智能机器人工作站仿真实时系统构建方法

张国栋[1] 洪荣晶[1,2] 方成刚[1,2]

[1](南京工业大学机械与动力工程学院 江苏 南京 211816)

[2](南京工大数控科技有限公司 江苏 南京 211800)

摘 要 为实现工业机器人工作站中机器人位姿的实时监控、周遭设备的伴随仿真,使用 Python 语言基于 RoboDK软件平台开发了具有与机器人实时数据交互功能、与 PLC 信号交互功能的系统。实现机器人位姿在线监控并实时模拟以及与周遭设备伴随仿真,并设计相关人机交互面板。测试效果表明,此两大功能有效解决了虚拟工作站与实际工作站之间实时仿真问题,实现了机器人工作站伴随仿真功能。它满足了数字化工厂的需要,并拓展了相关 RoboDK 软件功能。

关键词 RoboDK 机器人 PLC 通信 伴随仿真

中图分类号 TP249 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.12.001

CONSTRUCTION METHOD OF INTELLIGENT ROBOT WORKSTATION SIMULATION REAL-TIME SYSTEM

Zhang Guodong[1] Hong Rongjing[1,2] Fang Chenggang[1,2]

[1](*School of Mechanical and Power Engineering, Nanjing Tech University, Nanjing 211816, Jiangsu, China*)

[2](*Nanjing Gongda CNC Technology Co., Ltd., Nanjing 211800, Jiangsu, China*)

Abstract In order to realize the real-time monitoring of the robot's pose and the accompanying simulation of the surrounding equipment in the industrial robot workstation, the real-time data interaction function with the robots and the PLC signal interaction function are developed based on the RoboDK software platform in Python. We realized the robot pose online monitoring and real-time simulation, as well as with the surrounding equipment simulation, and designed the relevant human-computer interaction panels. The test results show that the two functions effectively solve the real-time simulation problem between virtual workstation and actual workstation, and realize the accompanying simulation function of robot workstation. It can meet the needs of digital factory, and expand the relevant RoboDK software functions.

Keywords RoboDK Robot PLC Communication Companion simulation

0 引 言

机器人虚拟仿真方便了人们对整个机器人工作站的布局规划以及整体预览[1]。在实际工业应用中,常用的离线编程软件 robotstudio、DELMIA、SprutCAM 都需要通过存储设备将离线程序导入到机器人中,这一操作不能及时验证机器人轨迹的正确性。在机器人离线编程的过程中,采用虚拟仿真与实际设备进行联动,可以更方便地监控周遭设备的运行状态。很多仿真软件例如 roboguide 虽然实现了与机器人的信号交互,但与周遭设备之间的联动运动却不是很好,甚至根本无法做到。

针对以上实际研究过程中的不足与存在的问题,本文使用 Python 语言基于 RoboDK 软件平台开发了与周遭设备之间的信号交互功能,以及与机器人之间实

收稿日期:2019 - 07 - 15。国家自然科学基金项目(51635003);江苏省科技成果转化专项资金项目(BA2017099)。**张国栋**,硕士生,主研领域:机器人与自动化。**洪荣晶**,教授。**方成刚**,副教授。

时数据交互功能。这两大功能在实际过程中有如下用途:(1) 通过 TCP/IP 协议实现与机器人通信,可以将运动指令发送给机器人,方便检查机器人轨迹的正确性,方便及时调整,并可以实时监控机器人运行状态。(2) 通过 Python-Snap7 模块实现与周遭设备进行通信,实现与周遭设备的联动,便于在项目实行时候查看整个工作站的运行情况。

1 机器人伴随仿真系统搭建

机器人伴随仿真系统具体流程框架如图 1 所示,主要是 Python 接口函数与机器人端实现数据交互,然后通过解包等相关流程,结合 RoboDK 中 Robolink、Robodk 相关模块,实现伴随仿真。

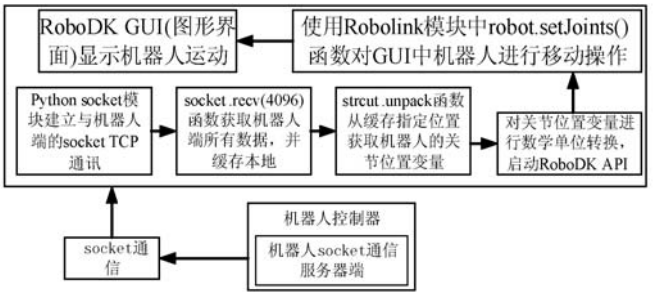

图 1 仿真框架结构图

1.1 创建机器人机构

机器人的正向运动学是旋转运动中从机器人关节变量空间到笛卡尔坐标系空间的运动变换。对于给定的一组关节变量,求取末端执行器的位置和方向是正向运动学的主要问题[2]。确定 UR 机器人的正向运动学,主要是为了在实时传输机器人的关节值之后可以准确获得相关机器人的一个实时姿态,从而在 RoboDK 的图形界面中实时显示机器人的姿态。

在 RoboDK 将机器人三维模型根据运动机构数分为 BASE、J1、J2、J3、J4、J5、J6 七个部分,保证机器人的基坐标系的位置处于工作站的原点。通过“实用程序”中“创建机构或者机器人”选项创建机器人,对于标准 6 轴机器人,通过其关节值和轴运动范围定义机器人。其中 UR 机器人的关节长度如图 2 所示。创建完成之后,机器人的 D-H 参数如表 1 所示。

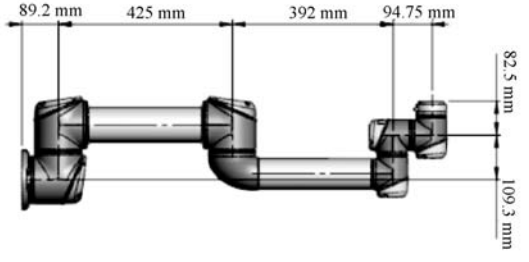

图 2 UR 机器人连杆长度

表 1 UR 机器人的 D-H 参数表

关节编号	关节变量 θ_i 最小值	关节变量 θ_i 最大值	α_i /deg	a_i /mm	θ_i /deg	d_i /mm
J1	-360	360	0	0	θ_1	89.20
J2	-360	360	90	0	θ_2	0
J3	-360	360	0	425.00	θ_3	0
J4	-360	360	0	392.43	θ_4	109.00
J5	-360	360	-90	0	θ_5	93.65
J6	-360	360	90	0	θ_6	82.00

根据 D-H 参数方法,得到机器人的相邻关节 n 到 $n-1$ 的位姿变换为:

$$^{n-1}A(\theta_n) = \mathrm{Rot}(x_{n-1},\theta_n)\mathrm{Trans}(0,0,d_n)\mathrm{Trans}(a_n,0,0)\mathrm{Rot}(x_n,\alpha_n) \quad (1)$$

式中:a_n 和 d_n 为平移距离;θ_n 和 α_n 为旋转角度。

对于六轴机械手,机器人法兰末端到机器人的基坐标系的变换矩阵为:

$$\boldsymbol{T}_n = {}^0A_1\,{}^1A_2\,{}^2A_3\,{}^3A_4\,{}^4A_5\,{}^5A_6$$

式中:${}^0A_1\,{}^1A_2\,{}^2A_3$ 确定末端法兰的位置;${}^3A_4\,{}^4A_5\,{}^5A_6$ 确定末端法兰的姿态。

1.2 通信连接获取机器人位姿信息

1.2.1 Socket 通信获取机器人信息

Socket 是 TCP/IP 网络通信中最为常用的一个 API(应用程序接口),任何网络通信都是通过 Socket 来完成的。本文所使用的 UR 机器人在 TCP/IP 协议的基础上,提供了丰富的端口用于与外部设备进行交互[3]。其中 UR 机器人 30003 为实时反馈端口,与客户端信息交流的频率为 125 Hz,通过此端口每次收到的数据为 1 044 个字节,以标准网络格式排列。主要反馈内容包括机器人的关节目标、速度、加速度、电流、扭矩等值。在 Python 中建立通信客户端,对机器人控制器中的数据进行实时读取,使用到的 Python 中提供的 Socket 模块对象函数主要如表 2 所示。

表 2 Socket 对象函数

函数	描述
socket. create_connection ((ROBOT_IP, ROBOT_PORT))	与机器人控制器建立连接,其中:ROBOT_IP 为机器人 IP,ROBOT_PORT 为机器人选择的通信端口号
socket. recv(bufsize[, flags])	从套接字接收数据,最大数量的数据由 bufsize 指定
socket. close()	关闭套接字

客户端与服务器端(机器人端)连接时的时序图如图 3 所示。

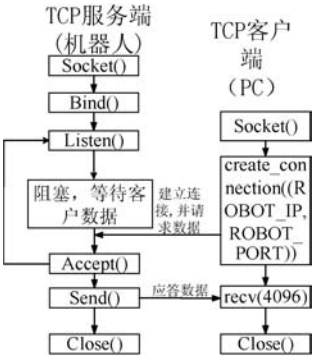

图 3　面向连接 TCP 的时序图

通过 Socket 连接之后,客户端接收由机器人 30003 端口发送出来的机器人所有状态信息,返回信息以字节形式返回。其中客户端接收机器人的主要信息如下:

dic = {'MessageSize': 'i', 'Time': 'd', 'q target': '6d', 'qd target': '6d', 'qdd target': '6d','I target': '6d',

'M target': '6d', 'q actual': '6d', 'qd actual': '6d', 'I actual': '6d', 'I control': '6d',

'Tool vector actual': '6d', 'TCP speed actual': '6d', 'TCP force': '6d', 'Tool vector target': '6d',

'TCP speed target': '6d', 'Digital input bits': 'd', 'Motor temperatures': '6d', 'Controller Timer': 'd',

'Test value': 'd', 'Robot Mode': 'd', 'Joint Modes': '6d', 'Safety Mode': 'd', 'empty1': '6d', 'Tool Accelerometer values': '3d',

'empty2': '6d', 'Speed scaling': 'd', 'Linear momentum norm': 'd', 'SoftwareOnly': 'd', 'softwareOnly2': 'd', 'V main': 'd',

'V robot': 'd', 'I robot': 'd', 'V actual': '6d', 'Digital outputs': 'd', 'Program state': 'd', 'Elbow position': '3d', 'Elbow velocity': '3d'}

1.2.2　struct 函数解析数据

在 TCP/IP 协议进行诸如 Int、char 之类的数据传输的时候,服务器端需要某种机制将某些特定的结构体的类型打包成二进制流的字符串后进行网络传输,而客户端也应该经过某种机制进行解包还原原来的结构体数据[4]。从机器人控制器端发送来的 1 044 个字节,对其进行解包获得相应的数据,对照机器人 30003 端口发送的数据表得知前 256 个字节的排列顺序以及相关内容如表 3 所示。

表 3　实时反馈数据包

字节顺序	内容
1 ~ 4	整段数据包的字节数
5 ~ 12	控制器控制时间
13 ~ 255	目标点的关节位置等信息
256	实时关节位置

通过 struct. unpack_from( ) 对接收到的机器人的信息进行解包,其流程如图 4 所示。

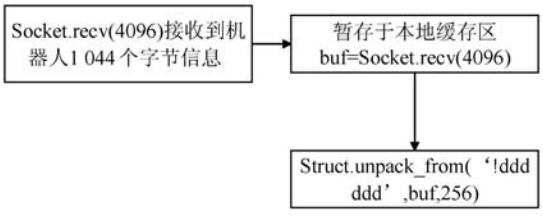

图 4　数据解包流程

Struct. unpack_from( ) 函数中参数分别表示:格式化字符串格式、缓冲区、偏移量。函数定义方法如下:根据格式化字符串格式从位置偏移处开始从缓冲区解包。缓冲区大小(以字节为单位,减去偏移量)至少为格式所需要的大小。图 4 表示从缓存区第 256 个字节开始解包,解包数据格式为"! dddddd",即 6 个整数,按网络字节顺序进行排列,即得到机器人 6 个实时关节值。

1.3　机器人模型运动仿真

通过 1.1 节、1.2 节获得机器人的实时关节数据。机器人传输过来的值为 UR 所规定的格式,均为弧度制。需要对其利用相关的数学公式转化为度数,然后利用 Robolink 模块对应函数,实现对模拟环境中机器人的动作控制。

程序中可设置刷新时间以及进行多线程编程,可以让机器人进行数据获取的同时执行模拟环境中机器人移动操作。在 RoboDK 中,通过其提供的基于 Python 的 API,将工业机器人虚拟环境中的三维运动与实际相结合。用于 Python 的 RoboDK API 分为以下两个模块:Robolink 模块,Robodk 模块。Robolink 模块主要功能为:检索 RoboDK 工作站树中的任何对象,由 Item 对象表示;根据 Robolink. Item 类对该项执行不同的操作。本次运用到的 Robolink. ltem 类函数主要是 setJoints( ),其主要说明为:设置机器人或目标的当前关节。如果设置了机器人关节,则将在屏幕上更新机器人位置。

对从上述 Socket 通信和解包所得到的真实机器人的关节值进行相关数学转换,转换程序如下:

```
def on_packet(packet):
    global ROBOT_JOINTS
    #从数据包中检索所需信息
    rob_joints_RAD = packet_value(packet,UR_GET_JOINT_POSITIONS)
    #print(rob_joints_RAD)
    ROBOT_JOINTS = [ji * 180.0/pi for ji in rob_joints_RAD]
```

转换完成之后得到的机器人的关节值为度数,将变量放入到 setJoints 函数中即可。在此之前,需要获

取已经配置好的机器人名称,其中更新位姿的主要流程如图 5 所示。

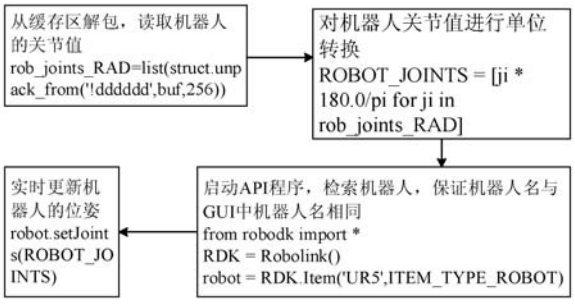

图 5 更新位姿流程

上述过程完成之后,在 RoboDK 客户端实时获取实际机器人的末端位姿,在 UR5 机器人控制器中固定 IP,启用机器人以太网通信。通过 UR 机器人控制面板 polyscope 面板控制机器人运动,即可在 RoboDK 中实时显示机器人位姿,如图 6 所示。其中,左端为离线编程的环境中机器人位姿,右端为机器人 polyscope 控制面板。可以看出,实际机器人的关节值实时传递给虚拟环境中的机器人,到达位姿一致,且可以实时更新,保证机器人控制面板和虚拟机器人面板的关节值均保持相同。

图 6 实验结果图

2 机构仿真系统搭建

在工业机器人的离线编程的实际应用中,机器人常会与周遭设备进行互动,例如焊接时,焊接机器人往往需要与变位机设备进行互动,如果在仿真场景中实时反馈变位机的位姿,对机器人的离线编程具有很大的帮助。在自动化系统之中,很多设备都是由 PLC 监控其运行,通过 PLC 可以知道当前设备的一个运行状态、位置。可以在虚拟场景中得知 PLC 相关状态的信号,然后将这些信号通过 RoboDK 中 Robolink、Robodk 两大模块,虚拟出实际场景中运动机构的动作。机构半实物仿真相关过程如图 7 所示。

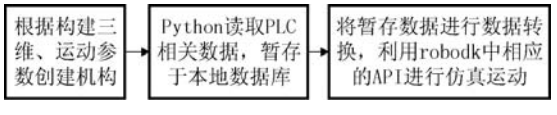

图 7 机构半实物仿真过程

2.1 Python 读取 PLC 数据

Python 与西门子 PLC 进行通信,主要是靠 snap7 套件。snap7 是一个开源的 32/64 位多平台以太网通信套件[5],主要用于与西门子 S7 系列 1200、1500 的 PLC 进行本地连接。Python-snap7 是 snap7 库的Python 包装器。通过 RoboDK 中嵌套的 Python API 接口,安装好 Python-snap7 模块、snap7 模块。完成环境搭建工作之后,方可与相关的 PLC 之间进行通信。

结合 Python-snap7 的模块函数的相关分析以及相关文档,Python-snap7 库中的函数主要类方法是 read_area 和 write_area[6],函数中主要的函数参数为需要提供 PLC 的区域地址(area、dbnumber)、起始偏移地址(start)、读和写的数据长度(size/data)。PLC 能提供如下信息:PLC 的存储区通过 tag 的形式与存储区间关联,分为输入(I)、输出(O)、位存储区(M)、数据块(DB)。程序在访问对应的(I/O)tag 时,是通过访问 CPU 中的过程影响存储区对应的地址进行操作的。函数参数中 area、size 的取值与数据区域、数据类型的对应关系如表 4 所示。

表 4 函数参数对应关系表

基本数据类型	size 值	数据区域	area 键	area 值
布尔型	1	输入存储区	PE	0x81
字节型	1	输出存储区	PA	0x82
字型	2	内部标志位存储区	MK	0x83
双字型	4	数据块	DB	0x84
实数型	4	计数器	CT	0x1C
计数器型	2	定时器	TM	0x1D

从 PLC 的用户手册可知,对于 M3.4,对应的就是 MK(0x83),对应的起始地址是 3,对应的 bit 位为 4,即数据类型为 bit。在读取该位即调用函数的时候,其变量需要设置为 area = snap7. snap7types. areas. MK, dbnumber = 0, start = 3. 3, size = 1,即可读取关于 M3.4 的状态。为了更好地对 PLC 状态进行读写,现通过 Python 语言实现了 PLC 程序监控测试面板,并且通过 RoboDK 中嵌入的 API 接口将面板嵌入其中,如图 8 所示,通过该交互界面可以实现以下功能:

(1) PLC 状态读取。通过 GUI 界面中输入 PLC 的 IP,确定其机架号、插槽号;并且设置相关的读取数据区域,其中读写数据的参数主要由表 4 到表 6 决定,设置好相关参数之后,点击“查询数据”按钮,即可在最下方文本显示框中看到相关数据。

(2) PLC 数据写入。先设置好要写入的数据区域(基本步骤参见第一步),然后设置要写入的数据,如要写入 MB0 中的 M0.3、M0.4 为 1,设置好读写区域之后,

将准备发送的数据写为“2#0001 1000”或“24”即可。

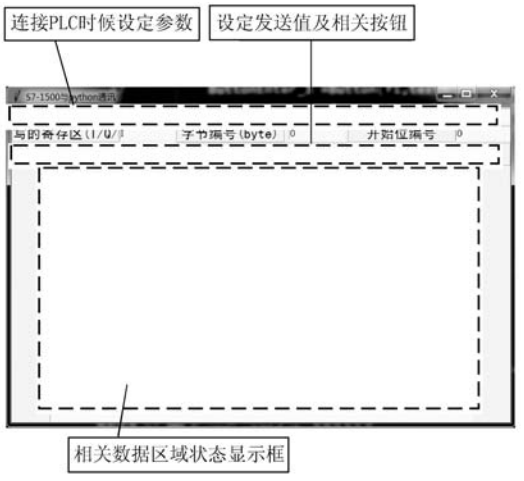

图 8　PLC 程序监控面板

2.2　设备仿真

将上述功能面板进行精简化之后,所得 PLC 信号监控面板如图 9 所示。为了方便对机构的控制,通过博途软件进行 PLC 程序的编写,将梯形图程序下载到 PLC 中之后,通过 PLC 信号设置精简面板,设置相对应的 PLC 输入信号以及监控相关输出信号,驱动相应机构运动,并且设置相关的仿真机构的运动。本次仿真机构为变位机,其驱动程序如图 10 所示。

图 9　PLC 信号设置精简面板

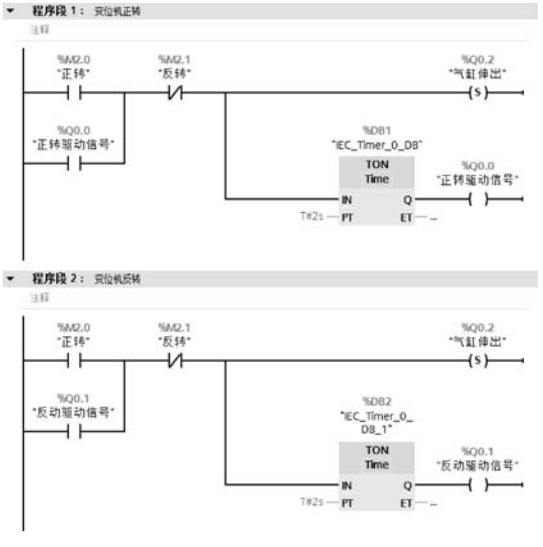

图 10　变位机驱动程序

变位机运动由三个电磁阀控制,其中:阀 1、阀 2 分别控制变位机正转、反转;阀 3 控制压紧气缸的伸缩,并且有一光电传感器开关为变位机运动到位的标记信号。M2.0、M2.1 控制变位机的正反转,如表 5 所示。对 PLC 进行值写入时,直接对 MB2 进行操作即可。

表 5　变位机状态与 MB2 取值对应表

变位机状态	M2.0 取值	M2.1 取值	MB2 取值
正转	1	0	1
反转	0	1	2
零位	0	0	0
	1	1	3

通过变位机的三维创建其机构,如图 11 所示。设置其变位机机构回转速度和回转加速度与真实机构一致,将仿真机构的驱动信号设置为 PLC 的等待信号。程序结构如图 12 所示。

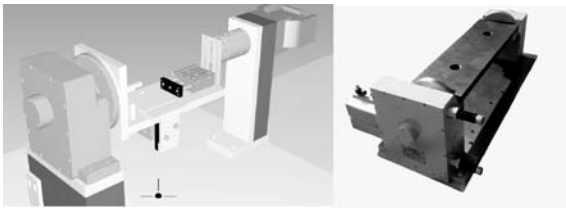

图 11　创建变位机机构

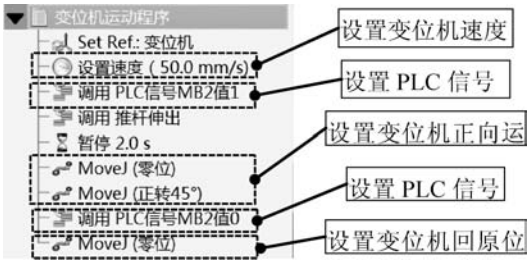

图 12　变位机运动程序

通过驱动该程序,设置 MB2 信号为 1,真实变位机正转,同时变位机仿真程序也开始运行。设置 MB2 信号为 0,真实变位机回零位,同时仿真程序驱动虚拟变位机回零位[7]。

3　结　语

本文通过 RoboDK 中 Python 的 API 接口,实现机器人关节目标值的读取,并且利用了 RoboDK API 接口中 Robolink 模块,完成了从实际机器人控制虚拟机器人的伴随仿真。拓展了该软件的新功能,并且可在其他支持 TCP/IP 通信的机器人上进行相关测试。该功能可在离线环境中实时观测机器人的位置,便于机器人位置的监控,且可以方便机器人创建相对工件坐标系。对于离线轨迹的创建也较为方便,在离线仿真中意义重大。同时也实现与 PLC 控制系统之间的信号交互,可实现与实际设备之间的伴随仿真。目前该

(下转第 12 页)

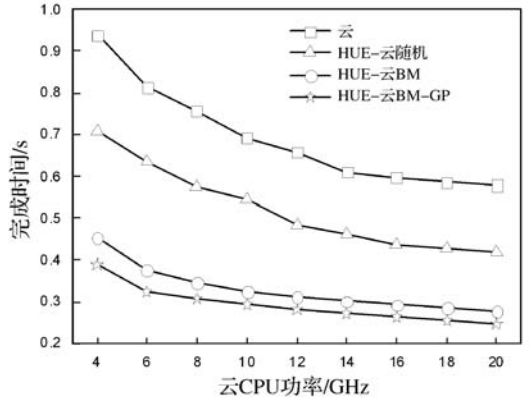

图 5 不同卸载方案的完成时间对比(CUE = 70)

在图 6 中,通过详细检查特定 CUE 任务的卸载延迟和计算时间,进一步了解不同方案的性能。通过分析不同方案在本地、HUE 与云的计算时间和卸载时间发现,计算时间在不同的处理器之间很平衡,并且卸载延迟时间在总时间中所占比例不高。

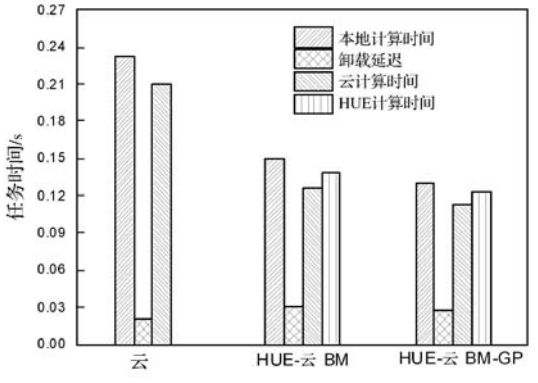

图 6 不同卸载方案的计算时间和卸载延迟对比

5 结 语

本文提出一个将计算卸载到边缘云和移动对等端的通用框架。该设计旨在保持应用程序延迟需求的同时,最大限度地减少总计算时间。数值结果表明,通过带宽激励,将计算卸载到边缘云和移动对等端的方案是可行的。与将计算卸载到边缘云方案相比,该方案具有较大的性能增益,在卸载延迟时间没有明显增加的情况下,总计算时间可以减少 35% ~40% 。

参 考 文 献

[1] Abbas N, Zhang Y, Taherkordi A, et al. Mobile edge computing: A survey[J]. IEEE Internet of Things Journal, 2018, 5(1): 450 - 465.

[2] Tran T X, Pompili D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(1): 856 - 868.

[3] Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628 - 1656.

[4] 于博文, 蒲凌君, 谢玉婷,等. 移动边缘计算任务卸载和基站关联协同决策问题研究[J]. 计算机研究与发展, 2018, 55(3): 537 - 550.

[5] Mao Y, You C, Zhang J, et al. A survey on mobile edge computing: The communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322 - 2358.

[6] 柳兴, 李建彬, 杨震, 等. 移动云计算中的一种任务联合执行策略[J]. 计算机学报, 2017, 40(2): 364 - 377.

[7] 张文柱, 曹琲琲, 周雪婷. 移动云环境下高能效的移动终端计算迁移策略[J]. 西安电子科技大学学报(自然科学版), 2017, 44(3): 175 - 180.

[8] 曹傧, 梁裕丞, 罗雷, 等. ad hoc 云环境中分布式博弈卸载策略[J]. 通信学报, 2017, 38(11): 24 - 34.

[9] Ti N T, Le L B. Computation offloading leveraging computing resources from edge cloud and mobile peers[C]//2017 IEEE International Conference on Communications(ICC),2017.

[10] Du J, Zhao L, Feng J, et al. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee[J]. IEEE Transactions on Communications, 2018, 66(4): 1594 - 1608.

(上接第 5 页)

系统功能测试阶段已过,处于程序优化阶段,但基本功能已经实现。今后将继续优化该系统的两大功能,进一步完善 RoboDK 相关拓展功能[8-9]。

参 考 文 献

[1] 邓华健. 机器人离线编程系统的开发及其应用[D]. 广州:广东工业大学,2017.

[2] 李双双. 工业机器人建模、运动仿真与轨迹优化[D]. 呼和浩特:内蒙古大学,2012.

[3] 俊杰. 基于无线网络的机器人远程控制系统集成平台研究[D]. 北京:北京交通大学,2012.

[4] 罗瑜. 远程监控系统中网络通信的研究与实现[D]. 武汉:武汉理工大学,2006.

[5] 魏学舟,刘涛. 基于 Snap7 的 PLC 上位机监控软件开发[J]. 设备管理与维修,2018(14):129 - 131.

[6] 李华,安磊. 利用 Python Snap7 实现上位机与西门子 PLC 的通信[J]. 数字化用户,2018,24 (28):20.

[7] 王博,黎柏春,杨建宇,等. 智能制造系统的 6R 工业机器人仿真和监控平台[J]. 哈尔滨工程大学学报,2019,40(2):365 - 373.

[8] 刘和彬. 基于 VR 的工业机器人任务仿真与实时监控研究[D]. 沈阳:东北大学,2013.

[9] Rossmann J, Hempe N, Emde M, et al. A real-time optical sensor simulation framework for development and testing of industrial and mobile robot applications [C]//ROBOTIK 2012,7th German Conference on Robotics,2012.