计算机科学
COMPUTER SCIENCE

自主机器人的伴随观察模式及其软件实现框架

薛元洲, 杨硕, 毛新军

引用本文

薛元洲, 杨硕, 毛新军.自主机器人的伴随观察模式及其软件实现框架[J]. 计算机科学, 2023, 50(7): 1-9.

XUE Yuanzhou, YANG Shuo, MAO Xinjun. Adjoint Observation Schemes and Software Implementation Framework for Autonomous Robots [J]. Computer Science, 2023, 50(7): 1-9.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.221200020

自主机器人的伴随观察模式及其软件实现框架

薛元洲[1,2]　杨　硕[3]　毛新军[1,2]
1 国防科技大学计算机学院　长沙 410073
2 复杂系统软件工程湖南省重点实验室　长沙 410073
3 国防科技大学系统工程学院　长沙 410073
(yzxue@nudt.edu.cn)

摘　要　自主机器人是一类运行于开放环境下、可自主决策和执行其自主行为的信息物理系统,它根据任务需求进行决策产生行为策略并调度执行。环境状态的动态变化性常常导致规划的行为策略不再适用于当前环境,使得行为执行的结果不符合预期,从而影响自主机器人的任务实现。上述问题对自主机器人软件的行为决策和软件构造均提出了更高的要求。一方面,自主机器人需在行为策略执行过程中加强对环境状态及其变化的观察,并基于观察的结果及时、灵活地调整行为决策,提升机器人的观察模式及行为决策算法的复杂度。另一方面,上述观察、决策、执行行为的复杂交互提升了软件构件抽象及数据交互的复杂性,如何抽象机器人的传感、决策、效应等软构件功能,并提供相适配的软件架构,成为自主机器人软件构造面临的重要挑战。针对上述挑战,首先提出自主机器人伴随行为的思想,显式定义观察与效应行为之间的伴随交互关系,根据行为执行不同阶段提出前提伴随观察模式和目标伴随观察模式,以提升自主机器人对环境变化的感知能力和决策调整能力。其次,开发了一款基于多智能体系统的自主机器人软件开发框架 AutoRobot,该框架将机器人的传感器、效应器及规划器抽象为一组自主的软件智能体,智能体间通过自主决策和协同实现上述伴随观察模式。AutoRobot 框架针对不同角色智能体设计和封装了一组可重用的软件组件,可有效支持自主机器人软件的复用和高效开发。最后,开展了仿真环境下的实验分析,通过与 ROSPlan 和 DESPOT 两种自主机器人任务规划和执行方法进行对比,验证了基于伴随观察模式的任务规划与执行的高效性和有效性。

关键词:自主机器人;开放环境;伴随观察模式;任务规划;任务执行
中图法分类号　TP311

Adjoint Observation Schemes and Software Implementation Framework for Autonomous Robots

XUE Yuanzhou[1,2], YANG Shuo[3] and MAO Xinjun[1,2]
1 College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China
2 Key Laboratory of Software Engineering for Complex Systems, Changsha 410073, China
3 College of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Abstract　Autonomous robot is a kind of cyber-physical system which operates in an open environment and can make and execute its behavior autonomously. It generates and executes effective plans according to the task. The dynamic change of environment state often results in that the planned behavior strategy is no longer applicable to the current environment and the results of behavior execution are not in line with expectations, thus affecting the task achievement of autonomous robots. The above problems pose challenges to both the behavioral decision-making and the construction of autonomous robot software. On the one hand, autonomous robots need to strengthen the observation of environment states and their changes during the execution of its behavior strategy, and adjust the behavior strategy in a timely and flexible manner based on the observation results, which improves the complexity of the robot's observation scheme and behavioral decision-making algorithm. On the other hand, the complex interactions of observation, decision-making and behavior execution enhances the complexity of software component abstraction and data interaction, and how to abstract the functions of software components such as sensing, decision-making and actuating of robots and provide suitable software architecture has become an important challenge for the software construction of autonomous robots. In view of the above challenges, the idea of adjoint behavior of autonomous robots is proposed, the adjoint interaction between observation and actuating behavior is clearly defined, and two adjoint observation schemes, conditional adjoint observation scheme and objective adjoint observation scheme, are proposed according to different stages of behavior execution, so as to im-

到稿日期:2022-12-04　返修日期:2023-03-27
基金项目:国家自然科学基金(62172426)
This work was supported by the National Natural Science Foundation of China(62172426).
通信作者:杨硕(yangshuo11@nudt.edu.cn)

prove the sensing of environmental changes and decision-making adjustment ability of autonomous robots. Secondly, AutoRobot, an autonomous robot software development framework based on multi-agent system, is developed, which abstracts the robot's sensors, actuators and planners into a set of autonomous software agents, and the agents implement the above adjoint observation schemes through autonomous decision-making and collaboration. The AutoRobot framework designs and packages a set of reusable software components for different agent types, which can effectively support the reuse and efficient development of autonomous robot software. Finally, an experimental analysis is carried out in the simulation environment, and the efficiency and effectiveness of task planning and execution based on the adjoint observation schemes are verified by comparing it with ROSPlan and DESPOT, two autonomous robot task planning and execution methods.

Keywords Autonomous robots, Open environments, Adjoint observation schemes, Task planning, Task execution

1 引言

自主机器人是一类运行在开放环境中，无需人为介入与干预、自主决策并实施行为以完成所赋予的任务和目标的机器人[1]。自主机器人根据任务需求自主决策产生有效的行为策略，该行为策略由一组有序组织的机器人行为构成。自主机器人通过调度其不同的效应器执行该行为策略，从而成功达成任务需求。但是，在开放环境中的行为策略执行可能会遇到两类问题：行为策略失效和行为执行效果不达预期。行为策略失效指预先决策的行为策略在环境变化之后无法执行；行为执行效果不达预期指由于环境的变化，行为执行之后未能达到其指定的行为效果。这两类问题要求自主机器人能够在运行时观察环境变化及行为策略的执行状态，并根据观察结果及时、灵活地调整其行为策略。

目前部分工作对自主机器人对环境的观察侧重于行为执行的运动控制阶段，在自主机器人控制其效应器改变环境状态的同时接收传感器反馈的信息，并据此判断控制效应器的过程是否出现差错[2-3]。还有部分工作侧重于观察行为执行前后的环境状态，并以此判断该行为的执行是否符合预期[4]。但这些工作将机器人对环境的观察表示为接收传感器信息并输出当前环境状态，而没有考虑效应行为对观察行为的辅助作用。具体地，自主机器人在执行观察行为的过程中可以通过执行某些效应行为来改变环境状态，从而获取更多的信息。例如，为了判断桌子上是否有某物体，自主机器人可以在桌子周围转一圈，从多个角度观察目标位置以获得更可靠的观察结果。

另一方面，自主机器人行为的决策、执行、观察等过程均需由相应的软件构件实现，行为决策、执行与观察等过程之间交互关系的复杂化，提升了自主机器人软件构件的抽象及数据交互的复杂性，进而提高了自主机器人软件开发的复杂度。因此，抽象、设计和开发自主机器人软件以实现上述过程变得相当困难。

为了应对上述挑战，本文首先提出了自主机器人伴随行为的思想，显式定义了自主机器人效应行为和观察行为之间的交互，并提出了两种伴随观察模式（Adjoint Observation Schemes，AOS），包括前提伴随观察模式和目标伴随观察模式。其中，前提伴随观察模式主要用于检测环境变化导致的行为策略失效的问题，目标伴随观察模式用于检测行为执行不达预期的问题。在此基础上，本文开发了一款基于多智能体系统的软件开发框架 AutoRobot，将自主机器人的传感器、效应器及规划器抽象为一组自主的软件智能体，智能体间通过自主决策和协同实现上述的伴随观察模式。AutoRobot 提供了一套可复用的软件组件，开发人员可以将其应用到不同的自主机器人任务之中，从而降低自主机器人软件的开发复杂度，提高开发效率。

2 相关工作

本节将从自主机器人的观察模式以及自主机器人软件开发框架和平台两方面阐述现有的相关工作。

2.1 自主机器人的观察模式

在机器人领域，行为执行过程中机器人对环境的观察有两类主要模式：持续性观察和间断性观察。

持续性观察指在机器人执行任务的整个过程中，持续不断地观察环境收集信息，并在遇到意外时进行策略调整以确保任务目标的完成。Dondrup 等[5]实现了一个有限状态机来并行地观察环境以检查效应行为的前提条件和预期效果是否满足，并根据机器人可以采取的效应行为来生成恢复行为。Yang 等[6]采用行为树结构来快速响应随时可能发生的环境变化。这些结构体现了观察和效应行为的并行和串行伴随关系，从而能够支持稳健的行为策略执行。Noreils 等[7]设计了一个用于行为监控与异常检测的模块，该模块从机器人的传感器不断接收信息，监控当前环境状态是否与预期一致，在环境状态与预期状态不一致时生成若干效应行为来进行状态恢复。Mukherjee 等[8]和 Kase 等[9]结合机器学习技术来将图像信息转换为机器人可理解的知识，从而判断任务执行状态是否出现不合预期的变化。Kaelbling 等[10]提出了一套混合规划—感知的方法，先根据已有知识生成线性的行为策略，在行为执行过程中监控可能出现的偏差，并及时进行重规划以产生新的行为策略。Cashmore 等[11]提出了一套名为 ROSPlan 的软件框架，提供了可定制的环境状态监控接口，来持续监控环境中某些状态并及时更新到自主机器人的知识库中。

间断性观察不需要时时刻刻对环境保持观察，而是在行为策略执行过程的部分关键时刻对环境进行观察。Coruhlu 等[12]提出在执行行为策略一部分之后生成预期环境状态并观察实际的环境状态，以检查其与预期环境状态是否存在差异。Rao 等[13]将概率规划器应用于自主机器人软件系统，在效应行为执行之后对环境进行观察，判断效应行为的执行结果，根据执行结果选择后续的效应行为。Suarez 等[14]提出了观察与效应的循环，收集环境信息以确定当前状态，并根据当前状态执行相应的效应行为从而达成任务目标。在部分可

观察的环境中，有相当多的工作[15-16]提出规划、执行、观察的控制环路，在每次效应行为结束后获取环境信息，并使用最新的环境状态进行规划，获取下一步应执行的行为。

上述工作通常将观察行为视为通过机器人传感器被动接收环境信息，而不会引起生任何环境状态变化的行为，这阻碍了机器人灵活地探索环境以进行更多更好的观察。通过被动传感获得的信息相当有限，并不足以使机器人准确地监控行为策略的执行状态。例如，在物体抓取任务中，机器人在执行抓取行为后试图检查它是否握住物体，但物体可能由于机械臂的遮挡而离开机器人的视野，因此它不能仅用头部摄像头直接观察物体的状态。本文提出了两种伴随观察模式，使观察行为和效应行为能更好地协作。伴随观察模式属于间断性观察，通过伴随观察模式，机器人可以通过效应行为来改变环境状态从而获得更多的观察结果，以实现更有效的行为执行监控。

2.2 自主机器人软件开发框架和平台

自主机器人软件是自主机器人的核心，其开发、部署、运行、维护需要相应的开发框架和平台支撑。本节从自主机器人的软件开发框架和开发平台两方面介绍现有的相关工作。

目前学术界和工业界使用最广泛的自主机器人软件开发框架是以 ROS[17] 和 YARP[18] 为代表的构件式软件开发框架，它们将自主机器人软件按功能划分为一个个软构件，每个构件实现其特定的功能，构件可以运行在不同的硬件平台上，通过开发框架提供的通信机制进行协作。以 ROS 为例，ROS 将机器人软件拆分为一个个节点(Node)，并提供跨平台的通信机制，允许运行在不同物理主机上的节点通过网络组成一个完整的软件系统[19]。得益于其开放性，ROS 已经形成了一个庞大的生态系统，包括统一的硬件抽象、常用的算法，以及一系列用于构建、测试和日志管理的跨平台工具。它还集成了一套全面的开源软件库，包括建图、运动控制、导航等。

另一种流行的机器人开发框架是以行为树[20-21]和状态机[22]为代表的行为式开发框架。行为式软件开发框架通过某种结构(如有限状态机、树状结构等)指定自主机器人执行的行为，提供模块化的设计和定义良好的软件组件，开发人员可以自由组合这些模块从而构建出特定的机器人行为。目前较为流行的行为式开发框架包括 BehaviorTree. CPP[1)]，SMACH[2)]，SMACC[3)]等。但随着任务复杂度的提高和软件规模的增大，表示自主机器人行为将变得更为复杂，难以设计、实现和验证其内部结构[23]。

自主机器人软件开发平台辅助开发人员开展自主机器人软件的开发活动，如设计、编码、构建、测试和运行等，可提高开发效率和质量。具有代表性的开发平台包括 Choregraphe[24] 和 vscode-ros[4)] 等。Choregraphe 支持的自主机器人硬件平台包括 NAO 和 Pepper，它支持自主机器人软件的可视化开发，提供了多种通用功能组件来辅助开发者开发复杂的自主机器人软件，并提供自主机器人软件的设计、编码和仿真运行支持。vscode-ros 基于 ROS，其功能包括环境配置、依赖包管理、代码编译、节点调试、支持多种编程语言等。它帮助开发者完成自主机器人软件开发过程中的繁琐操作，但只是对 ROS 功能的简单封装，并不能提供自主机器人软件复杂行为模式的封装和开发支持。

上述工作提供了关于自主机器人不同功能组件之间通信交互的基本抽象，但很少有人针对自主机器人的复杂观察模式和决策机制提供领域定制的软件构件抽象和开发框架，导致自主机器人软件的开发复杂度较高。本文提出了一种用于自主机器人软件开发的多智能体软件开发框架 AutoRobot。该开发框架基于构件式开发框架 ROS，并在 ROS 的基础上提供了若干可复用的软件组件，将组件之间的通信进行了抽象和封装，并提供了可扩展的接口。开发者能复用这些软件组件，并根据任务需要对开发框架进行定制，即可实现用于特定任务的自主机器人软件，这可以大大简化自主机器人软件开发的过程，降低开发复杂度，提高开发效率。

3 自主机器人的伴随观察模式

本节建立了自主机器人的模型，包括自主机器人的伴随观察模式以及支持伴随观察模式的规划和调度算法。

3.1 自主机器人模型

自主机器人配备了各种各样的效应器和传感器，效应器用于改变自身或周围环境的状态，如机械臂、马达等，传感器用于从环境中获取信息，如摄像头、激光雷达等。这使得自主机器人可以执行两种行为：效应行为和观察行为。其定义如下。

定义 1(效应行为) 自主机器人通过效应器改变环境状态来支持任务完成的行为。一个效应行为 α 定义为二元偶 $\alpha=\langle C,E\rangle$，其中，$C$ 表示该效应行为执行的前提条件集合，E 表示该效应行为成功执行后能达到的预期效果集合。前提条件集合由若干谓词组成，代表着此效应行为执行时这些谓词代表的关系都应该满足。预期效果集合由若干描述动作效果的谓词组成，描述了此效应行为成功执行后的环境状态。

定义 2(观察行为) 自主机器人通过传感器从环境中获取信息的行为。一个观察行为 β 定义为三元偶 $\beta=\langle p,r,a\rangle$，其中，$p$ 是一个表示该观察行为的观察目标的谓词，r 表示观察到的目标状态，a 表示与该观察行为协作执行的效应行为，机器人可以通过执行此效应行为改变环境状态，从环境中获取更多信息。例如，观察行为 $\beta=\langle in\_hand(obj),r,rotate\_arm\rangle$，表示此观察行为的观察目标是 $in\_hand(obj)$，即“观察物体是否在机械臂中”；其伴随执行的效应行为为“$rotate\_arm$”，表示自主机器人在执行此观察行为时会转动机械臂，将物体移动到机器人的摄像头前，以排除机械臂对物体的遮挡。自主机器人可执行的所有观察行为构成观察行为集合 B。

自主机器人的行为策略 S 通常由若干个效应行为有序构成，即 $S=\langle\alpha_1,\alpha_2,\cdots,\alpha_n\rangle$，自主机器人按顺序执行行为策略中的每个效应行为，最终获得任务完成状态 R，$R\in\{success,failure\}$，表示任务是否成功完成。

3.2 伴随观察模式

我们将观察行为与效应行为的交互关系称为自主机器人

1) https://github.com/BehaviorTree/BehaviorTree.CPP

2) https://github.com/ros/executive_smach

3) https://github.com/robosoft-ai/SMACC

4) https://github.com/ms-iot/vscode-ros

的伴随观察模式。针对自主机器人执行任务过程中可能出现的行为策略失效和行为效果不及预期的问题，我们提出两类伴随观察模式，即前提伴随观察模式和目标伴随观察模式。

3.2.1 前提伴随观察模式

自主机器人的行为策略包含若干有序的效应行为，若某个效应行为执行时其前提条件未完全满足，则效应行为的执行是无效甚至不安全的，最终会导致任务失败。例如，自主机器人执行某一物体抓取行为$\alpha_1=\langle C,E\rangle$，其中，$C=\{on\_table(obj,table)$，表示“物体在桌子上”；$E=\{in\_hand(obj),noton\_table(obj,table)\}$，表示“物体不在桌子上而在机械臂中”。在开放环境中，由于未知的环境变化（如人为移动了该物体），前提条件不再满足，此时执行“抓取物体”行为并不能成功将物体抓起来，因此不能达成任务。自主机器人需要在执行“抓取物体”效应行为之前判断其前提条件“物体在桌子上”是否满足。

前提伴随观察模式是检查效应行为的前提条件是否满足的一类通用模式。判断前提条件是否满足的方式是找到对应的观察行为，通过执行观察行为从环境中收集信息，判断此前提条件是否满足。在效应行为执行之前，调度器调度执行观察行为收集环境信息，判断效应行为的前提条件是否满足，进而判断行为策略是否失效。在前提伴随观察模式中，自主机器人通过伴随执行效应行为对应的观察行为，获得相应的观察结果，将观察结果汇总并反馈给调度器，从而判断当前效应行为是否失效，调度器决定此效应行为是否应该被调度执行。在执行观察行为β_i时，其对应的效应行为$\beta_i(a)$也会被执行，从而能够改变环境状态，获得更完整更可靠的观察结果。

用于检测效应行为α的前提条件是否都得到满足的若干观察行为被称为前提观察行为集pre_α，其中每个观察行为的观察目标对应于该效应行为的某个前提条件，即$pre_\alpha=\{\beta|\beta(p)\in\alpha(C)\}$。$pre_\alpha$先于效应行为$\alpha$执行，以确保效应行为执行之前其前提条件能得到满足，如图1所示。前提伴随观察模式中，效应行为与观察行为的调度顺序为：1）调度执行pre_α；2）从环境中获取观察结果，反馈给调度器；3）调度器判断效应行为的前提条件是否满足，如果满足，则调度执行该效应行为。

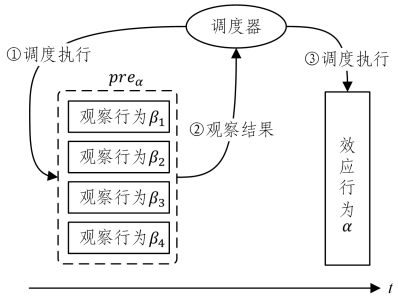


图1 前提伴随观察模式

Fig. 1 Conditional adjoint observation scheme

3.2.2 目标伴随观察模式

开放环境中效应行为的执行并不总是能获得符合预期的行为效果。例如，效应行为α_1的预期效果之一是“物体在机械臂中”$in\_hand(obj)$。但由于机器人传感器会有识别误差，或者在抓取过程中物体从机械臂中不慎掉落，行为执行的预期效果可能得不到满足。因此，自主机器人有必要对行为的执行结果进行检验，从而判断行为的执行是否符合预期。

目标伴随观察模式的内涵为：自主机器人执行效应行为后，通过执行相应的观察行为来从环境中获取信息，从而检验该效应行为的预期效果是否达到。目标伴随观察模式对应的用于检查行为预期效果的若干观察行为被称为目标观察行为集$post_\alpha$，其中每个观察行为负责检验效应行为的某一个预期效果是否达到，即$post_\alpha=\{\beta|\beta(p)\in\alpha(E)\}$。$post_\alpha$在效应行为$\alpha$执行之后再执行，从环境中收集信息，判断效应行为的预期效果是否都达到并将结果反馈给调度器和知识库，如图2所示。在目标伴随观察模式中，效应行为α与$post_\alpha$按以下顺序进行调度：1）调度执行α；2）调度执行$post_\alpha$；3）从环境中获取观察结果并更新到知识库中；4）判断效应行为的预期效果是否达到，如果未达到则重新调度执行α。

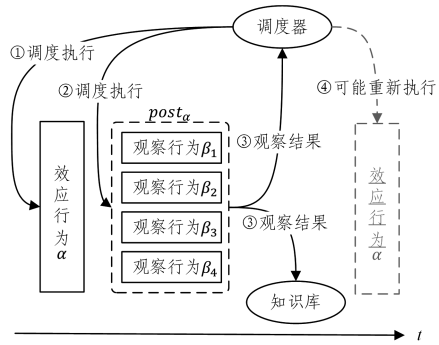


图2 目标伴随观察模式

Fig. 2 Objective adjoint observation scheme

采用上述两类伴随观察模式后，自主机器人能够在效应行为执行之前及时检测到环境变化导致的行为策略失效，从而能够尽早调整行为策略以适应变化后的环境。在执行效应行为之后，自主机器人能够对环境进行观察以检测效应行为是否达成预期效果，从而能够检测到效应行为失效可能导致的任务失败。另外，应用伴随观察模式有利于通过效应行为来改变环境，获得更完整更可靠的观察结果，从而能更准确地判断任务执行状态。

3.3 伴随观察模式的规划和调度算法

本节介绍支持伴随观察模式所需要的相关算法。

3.3.1 伴随观察行为规划算法

行为策略S由任务规划器POPF产生，包含若干用于完成任务的效应行为。伴随观察行为规划算法Adjoint-Plan(α)根据行为策略中每一个效应行为α计算出其所需的前提观察行为集pre_α和效果观察行为集$post_\alpha$，自主机器人执行这些观察行为来判断效应行为的前提条件是否满足以及是否达到预期效果，其流程如算法1所示。

算法1 伴随观察行为规划算法

输入：效应行为 α

输出：前提观察行为集 pre_α，效果观察行为集 $post_\alpha$

初始化：$pre_\alpha\leftarrow\emptyset$，$post_\alpha\leftarrow\emptyset$

1. function AdjointPlan(α)
2. for c_i in α(C) do
3. for β_j in Bdo

```
4.          if βj(p)=ci do
5.            add βj into pre_α
6.            break
7.          end
8.        end
9.      end
10.     for ei in α(E) do
11.       for βj in Bdo
12.         if βj(p)=ei do
13.           add βj into post_α
14.           break
15.         end
16.       end
17.     end
18.     return(pre_α,post_α)
19. end
```

针对特定效应行为的前提条件集合 $\alpha(C)$中的某个条件 c_i，在观察行为集合中选择一个观察行为β_j，使它的观察目标 $\beta_j(p)$与c_i相等，这个观察行为即为此条件对应的观察行为。在任务定义时，出现在效应行为的前提条件和预期效果中的所有谓词都应该定义相应的观察行为。遍历前提条件集合 $\alpha(C)$并找到所有对应的观察行为，这些观察行为就组成了此效应行为的前提观察行为集 pre_α（第 2－9 行）。同样地，遍历效应行为的预期效果集合 $\alpha(E)$并找到所有相应的观察行为，这些观察行为组成了此效应行为的目标观察行为集 $post_\alpha$（第 10－17 行）。最后，算法返回 pre_α和 $post_\alpha$。

3.3.2 观察行为与效应行为调度算法

生成 pre_α和 $post_\alpha$后，需要相应的行为调度算法调度自主机器人的效应行为和观察行为的执行，并且在检测到效应行为不可执行或执行效果不达预期时进行行为策略的调整。

观察行为调度算法用于调度一组观察行为 *ObsSet* 的执行，从环境中获取这些观察行为的观察结果，汇总并返回观察结果 *ObsRes*，如算法 2 所示。其中，算法的输入为观察行为组成的集合，即 $ObsSet=\{\beta_1,\beta_2,\cdots,\beta_n\}$，算法会调度执行这些观察行为；算法的输出$ObsRes\in\{\text{true},\text{false}\}$，表示这些观察行为的观察结果是否都为 true。

算法 2 观察行为调度算法

```
输入：观察行为集合 ObsSet
输出：观察结果 ObsRes
1. functionscheduleObs(ObsSet)
2.    for βi in ObsSet do
3.      sense βj from environment,result is rj
4.      if rj=false do
5.        return false
6.      end
7.    end
8.    return true
9. end
```

效应行为调度算法负责按顺序调度行为策略中每一个效应行为的执行，如算法 3 所示。在调度每一个效应行为时，会使用伴随行为规划算法产生 pre_{α_i}和 $post_{\alpha_i}$（第 4 行）。然后按顺序执行 pre_{α_i}，α_i，$post_{\alpha_i}$。

算法 3 效应行为调度算法

```
输入：行为策略 S，允许等待次数 NC，允许重试次数 NE
输出：任务完成状态 R
1. function schedule(S,NC,NE)
2.    for αiin S do
3.      nc←0,ne←0
4.      pre_αi,post_αi←AdjointPlan(αi)
5.      while nc<NC do
6.      if scheduleObs(pre_αi)=true do
7.        break
8.      else
9.        nc←nc+1
10.     end
11.   end
12.   if nc≥NC do
13.     S← task plan for a new strategy
14.     continue
15.   end
16.   while ne<NE do
17.     execute αi
18.     if scheduleObs(post_αi)=true do
19.       break
20.     else
21.       ne←ne+1
22.     end
23.   if ne≥NE do
24.     return failure
25.     end
26.   end
27.   return success
28. end
```

通过算法 2 调度执行 pre_{α_i}中的每一个观察行为，自主机器人能够从环境中获取并汇总对应的观察结果（第 6 行），判断其前提条件是否全部满足。如果存在前提条件不满足就继续等待，直到其前提条件全部满足。如果等待次数超过算法参数指定的允许等待次数 *NC*，则认为此行为策略已经不再有效，需要重新决策以获取新的行为策略（第 20 行）。

自主机器人执行效应行为之后，会调度执行 $post_{\alpha_i}$中每一个观察行为来判断其预期效果是否全部达到（第 18 行）。如果存在某一效果未达到，则认为环境状态改变使得此效应行为的执行失败，需要重新执行此效应行为。如果失败次数超过允许重试的次数 *NE*，则返回任务失败。行为策略中所有的效应行为均执行成功后，任务完成。

4 基于多智能体系统的软件开发框架 AutoRobot

为了支持自主机器人软件的设计和实现，辅助开发者对自主机器人代码的复用和高效开发，本节介绍一个用于自主机器人的基于多智能体系统的软件开发框架 AutoRobot，通过复用开发框架中部分通用功能组件并给不同的自主机器人任务提供扩展接口，从而降低自主机器人软件的开发复杂度。

4.1 自主机器人软件的通用抽象软件架构

图 3 给出了自主机器人软件的通用架构。规划器从知识

库中获取当前环境状态和任务目标信息，进行任务规划以产生行为策略；调度器接收行为策略，调度行为策略中的每一个效应行为进行执行，并且产生效应行为对应的伴随观察行为，在执行效应行为前后执行相应的伴随观察行为对环境进行观察；传感器负责驱动自主机器人的传感器硬件，执行观察行为，从环境中获取信息并更新到知识库中；效应器负责驱动自主机器人的效应器硬件，执行效应行为来改变自身或周围环境的状态。

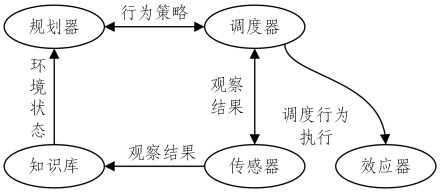


图 3 自主机器人通用软件架构

Fig. 3 Generic software architecture of autonomous robots

4.2 AutoRobot 软件开发框架

自主机器人内部部件独立运行，又根据任务执行情况、环境状态的改变等和其他部件进行频繁的交互，因此，我们将自主机器人内部的规划器、调度器、传感器和效应器软件组件使用软件智能体进行抽象，利用软件智能体的独立性和自主性实现各组件的功能，并利用各软件智能体之间的通信、合作、调度等能力共同组成自主机器人软件，以实现自主机器人功能。

AutoRobot 的整体架构如图 4 所示。该框架使用软件智能体来抽象和管理自主机器人的各种行为，使用 ROS 节点来具体执行自主机器人的行为。

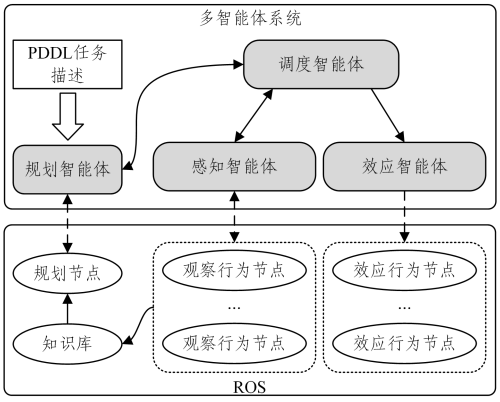


图 4 基于多智能体的软件开发框架架构图

Fig. 4 Architecture of software development framework based on MAS

AutoRobot 中共包含 4 个智能体：规划智能体、调度智能体、效应智能体和观察智能体。其中，规划智能体根据自主机器人的任务目标和初始环境进行任务规划得到完成任务的效应行为序列；调度智能体按顺序调度并执行每一个效应行为。我们将伴随观察模式的逻辑在调度智能体中实现，在每个效应行为调度的过程中，先使用算法 1 规划得到该效应行为的伴随观察行为集合，再使用算法 2 和算法 3 进行效应行为及其伴随观察行为的调度与执行。为了保证多智能体系统的通用性和可复用性，效应智能体和观察智能体并不实际控制机器人，而是将行为执行指令转发给相应的 ROS 节点，由 ROS 节点控制机器人的具体硬件。效应智能体接收调度智能体的行为执行指令，并通过与相应的 ROS 节点交互以进行实际的行为执行。观察智能体接收调度智能体的伴随观察请求，将执行观察行为的指令发送给相应的 ROS 节点执行，将观察结果汇总并反馈给调度智能体。

ROS 节点用于执行具体的计算和驱动任务。其中，规划节点用于执行具体的任务规划算法，知识库节点维护自主机器人对世界的认识，存储知识并根据对观察行为的观察结果对知识库进行更新以支持任务的规划。效应行为节点和观察行为节点分别用于实际执行效应行为和观察行为。

4.3 软件开发框架的复用与扩展

得益于功能组件的解耦和扩展接口的设计，基于多智能体系统的软件开发框架能够很好地复用到不同的自主机器人任务中，并且开发人员可以使用框架提供的接口对框架功能进行扩展以适用于特定的自主机器人任务。

4.3.1 功能组件的复用

在 AutoRobot 软件开发框架中，由于软件智能体、规划节点和知识库节点与具体的机器人硬件无关，如图 5 中灰色部分所示，因此可以在不同的机器人平台和任务中直接复用。这种开发方式能辅助自主机器人软件的设计和构造，并且可以降低自主机器人软件开发的复杂度，从而提高开发效率。

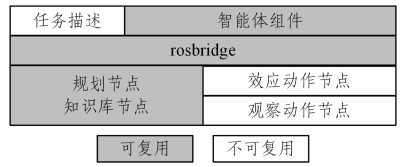


图 5 开发框架中的可复用部分和不可复用部分

Fig. 5 Reusable and non-reusable components in development framework

4.3.2 软件框架的扩展

不同的自主机器人任务具有不同的任务描述，不同的机器人能够执行的效应行为和观察行为也有所不同，因此，开发框架中的任务描述、效应行为节点和观察动作节点是不可复用的，如图 5 中白色部分所示。为了使开发框架能够应用于各种自主机器人任务中，AutoRobot 提供了针对效应行为节点和观察行为节点的扩展接口。该接口将智能体与 ROS 节点的通信功能进行了封装，使开发者无需关心不同组件间的通信细节而只需要聚焦于具体功能的实现。以编写一个新的观察行为节点为例，开发者需要编写一个 C++类继承 *SensorInterface* 并实现其中的 *concreteCallback* 函数。该函数包含一个参数 *msg*，表示此观察行为需要观察的目标，函数的返回值表示该目标的观察结果。开发者只需要在此函数中编写将传感器数据转换为机器人知识的代码逻辑，并且将观察结果通过函数返回值返回即可。

5 实验

5.1 实验场景设置

为了评估基于伴随观察模式(AOS)的任务规划与执行方法的高效性和有效性，我们将基于伴随观察模式的任务规划和执行方法与 ROSPlan[25] 和 DESPOT[15] 在仿真环境中进行

了实验对比。ROSPlan 和 DESPOT 分别采用持续性观察模式和间断性观察模式，并且均在互联网上开源，是自主机器人领域使用较为广泛的方法。ROSPlan 使用 POPF 任务规划器，维护一个先规划后执行的控制流，按顺序调度和执行计划中的每个效应行为，以期完成机器人任务。ROSPlan 提供了持续性行为监控的接口，能够对开发者指定的某些知识进行监控。ROSPlan 对于静态和完全可观察环境中的机器人任务非常有效，但由于没考虑开放环境的部分可观察性，通常无法很好地应用于开放环境中。DESPOT 是目前最有效的部分可观察马尔可夫决策过程的在线规划器之一，其能够与 ROS 很好地协同工作。它维护一个“规划-执行-观察”的迭代控制环路，可以通过每个效应行为后的观察行为来观察环境变化。然而，DESPOT 在进行任务规划时的计算量太大，导致计算效率较低，尤其是在复杂的任务规划问题上。

仿真实验场景如图 6 所示，场景中有一台 TIAGo 机器人，它配备了激光雷达、RGB-D 摄像头、机械臂和移动底座。机器人运行在一个咖啡馆中，需要在 1 号桌子处拿起一个物体，然后把它送到 2 号桌子上。机器人需要抓取物体，在没有碰撞的情况下成功导航到目标位置并放置物体才能完成此任务。我们在 Gazebo 仿真环境中进行了这项实验。在 2 号桌子附近的地方有一个客人正在移动，出于安全原因，机器人应该等待客人离开后再将物体放下。

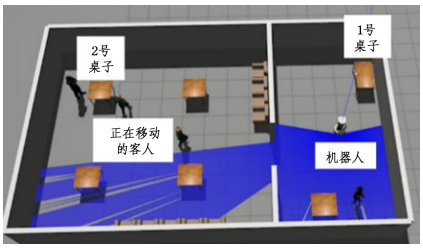


图 6 实验场景展示

Fig. 6 Simulation environment

5.2 实验结果分析

5.2.1 任务规划与完成的高效性分析

本小节通过比较 ROSPlan，DESPOT 和 AOS 这 3 种任务规划与执行方法在自主机器人物体运送任务中消耗的任务规划时间与任务完成时间，验证 AOS 在任务规划和任务完成方面的高效性。

图 7 给出了 ROSPlan，DESPOT 和 AOS 在同一自主机器人任务中的任务规划所消耗的时间。

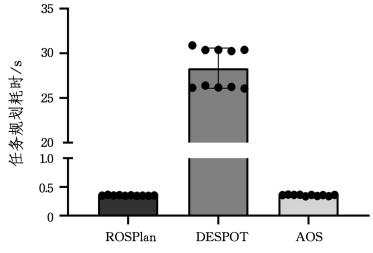


图 7 任务规划耗时对比

Fig. 7 Time costs in task planning

ROSPlan 与 AOS 所需的任务规划时间大致相当，相比之下，DESPOT 需要更多的时间来进行任务规划。AOS 消耗的规划时间包括任务规划以及伴随行为规划，通过与只有任务规划的 ROSPlan 进行规划时间消耗的对比可知，伴随观察行为规划消耗的时间维持在较低水平。伴随行为规划算法的流程如算法 1 所示，其主要耗时点在于遍历自主机器人的观察行为集合。而自主机器人的观察行为集合大小受限于其配备的传感器数量，一般不超过 20，因此遍历观察行为集合耗时很少。

图 8 给出了 3 种方法在自主机器人物体运送任务中的任务完成所需时间。ROSPlan 消耗的任务完成时间最少，原因是它没有在行为执行失败后进行任务执行恢复。由于传感器数据或者视觉识别算法可能存在误差，机器人识别到的物体位置可能和物体的实际位置并不完全一致，这种误差将导致机器人执行抓取行为后达不到预期效果。DESPOT 和 AOS 通过观察行为检测到这种异常情况后将重新识别并抓取，直到成功抓取到物体。观察行为的执行和部分效应行为的重新执行增加了整个任务的执行完成时间。

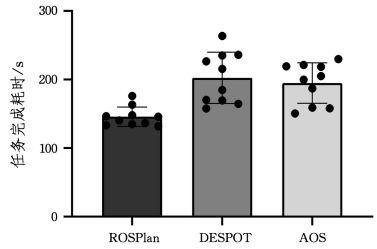


图 8 任务完成耗时对比

Fig. 8 Time costs in task execution

5.2.2 任务执行有效性分析

本小节从定量和定性两个方面来比较 3 种方法在同一自主机器人任务中的任务完成成功率，验证 AOS 的任务执行的有效性。

图 9 给出了 3 种方法在 10 次自主机器人物体运送任务中的成功率。由于在任务的执行过程中通过观察行为进行了行为策略执行监控以及异常情况恢复，AOS 与 DESPOT 比没有恢复行为的 ROSPlan 取得了更高的任务成功率。

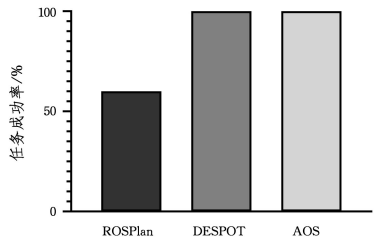
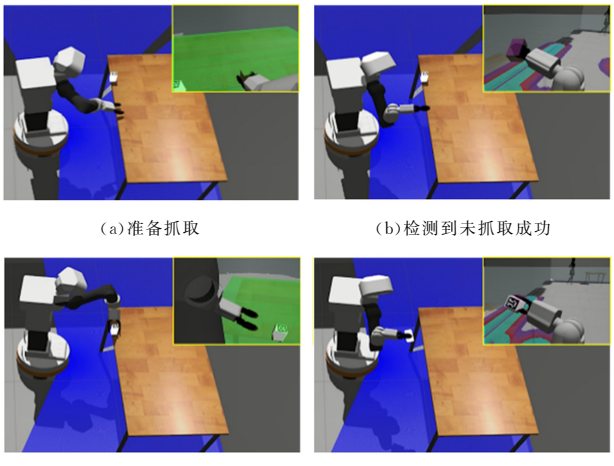


图 9 物体运送任务的成功率

Fig. 9 Success rate of object delivery task

自主机器人物体运送任务的执行可以分为 3 个阶段：抓取物体、运送物体和放下物体。图 10 给出了抓取阶段的部分截图。在使用 AOS 的任务执行过程中，机器人抓取物体后会通过观察行为检查行为的预期效果是否达成，如果预期效果未实现，机器人会重新执行该效应行为直到成功达成预期效果（见图 10(c)和图 10(d)）。值得注意的是，在使用基于伴随观察模式的方法时，机器人执行观察行为时会执行相应的效应行为来改变环境状态以获得更完整、更可靠的观察结果。

在此实验场景中，自主机器人将机械臂移动到摄像头前，以防止遮挡并获得更清晰的图像（见图 10(b)）。



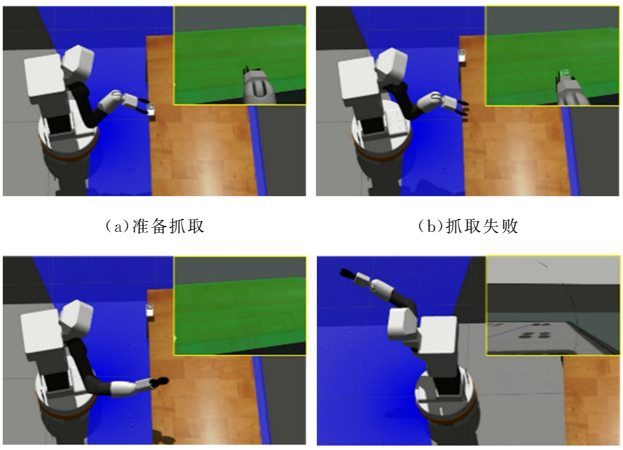
(a)准备抓取 (b)检测到未抓取成功

(c)重新抓取 (d)抓取成功

图 10 抓取阶段部分场景截图：AOS

Fig. 10 Screenshots of pick phase of AOS

图 11 给出了在相同任务中使用 ROSPlan 的部分截图。ROSPlan 在抓取行为执行完毕之后，不会检查抓取行为是否成功，而是直接执行后续的效应行为，而效应行为的预期效果未达成会影响后续效应行为的实施，最终导致任务执行失败。



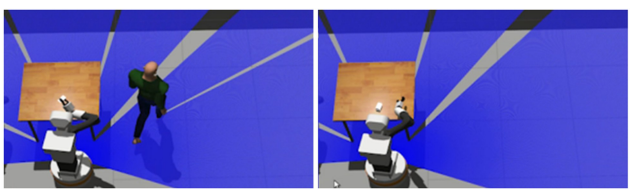
(a)准备抓取 (b)抓取失败

(c)未检测抓取结果 (d)继续执行任务

图 11 抓取阶段部分场景截图：ROSPlan

Fig. 11 Screenshots of pick phase of ROSPlan

图 12 给出了使用 AOS 的任务执行阶段在放下物体时的部分截图。由于“放下物体”效应行为的前提条件之一是机器人周围没有正在运动的客人，前提伴随观察模式会使机器人在放下物体之前等待客人的离开。

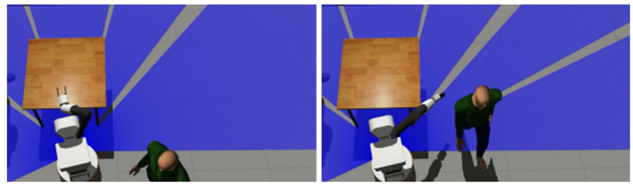


(a)等待客人离开 (b)放下物体

图 12 放置物体阶段部分场景截图：AOS

Fig. 12 Screenshots of place phase of AOS

图 13 给出了使用 ROSPlan 和 DESPOT 的任务执行过程阶段放下物体时的部分截图。



(a)放下物体 (b)客人未离开

图 13 放置物体阶段部分场景截图：ROSPlan，DESPOT

Fig. 13 Screenshots of place phase of ROSPlan and DESPOT

DESPOT 对行为的建模中不包含前提条件的限制，而 ROSPlan 在执行过程中不会检查效应行为的前提条件是否满足。在效应行为的前提条件不满足的情况下执行效应行为的结果是不可预知的，如客人在移动时机器人放下物体可能导致碰撞的发生（见图 13(b)）。

5.2.3 实现方法对比分析

本小节对实验中用到的 3 种方法进行对比，从不同方面分析实现方法对任务执行效果的影响，如表 1 所列。

表 1 ROSPlan，DESPOT 和 AOS 对比

Table 1 Comparison among ROSPlan, DESPOT and AOS

	ROSPlan	DESPOT	AOS
规划方式	任务规划	任务规划	任务规划＋伴随观察规划
观察模式	持续性监控	间断性监控	间断性监控
观察方式	传感器数据	传感器数据	效应行为＋传感器数据
异常恢复	无	有	有

从规划方式上分析，ROSPlan 和 DESPOT 只进行了任务规划来获得行为策略，AOS 在任务规划的基础上增加了伴随观察行为规划，来获得用于监控效应行为执行对应的观察行为。从观察模式上分析，ROSPlan 采用持续性监控，而 DESPOT 和 AOS 只在效应行为执行前后对环境进行观察，其观察更有针对性。在观察方式上，ROSPlan 和 DESPOT 对环境的观察局限于接收传感器数据并更新知识库，而 AOS 的观察行为能够通过效应行为来改变环境状态从而获得更完整、更可靠的观察结果。在异常情况恢复上，ROSPlan 检测到行为策略失效或效应行为执行失败后并不会进行行为策略的调整，这在一定程度上限制了其任务完成的成功率；而 DESPOT 和 AOS 能够重新执行某些效应行为或者进行重新规划以产生新的行为策略，这种方式虽然增加了任务完成时间，但能够提高任务完成的成功率。

结束语 针对动态环境中自主机器人行为策略失效和行为结果不符合预期的问题，本文提出了前提伴随观察模式和目标伴随观察模式来描述和刻画效应行为与观察行为的交互关系，加强自主机器人对其运行环境状态变化的感知能力，并在检测到上述两个问题时进行行为策略的调整。在此基础上，本文提出了相应的伴随行为规划算法和行为调度算法用于伴随观察行为的生成和调度。在仿真环境中将本文方法与 ROSPlan 和 DESPOT 进行对比，实验结果验证了伴随观察模式在任务规划层面的高效性以及任务完成方面的有效性。本文开发了一款基于多智能体系统的软件开发框架 AutoRobot，支持开发者在不同的任务和不同的自主机器人上快速实现伴随观察模式。该开发框架用若干软件智能体来抽象行为规划和调度的过程，并采用 ROS 节点来产生控制信号控制

自主机器人硬件的运作。该软件框架中的所有智能体可以在不同的自主机器人任务中直接复用。通过构建新的效应行为节点和观察行为节点，开发者可以扩展此框架以适用于某个特定的自主机器人任务。

参 考 文 献

[1] MAO X J. A Systematic Review on Software Engineering for Autonomous Robot[J]. Chinese Journal of Computers, 2021, 44(8): 1661-1678.

[2] THODUKA S, GALL J, PLÖGER P G. Using Visual Anomaly Detection for Task Execution Monitoring[C]// 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2021: 4604-4610.

[3] MIGIMATSU T, BOHG J. Object-Centric Task and Motion Planning in Dynamic Environments[J]. IEEE Robotics and Automation Letters, 2020, 5(2): 844-851.

[4] CUI G, SHUAI W, CHEN X. Semantic Task Planning for Service Robots in OpenWorlds[J]. Future Internet, 2021, 13(2): 1-19.

[5] DONDRUP C, PAPAIOANNOU I, NOVIKOVA J, et al. Introducing a ROS based planning and execution framework for human-robot interaction[C]// Proceedings of the 1st ACM SIGCHI International Workshop on Investigating Social Interactions with Artificial Agents. New York, NY, USA, Association for Computing Machinery, 2017: 27-28.

[6] YANG S, MAO X, WANG S, et al. Towards Adjoint Sensing and Acting Schemes and Interleaving Task Planning for Robust Robot Plan[C]// 2021 IEEEInternational Conference on Robotics and Automation(ICRA). 2021: 13791-13797.

[7] NOREILS F R, CHATILA R G. Plan execution monitoring and control architecture for mobile robots[J]. IEEE Transactions on Robo-tics and Automation, 1995, 11(2): 255-266.

[8] MUKHERJEE S, PAXTON C, MOUSAVIAN A, et al. Reactive Long Horizon Task Execution via Visual Skill and Precondition Models[C]// 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). 2021: 5717-5724.

[9] KASE K, PAXTON C, MAZHAR H, et al. Transferable Task Execution from Pixels through Deep Planning Domain Learning [C]// 2020 IEEE International Conference on Robotics and Automation(ICRA). Paris, France, IEEE, 2020: 10459-10465.

[10] KAELBLING L P, LOZANO-PÉREZ T. Integrated task and motion planning in belief space[J]. International Journal of Robotics Research, 2013, 32(9/10): 1194-1227.

[11] CASHMORE M, FOX M, LONG D, et al. Opportunistic planning in autonomous underwater missions[J]. IEEE Transactions on Automation Science and Engineering, 2017, 15(2): 519-530.

[12] CORUHLU G, ERDEM E, PATOGLU V. Explainable Robotic Plan Execution Monitoring Under Partial Observability[J]. IEEE Transactions on Robotics, 2021: 1-21.

[13] RAO D, HU G, JIANG Z. PRobPlan: A Framework of Integrating Probabilistic Planning Into ROS[J]. IEEE Access, 2020, 8: 106516-106530.

[14] SUAREZ R, BASANEZ L, ROSELL J. Using configuration and force sensing in assembly task planning and execution[C]// Proceedings of IEEE International Symposium on Assembly and Task Planning. 1995: 273-279.

[15] YE N, SOMANI A, HSU D, et al. DESPOT: Online POMDP Planning with Regularization[J]. Journal of Artificial Intelligence Research, 2017, 58: 231-266.

[16] GRADY D K, MOLL M, KAVRAKI L E. Extending the Applicability of POMDP Solutions to Robotic Tasks[J]. IEEE Transactions on Robotics, 2015, 31(4): 948-961.

[17] QUIGLEY M, CONLEY K, GERKEY B, et al. ROS: an open-source robot operating system[C]// ICRA Workshop on Open Source Software. 2009.

[18] METTA G, FITZPATRICK P, NATALE L. YARP: Yet Another Robot Platform[J]. International Journal of Advanced Robotic Systems, 2006, 3(1): 43-48.

[19] HELLMUND A M, WIRGES S, TAŞ Ö Ş, et al. Robot operating system: A modular software framework for automated driving[C]// 2016 IEEE 19th International Conference on Intelligent Transportation Systems(ITSC). 2016: 1564-1570.

[20] ROVIDA F, GROSSMANN B, KRÜGER V. Extended behavior trees for quick definition of flexible robotic tasks[C]// 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). 2017: 6793-6800.

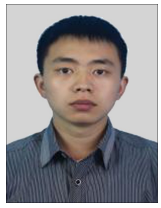
[21] COLLEDANCHISE M, NATALE L. On the Implementation of Behavior Trees in Robotics[J]. IEEE Robotics and Automation Letters, 2021, 6(3): 5929-5936.

[22] SHPIEVA E, AWAAD I. Integrating task planning, execution and monitoring for a domestic service robot[J]. Information Technology, 2015, 57(2): 112-121.

[23] PAXTON C, RATLIFF N, EPPNER C, et al. Representing Robot Task Plans as Robust Logical-Dynamical Systems[C]// 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS). Macau, China, IEEE, 2019: 5588-5595.

[24] POT E, MONCEAUX J, GELIN R, et al. Choregraphe: a graphical tool for humanoid robot programming[C]// The 18th IEEE International Symposium on Robot and Human Interactive Communication(ROMAN 2009). 2009: 46-51.

[25] CASHMORE M, FOX M, LONG D, et al. ROSPlan: Planning in the Robot Operating System[C]// Proceedings of the International Conference on Automated Planning and Scheduling. 2015: 333-341.



XUE Yuanzhou, born in 1997, postgraduate, is a student member of China Computer Federation. His main research interest is robot software engineering.

YANG Shuo, born in 1992, Ph.D, lecturer, is a member of China Computer Federation. His main research interests include intelligent robot software engineering, software engineering theory and methods and robot behavior modeling and task decision-making.

（责任编辑：李亚辉）