Proceedings of 2018 IEEE
International Conference on Mechatronics and Automation
August 5 - 8, Changchun, China

Development of Hardware-in-Loop Simulation Platform for Collaborative Robots Based on LinuxCNC and V-rep

Li Tongtong, Yang Tao
Beijing Institute of Precision Mechanical and Electrical Control Equipment
Beijing, China
litongbit@163.com, yangtaoyt507@126.com,

Yang Zelin , Liu Shuxuan, Li Jianming
Beijing Institute of Precision Mechanical and Electrical Control Equipment
Beijing, China
hn_yangzl@163.com, liushuxuan808@126.com

***Abstract* - Cooperative robots are currently the leading research and development direction of the robotics industry.Developing robotic systems often requires a combination of software and hardware. However, due to its complexity, making a real robot itself proposes great challenges to developers. This article describes a hardware-in-the-loop simulation control development platform based on Linux CNC and V-rep simulation software. In the real-time Linux system, we build a general-purpose, modular cooperative robot control system; and we establish a robot model in the V-rep. Data communication between them can be achieved through data interaction module. Thanks to the modularity of the control system, developers are free to add algorithmic modules. In this paper, a six-degree-of-freedom lightweight robotic arm model is taken as an example to serve as a virtual controlled object of the above cooperative robot control system. Afterwards, this paper establishes the forward and inverse kinematics algorithm and dynamics algorithm of the robot separately, and verifies the validity of this hardware-in-loop simulation development platform.**

Index Terms* - Cooperative robots，*Hardware in loop simulation, Kinematics and Dynamics Modelling.

I. Introduction

Cooperative robot has become one of the key areas of research in academia and industry, and some scholar have carried out some work in different applications [1-2]. In the process of robotics research, both mechanical structures and control algorithms need to be verified through experimental procedures.But the real construction of a robotic system requires multidisciplinary professionals such as machinery, electrical engineering, control, and automation to invest a lot of material and financial resources, and it is difficult to produce results in a short period of time, which has become a major obstacle to academic researchers [3]. When the researchers validate the robot control algorithm through experiments, the engineering implementation problem is a major obstacle for researchers. Therefore, simulation becomes one of the important methods in the development of robotics technology [4]. It can verify scientific theories and avoid engineering problems. However, total simulation seems to lose the significance of theoretical research. Therefore, many scholars at present use hardware in loop simulation to verify the validity of the theoretical algorithm [5-8].For the research and development of the control algorithm, the engineering implementation of the algorithm mainly depends on the control system if we can get the information of the robot body, such as position, speed, acceleration, current and other information. In this paper, a modular robot control system is developed based on Linux system. The robot body uses V-rep software to simulate and information exchange between them can be done through TCP/IP, which not only meets the need for verification of control algorithms, but also avoids many problems caused by the implementation of the robot arm body engineering. At the same time, a 6-DOF robot kinematics module and a dynamic module are added [9-10].

LinuxCNC is one of robot control system software that runs on most standard PCs loaded with a Linux operating system and supports G code programming and control automation equipment. In the traditional field can control such as milling machines, lathes, 3D printers, laser cutting machines and other equipment. With the development of the robotics industry, LinuxCNC is also expanding into the robotics industry. At this stage, it has been able to successfully control equipment such as tandem robots and hexapod robots [11].

V-rep is an open source simulation software for robot simulation developed rapidly in recent years. It integrates a large number of commonly used functions of robots, including kinematics calculations and dynamic calculations [12]. V- rep is used for rapid algorithm development, factory automation simulation, rapid prototype verification, robot-related education, remote monitoring, and double safety inspection. V- rep has an integrated development environment based on a distributed control architecture: each object/model can be controlled individually through embedded scripts, plug-ins, ROS nodes, remote API clients, or custom solutions. And it has very rich external interface which can be written in Python.

Therefore, V-rep can use python interface to communicate with LinuxCNC, and can provide robot body in simulation environment for LinuxCNC robot control system for controller debugging.

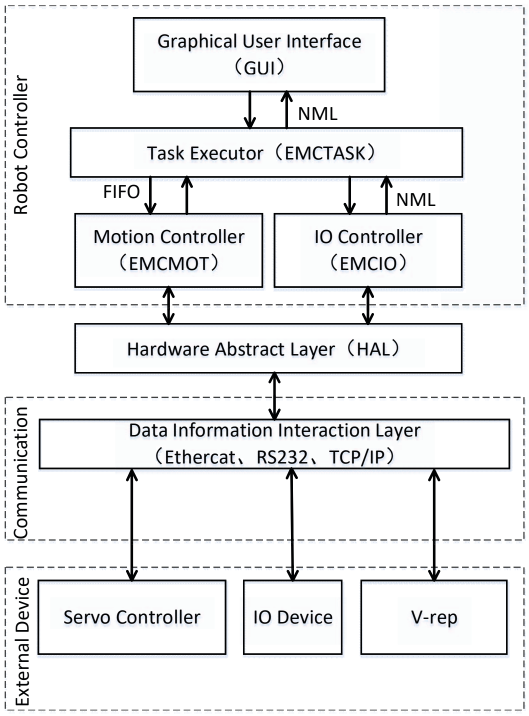
The structure of the article is as follows: The second chapter mainly introduces the establishment of LinuxCNC control system, the establishment of V-rep simulation model and the establishment of interactive interfaces. The third chapter is the realization of the kinematics and dynamics module of the 6-DOF manipulator. The fourth chapter is the experimental part to verify the operability of the system.

II. Cooperative robot control system

As a core component of the robot system, the robot control system is developed in an open, real-time, modular, intelligent, general-purpose, and low-cost design philosophy. It is dedicated to solving common problems of robot control systems and providing real-time. The stable system platform provides a unified control system platform for robot arm products of different configurations and different requirements.

The control system is mainly composed of four major parts, including a graphical user interface (GUI), a task executor controller (EMCTASK), a motion controller (EMCMOT) and a discrete IO controller (EMCIO), like in fig.1 as follow.

Graphical user interface (GUI) supports Jog speed control mode and MDI G code programming control mode. It is responsible for accepting user-set data and transferring it to the task controller. At the same time, data returned from the task controller is updated on the human-machine interface. Presented to the user, the Real-Time Operator Interface (RTAI) is used to provide the best real-time performance. The patched kernel real-time interface allows users to write out applications under demanding real-time requirements.


Robot Controller
Graphical User Interface (GUI)
NML
Task Executor (EMCTASK)
FIFO
NML
Motion Controller (EMCMOT)
IO Controller (EMCIO)
Hardware Abstract Layer (HAL)
Communication
Data Information Interaction Layer (Ethercat、RS232、TCP/IP)
External Device
Servo Controller
IO Device
V-rep
Fig.1 The architecture of control system

The task controller (EMCTASK) is the system hub and receives the motion instructions or G code instructions of the human-machine interface. The generated motion-related instructions are sent to the motion control through the shared memory in the real-time control system (RCS) library and NML (Neutral Message Language).

The motion controller (EMCMOT) is a real-time module that is responsible for completing the trajectory planning, forward and inverse kinematics operations, and finally completing the interpolation calculation of the position instructions of each axis. The position command is sent to the hardware through the HAL pin. On the other hand, the status of Motion is also fed back to the task controller (EMCTASK) in real time. At the same time, an error handling mechanism was established between different IO modules and the task controller, and the error was returned to the task controller for further processing.

The discrete IO controller is mainly responsible for handling the input and output control such as emergency stop, control power, and power, etc. It is a cyclical task that is responsible for outputting IO control through the HAL pin after receiving an IO command from the Task and from the HAL pin. Read the IO status of the feedback and update it to TASK.

The hardware abstraction layer (HAL) technology is also used in the entire software architecture of LinuxCNC. The hardware abstraction layer abstracts the system hardware into a software interface. Through the operation of the pins, the communication between the system and the hardware is realized, the internal details of the hardware are hidden, and the programming and use processes are simplified and hierarchical. The introduction of the HAL mechanism brings many advantages to LinuxCNC, such as modularization, unified driving, and modular implementation of software algorithms, all of which are convenient for researchers. For example, LinuxCNC can communicate with external hardware through Ethercat, RS232, Socket, and other communication interfaces. In this article, socket technology is used to communicate with V-rep simulation software, which will be introduced in later chapters.

III. Building Simulation environment

A. *Modelling in V-rep*

Establishing a robot model in V-rep can be divided into the following sections:

(1) Establishing kinematic model;
(2) Establishing dynamic model;
(3) Completing control algorithm code, such as threading or non-threaded scripts;
(4) Completing external interface program when needed.

There are generally two methods for building a kinematics model by V-rep. One is to model it by its own 3D software; the other is to convert the file format that can be opened by V-rep to build after the model is built by other 3D software, and then perform kinematics construction. The latter method is often used. Because V-rep computes the dynamic model, it will consume a lot of memory and CPU of the PC, which makes the simulation process very long. Therefore, when using the three-dimensional software to model it, it does not use a three-dimensional model with a real structure, but only guarantees the V-rep model. Consistent with the external physical contour of the real model, the internal structure is simplified as much as possible, and the dynamic parameters are evaluated by other three-dimensional models. In this paper, the three-dimensional model of the six-degree-of-freedom manipulator is firstly designed by SolidWorks, simplified to the ".stl" format, and a six-freedom manipulator simulation model is established by V-rep. Its D-H parameters are as follows:

TABLE I
D-H PARAMETERS

i	a(mm)	α(rad)	d(mm)	θ(rad)
1	0	0	$d_1 = 123.5$	θ_1
2	0	$\alpha_1 = 1.57$	0	θ_2
3	$a_2 = 436$	0	$d_3 = 32.5$	θ_3
4	$a_3 = 436$	$\alpha_3 = 3.14$	$d_4 = 136.4$	θ_4
5	0	$\alpha_4 = -1.57$	$d_5 = 136.4$	θ_5
6	0	$\alpha_5 = 1.57$	$d_6 = 86.4$	θ_6

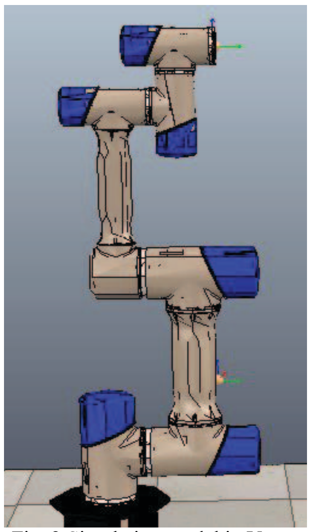


Fig. 2 Simulation model in V-rep

B. *Communication between LinuxCNC and V-rep*

The linuxCNC communicates with V-rep through the Socket interface. Socket is a kind of TCP/IP communication protocol. This article uses python code to write programs between linuxCNC and V-rep. The data exchange structure between the two is shown in the following figure. The robot control system sends the joint position command to V-rep after inverse kinematics calculation; at the same time, V-rep feeds back the current position and velocity of each joint of the robot control system, acceleration, output torque and so on.

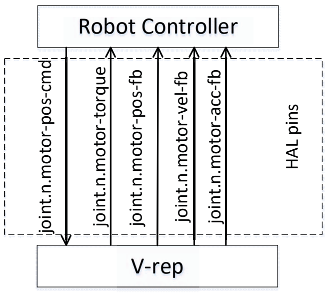


Fig.3 Communication between controller and V-rep

IV. KINEMATICS AND DYNAMICS *MODELLING*

A. *Kinematics Modelling*

It is known that the pose transformation matrix of the current mast frame coordinate system with respect to the previous bar coordinate system is:

$$ {}^{i-1}_{\ \ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

By coordinate transformation, we can get the pose matrix of the TCP in the world coordinate system:

$$ {}^0_6T = {}^0_1T * {}^1_2T * {}^2_3T * {}^3_4T * {}^4_5T * {}^5_6T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

In which:

$$n_x = c6 * (s1 * s5 + c234 * c1 * c5) + s234 * c1 * s6$$
$$n_y = s234 * s1 * s6 - c6 * (c1 * s5 - c234 * c5 * s1)$$
$$n_z = s234 * c5 * c6 - c234 * s6$$
$$o_x = s234 * c1 * c6 - s6 * (s1 * s5 + c234 * c1 * c5)$$
$$o_y = s6 * (c1 * s5 - c234 * c5 * s1) + s234 * c6 * s1$$
$$o_z = -c234 * c6 - s234 * c5 * s6$$
$$a_x = c234 * c1 * s5 - c5 * s1$$
$$a_y = c1 * c5 + c234 * s1 * s5$$
$$a_z = s234 * s5$$
$$p_x = d_3 * s1 - d_4 * s1 + d_5 * s234 * c1 + a_3 * c23 * c1 + a_2 * c1 * c2 - d_6 * c5 * s1 + d_6 * c234 * c1 * s5$$
$$p_y = d_4 * c1 - d_3 * c1 + d_6 * (c1 * c5 + c234 * s1 * s5) + d_5 * s234 * s1 + a_3 * c23 * s1 + a_2 * c2 * s1$$
$$p_z = d_1 - d_5 * c234 + a_3 * s23 + a_2 * s2 + d_6 * s5 * s234$$

In which:

$$s_m = \sin\theta_m,$$
$$c_m = \cos\theta_m,$$
$$s_{mn} = \sin(\theta_m + \theta_n),$$
$$c_{mn} = \cos(\theta_m + \theta_n),$$
$$s234 = \sin(\theta_2 + \theta_3 - \theta_4),$$
$$c234 = \cos(\theta_2 + \theta_3 - \theta_4)$$

Using the (2,3) elements of the left and right sides of the Eq.(2) to be equal, the angle of the joint 1 can be obtained:

$$ {}^0_1T^{-1} * {}^0_6T = {}^1_2T * {}^2_3T * {}^3_4T * {}^4_5T * {}^5_6T \tag{2} $$

$$q1 = \operatorname{atan2}\left(d_6 * a_y - p_y, d_6 * a_x - p_x\right) - \operatorname{atan2}\left(d_3 - d_4, \pm\sqrt{(d_6 * a_y - p_y)^2 + (d_6 * a_x - p_x)^2 - (d_3 - d_4)^2}\right)$$

Using the (2, 1), (2, 2), (2, 3) elements of the two matrices in Eq.(2) is equal, we get:

$$s5 = \pm\sqrt{(n_y * c1 - n_x * s1)^2 + (o_y * c1 - o_x * s1)^2}$$
$$c5 = a_y * c1 - a_x * s1$$
$$q5 = \operatorname{atan2}(s5, c5)$$

Using the (2,1) elements of the two matrices equal:

$$\cos(q6) = \frac{n_x * s1 - n_y * c1}{s5}$$
$$\sin(q6) = \frac{-o_x * s1 + o_y * c1}{s5}$$
$$q6 = \operatorname{atan2}(\sin(q6), \cos(q6))$$

The second rod coordinate system is available in the matrix transformation of the fourth rod coordinate system.

$${}^0_1T^{-1} * {}^5_6T^{-1} * {}^4_5T = {}^1_2T * {}^2_3T * {}^3_4T$$

Using the (1, 4) (3, 4) elements of the two matrices equally available:

$$p_{xx} = p_x * c1 - d_6 * \left(a_x * c1 + a_y * s1\right) - d_5 * \left(c6 * \left(o_x * c1 + o_y * s1\right) + s6 * \left(n_x * c1 + n_y * s1\right)\right) + p_y * s1$$

$$p_{zz} = p_z - d_1 - a_z * d_6 - d_5 * (o_z * c6 + n_z * s6)$$

$$d = (p_{xx}^2 + p_{zz}^2 + a_2^2 - a_3^2)/2a_2$$

$$q2 = \mathrm{atan2}(p_{xx}, -p_{zz}) - \mathrm{atan2}\left(d, \pm\sqrt[2]{p_{xx}^2 + p_{zz}^2 - d^2}\right)$$

Through coordinate transformation, 6-joint transformation matrix in 2-joint coordinates:

$${}^1_2T^{-1} * {}^0_1T^{-1} * {}^0_6T = {}^2_3T * {}^3_4T * {}^4_5T * {}^5_6T \quad (3)$$

The (1,3)(2,3) elements of the two matrices in Eq. (3)are equal and the (1,4)(2,4) elements are equal:

$$c3 = \left(\left(p_z * s2 - d_1 * s2 + p_x * c1 * c2 + p_y * c2 * s1\right) - (a_2 + d_5 * \sin(3-4) + d_6 * \cos(3-4) * s5)\right)/a_3$$

$$s3 = \left(\left(p_z * c2 - d_1 * c2 - p_x * c1 * s2 - p_y * s2 * s1\right) - (-d_5 * \cos(3-4) + d_6 * \sin(3-4) * s5)\right)/a_3$$

$$q3 = \mathrm{atan2}(s3, c3)$$

And then:

$$q4 = q3 - \mathrm{atan2}(\sin(3-4), \cos(3-4))$$

Joints 1, 5, and 2 have two sets of solutions for each joint. The robotic arm has a total of 8 sets of analytical solutions.

B. *Dynamics Modelling*

According to the rigid body dynamic model method [13], the robot is divided into 6 rigid bodies according to the physical rotation surface cut, which is called connecting rod 1 to connecting rod 6, and the symbol Link-1. Establish a fixed link coordinate system consistent with the kinematic coordinate system, as shown in Figure 2, all joint rotation axis is defined as Z axis, Z axis + direction from the motor to the harmonic direction, the symbol Bi. The coordinate system COGF of the connecting rod is established. The origin of the coordinate system is at the COM (Center of Mass) of the connecting rod, and the direction of the coordinate system is consistent with the direction of the corresponding connecting rod fixed coordinate system.

The model dynamics parameters are as follows:

(1) Mass of connecting rod: m_i;

(2) COM vector: the coordinate representation of the link-I center of mass in the link coordinate system B_i;

(3) Inertia tensor of connecting rod: The inertial tensor matrix of the connecting rod is established based on the coordinate system of the connecting rod center of mass;

(4) Coupling link coordinate system position vector: The position of the original link coordinate system is represented in the upper link coordinate system;

(5) Coupling coordinate system attitude vector of link: The attitude of the origin coordinate system of the link in the upper level link coordinate system;

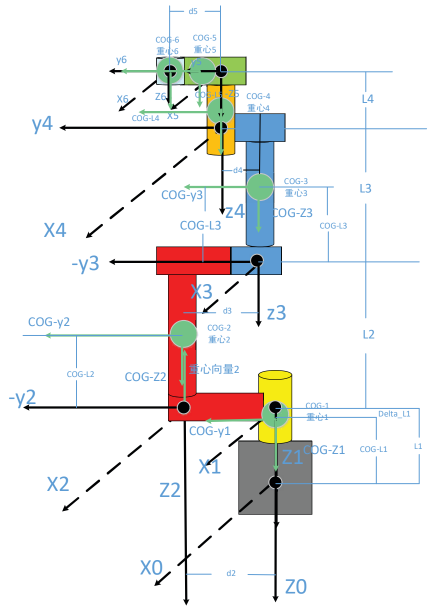


Fig. 4 Dynamics model

Using Newton's Euler formula, the inertia force and moment of inertia acting on the center of the connecting rod can be calculated:

$$F_i = m\dot{v}_{C_i}$$

$$N_i = {}^{C_i}I\dot{\omega}_i + \omega_i \times {}^{C_i}I\omega_i$$

The origin of the coordinate system C_i is located at the center of mass of the connecting rod, and the orientation of each coordinate axis is the same as that of the original link coordinate system.

Let f_i be the force acting on connecting link i by connecting rod i-1, and n_i the moment acting on connecting rod i by connecting rod i-1. Add all the forces acting on the link i to get the force balance equation:

$${}^iF_i = {}^if_i - {}_{i+1}^{i}R^{i+1}f_{i+1}$$

$${}^iN_i = {}^in_i - {}^in_{i+1} + \left(-{}^iP_{C_i}\right) \times {}^if_i - \left({}^iP_{i+1} - {}^iP_{C_i}\right) \times {}^if_{i+1}$$

In the formula, ${}_{i+1}^{i}R$ denotes the coordinate transformation matrix between the i+1 link and the i link, and ${}^iP_{C_i}$ denotes the bar center of mass in the bar coordinate system. The coordinates below, ${}^iP_{i+1}$, indicate the coordinates of the origin of the i+1 bar coordinate system in the coordinate system of the i bar.

Using the results of the force balance matrix and the additional rotation matrix, the above equation can be written as:

$${}^iN_i = {}^in_i - {}_{i+1}^{i}R^{i+1}n_{i+1}\left(-{}^iP_{C_i}\right) \times {}^iF_i - {}^iP_{i+1} \times {}_{i+1}^{i}R^{i+1}f_{i+1}$$

Finally rearrange the force and moment equations to form an iterative relationship between the individual rods:

$${}^if_i = {}_{i+1}^{i}R^{i+1}f_{i+1} + {}^iF_i$$

$${}^in_i = {}^iN + {}_{i+1}^{i}R^{i+1}n_{i+1}\left({}^iP_{C_i}\right) \times {}^iF_i + {}^iP_{i+1} \times {}_{i+1}^{i}R^{i+1}f_{i+1}$$

In statics, the joint torque can be obtained by calculating the component of the moment in the axial direction that a connecting rod applies to the adjacent rod:

$$\tau_i = {}^{i}n_i \hat{Z}_i$$

After finishing, the six-degree-of-freedom dynamic equation (without friction model) can be obtained:

$$\tau = M(q)\ddot{q} + B(q,\dot{q}) + G(q) \qquad (4)$$

Among them, q = [q1, q2, q3, q4, q5, q6] represents the position value of each joint. $\dot{q}$,$\ddot{q}$ represents the joint speed and acceleration. M(q) represents the mass matrix of the robot arm, B(q,$\dot{q}$) represents the centrifugal force and Coriolis force components, and G(q) is the gravity vector.

The formulae are too long to display. The following Fig.5 shows the torque curves of joint 2, 3, 4 under the static state from -80 degrees to 80 degrees.

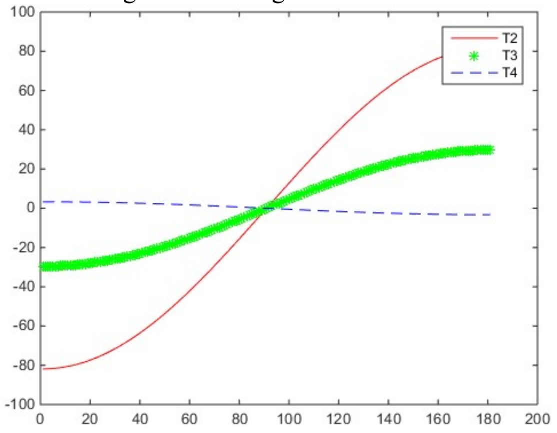


Fig.5 Output torque of joint 2, 3, 4

V.EXPERIMENT

The robot controller hardware used in this article is a 3.5-inch single-board computer module using the X86 architecture. It uses Intel I5-2410U CPU, onboard 4G memory, RTL81111.1 network card, operating system is Ubuntu12.04, running RTAI3.4.9 real-time kernel. The robot joint space trajectory planning period is 1 ms, the kinematics calculation period is 10 ms, and the dynamic calculation period is 10 ms.

The controller supports G code programming and verifies the feasibility of this hardware-in-the-loop simulation platform by executing the following procedure.

```
#1 = 0
o101 while [#1 LT 1000]
    G0 X100 Y-450 Z-280 A10 B75 C-75
    G0 X0     Y-450 Z-280 A10 B75 C-75
    G0 X-100 Y-450 Z-280 A20 B75 C-75
    G0 X-200 Y-450 Z-280 A30 B75 C-75
    G0 X-100 Y-450 Z-280 A20 B75 C-75
    G0 X100    Y-450 Z-280 A0 B75 C-75
    G0 X200    Y-450 Z-180 A10 B75 C-75
    G0 X200 Y-450 Z-280 A10 B75 C-75
  #1 = [#1+1]
o101 endwhile
```

Fig.6 is a video capture of the hardware-in-the-loop simulation. The cooperating robot arm acts according to the joint commands of the control system. At the same time, it records the position, velocity, acceleration and torque information of each joint.

Fig.7 is the position information of each joint. Fig.8 and Fig.9 represent the position and attitude information of the end of the arm.

The six graphs in Fig.10 represent the torque output values and joint estimated torque values for each joint of the cooperating arm. Among them, the red dashed line represents the dynamics estimate, and the solid black line represents the joint torque output measured in the simulation.

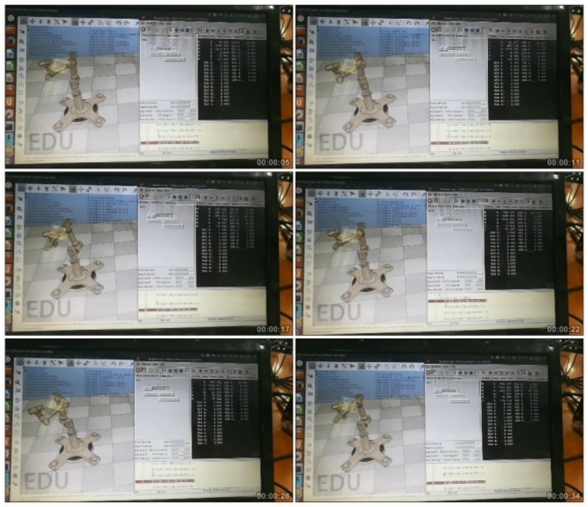


Fig. 6 Hardware in loop simulation process

The six graphs in Fig.11 represent the error between the measured joint torque output and the estimated dynamics of the joints of the cooperating manipulator. Compared with the torque amplitude of the previous figure, the proportion of the error is very small. The correctness of the dynamic algorithm was verified.

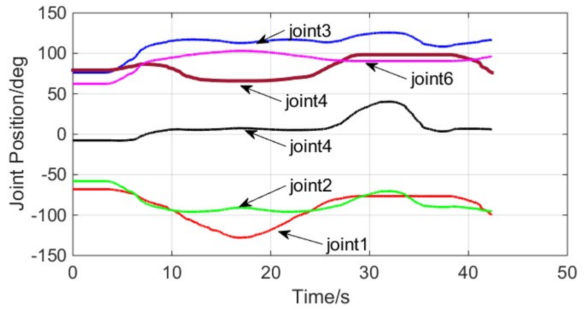


Fig. 7 Position of joints

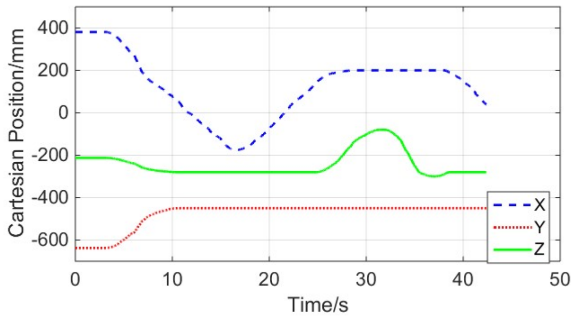


Fig.8 Cartesian position of robot end effector

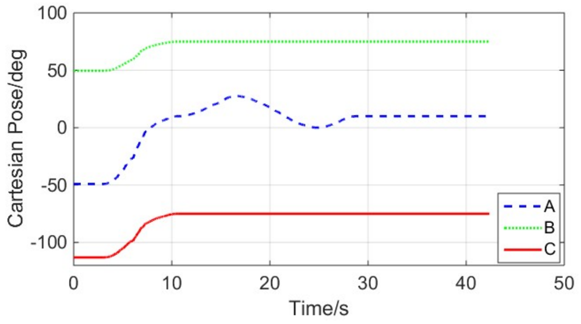


Fig.9 Cartesian pose of robot end effector

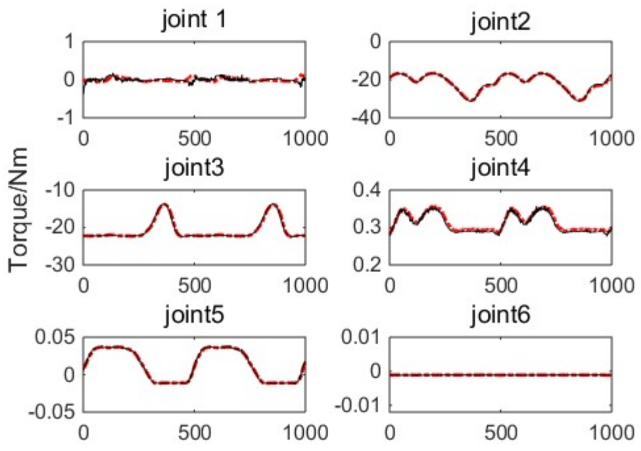


Fig.10 Actual and estimated value of joints' output torque

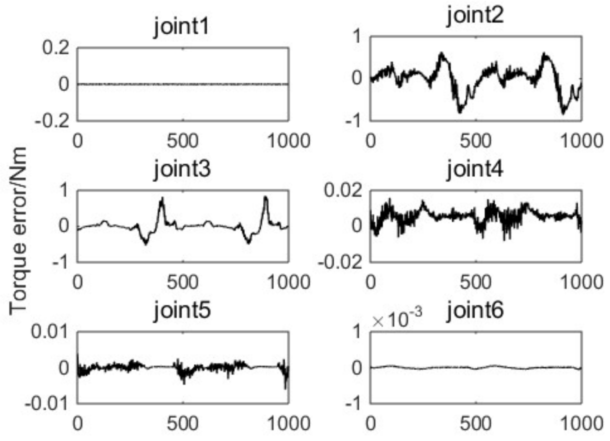


Fig.11 Error between actual and estimated output torque

CONCLUSIONS

This paper introduces the architecture and working principle of Linux CNC, and puts forward a hardware-in-loop simulation and research and development platform based on LinuxCNC and V-rep, which consists of a cooperative robot control system and a V-rep robot virtual model. The kinematics model of the six-degree-of-freedom manipulator is established and its analytical solution is obtained. At the same time, the dynamic model of the six-degree-of-freedom manipulator was established and compared with the V-rep simulation results. The results show that this hardware-in-loop simulation platform has the ability to develop and verify cooperative robotic arm control algorithms. At the same time, this paper also lays a theoretical foundation for the collaborative robot drag teaching and collision protection.

REFERENCES

[1] Chawda V, Niemeyer G. Toward torque control of a KUKA LBR IIWA for physical human-robot interaction[C]. International Conference on Intelligent Robots and Systems. IEEE, 2017:6387-6392.

[2] Liu, J., Fan, Q., Yang, T., Li, K. and Huang, Q. "A space robot manipulator system: designed for capture"[J]. Mechatronics and Automation, Vol. 5, Nos. 2/3, pp.125–132.

[3] Liu Yang, Sun Wei. "Research Status and Technical Development of Cooperative Robots"[J]. Journal of North China University of Technology, vol.29, no.2 pp. 76-85, 2017.

[4] Alfs S, Ivlev O, Martens C, et al. Simulation tool for kinematic configuration control technology for dexterous robots[C]. Industrial Electronics Society, 1999. IECON '99 Proceedings. the, Conference of the IEEE. IEEE, 1999:430-435 vol.1.

[5] Ogan R T. "Hardware-in-the-Loop Simulation". Transportation Research Part C Emerging Technologies, 2006, 12(1):73-89.

[6] Krenn R, Schaefer B. Limitations of Hardware-in-the-Loop Simulations of Space Robotics Dynamics using Industrial Robots[C]. Isairas Symposium, Noordwijk, the Netherlands. DLR, 1999.

[7] Carufel J D, Martin E, Piedboeuf J C. Control strategies for hardware-in-the-loop simulation of flexible space robots[J]. Control Theory and Applications, IEE Proceedings -, 2000, 147(6):569-579.

[8] Temeltas H, Gokasan M, Bogosyan S, et al. Hardware in the loop simulation of robot manipulators through Internet in mechatronics education[C]. IECON. IEEE, 2002:2617-2622 vol.4.

[9] Capurso M, Ardakani M M G, Johansson R, et al. Sensorless kinesthetic teaching of robotic manipulators assisted by observer-based force control[C]. IEEE International Conference on Robotics and Automation. IEEE, 2017:945-950.

[10]Liu J, Huang Q, Yang T, et al. Compliant control of a space robot with multi arms for capturing large tumbling target[C]// IEEE International Conference on Mechatronics and Automation. IEEE, Takamatsu, Japan, 2017:1859-1864.

[11]Li B C, Yu T, Liu P. kinematics simulation and control system design of the three DOF parallel mechanism[C]. Joint International Mechanical, Electronic and Information Technology Conference. 2015.

[12]Rohmer E, Singh S P N, Freese M. V-REP: A versatile and scalable robot simulation framework[C]. Ieee/rsj International Conference on Intelligent Robots and Systems. IEEE, 2013:1321-1326.

[13]Liu J, Huang Q, et al. Whole-body compliance for multi-arm space robotic capturing of large tumbling target in connection compliant phase [J]. Advances in Mechanical Engineering, 2018, Vol. 10(4) 1–19.