物品检测与抓取机器人

测试说明书

STD102

V5.0

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 2.3/2.4 测试
17373517	王正达	完成 2.1/2.2 测试
15241035	彭文鼎	完成 3.1/3.2/非功能性测试
17373288	尹俊成	完成 1.1 测试/领导非功能性测试

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	2020.5.26	王正达	杨开元	本小组软件设计说明初稿
V2.0.0	2020.5.26	彭文鼎	尹俊成	增加语音识别测试说明
V3.0.0	2020.5.26	彭文鼎	尹俊成	增加导航等测试说明
V4.0.0	2020.6.2	王正达	杨开元	增加非功能性测试等说明
V5.0.0	2020.6.8	王正达	杨开元	本小组测试说明书终稿

目录

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 主要功能和非功能性需求

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

本文档为软件测试说明文档，主要由范围、任务概述、测试准备、测试用例、测试结果、测试结果分析六部分组成。

概括来说，本文档将描述软件测试的整个过程，对测试用例进行介绍，分析测试结果，针对测试中发现的问题进行修改，完成代码测试工作。

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

编号	标题	版本	发行日期
1	启智 ROS 机器人开发手册	V1.1.0	2018.11.09
2	SDP102	V2.0.0	2020.03.24
3	SRS102	V3.0.0	2020.05.17
4	SDD102	V3.0.0	2020.05.19

2. 任务概述

2.1 功能需求

应用名称	功能需求名称	功能需求标号	功能需求描述
机器人移动	键盘控制机器人移动	0.1	可以使用键盘控制机器人的移动
语音交互	语音指令识别	1.1	可以识别语音并转化为指令
自主前往物体位置	构建与存储地图	2.1	可以构建环境地图并保存
	设定航点	2.2	可以预设一些自定义航点，保存航点位置与名称
	路径规划	2.3	机器人可根据指定航点名称自主进行路径规划
	导航与自主避障	2.4	机器人可根据路径规划结果自主前往目标航点，路途中动态避障
目标物体抓取	识别目标物体	3.1	机器人在到达目标航点后可自主识别目标物体位置
	抓取目标物体	3.2	机器人在识别到目标物体后可自主抓取目标物体

2.2 非功能需求

非功能需求名称	非功能需求编号	非功能需求描述
可维护可拓展	4	机器人系统应便于持续维护和灵活扩展
可靠性	5	机器人系统应可靠耐用
易用性	6	机器人系统应易于学习和使用
及时响应	7	机器人系统应可以对指令做出及时的响应

3. 测试准备

需求名称	硬件环境	软件环境	测试人员	时间分配	测试数据准备
键盘控制移动	个人笔记本	Ubuntu16.04+ ROS Kinetic+ Team102 code+ 课程组仿真环境	尹俊成	5.30 前	
语音指令识别			尹俊成	5.30 前	随机语音+语音指令
构建与存储地图			王正达	6.1 前	不同环境设置
设定航点			王正达		不同环境中多组随机航点
路径规划			杨开元	6.2 前	同上
导航与自主避障			杨开元		同上+随机障碍
识别目标物体			彭文鼎	6.8 前	多组航点位置与物体
抓取目标物体			彭文鼎		
可维护可拓展			全体人员	6.8 前	/
可靠性					所有上述测试+多次集成测试
易用性					/
及时响应					所有上述测试

4. 测试用例（模块测试）

4.0 测试用例 0（TC0）

用例名称：键盘控制移动黑白盒测试

用例标识：TC0

对应需求：0.1

实际应用场景：超市管理人员键盘操控机器人运动

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好；

输入：

通过键盘输入多组不同指令序列。

预期的输出：

相应分支测试完毕输出提示信息，详见../test/log/log0.csv。

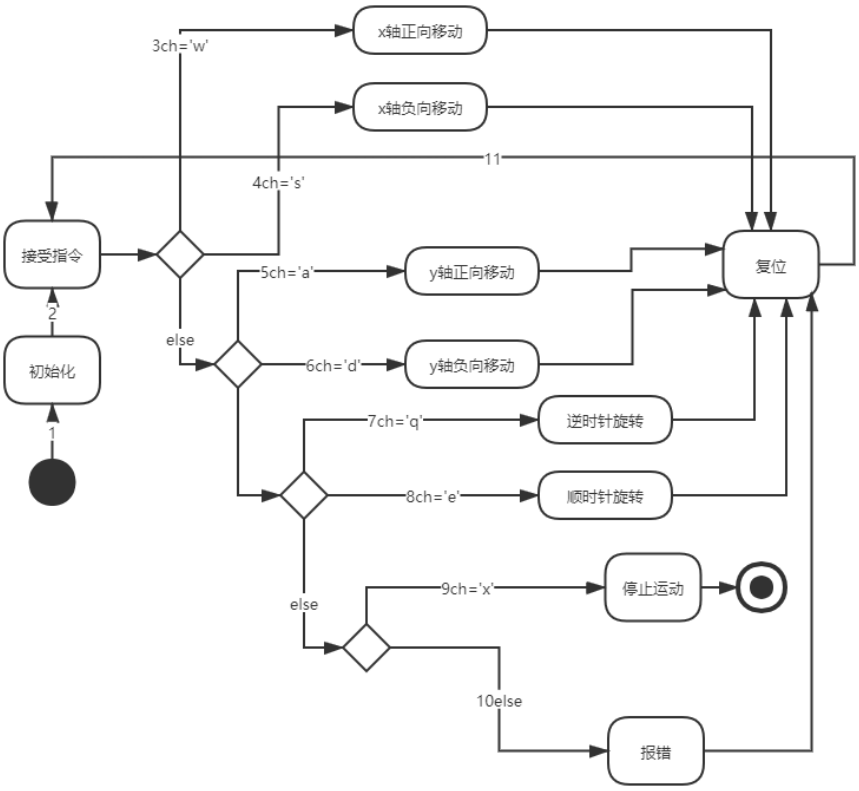
评价准则：

机器人完成预期的运动，控制台显示相应输出。

流程：

1. 运行机器人仿真环境。
2. 启动 rushbot_keyboard_ctrl_test 节点.
3. 手动在控制台输入指令序列。
4. 观察机器人在仿真环境内运动结果及控制台输出。
5. 进行对比评价。

模块状态图：



分支与输出对应关系：

分支编号	控制台输出
1	1TC0Start
2	ready4command
3	3w_over
4	4s_over
5	5a_over
6	6d_over
7	7q_over
8	8e_over
9	9x_over
10	10wrong_over
11	11reset_over

测试记录：

见../test/log/log0.csv。

4.1 测试用例 1（TC1）

用例名称：语音指令功能黑盒测试

用例标识：TC1

对应需求：1.1

实际应用场景：顾客语音输入所需商品名称

条件或状态：

ROS 系统运行状态良好；相关代码提前编译完毕。

输入：

输入的逻辑为：输入字符串 A，测试机器人能否检测出关键词 B

举例如下：

(1) A = “please start your show”, B = “start”

(2) A = “stop now”, B = “stop”

预期的输出：

若 A 中都含有 B，程序打印 success；否则程序打印 fail。

评价准则：

是否正确识别字符串

流程：

运行语音识别模块，测试人员按次序说出字符串 A 对应的内容，同时检查程序输出并记录。A 中每个测试样例重复三次。

测试记录：

见../test/log/log1.csv

4.2 测试用例 2（TC2）

用例名称：建图功能黑盒测试

用例标识：TC2

对应需求：2.1

实际应用场景：超市管理人员键盘操控机器人绕场一周完成建图

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好；相关代码提前编译完毕。

输入：

通过添加障碍物设置多组不同环境地图。

预期的输出：

机器人顺利完成地图的建立和保存。

评价准则：

机器人建图完成后，打开建立好的地图与原设置地图进行对比。

流程：

1. 运行机器人仿真环境。
2. 设置地图。
3. roslaunch wpb_home_tutorials gmapping.launch 启动机器人 gmapping 建图。
4. rosrun robot_sim_demo keyboard_vel_ctrl 启动机器人运动功能，键盘操控机器人在地图内运动一周。
5. 运动结束，rosrun map_server map_saver -f map 保存地图。
6. 地图对比进行评价。
7. 回到步骤 2。

测试记录：

见../test/log/log2.docx 或 ../test/log/log2.pdf

4.3 测试用例 3（TC3）

用例名称：设置航点黑盒测试

用例标识：TC3

对应需求：2.2

实际应用场景：超市管理人员在建好的地图中标记货架

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好；相关代码提前编译完毕。机器人建图完毕。

输入：

多组（航点名称，航点位置）。

预期的输出：

机器人顺利完成航点的设置和保存。

评价准则：

无报错完成航点设置。

流程：

1. 运行机器人仿真环境。
2. 完成 TC1 测试。
3. roslaunch waterplus_map_tools add_waypoint.launch 启动航点添加功能。
4. 在显示的地图中选取指定位置。
5. 在 Rviz 工具栏中点击“Add Waypoint”按钮添加航点。
6. 重复步骤 4、5 多次添加。
7. 保持 add_waypoint.launch 终端开启，运行 rosrun waterplus_map_tools wp_saver 进行航点保存。
8. nano waypoints.xml 打开航点文件，查看航点添加结果并修改航点名称。
9. 回到 add_waypoint 所在终端窗口。按下“Ctrl+C”退出脚本，然后重新启动 add_waypoint 脚本。查看航点名称修改结果。

测试记录：

见../test/log/log3.csv

4.4 测试用例 4（TC4）

用例名称：目标点导航黑盒测试

用例标识：TC4

对应需求：2.3、2.4

实际应用场景：机器人自主导航前往货架

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好；相关代码提前编译完毕。机器人建图完毕，地图文件已保存至相关目录。

输入：

目标点位置（正确目标点或异常目标点）

预期的输出：

机器人导航结果：成功或失败

评价准则：

准确及时到达所有正确目标点，对异常目标点做出响应。

流程：

1. roslaunch wpr_simulation wpb_simple.launch Gazebo

   运行 Gazebo 测试环境

2. roslaunch rushbot nav_test.launch

   开启 AMCL 和 move_base，完成导航相关节点加载

3. 若 Gazebo 和 Rviz 中机器人位置不一致，需要对机器人初始位置微调

4. rosrun rushbot rushbot_nav_goal

   启动测试节点并输入以地图为坐标系的终点目标

5. 机器人规划路径并移动

测试记录：

见../test/log/log4.csv

4.5 测试用例 5（TC5）

用例名称：多航点巡航黑盒测试

用例标识：TC5

对应需求：2.3、2.4

实际应用场景：机器人自主导航前往一系列货架

输入：

无

预期的输出：

机器人巡航途径目标点

评价准则：

路径规划正确，按顺序依次前往目标点

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好；相关代码提前编译完毕。

机器人建图完毕，地图文件已保存至相关目录；航点文件已保存至相关目录。

流程：

1. roslaunch wpr_simulation wpb_simple.launch Gazebo
   运行 Gazebo 测试环境
2. roslaunch rushbot nav_test.launch
   开启 AMCL 和 move_base，完成导航相关节点加载
3. 若 Gazebo 和 Rviz 中机器人位置不一致，需要对机器人初始位置微调
4. rosrun waterplus_map_tools wp_nav_test
   启动测试节点
5. 机器人巡航移动

4.6 测试用例 6（TC6）

用例名称：物品识别测试

用例标识：TC6

对应需求：2.3 2.5

条件或状态：

前置依赖：项目仓库下 models 目录下所有文件夹应放置于本机~/.gazebo/models 目录下

流程：

1. roslaunch wpr_simulation wpb_simple.launch Gazebo 运行环境
2. roslaunch rushbot obj_detect.launch 开启 rviz 并进行相关配置 这里没有开启 AMCL 和 move_base，不能进行导航
3. 在 Gazebo 机器人前方 2m 以内摆放一张 cafe_table，在上方放置一个 coke_can，并将 coke_can 的 Property 中 pose 的 y 值调整为桌面高度：0.77
4. 将 rviz 边栏中 Kinect 订阅主题改为 /kinect2/sd/points 改动之后，rviz 中机器人面前应有图像出现
5. roslaunch rushbot all_servers.launch 启动检测抓取节点
6. 机器人寻找平面后靠近平面检测物体

评价准则：

机器人是否正确识别物品

4.7 测试用例 7（TC7）

用例名称：可靠性测试

用例标识：TC7

对应需求：5

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好。

测试方法：

我们将整个机器人系统从开始建图到最终完成导航看作一次完整的运行，利用 test_modules.py 文件将全过程放到一个总控文件中进行运行，运行过程中输出相应的提示信息。

评价准则：

成功完整运行的次数占总测试次数的比例。

4.8 测试用例 8（TC8）

用例名称：及时响应测试

用例标识：TC8

对应需求：7

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好。

测试方法：

我们将对于及时响应这一非功能性需求的测试融入到之前进行的所有测试中，对值得研究的如下几项时间进行分别观察记录：

- 测试人员发出语音指令后，机器人完成识别所需时间。
- 测试人员给出开始导航指令后，机器人从待机到开始移动所需的时间。

评价准则：

1. 小组成员个人感受。

2. 如实记录数据，未来可与其他小组的机器人系统进行对比。

4.9 测试用例 9（TC9）

用例名称：抓取目标物体功能测试

用例标识：TC9

对应需求：3.2

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好。

说明：

由于仿真环境原因，我们的抓取只是在检测到物体并确定位置之后等待一会儿，随即输出抓取成功提示，因而没有相关测试。

4.10 测试用例 10（TC10）

用例名称：机器人服务系统测试

用例标识：TC10

对应需求：2.3 2.4 3.1 3.2

条件或状态：

机器人仿真环境部署完毕；ROS 系统运行状态良好。

机器人建图、航点设置、商品模型保存等准备性工作完成。

测试方法：

1.通过 rushbot_main.launch 文件启动整个服务。

2.为机器人在仿真环境内设置初始位置，机器人进入等待状态。

3.语音输入想要获得的商品名称，机器人会自主导航前往抓取商品。

4.在机器人运动路径上动态添加障碍物。

5.等待机器人获得商品并返回。

6.重复 3-5 步。

评价准则：

1.机器人是否正确识别商品名称。

2.机器人是否正确规划完成路径规划。

3.机器人是否可以完成动态避障。

4.机器人是否可以成功进行物体识别。

5.机器人是否可以返回初始位置。

5. 测试结果

5.0TC0 测试结果与评估

5.0.1 实际结果

可见../test/log/log0.csv

	输入	输出
1	wsadqe	1TC0Start 2ready4command 3w_over 4s_over 5a_over 6d_over 7q_over 8e_over

2	wrtyuiopfghjklzcvbnm	1TC0Start 2ready4command 3w_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over
3	w1234567890-=[]\;',./	1TC0Start 2ready4command 3w_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over 10wrong_over

4	wx	1TC0Start 2ready4command 3w_over 9x_over

5.0.2 完成状态：所预期的

5.0.3 测试说明

白盒测试：共 4 个样例完成了该程序模块 100%分支覆盖与 100%语句覆盖。

黑盒测试：共 4 个样例包括键盘所有可能输入（不包括序列顺序）。

5.1 TC1 测试结果与评估

5.1.1 实际输出:

	测试样例	关键词	输出结果
1	please start your show	start	success
2	what a pity	start	fail
3	start now stop	start	success
4	stop now start	start	fail
5	please stop now	stop	success
6	stop by now	stop	success
7	what a pity	stop	fail
8	please stop	stop	success
9	follow me now	follow	success
10	now follow me	follow	success
11	now me follow	follow	success
12	what a pity	follow	fail

5.1.2 完成状态:所预期的

5.2 TC2 测试结果与评估

5.2.1 实际结果:

机器人顺利完成了预想的地图建立和保存。

地图及建图结果见 ../test/log/log2.pdf

5.2.2 完成状态:所预期的

5.2.3 测试说明

通过多组黑盒测试，确定了建图功能的稳定性。

5.3 TC3 测试结果与评估

5.3.1 实际结果:

	航点位置	航点说明	航点设置结果
1	(2.00,1.00)	第一象限航点 1	成功
2	(6.00,7.00)	第一象限航点 2	成功
3	(-2.00,-1.00)	第二象限航点 1	成功
4	(-6.00,-7.00)	第二象限航点 2	成功
5	(2.00,-1.00)	第三象限航点 1	成功
6	(6.00,-7.00)	第三象限航点 2	成功
7	(-2.00,1.00)	第四象限航点 1	成功
8	(-6.00,7.00)	第四象限航点 2	成功
9	(200.00,200.00)	地图外航点 1	失败
10	(-200.00,-200.00)	地图外航点 2	失败

5.3.2 完成状态:所预期的

5.3.3 测试说明

通过设立各种正常航点、边界航点、异常航点，我们完成了对于航点设置的测试。

5.4 TC4 测试结果与评估

5.4.1 实际结果

	起始位置	目标点	用例说明	响应时间	导航结果
1	(0, 0)	(1, 0)	+x 方向直线无障碍物	13s	成功

2	(1, 0)	(0, 0)	-x 方向直线无障碍物	15s	成功
3	(0, 0)	(0, 1)	+y 方向直线无障碍物	11s	成功
4	(0, 1)	(0, 0)	-y 方向直线无障碍物	11s	成功
5	(0, 0)	(4, 1)	近距离避障第一象限	39s	成功
6	(4, 1)	(2, 5)	近距离避障第二象限	37s	成功
7	(2, 5)	(-3, 2)	近距离避障第三象限	38s	成功
8	(-6, 8)	(-5, 5)	近距离避障第四象限	26s	成功
9	(0, 0)	(6, 1)	远距离避障第一象限	47s	成功
10	(6, 1)	(-1, 5)	远距离避障第二象限	1m 6s	成功
11	(5, 3)	(0, 0)	远距离避障第三象限	52s	成功
12	(-5, 5)	(5, 3)	远距离避障第四象限	1m 15s	成功
13	(5, 3)	(200, 200)	地图外点	39s	失败，机器人原地不动
14	(5, 0)	(5, 0)	障碍物内部	52s	失败，机器人原地不动

5.4.2 完成状态：所预期的

5.4.3 测试说明

通过设定不同位置的目标点，我们分别测试了近距离避障、远距离避障、非法目标点等等情况。在测试过程中，近距离避障大约能在 20 秒内完成响应并返回结果，远距离避障可以在 1 分钟内返回结果。对于非法目标点，机器人大约在 1 分钟以内完成相应，在这期间机器人原地不动。

5.5 TC5 测试结果与评估

5.5.1 实际结果

机器人在三个航点之间巡航移动，具体结果见视频。

5.5.2 完成状态：所预期的

5.6 TC6 测试结果与评估

5.6.1 实际结果

机器人识别出目标物体，具体结果见视频。

5.6.2 完成状态：所预期的

5.7 TC7 测试结果与评估

5.7.1 实际结果

Available tests:

0: gmapping

1: add_waypoint

2: navigation

3: voice_recognition

Please input test id:

0

CompletedProcess(args="roslaunch wpr_simulation wpb_simple.launch,

returncode=0)

...

5.7.2 完成状态：成功率：28/30

5.8 TC8 测试结果与评估

5.8.1 实际结果

经我们实际测量，语音识别模块响应时间大概稳定在 0.4 秒左右；短距离导航相应并完成大约稳定在 20 秒以内，长距离导航响应并完成时间大概稳定在 1 分钟以内。

5.8.2 完成状态：所预期的

5.9 TC9 测试结果与评估

5.9.1 说明

由于仿真环境原因，我们的抓取只是在检测到物体并确定位置之后等待一会儿，随即输出抓取成功提示，因而没有相关测试。

5.10 TC10 测试结果与评估

5.10.1 实际结果

经多次测试，机器人均可完整正确地执行整个服务流程。

5.10.2 完成状态：所预期的

6. 测试结果分析

6.1 对被测试软件的总体评估

从功能性层面来看，被测试软件已经可以满足所有需求功能。

从非功能性层面来看，被测试软件可以较为高效稳定地执行所有的功能。

6.2 测试环境的影响

- 由于疫情原因，机器人只能通过仿真环境进行模拟，与真实状态有较大差异：一方面，仿真环境由于电脑计算资源限制，反应比较慢，有时候会甚至出现崩溃；另一方面，仿真环境始终还是无法模拟真实环境中的所有情况，因此有机会的话建议还是加入真实环境中的测试。
- 机器人测试在虚拟机运行条件下和双系统运行条件下在运行速度和稳定性层面均有一定差异。

6.3 改进建议

- 在多平台多次测试：增加软件健壮性、可移植性
- 增加更多 debug 信息，便于调试和复现 bug