Computer Science and Application 计算机科学与应用, 2019, 9(11), 2065-2076
Published Online November 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.911232

Hans汉斯

Implementation of Control System for Multi-User and Multi-Robot Based on Cloud Platform

Xuling Jin[1], Jianyong Feng[2], Yunpeng Shen[1], Yingjian Cao[1], Jian Ye[2]

[1]Beijing University of Civil Engineering and Architecture, Beijing
[2]Institute of Computing Technology Chinese Academy of Sciences, Beijing
Email: jye@ict.ac.cn

Received: Nov. 1st, 2019; accepted: Nov. 12th, 2019; published: Nov. 19th, 2019

Abstract

Along with the complexity and diversification of the robot service scene, the numerous data processing and analysis problems collected during the robot work process also need to be solved. To this end, a collaborative system of multi-user and multi-robot based on cloud platform is designed. The system transmits data through the http protocol, and the user sends a text command to the server through Android speech recognition, and the robot acquires the user's command from the server, then executes and returns the result. The integration of cloud technology and multi-robot systems makes multi-robot systems feature improved energy efficiency, high real-time performance and low cost.

Keywords

Cloud Platform, Android, Robot Operating System (ROS)

基于云平台的多用户多机器人的控制系统实现

靳旭玲[1]，冯建勇[2]，沈云鹏[1]，曹英健[1]，叶　剑[2]

[1]北京建筑大学，北京
[2]中国科学院计算技术研究所，北京
Email: jye@ict.ac.cn

收稿日期：2019年11月1日；录用日期：2019年11月12日；发布日期：2019年11月19日

文章引用: 靳旭玲, 冯建勇, 沈云鹏, 曹英健, 叶剑. 基于云平台的多用户多机器人的控制系统实现[J]. 计算机科学与应用, 2019, 9(11): 2065-2076. DOI: 10.12677/csa.2019.911232

摘　要

伴随着机器人服务场景的复杂化、多样化，机器人工作过程中采集的数量种类繁多的数据处理与分析问题也亟待解决。为此，设计一个基于云平台的多用户多机器人的协同工作系统。该系统通过http协议进行数据传输，用户通过Android语音识别将文本命令发送到服务器，机器人从服务器获取用户的命令，执行并将结果返还。将云技术和多机器人系统的集成使得多机器人系统具有改进能源效率、实时性高、成本低的特点。

关键词

云平台，Android，机器人操作系统(ROS)

1. 引言

机器人技术在过去几十年中取得了重大进展，越来越多的机器人也出现在大众视野中，如工业机器人，科研机器人，家庭机器人如扫地机器人等。机器人在现实生活中投入使用极大地便利了人类生活。大部分应用场景中的机器人有一个相同的特点，那就是计算、内存和传感都集成在一个单机系统中，机器人执行任务时，只能利用本地机器人系统中的程序以及资源。这种模式有一个缺点，由于机载硬件和计算资源的限制，当超出预料之外的情况发生或信息资源不足时，则会影响任务的执行。为了解决这个问题，研究人员不断探索，2010 年卡基梅隆大学的 James Kuffner 教授在 Humanoids 国际会议上首先提出了云机器人[1]的概念，获得了大量研究学者的赞同。云机器人依靠云平台[2]上的数据或者代码支持它的操作，并实现自动化[3]，而不是将传感、计算和内存集成到机器人单机系统。它能够借助互联网共享机器人之间的数据[4]，协同工作[5]，打破单一机器人的硬件瓶颈[6]，为云机器人提供了更高的计算能力和存储空间[7]。对于机器人单机系统难以完成的任务[8]，例如需要大量计算的场景识别、数据分析、语音合成等工作[9]，可以借助服务器完成后由机器人访问任务信息，云机器人不需要运算复杂的任务。本研究的主要贡献：1) 对于当前没有一个完整实用的系统实现任务的完整流程机器人系统环境，提供了一种从用户发布任务到任务完成，最后任务结果反馈给用户的闭环形式。2) 依托云服务器构建了机器人云平台，该技术利用云平台中集成的模型为连接该平台的机器人按需提供数据资源，软件管理不必花费大量时间进行调整，并可以快速适配和添加服务。本文对机器人云平台的架构设计进行了一定的探讨，并着力研究更好地实现云平台、机器人与 Andorid 应用的协同配合，完成用户任务。整个系统大致分为云服务器、Android 应用[10]和云机器人三个部分。

2. 系统分析与设计

系统架构

本系统需要完成用户通过 Android 端将任务发布到服务器，服务器将任务转化为机器人的执行动作，将命令传递给相应的机器人，机器人在完成相应任务后发送相应的反馈结果，最终将结果展示到 Android 端供用户查看[11]。基于以上设计，规划出系统框架结构如图 1 所示。

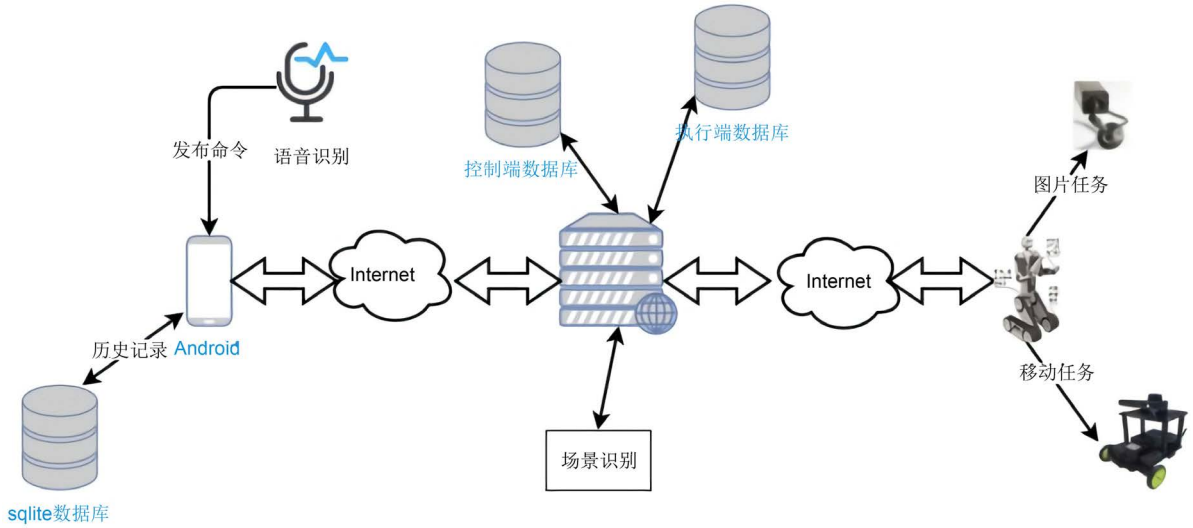


Figure 1. Architecture of multi-user and multi-robot control system
图 1. 多用户多机器人控制系统架构

系统各模块职责功能如下所述：

Android：为用户提供最直接的交互服务。从服务器获取机器人状态，执行任务状态，机器人反馈的各种数据，向服务器发送命令请求，展示文字、图片等供用户浏览。

服务器：安卓端用户与机器人用户的注册和管理；任务文本的处理，将字符串解释成机器人需要执行的任务，将任务和必要参数发布到特定 API；目标识别与场景理解，对机器人上传的图片进行处理，返回目标识别与场景理解的结果；任务反馈，机器人完成任务后对安卓端返回任务执行结果。

机器人：机器人作为执行者通过 HTTP 协议与服务器通信获取任务，接受控制端的指令完成用户发布的任务并在完成任务时给服务器做出反馈。

3. 系统开发与实现

3.1. Android 端开发实现

本文基于 Android 系统开发终端平台[12]，设备选用 Android 端智能手机。逻辑上由发布命令模块、查询结果模块、历史记录模块、HTTP 通信模块。

Android 端主要功能有登录登出注册、发布命令、查询结果、历史记录等。

发布命令模块：此界面为 app 的主界面，主界面效果如图 2 所示由登录成功后跳转进入，集成百度语音识别，用来讲用户的语音命令转化为文本，使命令的发布更加简单，除去繁琐的打字功能[13]。发布命令的主界面中放置了 spinner 控件，实时刷新，点击后显示在线的机器人列表，供用户选择使用。在选择机器人时判断机器人是否为闲置状态，防止不同用户控制同一个机器人的问题。Textview 用来显示语音转化的文字，如与用户本意不同可再次识别更改。所有数据选择齐全点击发布按钮一个发布命令的流程就执行完毕了。当用户不需要再次使用机器人时可以点击右上角的 toolbar 控件释放被占用的机器人，这时其他用户才可以使用这个机器人。向右滑动主页面的左侧会显示菜单栏，点击里面的列表可以跳转到相应的模块。

查询结果模块：当发布完任务时界面会跳转到查询界面。该界面显示 9 个文本框用来显示任务的各项具体信息。当任务状态为“F”停止刷新网络请求，通知栏提示用户机器人任务完成。如果发送任务请求时为关于图片的任务，获取到图片地址后可查看大图，长按图片选择是否保存到本地。

Figure 2. Android main interface
图 2. Android 端主界面

历史记录模块：由发布命令模块的菜单栏进入，该模块由一个 Listview 控件，每个 item 中有四个 Textview 控件，分别显示用户、机器人、任务描述以及时间。Listview 分批量加载，每次加载 20 个提高加载速度。

HTTP 通信模块：该模块负责从服务器获取机器人状态、任务状态以及向服务器发送数据请求等。使用 Android 中的 okhttp 框架，通过使用 post、get，put 不同的方法发送登录时的用户名到各个服务器地址来实现不同的数据请求。

3.2. 服务器端实现

3.2.1. 主要数据模块

系统采用 Django + restframework 框架进行开发，篇幅限制，这里简要介绍云机器人平台的主要数据模块，有用户数据和任务数据。前者主要对用户基本信息已经行为信息进行记录，形成用户资源库，方便云平台随时进行查询；后者对每个被发布的任务详细信息进行描述，在任务的执行状态与完结状态中扮演着重要角色。

用户数据模块中，考虑到云机器人平台控制端用户与传统网站用户的差异，原有的 Django 中的用户不能满足本平台的需求，所以需要继承 Django 内部的抽象用户类，重写用户模块。将发布命令的控制者和接受命令的执行者都看作用户，与重写好的用户类建立一对一连接，在该类的基础上加入两种用户不同的特性。当然，应用中还设有一类管理数据信息的超级用户，这里不详细讨论。用户模块数据信息类图如图 3 所示。

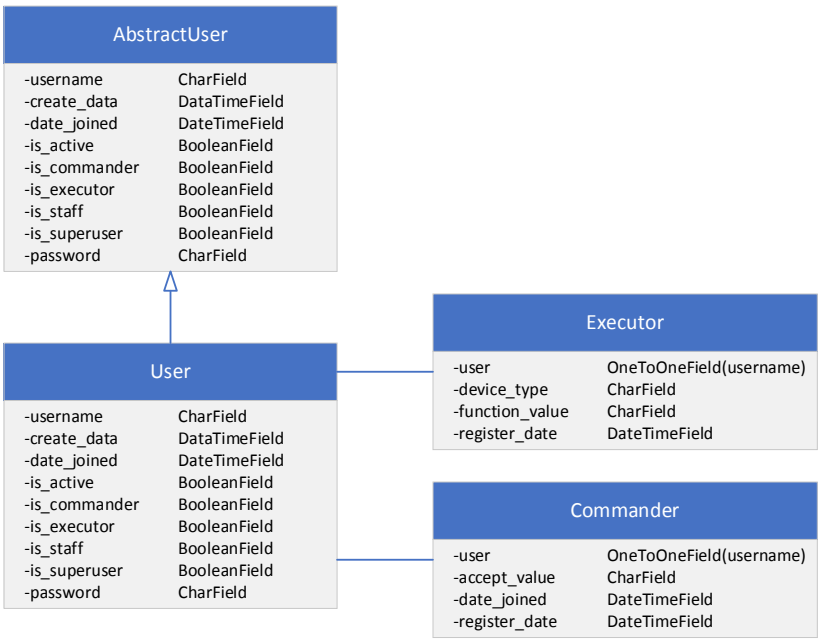


Figure 3. User module data information class diagram
图 3. 用户模块数据信息类图

考虑到后期用户数量的增多，任务执行过程中可能需要反复查询用户信息，如果用户表单占用系统资源过多，可能会影响系统的整体性能，所以将任务等待与任务执行过程中需要的信息分离出来，构建用户状态表如图 4 所示。

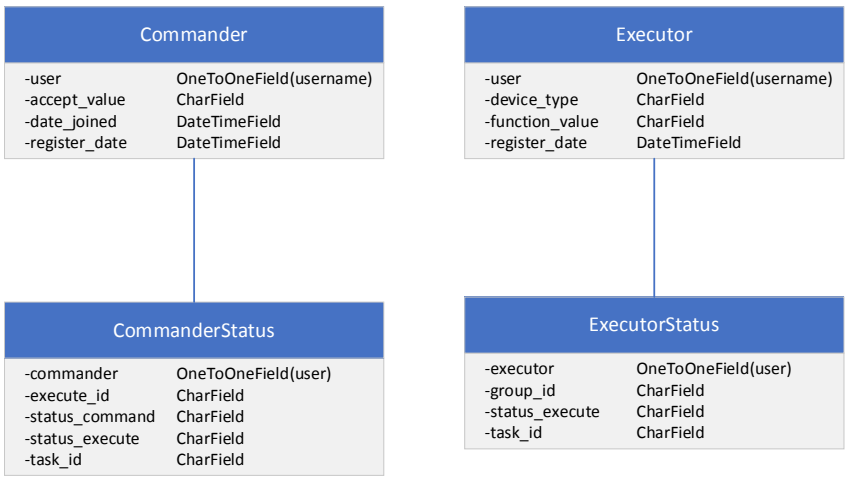


Figure 4. User state class diagram
图 4. 用户状态类图

任务管理表是整个数据系统的核心，任务的重要数据都存储在管理表中，任务类图如图 5 所示。控制端用户 id 和执行端用户 id 都存储在任务管理表中，通过用户 id 可以查询用户表和状态表。任务描述为控制端用户上传的用户上传任务的信息，云平台主要对任务描述进行解析。

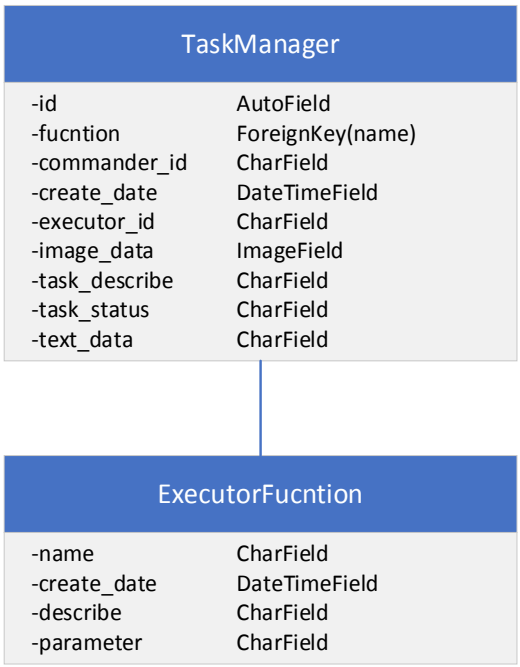


Figure 5. Task class diagram
图 5. 任务类图

3.2.2. 用户注册登录部分

安卓端注册方法为访问注册 url 并将注册信息以表单的形式填入，POST 到服务器端。

服务器返回注册结果。

注册发送的信息如下：

“username”: “user1”,

“password”: “mypassword”,

“equipment_type”: “android”,

“……”,

注册成功：返回状态码 200。

注册失败：返回状态码 404。

安卓端登录方法访问登录 url 并将账号以及密码以 Json 字符串的形式填入，POST 到服务器端。

服务器收到登录请求后查询数据库，验证登陆信息，若登录成功，返回 user_id，否则返回错误信息，若返回 user_id，安卓端用户需要将该 ID 存在本地，在任务发送过程中填入 url。登录成功后在控制端状态表结构(CommandUserStatus)中增加一个条目。

登录发送的信息如下：

“username”: “user1”,

“password”: “mypassword”,

登录成功：返回状态码 200，并获取 user_id。

登录失败：返回状态码 404。

ROS 机器人注册用户过程与安卓端用户注册大致相同相同，不过增加了机器人本体相关功能的描述。

3.2.3. ROS 机器人登录过程

访问登录 url 并将账号以及密码以表单的形式填入，POST 到服务器端。服务器收到登录请求后查询数据库，验证登陆信息，若登录成功，返回 user_id，否则返回错误信息，若返回 user_id，机器人端用户需要将该 ID 存在本地，在任务查询过程中填入 url。登录成功后执行端状态表结构(ExecuteUserStatus)中增加一个条目。

登录信息：

“username”: “robotuser1”,

“password”: “mypassword”,

登录成功：

返回状态码 200，并获取 robot_user_id。

登录失败：

返回状态码 404。

3.2.4. 任务发送过程

安卓控制端用户登陆后将语音转化成字符串，配合自身的控制端 ID，选择的执行端 ID，访问服务器为本用户分配的 url，以表单的形式 POST 任务信息。服务器返回接收命令结果，并决定是否创建任务。

任务发送信息如下：

“command_id”: “1”,

“execute_id”: “1”,

“command”: “command_string”,

服务器创建任务成功：返回状态码 200。

服务器创建任务失败：返回状态码 404。

任务查询过程：

安卓端用户发布任务到服务器后，查询控制端任务状态表。

任务成功完成后，如执行的任务有拍照功能，安卓端需要解析 Json 字符串后访问图片的 url 来显示结果图片。

3.2.5. ROS 机器人查询任务

ROS 机器人根据服务器分配的 robot_user_id 访问特定的 url，来查询自己当前任务 ID。

服务器在相应的 url 中返回机器人当前状态表。解析出需要的任务以及参数。

“function_task”: “take_picture”,

“parameter”: “None”,

机器人返回任务结果：

ROS 机器人返回执行的结果，结果通过 url 上传。

若上传成功，status_code 为 200。

commander 发布任务后，查询 commander_status 得到 task_id，根据 task_id 查询当前任务进度，查看 task 的内容更新 commander_status。

executor 查询 executor_status 是否有任务派送，如果 task_id 不为空，则说明有任务需要执行，根据 task_id 查询 task，并调用相关 function 和 parameter 执行，执行过程中不断查询 task 并更新 task (Update)。

3.3. 机器人端实现

ROS 端与服务器通信采用 HTTP 协议进行通信，ROS 的开发中使用了 ROS 基础的功能包 usb_cam 和 nav_test 作为摄像头和移动功能的驱动对功能进行二次开发。

ROS 端与服务器通信：ROS 端与服务器通信利用 Http 协议，利用轮询的方式查询任务，查询流程。如图 6，在开机自动登录后通过执行者的用户名获取执行者的状态和任务 ID (task_id)，在没有任务时任务 ID (task_id)为空，服务器接受任务会生成任务 ID (task_id)；通过 get 请求获取 task_id 后，会通过 task_id 进入任务页面，在任务页面通过 get 请求获取任务状态，任务状态为执行状态(Executing)执行任务并使用 put 请求上传服务器任务需要的数据对任务进行反馈(对上传状态进行监听)。

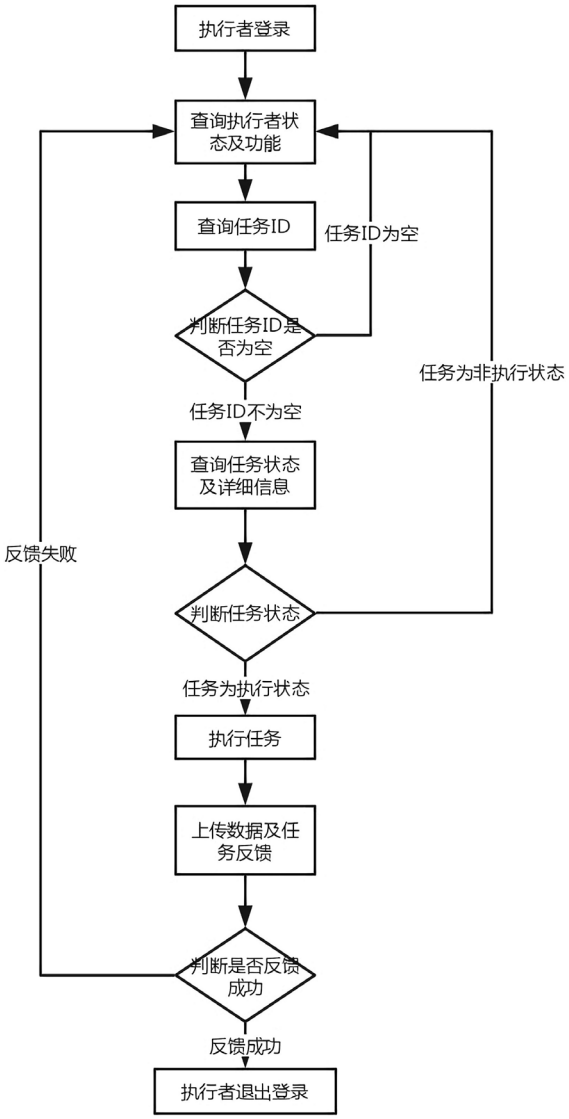


Figure 6. Robot end flow
图 6. 机器人端流程

移动机器人 ROS 端通信：机器人总通信架构如图 7。ROS 中通过节点(node)与节点之间的通信：包括话题(topic)和服务(service)两种方式。在不需要时效性的通信可以用话题，在机器人的移动过程中需要

机器人与控制同步就要用服务。

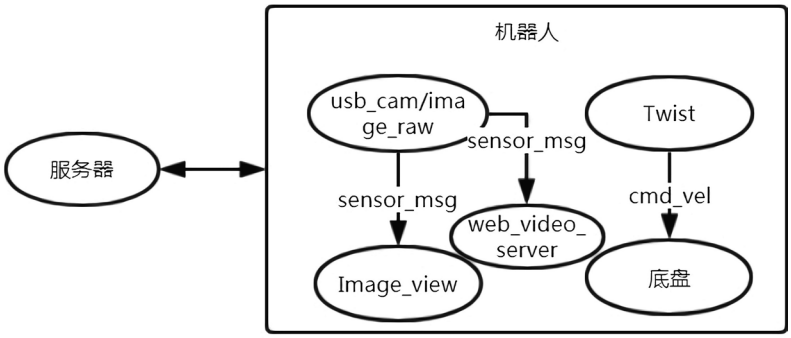


Figure 7. Robot communication architecture
图 7. 机器人通信架构

图像采集模块：图像的采集中基于 usb_cam 功能包，其中利用了 V4L2 技术打开摄像头的视频设备，通过数据结构和底层 V4L2 驱动接口把视频数据记录在内存中保存成一帧一帧的数据，创建视频数据节点/usb_cam/image_raw 将视频数据发布，image_view 通过订阅视频数据节点/usb_cam/image_raw，利用 image_transport、cv_bridge、opencv 获取消息(sensor_msg)的图像信息，并将图像保存通过网络通信将图片上传到服务器。

APP 控制移动模块：本文中机器人设计是基于 APP 端控制经服务器人为远程控制机器人，使机器人的移动时刻在用户的控制下，可以增加机器人的可控性方便用户在图像采集等功能时对角度进行控制，要求机器人与服务器时刻保持通信，本文采用轮询的方式更新机器人的移动状态(UP 代表前进，DOWN 代表后退，LEFT 代表左转，RIGHT 代表右转，STOP 代表停止)，其中利用控制发布节点 cmd_vel 的线速度(linear_speed)和角速度(angular_speed)达到定义的状态；APP 端更新移动状态时，ROS 端的移动状态便会改变。

自主避障移动模块：自主避障利用了 kinect 的深度摄像头生成的点云，其发布的点云通过 image_pipeline 转换成障碍物栅格分布图。nav_test 软件包启动底盘导航程序后会自动处理分析障碍物分布图，之后根据发布的目标的目标导航点自主移动如图 8。

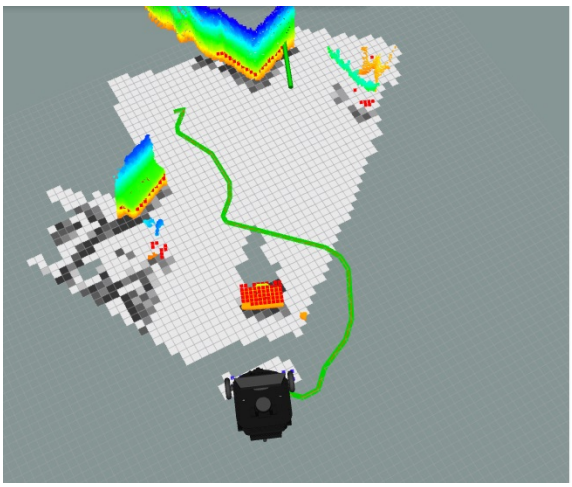


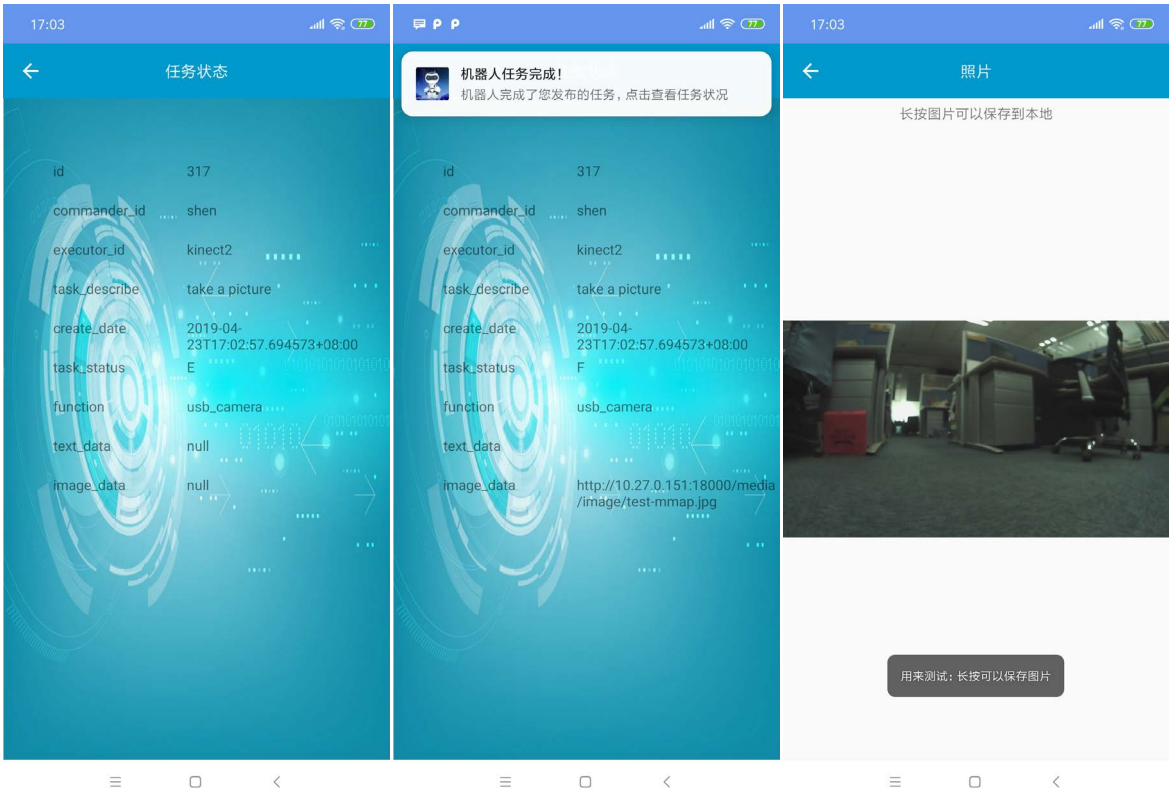
Figure 8. Robot autonomous obstacle avoidance navigation
图 8. 机器人自主避障导航

4. 系统实现效果

4.1. 软件

测试人员登录软件后，选择当前可执行任务的机器人，通过手机语音发布一个拍摄照片的任务，Android 端应用将语音识别结果与任务数据一同提交到服务器，并不断查询任务状态。Android 端跳转显示任务查询界面如图 9(a)，所示任务状态为“E”，表示任务正在执行，图片地址为“null”，代表照片还未上传到服务器。

当机器人执行完任务后服务器端更改任务表单状态，此时 Android 端应用也随之更新，如图 9(b)任务状态为“F”，这代表之前用户发布的任务已经完成，图片地址出现链接，点击之后可以查看机器人在当前位置使用机身携带的相机拍摄的图片，如图 9(c)所示，长按可以保存机器人拍摄的图片至本地。证明该系统能够为用户提供发布任务、查看结果的服务。



(a) (b) (c)

Figure 9. Task progress query
图 9. 任务进度查询

4.2. 硬件

本次测试使用的机器人型号为 XQ4-Pro，如图 10 所示，这是一款专门用于 ROS 开发的机器人，具有强大的开发计算能力，持久的电池续航能力和灵活的的运动能力。本文在 XQ4-Pro 上集成了相机、Kinect、机械臂、红外等多个硬件设备。

打开机器人电源后，机上搭载的操作系统开始启动，各项功能加载完成后机械臂执行相应动作代表机器人前期启动工作完成，并不断向服务器查询相关任务数据，随时准备执行任务。机器人接收到任务

Figure 10. XQ4-Pro robot
图 10. XQ4-Pro 机器人

数据后调用本地代码库，开始执行拍照任务，相机处的指示灯闪烁，代表此时正在拍摄照片，拍摄完成后图片数据上传到服务器，发布命令的用户可在 Android 应用查看任务状态。

5. 结束语

本设计方案通过 HTTP 协议对云机器人平台进行了设计与实现，并针对用户权限和不同的场景变化对机器人的权限进行管理，通过移动端和 PC 端对机器人控制。最后通过测试完成了移动端发布任务及机器人对任务进行移动和采集数据功能。本文设计的系统在云平台、机器人与 Android 应用的协同配合下，完成了用户发布任务，到任务执行结果反馈给用户的全部流程，使之形成一个闭环，并为多用户多机器人控制云平台系统的提供了一个架构参考。本系统可以扩展到机器人智能家居应用中，用户只需在手机上发布一个任务，机器人便会执行并返回结果，极大便利人们的生活。

项目基金

本课题得到国家重点研发计划课题(2017YFB1302400)、江苏省科技计划产业前瞻与共性关键技术竞争项目(BE2018084)、2018 北京高等学校高水平人才交叉培养“实培计划”面向云机器人的服务平台设计与实现(29041618014)、国家自然科学基金课题(61401040)、2018 年工业互联网创新发展工程项目。

参考文献

[1] 李波, 薛端, 黄鑫. 云机器人系统研究综述[J]. 计算机工程与应用, 2017, 53(17): 26-40.

[2] Zhang, H. and Zhang, L. (2019) Cloud Robotics Architecture: Trends and Challenges. 2019 *IEEE International Conference on Service-Oriented System Engineering* (*SOSE*), San Francisco, 4-9 April 2019, 362-3625. https://doi.org/10.1109/SOSE.2019.00061

[3] Sheta, A.F., Ghatasheh, N., Faris, H. and Rodan, A. (2019) Robotics Evolution: From Remote Brain to Cloud.

[4] Bozcuoğlu, A.K., Kazhoyan, G., Furuta, Y., Stelter, S., Beetz, M., Okada, K. and Inaba, M. (2018) The Exchange of Knowledge Using Cloud Robotics. *IEEE Robotics and Automation Letters*, **3**, 1072-1079. https://doi.org/10.1109/LRA.2018.2794626

[5] 尹磊, 周风余, 王玉刚, 袁宪锋. 云机器人前沿技术研究进展[J]. 龙岩学院学报, 2016, 34(2): 16-24.

[6] 龚正. 云机器人系统设计与关键功能实现[D]: [博士学位论文]. 南京: 南京航空航天大学, 2016.
[7] 洪臣, 史殿习. 云机器人架构和特征概述[J]. 机器人技术与应用, 2018, 186(6): 15-19.
[8] 胡春旭. 基于 ROS 的云机器人平台设计与实现[D]: [硕士学位论文]. 武汉: 华中科技大学, 2015.
[9] 张恒, 刘艳丽, 刘大勇. 云机器人的研究进展[J]. 计算机应用研究, 2014, 31(9): 2567-2575.
[10] 杨明极, 毕晶. 基于 Android 视频客户端的设计[J]. 电视技术, 2012, 36(3): 43-47.
[11] 倪红军. 基于 Android 平台的消息推送研究与实现[J]. 实验室研究与探索, 2014, 33(5): 96-100.
[12] 徐敏. Android 平台多媒体通信客户端研究与实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2011.
[13] 马志强. 基于 Android 平台即时通信系统的设计与实现[D]: [硕士学位论文]. 北京: 北京交通大学, 2009.