**【餐厅服务机器人】**

**软件设计说明书**

**【*SDD Team 104*】**

**【1.1】**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 机器人，你在干神魔？ | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 17373175 | 郑耀彦 | 参与4、5小节编写 |
| 17373005 | 黄月姣 | 参与3小节编写 |
| 17373510 | 王蒙 | 参与1、2、4、6、7小节编写 |
| 17373515 | 李任宇 | 参与1、2、4、6、7小节编写 |
| 17373173 | 叶泽峰 | 参与8小节编写 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0 | 2020.04.20 | 李任宇 | 叶泽峰 | 软件设计说明书初稿 |
| 2.0 | 2020.05.03 | 李任宇 | 叶泽峰 | 软件设计说明书二稿（详细设计） |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1 范围

1.1    项目概述

本项目拟在ros启智机器人的基础上开发一个能够为顾客运送产品的餐厅服务机器人。支持机器人自动行驶运行与用户手动操控运行，并提供语音控制等辅助服务帮助用户使用，机器人具体的运行功能如下。

主要功能：

1、机器人的主动控制

2、静态或动态障碍物避障

3、机器人利用传感器实时建立环境地图

4、机器人根据地图和自身的位置信息实现动态路径规划及导航控制

5、检测、识别并定位环境中的特定目标，动态接近目标物

6、抓取目标物

非功能性需求：语音控制、运行噪音小、能上坡

应用场景：餐厅服务、医疗服务

1.2    文档概述

本文档是《机器人，你在干神魔？》小组的软件设计说明书，项目内容组织如下所示，其中包括：需求概述、数据库设计、体系结构设计、接口设计、详细设计运行与开发环境和需求可追踪性说明。

1.3    术语和缩略词

|  |  |
| --- | --- |
| 缩略词 | 全称 |
| ROS | Robot Operating System |
| L | Low |
| M | Moderate |
| H | High |

1.4    引用文档

启智ROS版\_开发手册\_20181109

启智ROS机器人

2.需求概述

2.1 业务需求描述

本项目旨在设计一款用于餐饮行业辅助运送菜品的机器人，主要功能为帮助将一定量的物品自动从一个地点运送到指定地点，或者通过用户手动操控或者语音控制完成这一过程。（注：目前仅支持运送单个货物，多个货物的扩展工作可能会在后续版本中逐步完成）

初次使用或者重置系统后需要对机器人进行相关的配置，用户可手推或手柄控制机器人遍历业务实地场景，使用激光雷达及相关SLAM算法对地图进行建模并保存。可以保存常用起点、与固定地点在内存中简化后续操作。

用户使用机器人时，有三种操控方式。用户可以发出语音指令指定机器人回到地点、检测抓取饮料操作。执行回到地点操作时，机器人从当前位置自动规划路径回到规定地点。也可以通过手机或电脑连接机器人通过我们提供的方向控制板控制机器人前进或者通过填写行程表让机器人按照此表自动规划路线。

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。

2.2 数据需求

用户通过交互窗口注册或者登录账号，验证身份后可以通过交互窗口或者语音输入对机器人下达任务指令。机器人在用户验证身份后启动，采集周围环境的数据，根据用户下达的任务指令进行相应的操作。

数据范围描述：用户个人用户名、密码、id等基本信息；机器人收集的环境信息；用户指定的任务信息；用户的语音信息。

2.3 非功能需求

1. 性能指标

响应时间：

主要组成部分为指令识别时间、系统处理时间和指令执行时间。指令识别与系统处理规划应在10s内完成；执行时间根据具体路径和机器人速度而定。

功耗：

根据硬件实际功耗而定。

处理能力：

可以处理简单的语音指令。

可以在地形条件下完成较优的路径规划。

可以自主判断、躲避障碍并重新规划路线。

2. 质量属性

系统可用性：

可以实现物品抓取、运送等功能。

可以实现人工设置目标与简单的语音控制。

可移植性：

该机器人针对ROS系统。

完整性：

应实现的完整功能包括物品抓取与放下、自主路径规划、简单的语音识别、人工指令设置和避障功能。

效率：

根据实际算法效率和硬件效率而定。

2.4 功能需求

2.4.1 用户手动操控

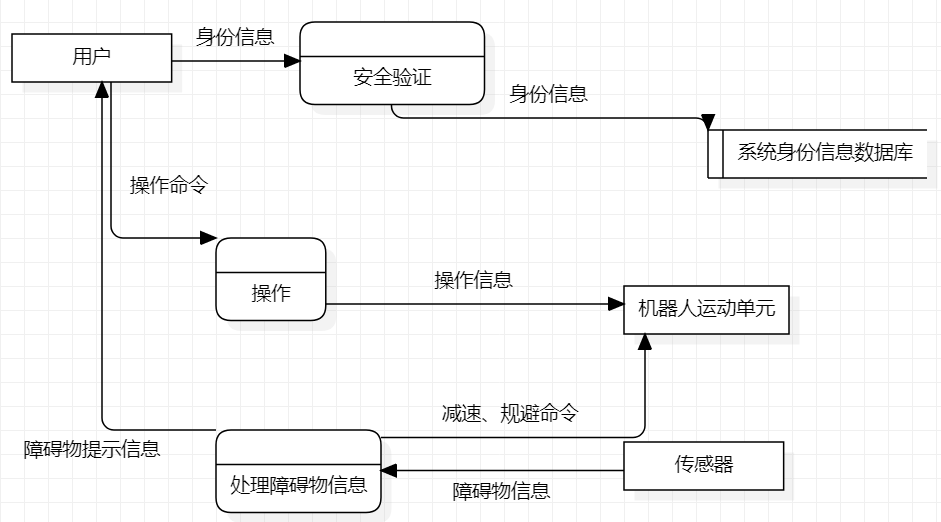


图1 手动操控数据流图

2.4.2 机器人自动运输

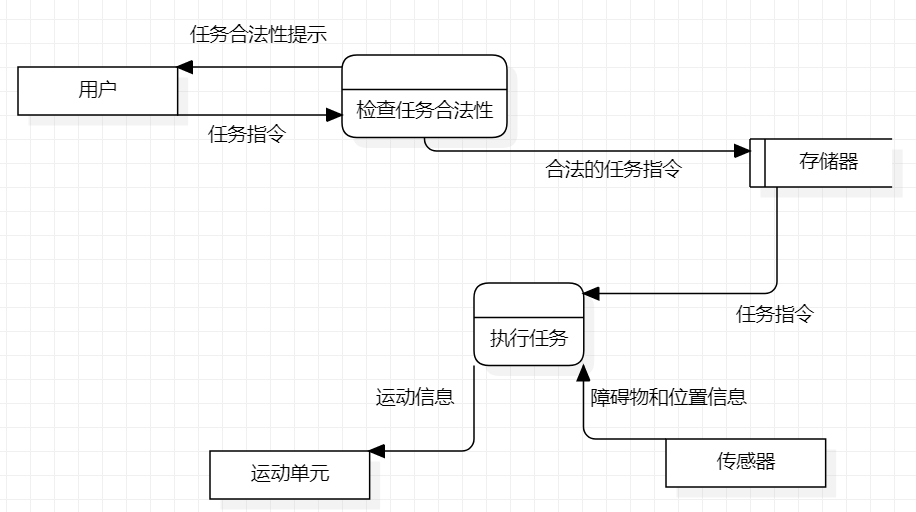


图2 自动运输数据流图

1. 数据库设计

本系统数据库包含地图数据库和用户数据库。各数据库/单元关系如下：

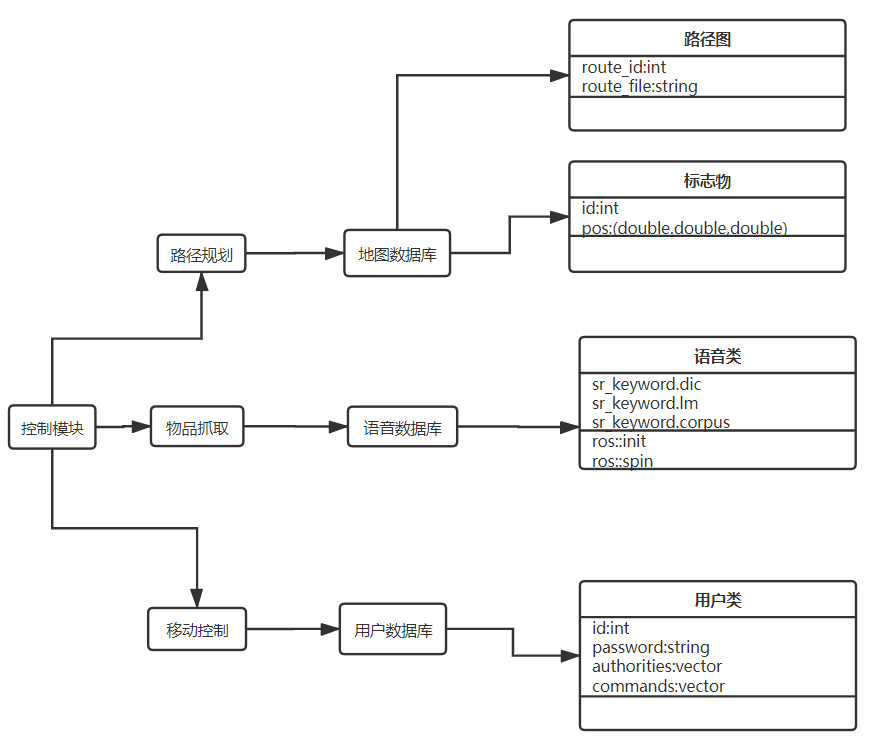


图3 数据库类图

3.1 地图数据库

地图数据库包含了地图信息，受依赖于路径规划模块。地图类的信息包括地图标识以及地图文件索引。地图类内包含的信息有路径和标志物信息。路径包含地图路径信息；标志物为建立地图信息时记录的特殊位置信息，例如餐桌等。

3.2 用户数据库

用户数据库管理用户信息，受依赖于控制模块。用户在通过验证后方可使用机器人执行命令。用户信息包括用户标识、验证密码、用户权限以及用户命令。用户通过标识与密码验证身份通过后，在自身权限内对机器人发出命令。

3.3 语音指令数据库

语音数据库包含语音识别模块中的关键词信息以及关键词与控制指令的键值关系，对应关系如下。用户可以通过关键词控制机器人完成一些特定指令。

|  |  |
| --- | --- |
| 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*** | 机器人停止运动 |
| ***Master*** | 用户所在位置 |
| ***Place N*** | 地点关键词 |
| ***Follow*** | 机器人跟随管理员 |
| ***Stop Following*** | 机器人停止建图跟随 |

1. 体系结构设计

4.1.总体结构

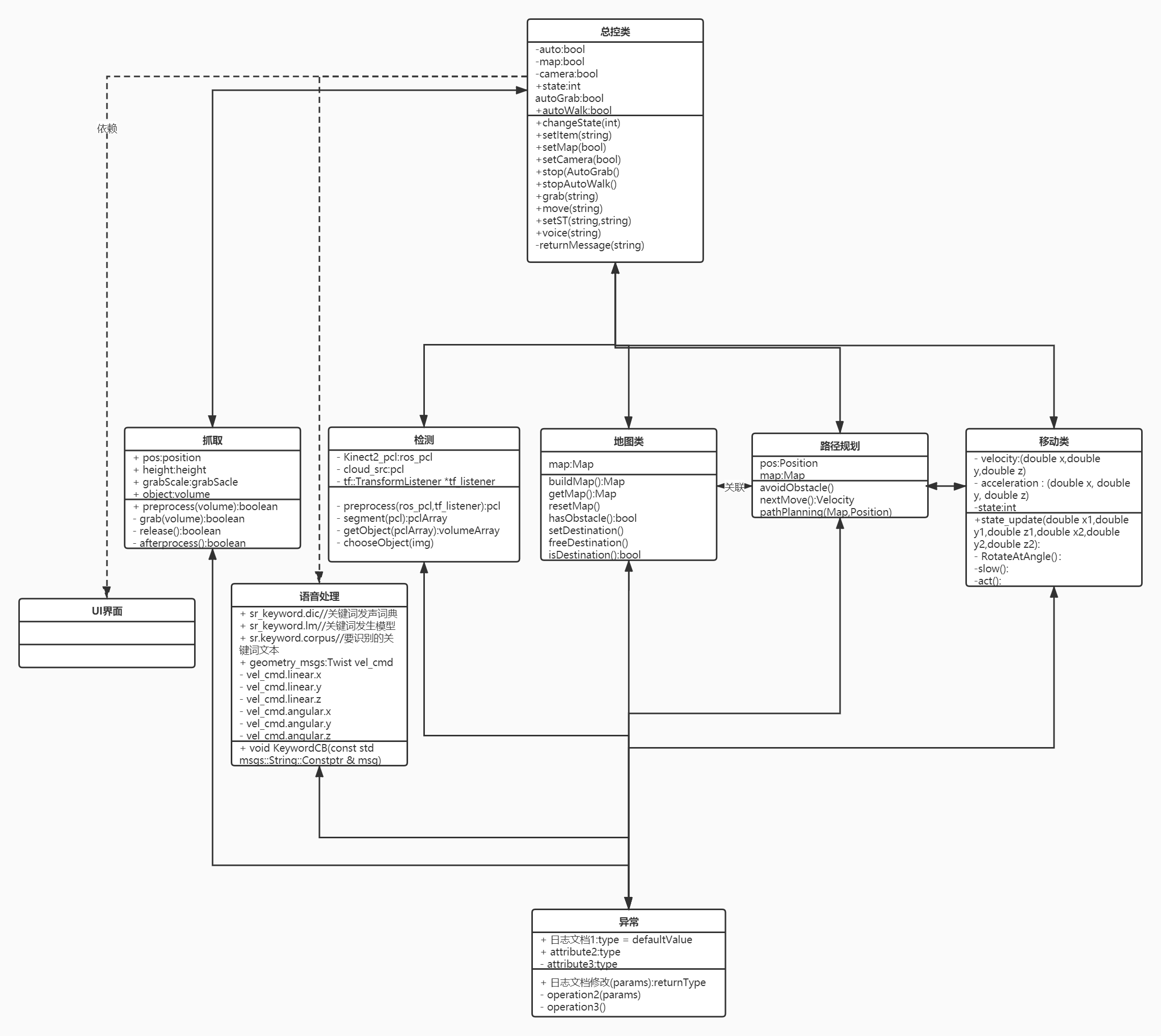


图4 总体结构图

系统的核心单元为总控模块，以及移动控制、地图管理、UI界面、语音识别、目标检测、目标抓取、自主路径规划多种功能的控制模块。附带异常处理模块。

UI界面模块获取来自机器人控制面板和Internet端的指令，向机器人总控模块发送消息。机器人控制面板需具备显示功能；Internet'端需要设置交互界面并具备处理语音和文本输入的功能。

机器人总控模块监听其他模块，解析指令，按需激活功能控制模块。

移动控制、地图管理、UI界面、语音识别、目标检测、目标抓取、自主路径规划作为控制模块调用硬件控制模块和传感器事件管理模块完成任务。硬件控制模块需具备控制机械臂和底盘的功能，同时可以接受手柄的操控；传感器事件管理模块通过调度器来接收各传感器的状态及信息、与事件相关联。同时这些功能需要具备一定的错误处理能力。

4.2.硬件体系结构

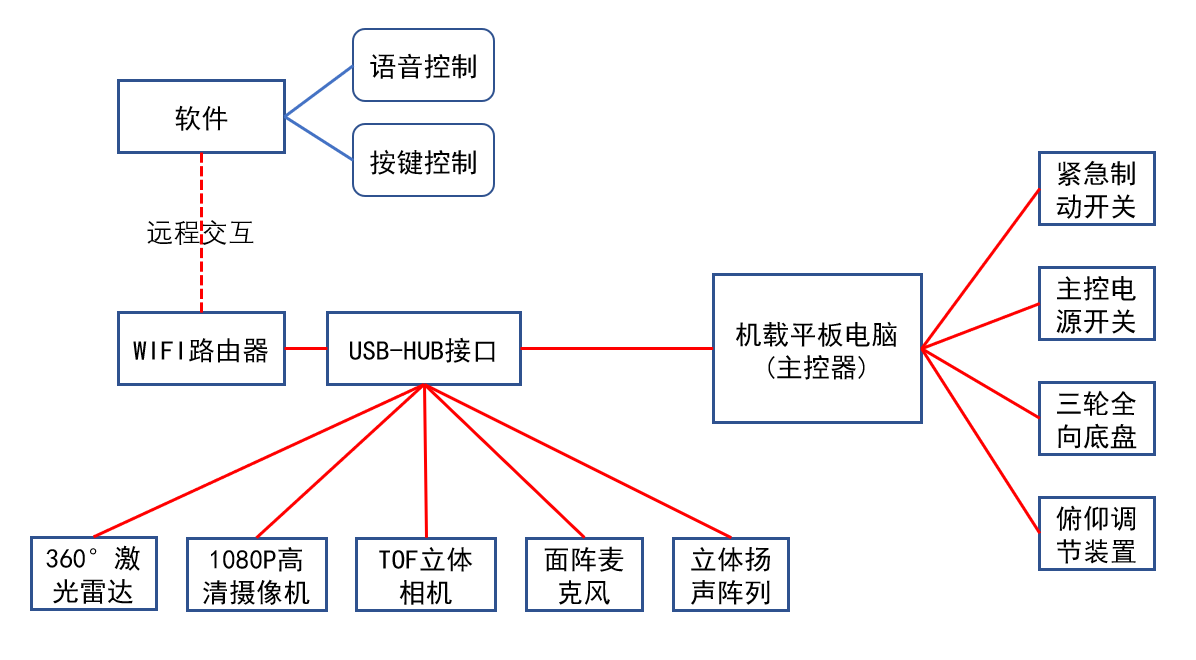


图5 硬件体系结构图

硬件体系结构主要考虑机器人自身的各个组件之间的关系和软件之间的远程交互。

用户的语音控制和按键控制可以通过远程交互控制机器人的开关和行为；同时机器人的雷达或深度相机可以构建出地图并识别障碍物并将信息传至软件界面，摄像头和麦克风捕捉到的环境情况可以通传达至用户。

1. 接口设计

5.1.用户界面设计

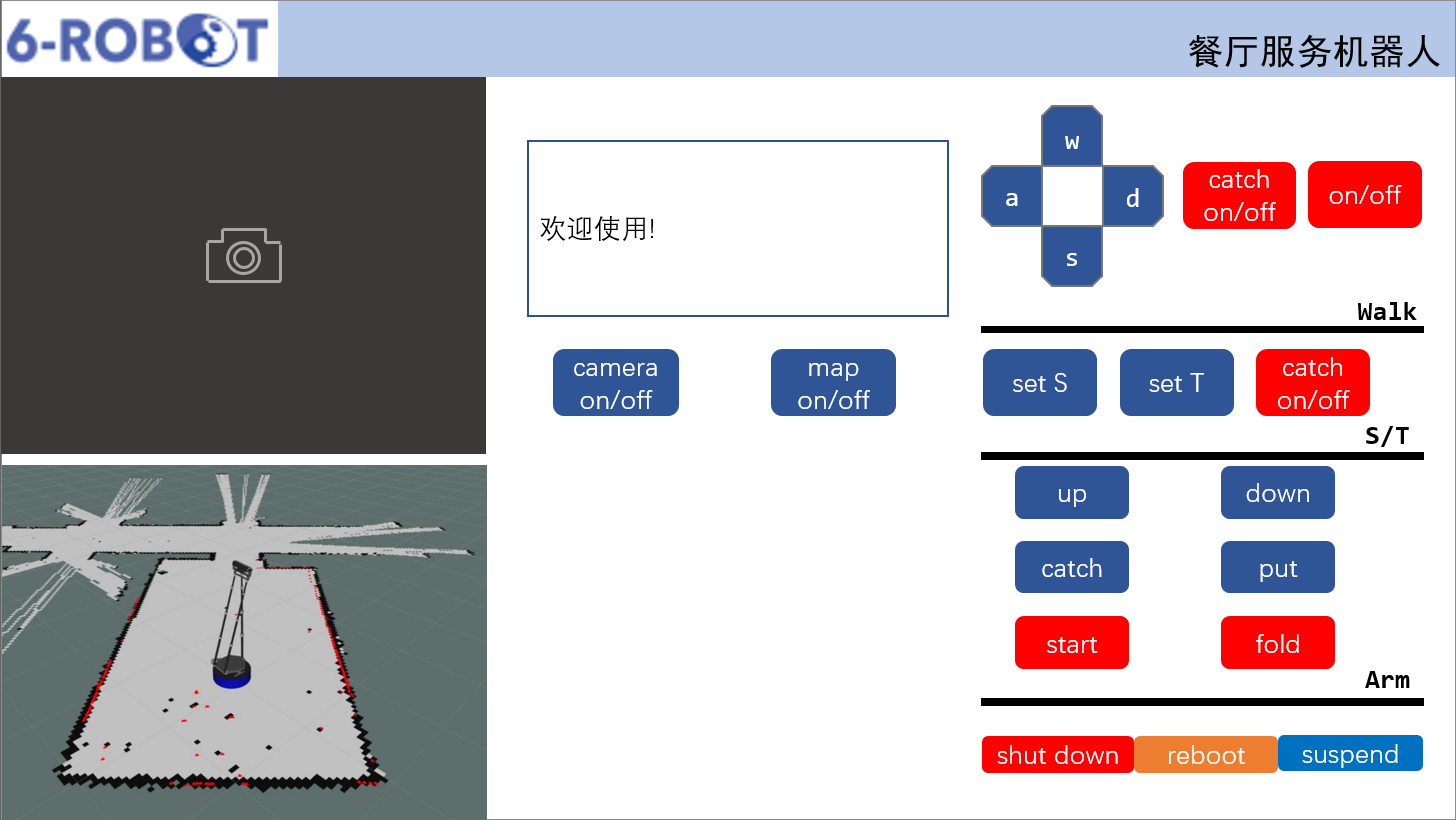


图6 用户界面示意图

如图，用户界面左部为机器人高清相机传来的画面和实时地图界面。界面中部为状态框和相机、地图的开关。界面右部为机器人行走、设置自动寻路、机械臂控制、总开关等按钮。

5.2.硬件接口

1、360°激光雷达：

可捕捉360°的深度信息，并使用SLAM算法构建地图以及识别障碍物，并将信息传递至用户界面和运动控制模块、障碍回避模块、地图管理模块、路径规划模块。

2、底盘：

可捕捉机器人的实时姿态，包括朝向、倾角、速度等信息，并将信息传递至用户界面和运动控制模块、障碍回避模块。

3、1080P高清摄像头：

可实时捕捉机器人所朝向的画面，并将画面传至用户界面。

4、TOF摄像头：

可以识别目标物体并将信息传至目标特征管理模块，帮助机器人识别并抓取目标物体。

5、面阵麦克风、立体扬声阵列：

可以使机器人所在环境和用户终端产生语音交互，帮助用户了解机器人所处环境。

6、USB接口：

连接各个硬件接口和网卡，与用户进行远程交互。

1. 详细设计

6.1．UI界面功能解释

此部分主要作为解释UI界面所提供的功能进行详细说明。现界面提供有服务自主路径规划功能、手动控制功能与识别与抓取功能。

界面图像：UI界面简略图在见第5节接口设计5.1节图像展示。

交互分析：

1、手动操控功能顺序图

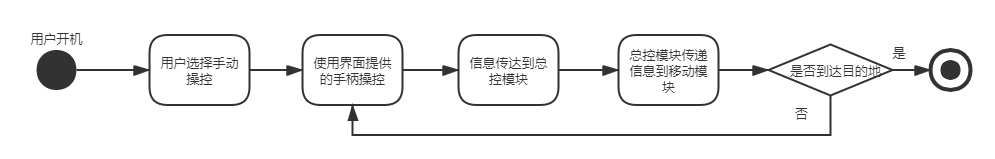


图7 手动操控功能顺序图

用户通过操控盘(包含方向盘、转向盘、速度控制盘)输入的信息会投递到总控模块类，信息标记为手动控制，总控模块将信息按照功能预定路线传递信息到下属模块类。总控模块类与移动模块类内部工作详情见后。

2、路径规划功能顺序图

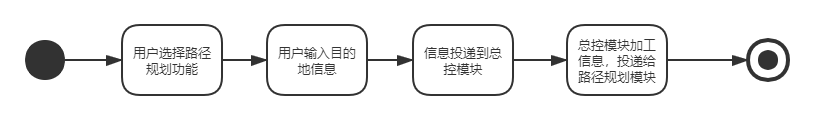


图8 路径规划功能顺序图

用户选择路径规划功能后，弹出对话框指定目的地位置，信息标记为路径规划交付给总控模块加工处理，总控模块处理后交付给路径规划模块类控制运动。总控模块类与路径规划模块类内部工作详情见后。

3、识别/抓取功能顺序图



图9 识别抓取顺序图

用户选择识别/抓取功能后，信息标记为识别/抓取投递到总控模块类，总控模块传递给识别/抓取模块类自动进行后续操作。识别/抓取内部工作细节设计见后。

6.2 移动控制模块

输入：移动速度、移动方向

输出：控制底盘运动的信息指令

功能：控制机器人的运动，使机器人可以按照所给输入速度和方向进行移动。当检测到有障碍物在指定距离时，会降低速度。类图如下:

movement为移动类，Pathmanager为路径规划模块，exceptionhandler为异常处理模块，自动控制功能下movement从Pathmanager模块接收信息，手动控制功能下movement从controller模块接收信息，异常信息投递给exceptionhandler模块。

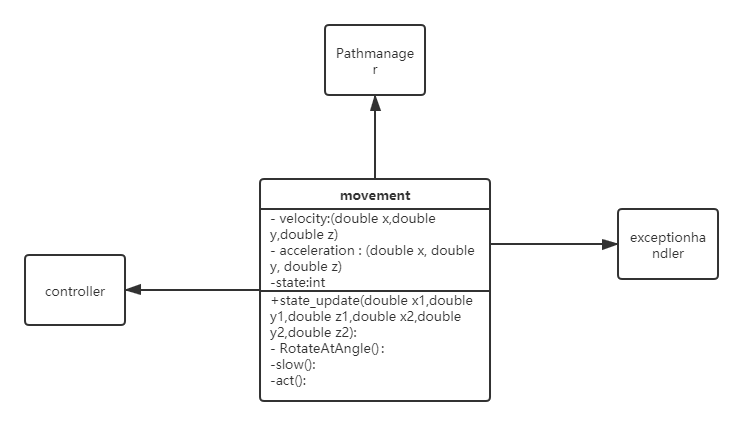


图10 运动控制模块类图

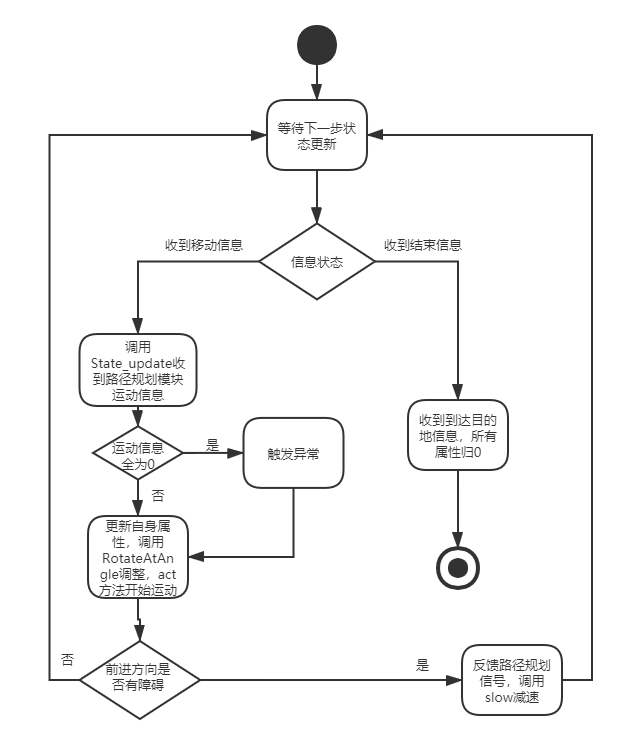


图11 movement内部工作顺序图

其中信息状态为state属性，1为收到移动信息，0为收到结束信息。当模块执行手动控制功能时(即收到信息来自总控模块)，收到路径规划模块发送信息会出发异常，反之亦然。

模块具体属性与方法详细说明如下：

velocity : (double x, double y, double z) 表示机器人当前速度, x表平移前进速度，负值代表后退；y表示平移向左运动，负值代表向右运动；z 代表机器人自转速度，正值左转，负值右转；x,y的单位为”米/秒”，z的单位为“弧度/秒”

acceleration : (double x, double y, double z) 表示机器人当前加速度，对应项与velocity相同

State\_update() : (double x1,double y1,double z1,double x2,double y2,double z2) 用于得到机器人的下一步的状态信息，x1,y1,z1代表velocity的属性值，x2,y2,z2代表acceleration属性值。

State: int 记录当前状态，0为停止运动状态，1为运动状态。在运动状态下， 如果状态更新时收到的速度、加速度全为0，判定为异常。

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。

Slow():当机器人运行前方有障碍物时，减缓速度

Act():按现有运动状态信息执行机器人运动

* 1. 地图管理模块

功能：记录地图建模信息并保存；给路径规划模块提供地图信息。

输入：激光雷达扫描结果。

输出：当前场景地图信息。

说明：当初次使用或重置系统后需要对机器人进行相关的配置，用户通过手推或手柄控制机器人遍历实地场景，使用激光雷达及相关SLAM算法对地图进行建模并保存。当地图建模完成后，提供自主路径规划模块查询地图的接口。

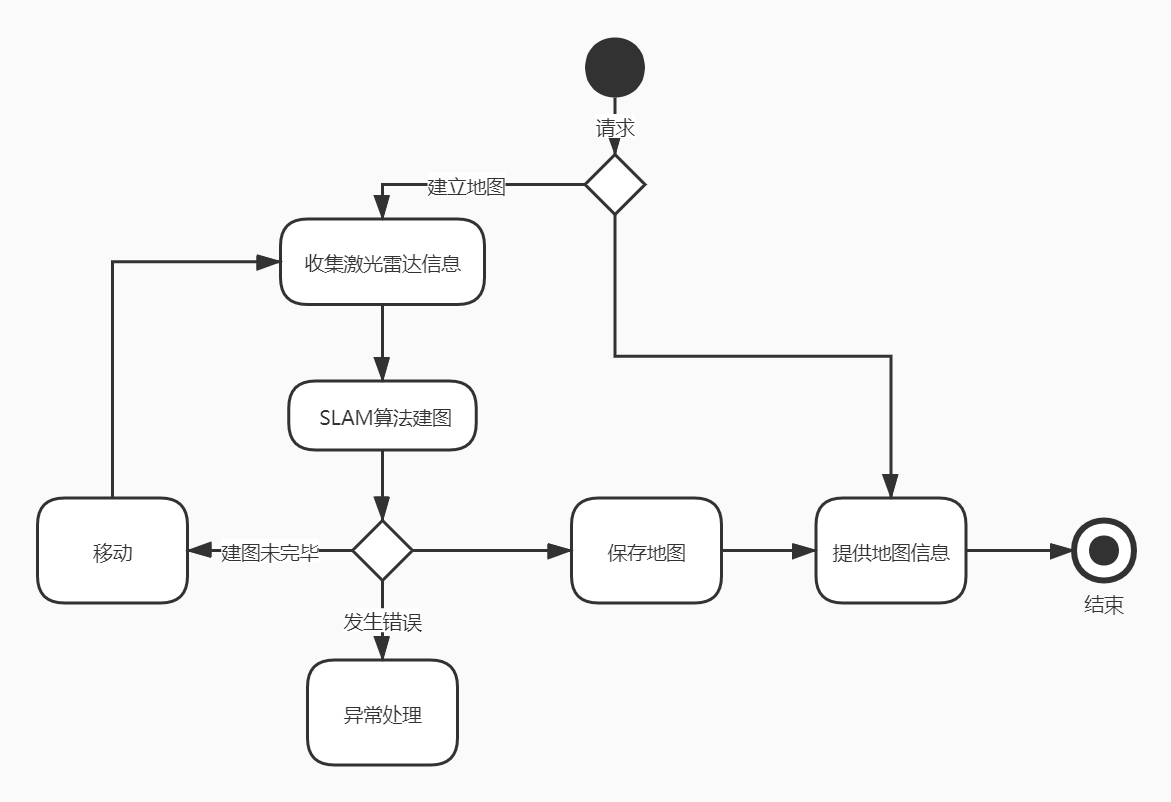


图12 地图管理模块流程图

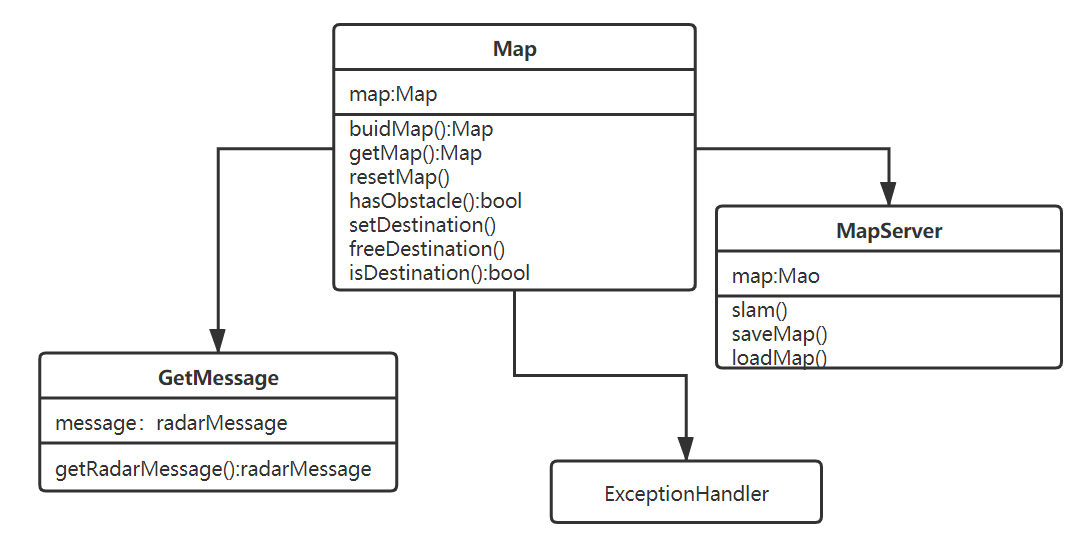


图13 地图管理模块类图

地图管理模块主类为Map，负责管理和提供地图信息。Map的buildMap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MapServer类。MapServer类根据Getmessage类传递的数据通过Hector SLAM程序建图，并使用ROS的map\_server包，将当前 SLAM 建好的图保存为名为“map”的地图。Map的getMap方法获取已保存的地图。ExceptionHandler处理异常情况。

* 1. 自主路径规划模块

输入:地图文件、机器人起点位置(没有输入则默认为当前位置)与终点目标位置。

输出: 机器人下一时刻的移动参数（方向、速度信息）。

功能：当依据机器人已经建立的地图数据库信息，根据用户设置的起点与终点自动规划最短且不会碰撞障碍物的路径。

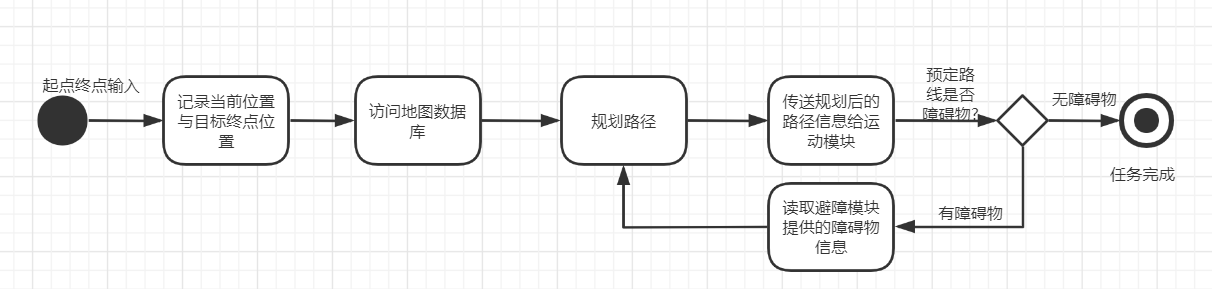


图14 自主路径规划模块流程图

实现过程：路径规划类的getPath()方法调用ROS的move\_base包。move\_base包通过订阅机器人位置信息、地图障碍等数据，规划出全局和局部路径，再将路径转化为机器人的速度信息。路径规划类再将速度信息提供给移动类以实现导航。其间需要接收传感器消息防止碰撞障碍物并调用异常处理类处理异常情况。类图如下：

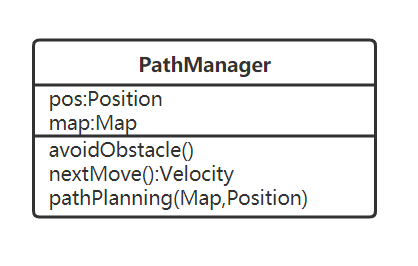


图15 路径规划类图

其中，pos和map属性记录当前位置和地图信息。avoidObstacle方法接收移动类的障碍物信息，若有障碍物则重新规划路径；nextMove方法调用pathPlanning提供下一次移动需要的速度信息。

* 1. 目标检测模块

功能：记录目标特征信息，并给目标抓取模块提供平面检测结果、物体检测结果等信息

设计：

目标检测功能的实现流程如下：

1.检测预准备

创建用于坐标转换的tf\_listenner；订阅Kinect2的点云数据；发布名为“obj\_marker”的主题，用于在Rviz中标注物品的空间位置；发布名为“segmented\_plane”的主题，用于在Rviz中显示检测出的平面的点云集合；发布名为“segmented\_objects”的主题，用于在Rviz中显示检测出的物品的点云集合。

此后调用ros::spin()挂起主线程，通过回调函数对相应数据进行分析处理

2.预处理

预处理包括点云坐标转换和点云格式数据转换。为了处理方便，使用tf\_listener将全部点云坐标转换到以机器人在地面投影中心为原点的坐标系；为了更好的操作点云，将ROS格式数据转换到了PCL格式数据cloud\_src

3.点云分析检测

使用PCL的分割对象segmentation将初始点云中的水平平面检测出来，平面的标号存储来planelndices->indices数组中，再通过while循环遍历平面，找出其高度符合我们要求的平面作为桌面。再将识别出的桌面点云从点云中剔除，并将标准平面上方一定距离之内的点云分离，作为物品点云集合，再使用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

对分割出来的每个物体进行体积统计，调用DrawBox()绘制其外接矩形，调用DrawText()在其上方显示物品标号

5.发布检测结果

通过步骤一发布的主题来完成，其他模块可以通过对应主题获取所需信息

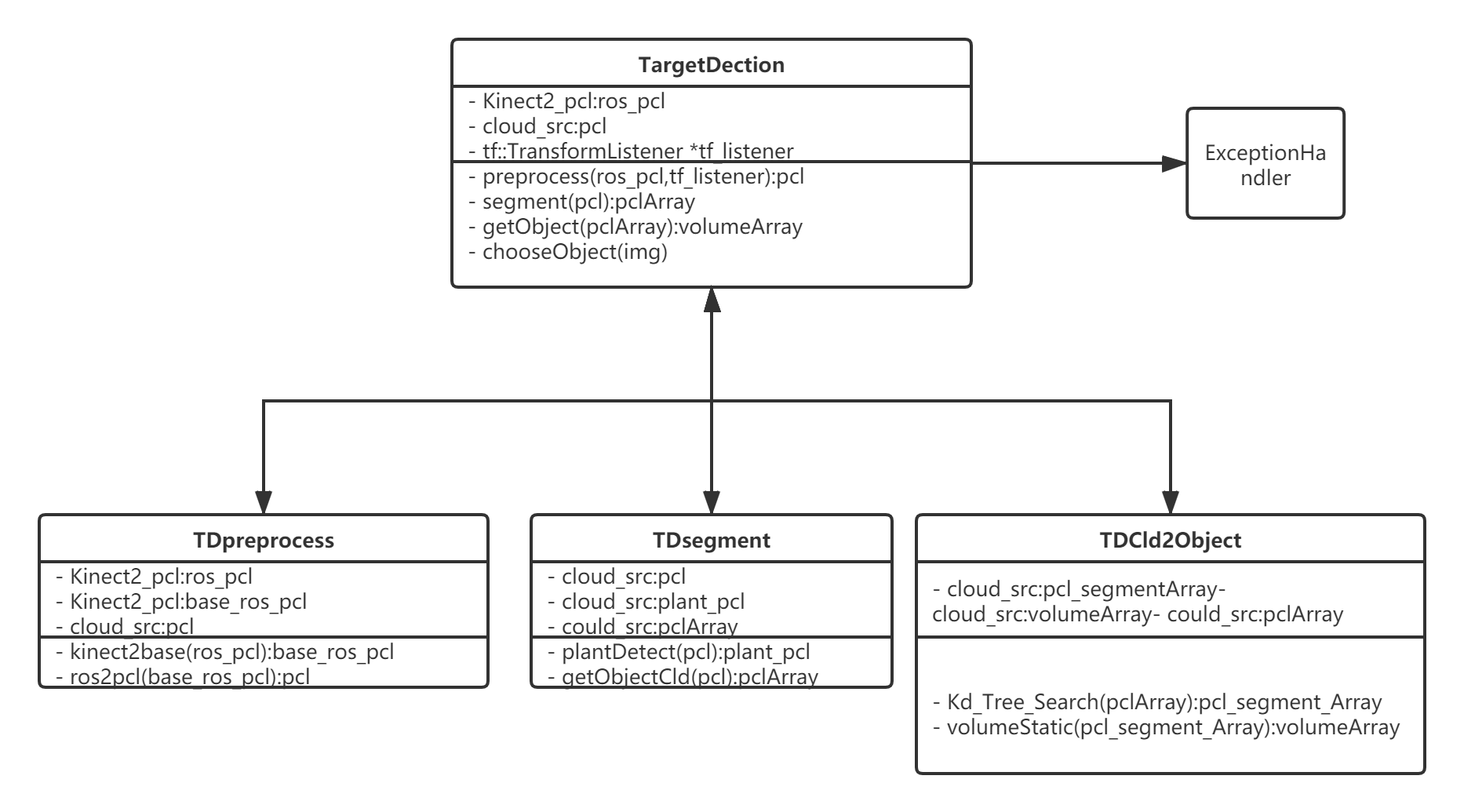


图16 目标检测模块类图

类图说明：主类TargetDection用来实现检测模块的功能。主类的preprocess方法将创建TDprocess对象，进行初始化操作，TDprocess对象先通过kinect2base方法对点云数据进行坐标转换，再通过ros2pcl方法进行点云数据格式转换；主类的segment方法将创建TDSegment对象，其通过plantDetect方法得到平面点云，之后利用getObjectCld方法筛去平面点云得到平面上方物体点云集合；主类的getObject方法将创建TDCld2Object对象，其通过getObject方法使用Kd-Tree方法对剩余点云数据进行近邻搜索查找，得到物体点云集合，之后调用voluneStatic方法完成从点云团簇到物品的转换，得到物品的形状、体积。

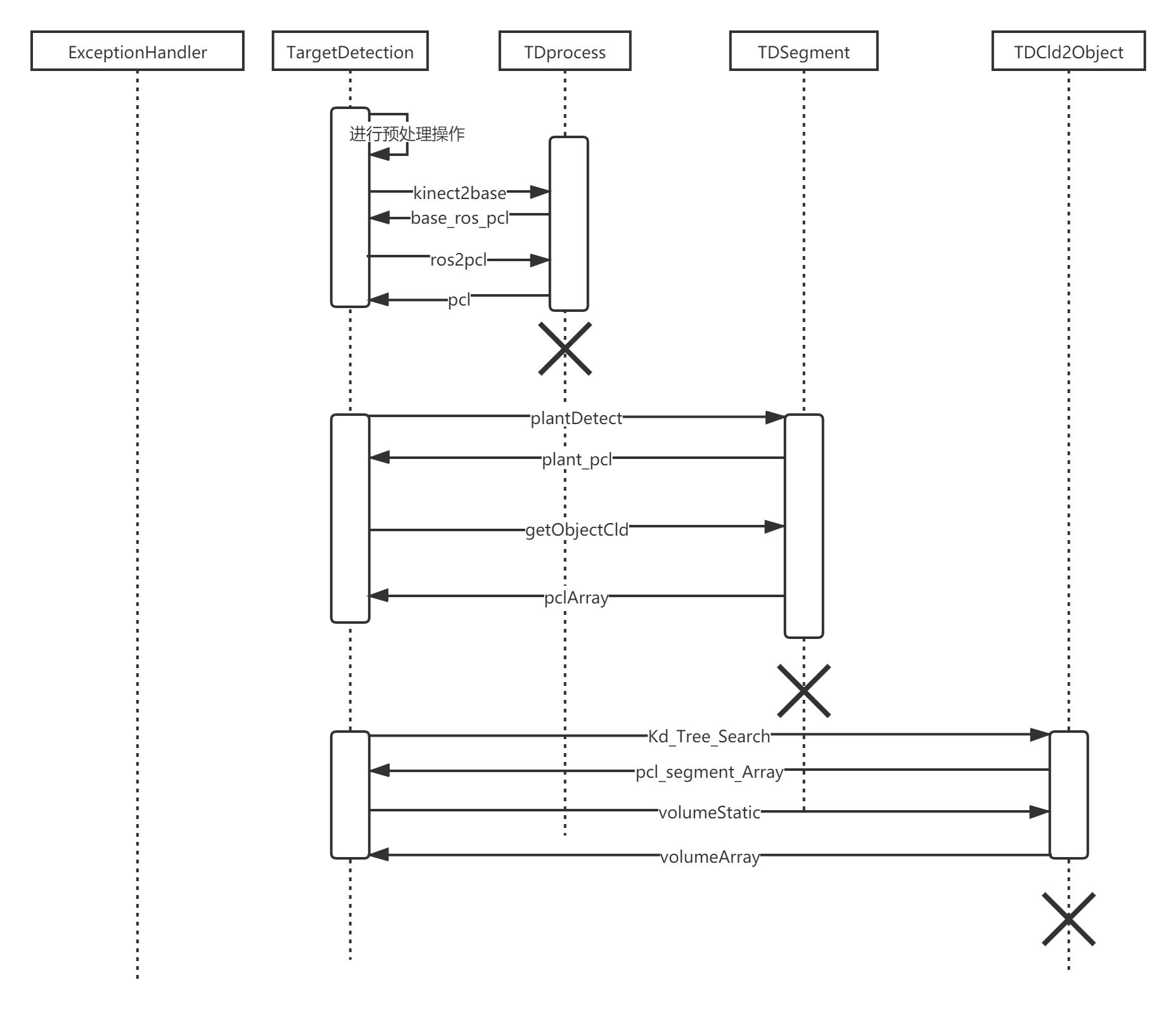


图17 目标检测功能时序图

* 1. 目标抓取模块

输入: 检测模块的平面识别以及物体检测结果/总控模块的控制信息

输出: 抓取是否成功的结果/无

功能：

(1)自动目标抓取:抓取模块利用检测模块的得到的结果，包括平面检测结果、物体检测结果。之后进行抓取操作、放回操作。在调用抓取模块之前需要进行一系列的适配操作，最主要的是调节机械臂参数。

(2)手动目标抓取:抓取模块根据总控模块给的指令进行移动、抓取、放置等操作。

自动目标抓取功能的实现流程如以下：

（1）抓取准备

在获得平面检测结果以及物体检测后，机器人调整与平面以及物体的距离和方向，进入最佳抓取状态。

（2）抓取物品

机器人伸出手臂，控制手爪闭合宽度以抓取物体，完成抓取任务。

（3）放置物品

机器人到达放置物品的指定地点之后，张开手爪，放开物品。

（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。这保证了机器人每次抓取之前的机械臂状态的一致性。

若在自动抓取过程中总控中心发出手动控制指令，则发生异常，暂停抓取，将错误发送给总控中心。

手动目标抓取功能的实现流程如下:

机械臂根据总控制类的指令进行移动、抓取、放置等功能

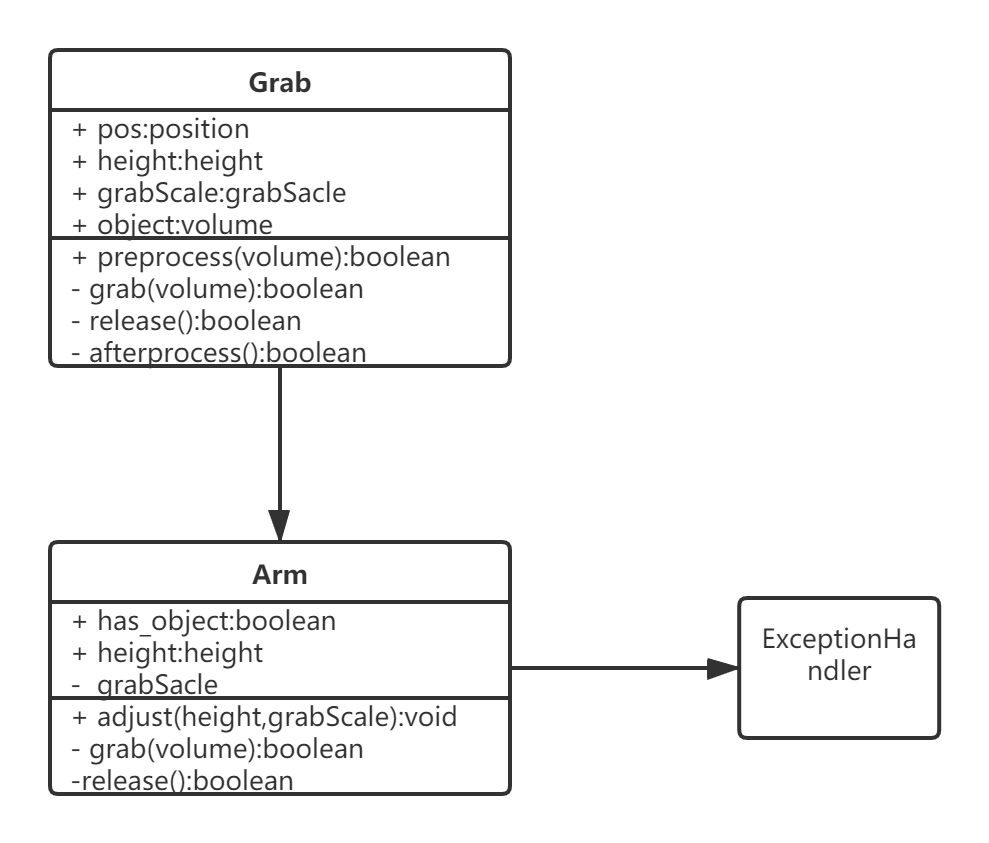


图18 目标抓取模块类图

类图说明：

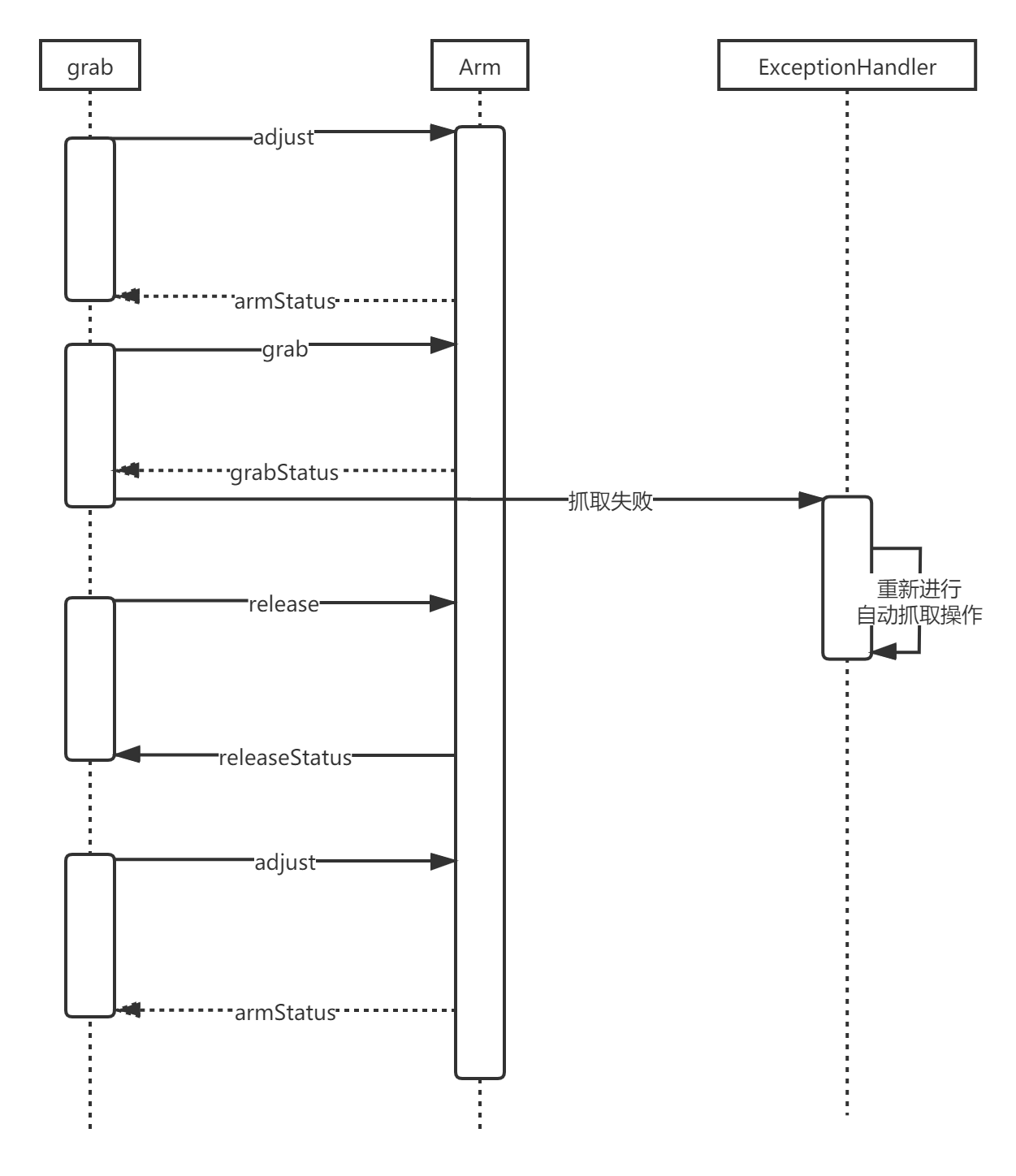
抓取模块的主类是Grab类。在自动目标抓取中，其中preprocess方法根据物体物体的形状大小位置，调用Arm类的adjust调节机械臂的状态，使其进入最佳抓取状态；其中grab方法通过调用Arm类的grab方法，对物体进行抓取；其中release方法通过调用Arm类的release方法放开所抓取的物体；在抓取后，可以通过主类的afterprocess方法返回抓取之前的状态，以便进行下一次抓取。在手动抓取中，主类grab通过handControl方法调用类Arm的moveByHand方法进行手动控制。

图19 自动目标抓取时序图

上图是自动目标抓取的流程图，手动目标抓取由于过于简单，略去。

* 1. 语音识别模块

功能：语音识别关键词，对用户的语音指令进行识别。在PocketSphinx的基础上，对关键词库进行扩展并存入数据库。并建立关键词与对应指令的对应映射关系，以达到用户通过关键词控制机器人的效果。机器人可以进行语音播报，与用户进行交互，令用户确认指令信息，以便实现更好地交互。

输入：用户输入的语音信息

输出：无

语音识别：对.corpus文件的内容进行修改，编辑成我们定制的关键词，并对源文件进行覆盖。重新launch，检查结果。此后启动pocketphinx的recognizer节点，即可讲识别出的语音内容转换为字符串形式发布。

语音播报：定义广播对象tts\_pub驱动机器人进行发声，配置机器人广播的一系列参数，并使用while循环和r.sleep()来控制机器人不停的发声，在while循环中调用ros::spinOnce()调用回调函数。

语音关键词如下：

|  |  |
| --- | --- |
| 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*** | 机器人停止运动 |
| ***Master*** | 用户所在位置 |
| ***Place N*** | 地点关键词 |
| ***Follow*** | 机器人跟随管理员 |
| ***Stop Following*** | 机器人停止建图跟随 |

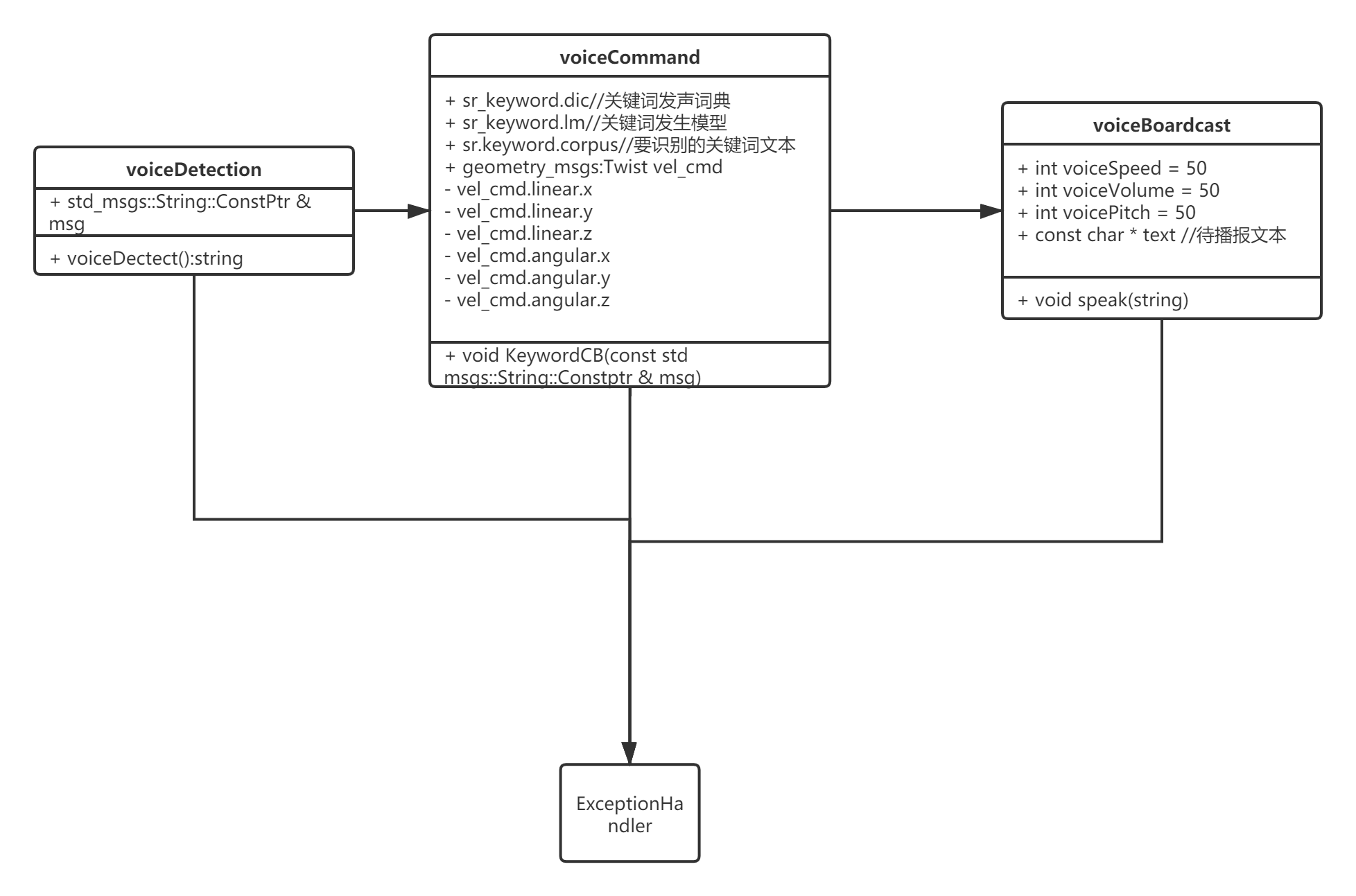


图20 语音识别模块类图

类图说明：其中voiceDetection类通过voiceDetect方法提取出语音中的关键词，发送给voiceCommand类，其通过KeywordCB回调函数对机器人底盘发出指令，并将关键词再发送给voiceBoardcast类，其通过speak方法向用户进行播报。

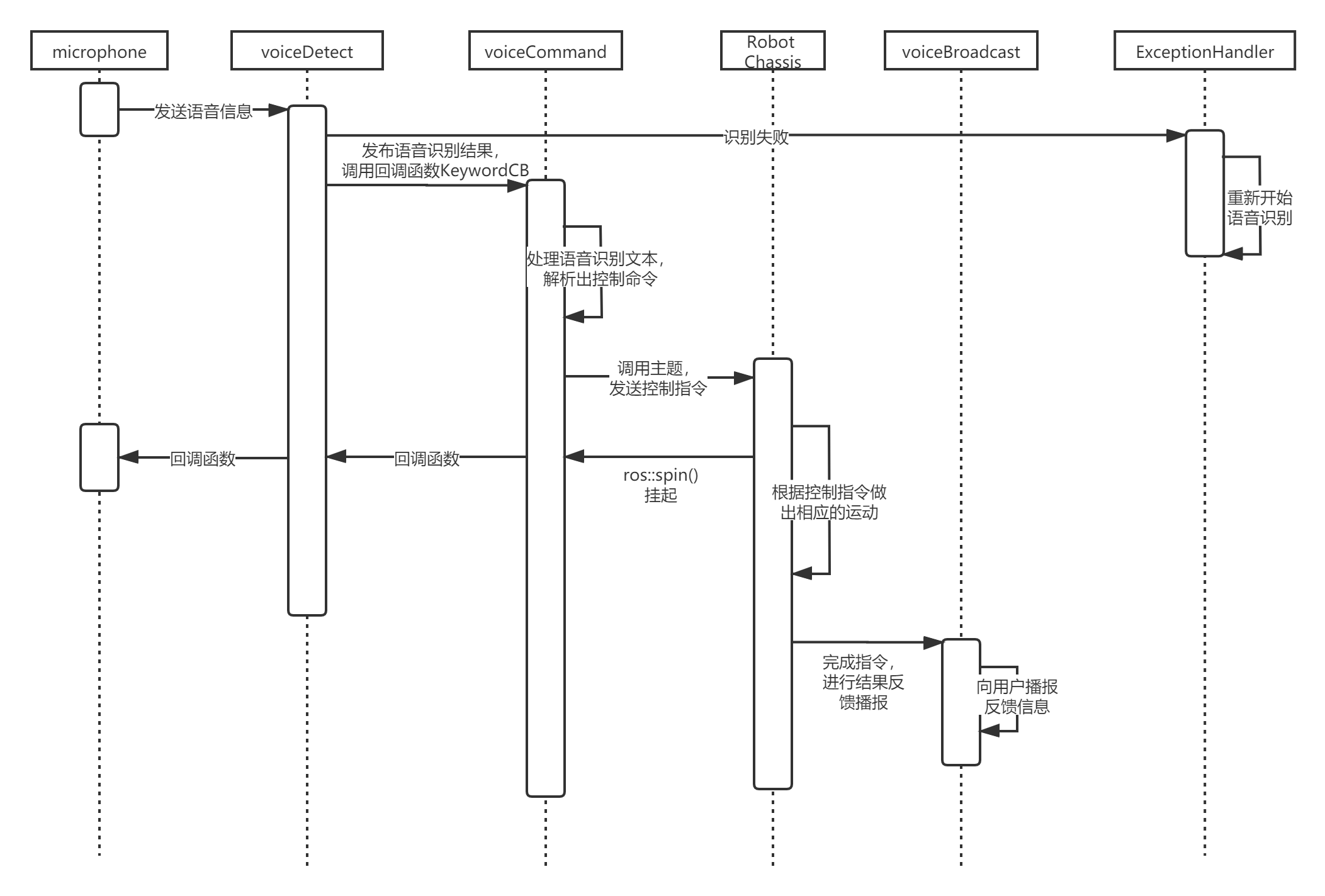


图21 语音识别模块时序图

* 1. 总控模块

功能简述：接收来自UI界面和语音识别模块传来的指令，并控制相应的识别抓取模块、地图模块、路径规划模块、移动模块执行操作。同时将机器人的状态返回指UI界面。

输入：来自UI界面和语音模块的指令

详细功能：

(1)对于地图显示的开关：

接收地图模块传来的实时地图并通过bool变量控制是否返回给UI界面。

(2)对于摄像头的开关：

控制摄像头硬件的开关。

(3)对于抓取的手动控制：

对抓取模块传达手动抓取的指令，包括手动抓取开关、手动控制抓取。

(4)对于移动的手动控制：

对移动模块传达手动控制移动的指令，包括手动控制开关、手动控制移动。

(5)对于自动寻路的控制：

对路径规划模块传达起点终点的设置，并且能指定是否在起点进行物品抓取。

(6)对于机器人的总开关：

可设置机器人关机、开机、挂起。

(7)对于抓取物品类型的指定

通过语音指定抓取物品，并记录在抓取模块中。

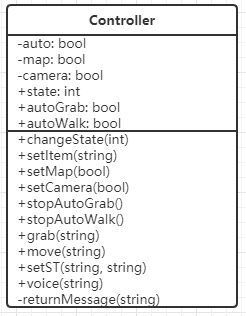


图22 总控模块类图

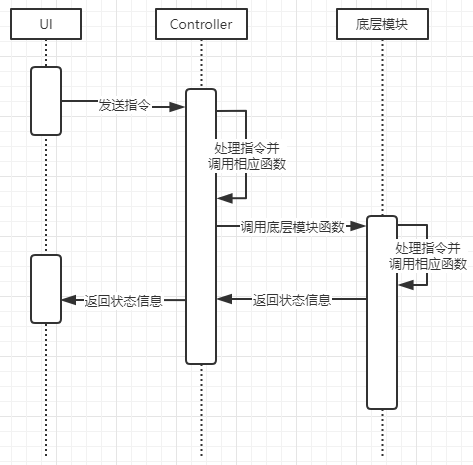


图23 总控模块时序图

* 1. 异常处理模块

功能简述：接收来自语音模块、抓取模块、地图模块、路径规划模块、移动模块的异常信息并指导相应模块做出处理，并将信息返回指UI界面。

输入：来自各个模块的异常信息

详细功能：

(1)语音模块：

未能成功识别(VoiceException)，无法识别的语句。

(2)抓取模块：

自动运行中(AutoException)，在机器人自动活动过程中接收到了手动抓取指令。

抓取失败(GrabFailedException)，未能成功抓取到物品。

(3)地图模块：

建图失败(MapFailedException)，通过slam算法建图出错。

(4)路径规划模块：

不能到达目的地(TargetFailedException)，目的地不可达。

超时(TimeLimitedException)，寻路超时。

(5)移动模块：

自动运行中(AutoException)，在机器人自动活动过程中接收到了手动移动指令。

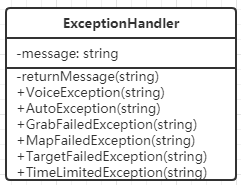


图24 异常处理模块类图

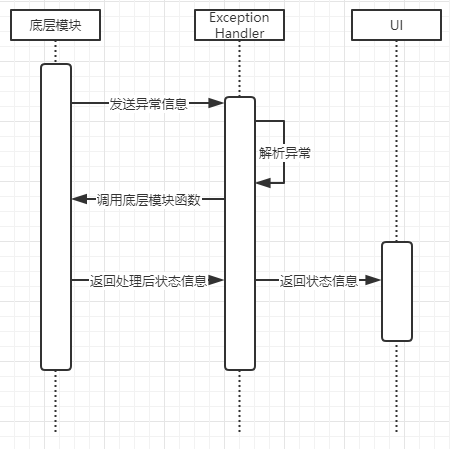


图25 异常处理模块时序图

1. 运行与开发环境
   1. 运行环境
      1. 硬件运行环境

本系统运行的计算机配置：

CPU：建议 1.2GHz Pentium III 处理器及以上。最低700MHz

内存：建议 RAM 要求2GB。

硬盘：建议20 GB 以上的可用空间。

显示器分辨率：最低Super VGA (800x600) 或更高分辨率的显示器（颜色设

置为32位真彩色）

其他：键盘、鼠标、2个以上USB接口。

* + 1. 软件运行环境

本系统运行的软件环境为Ubuntu 16.04的支持kinetic版本的ROS系统。

* 1. 软件环境

软件开发环境是 RoboWare Studio，软件开发语言为C++，工具使用基于 Visual Studio Code 开发的 ROS 专用 IDE。

1. 需求可追踪性说明

8.1 功能需求

本项目作为一款用于餐饮行业辅助运送菜品的机器人，主要功能为帮助将一定量的物品自动从一个地点运送到指定地点，或者通过用户手动操控或者语音控制完成这一过程。

（1）本系统要求机器人对地图进行建模并保存。实现这一目标的方法是使用启智ROS机器人自带的激光雷达，搭配SLAM算法完成建模工作。本系统中的地图管理模块将实现此功能，由数据库建立模块保存地图数据。

（2）本系统要求用户与机器人通过移动设备进行交互，并使用手柄对机器人进行操控。启智ROS机器人自带多种控制手柄支持，拥有丰富灵活的设置参数。由外部通信管理模块完成交互功能。

（3）本系统要求机器人对不同待抓取物品进行特征学习，以便后续物品的识别抓取。启智ROS机器人头部装备kinect2视觉传感器，彩色图像分辨率高达1920\*1080，用以图像获取；使用PCL的平面检测算法，可以准确识别物体，进行学习。由目标特征管理模块实现此功能。

（4）用户操控机器人时，机器人的动作主要为移动与抓取。启智ROS机器人拥有三轮全向底盘，可以安装机械臂满足抓取物品的需求。由目标抓取模块完成此功能。

（5）用户可以使用语音指令指定机器人的操作。启智ROS机器人自带阵列麦克风，可以用于采集正前方的实验数据；使用PocketSphinx语音识别包可以进行语音指令识别。由语音控制模块完成指令处理。

（6）机器人有自动规划路径移动到指定地点的功能。启智ROS机器人带有navigation导航功能，通过对ROS包修改增加控制速度，多点导航的功能，满足用户规划路径的需求。

8.2 非功能需求

部分非功能需求依托于机器人的具体硬件参数而满足，如功耗，可移植性，系统可用性等。以下仅说明需要通过系统设计方案满足的需求。

8.2.1 性能指标

（1）处理能力：

可以处理简单的语音指令，包括识别定义指令、识别并提示未定义指令。可以在平坦的地形条件下完成相对短的路径规划，规划过程限定在10s内。可以判断、躲避障碍并重新规划路线。

此要求将由语音控制模块，自主路径规划模块满足，我们将通过优化路径规划算法的方式，提升算法的速度，满足性能要求。

8.2.2 质量属性

（1）完整性：

SRS要求实现的完整功能包括以下几点：

1.物品抓取与放下：自主识别目标、判断距离；

2.路径规划：路径规划应在10s内完成，且路径为较优路径；

3.语音识别：识别固定的几条简单指令，并能识别出未定义指令；

4.避障功能：可识别阻挡运动的障碍物、停止并在10s内重新规划路线。

本系统将以上功能分割到不同模块中，由目标抓取，自主路径规划，语音控制，障碍回避模块分别完成，由总控模块对这些模块进行控制。

（2）健壮性：SRS要求机器人能够识别并提示出未定义指令。设置最大执行时长，避免没有可操作目标、地面情况导致的运动受阻或路径死循环。

该要求将由语音控制模块和总控模块完成。发生错误时，以上模块将捕获错误，停止工作，向用户发送提示，人工排查错误。