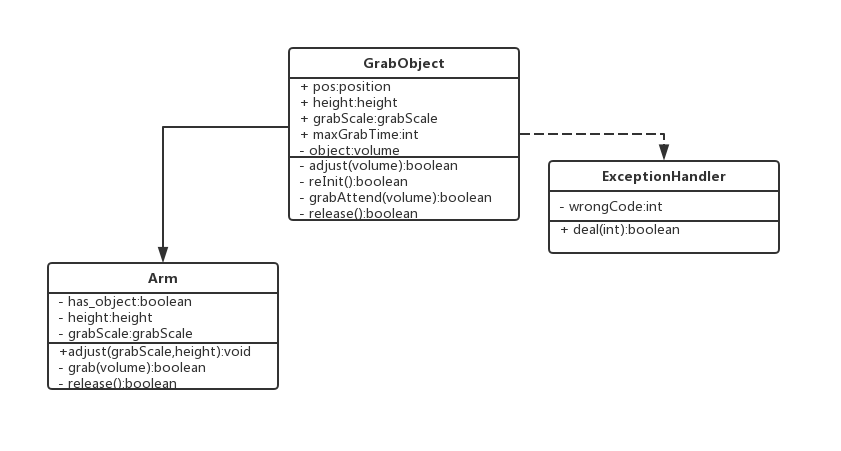
（3）放置物品

机器人到达放置物品的指定地点之后，如有需要会调整机械臂高度，然后张开手爪、放开物品。

这里的放置套路一样是固定的，机器人放置物品的地点是指定的，相当于指定了一个放置的空间位置，机器人会根据该位置来决定松开手爪的空间位置。如果该空间位置下没有桌子等，物品将会直接掉在地上。

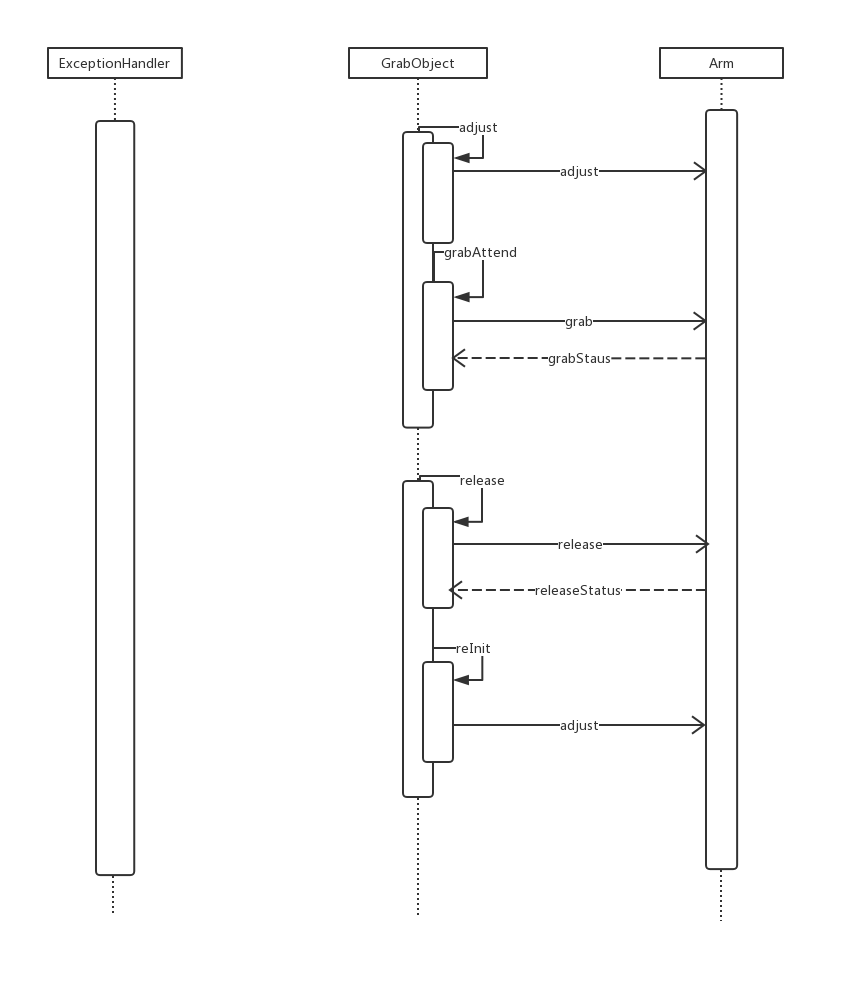
（4）抓取后的调整

完成抓取任务之后，机器人会调整机械臂到抓取之前的状态（也是默认的抓取待命状态）。这保证了机器人每次抓取之前的机械臂状态的一致性。



**图 6.4-1 抓取模块类图**

抓取模块的主类为GrabObject类。adjust方法根据物体的形状大小位置，调用Arm类adjust方法调节机械臂的高度、抓取参数。grabAttend方法不断尝试调用Arm类grab方法，直至grab成功或达到最大抓取数量。release方法调用Arm的release放开抓取物体。抓取的最后可以调用reInit方法回到抓取之前的状态，保持前后一致性。ExceptionHandler处理异常情况。



**图 6.4-1 抓取模块时序图**

## 运动模块

【功能】

运动模块主要负责控制机器人实现基本的运动，主要是平移运动和原地旋转运动。运动模块可以接收位移向量作为输入参数，然后规划机器人的速度和运动时间开始运动。另外运动模块需要接收激光雷达的测距信息，从而对障碍物做出急停反应。

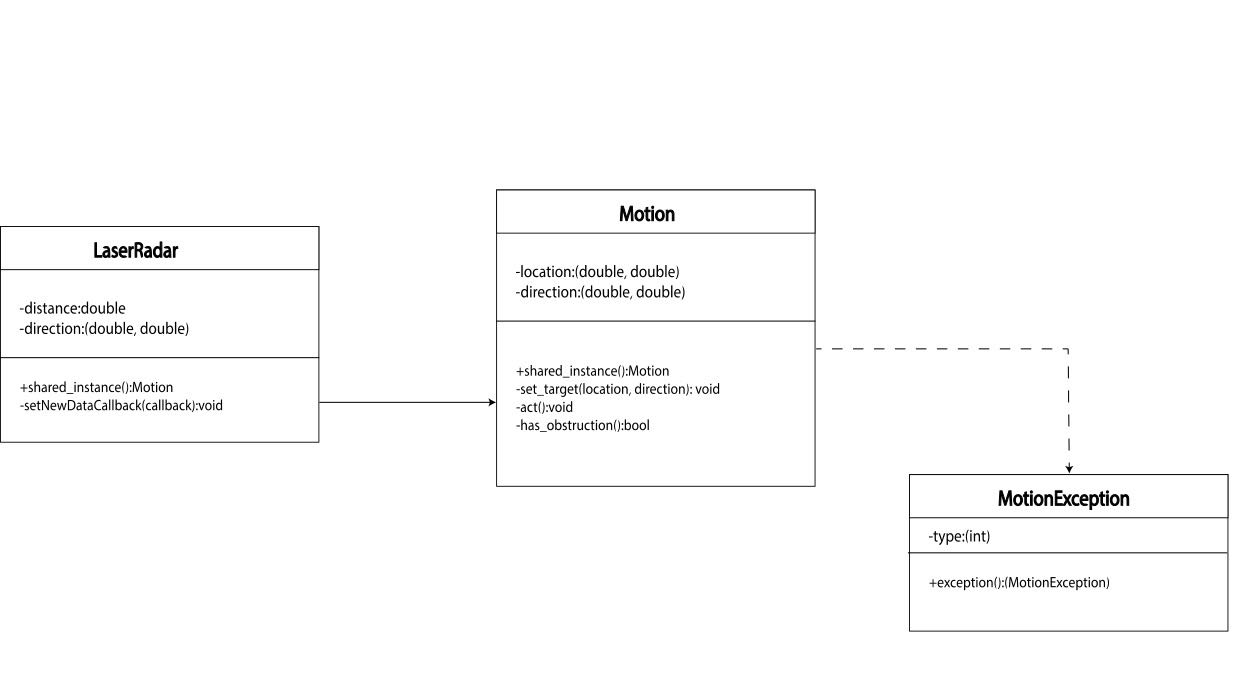
【输入】

位移向量，上层模块根据路径规划等需要，设置位移参数。

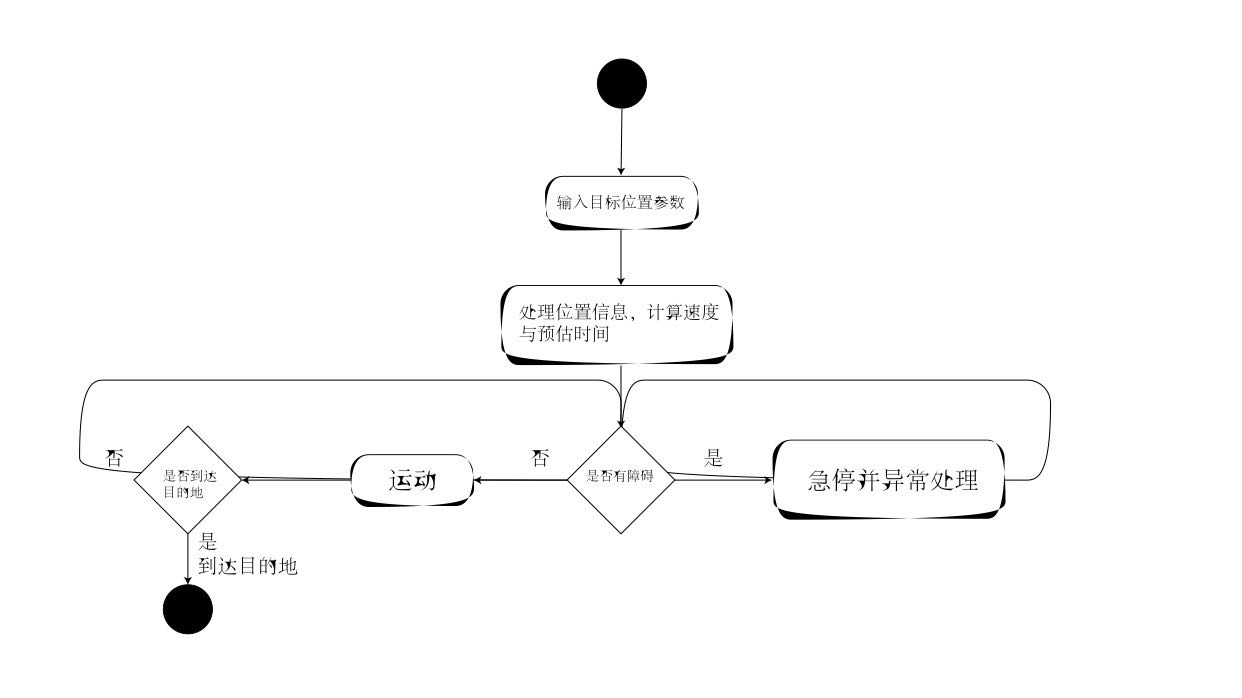
【输出】

运动控制指令，直接调用ros的内置模块。

【设计】

**图 6.5-1 运动模块类图**

模块的主类是Motion，它调用机器人控制系统的方法来操作机器人的运动。同时会通过LaserRadar类来检测运动路径上是否有障碍物，可以通过障碍物距离信息来判断，从而保证安全移动。另外还需要异常处理类MotionException，对运动过程中的异常进行处理。



**图 6.5-2 运动模块顺序图**

**6.6 建图模块**

【功能】

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。

【输入】

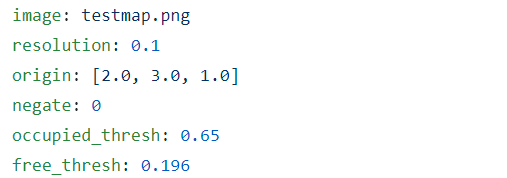
激光雷达扫描结果。

【输出】

当前场景的地图，保存为“map.pgm”和“map.yaml”。其中PGM为可查看的图片格式。YAML文件描述了地图元数据，并命名了图像文件。图像文件对占用数据进行编码。YAML文件包含：

* **image** : YAML文件的相对路径
* **resolution** : 地图的分辨率，单位是m / 像素。
* **origin** : 地图中左下角像素的二维表示，为（x，y，yaw），偏航为逆时针旋转（偏航= 0表示无旋转）。
* **occupied\_thresh** : 像素值高于此阈值被认为是完全占用。
* **free\_thresh** : 像素值低于此阈值被认为是完全空闲。
* **negate** : 是否应该反转白/黑，自由/占用语义（阈值的解释不受影响）

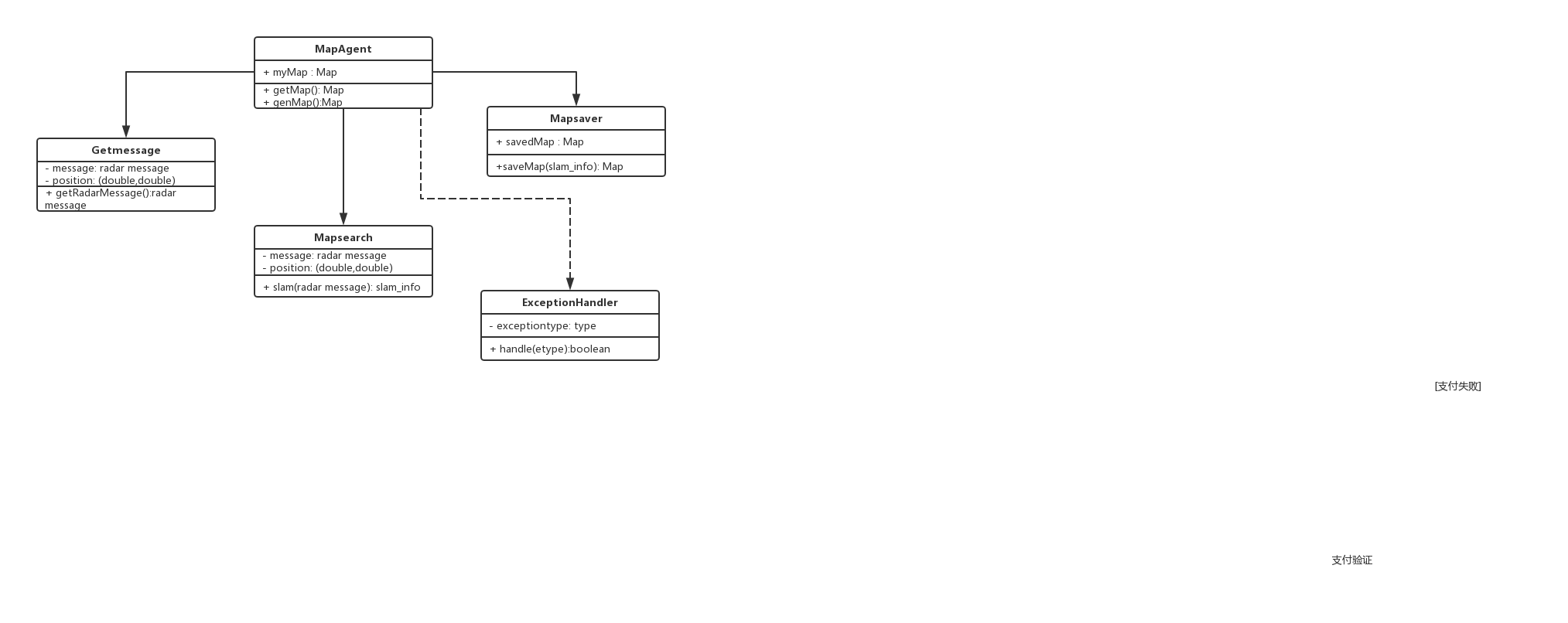
例如：



**【设计】**

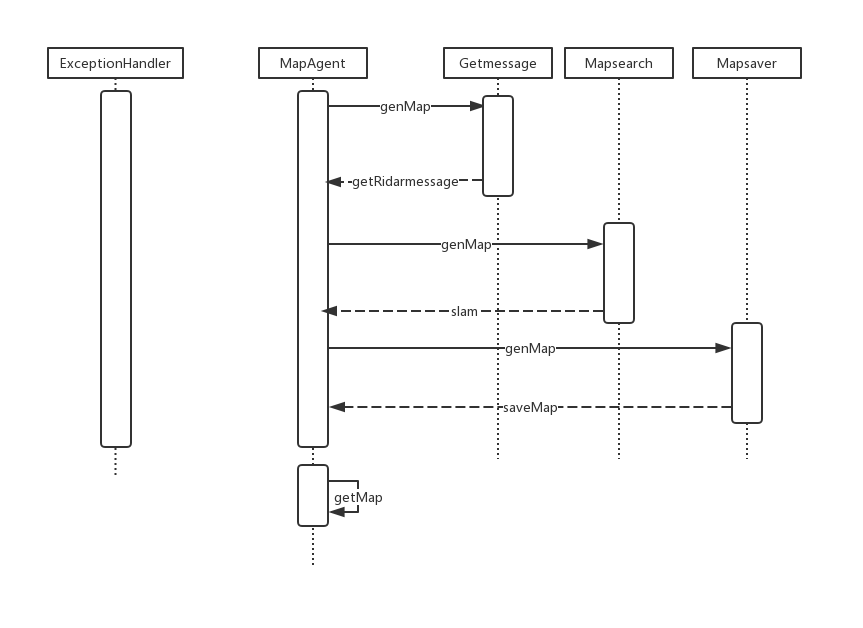
主类是MapAgent类。MapAgent的genMap方法首先调度Getmessage类的getRadarMessage方法获取激光雷达数据，将数据传递给Mapsearch类。Mapsearch类根据Getmessage类传递的数据通过Hector SLAM程序建图，将结果传递给Mapsaver类。Mapsaver使用ROS的map\_server包，将当前 SLAM 建好的图保存为名为“map”的地图。MapAgent的getMap方法获取已保存的地图ExceptionHandler处理异常情况。

类图如下：



**图 6.6-1 建图模块类图**

时序图：



**图 6.6-2 建图模块时序**

**6.7 路径规划模块**

【功能】

规划出一条路径，这条路径从机器人当前点出发，绕开安全边界，到移动目标位置结束。

【输入】

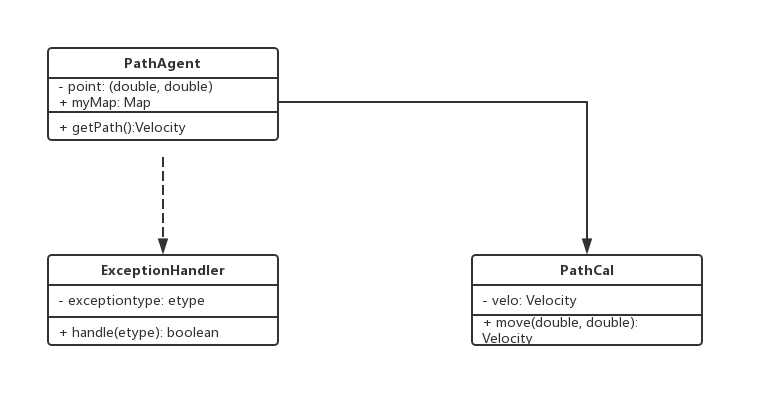
地图与目标位置。

【输出】

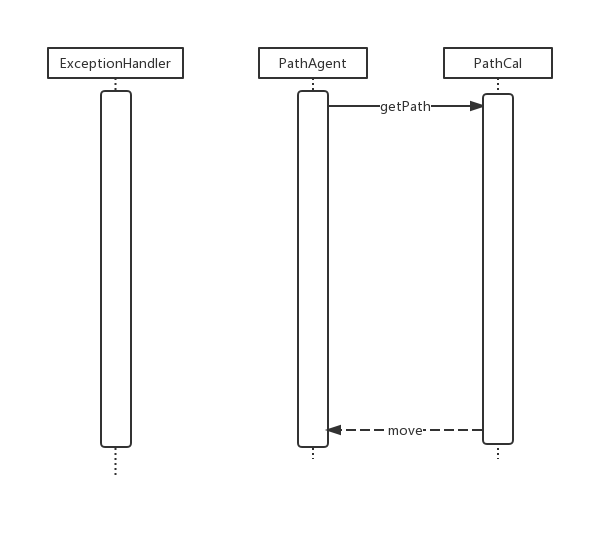
一条从当前位置到目标位置的路径（下个时刻的水平速度，垂直速度和角速度）。

【设计】

主类是PathAgent类。getPath方法调度PathCal类的move方法。move方法调用ROS的move\_base包根据地图和障碍物规划出到目标位置的一条路径。ExceptionHandler处理异常情况。说明：move\_base包已经实现根据实际情况选择根据地图的全局路径规划和根据下一位置是否有障碍物的局部路径规划，因此我们不用实现具体算法。类图如下：

**图 6.7-1 路径规划模块类图**

时序图：



**图 6.7-2 路径规划模块时序图**

# 运行与开发环境

## 运行环境

硬件环境：至少配备以下模块的启智ROS：

* 激光雷达 精准地探测周围的障碍物。
* 机械臂 夹取并运送目标物体。
* 集成编码器的电机 实现运动并获取精准的运动信息
* 摄像头 拍摄周围环境以进行识别
* 机载平板电脑（可选） 运行项目程序、处理传感器信息并发送控制命令。
* Kinect2 3D体感摄影机
* 其他相关的必要设施 如大小均等的轮子、三轮全向地盘、支架、电池与充电器等

软件环境：

* Ubuntu14.04 trusty或Ubuntu16.04 xenial
* ROS indigo 或 ROS kinetic
* IAI-Kinect2
* RPLidar激光雷达驱动程序
* 科大讯飞语音识别
* 以上软件包所依赖的必要运行库如python、X11等

## 开发环境

硬件环境：

* AMD64架构的计算机 用于编译运行项目源代码
* 其它任何可配备键盘或虚拟键盘等输入设备的计算机 用于编写代码与文档
* 无线路由器、网线 用以提供必要的网络环境与远程操控的基础
* 运行环境的必要硬件
* 其他相关的必要设施

软件环境：

* 运行环境的必要软件
* RoboWare Studio
* Rviz（ROS Visualization）
* 开发所依赖的各种运行库dev版的软件包

# 需求可追踪性说明

## 功能性需求

### 地图初始化

针对SRS文档涉及的地图初始化功能，本设计中借助机器人的激光雷达进行扫描、使用Hector SLAM算法进行建图。

激光雷达会将扫描结果保存到特定文件，通过提取该文件并使用Hector SLAM算法进行处理，我们就能得到地图信息；随后经过使用激光雷达来对机器人位置进行校准之后，机器人就得到了准确可用的地图、站在了地图的可知位置。

### 路径规划

针对SRS文档涉及的路径规划功能，由于机器人的使用场景对机器人的限制较宽松（空间较大，且不苛求执行效率），本设计中直接采用了ROS的Navigation模块。

Navigation模块主要包含定位和导航两个方面的功能。定位功能通过蒙特卡洛自适应定位算法实现，使用概率理论对机器人位置进行估计；导航功能通过move\_base包实现，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，同时还提供了导航参数的设置接口。

### 目标物检测抓取

针对SRS文档涉及的目标物检测抓取功能，本设计直接采用了ROS机器人的抓取模块，但是对检测模块做了一些补充。

本设计中，物品检测识别主要是通过Kinect2模块和PCL的平面检测算法实现。Kinect2模块使用Kinect2视觉传感器获取外界情形信息并以三维点云的形式存储，分析模块使用PCL的平面检测算法对三维点云进行分析，得到独立点云团簇，每个点云团簇视为一个物品。

本设计中，考虑到特定情况下，物品的空间位置信息可以通过其他形式补全，因此还引入了更为简洁的、二维层次上的图像识别技术来对前述三维检测算法形成补充。

### 用户交互

针对SRS文档中涉及的用户交互功能，由于场景需要，本设计中除了界面交互以外，还重点实现了机器人的语音交互功能。

机器人的语音交互功能包括语音识别和语音播报，本设计中以上两个功能都通过使用科大讯飞的语音模块实现。其中，实现语音播报功能可以直接使用科大讯飞的语音播报模块；实现语音识别功能，在使用科大讯飞的语音识别模块的基础上，还要定制个性化关键词，并在科大讯飞官网完成新的语音模型的训练。

## 非功能性需求

### 系统性能

机器人的通过语音播报或界面提示，对用户进行反馈。

### 系统可靠性

系统采用类似微内核的设计理念，外挂简单可靠的监测式守护进程，发现主程序响应时间不达标立刻采取紧急制动处理。

### 系统易用性

对用户的操作技能要求较低，路径规划模块允许地图校准存在一定误差；支持界面交互和语音交互，支持语音识别的关键词定制，提供更多的交互自由；使用关键词匹配的形式进行语音识别，降低用户交互难度；提示信息丰富，操作难度低。