IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 23, 2019, accepted November 6, 2019, date of publication November 11, 2019,
date of current version November 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952762

A Novel Semi-Autonomous Teleoperation Method for the TianGong-2 Manipulator System

CHONGYANG LI, ZAINAN JIANG, ZHIQI LI, CHUNGUANG FAN, AND HONG LIU
State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin 150001, China

Corresponding author: Zainan Jiang (jiangzainan@hit.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1300400, and in part by the National Natural Science Foundation of China under Grant 61803124.

ABSTRACT Semi-autonomous teleoperation based on Learning from Demonstration is an effective method of remote operation of space manipulators, especially in the scenarios with limited communication and repeated operation problems. However, the joint trajectories generated by those methods may not be suitable for space manipulator control. In this paper, we present a novel semi-autonomous teleoperation method, which has been used for the TianGong-2 manipulator system. The proposed method can not only reproduce trajectories for the tasks according to the current environment, but also generate the smoother and smaller joint control torques. To implement the method, we first collected kinesthetic demonstrations by the space manipulator teaching platform as prior knowledge. Then, based on these kinesthetic demonstrations, we designed the joint control commands with Dynamics Constraint Learning from Demonstration algorithm. We finally evaluated our method with the simulation and on-orbit experiment by locating the dexterous hand to the pre-screwing bolt. Our results show a significant reduction of joint control torque fluctuations and peak-to-peak values, and also can reduce energy consumption.

INDEX TERMS Learning from demonstration, linear quadratic tracking, space manipulator, teleoperation.

I. INTRODUCTION

In recent years, with the development of space technology and robotic technology, space manipulators are used more and more widely in space exploration missions, including on-orbit servicing, space stations assembling, fueling, spacecraft on-orbit inspecting, and so on [1]–[5]. With the help of space manipulators, not only the risks of extra-vehicular working can be reduced, but also space exploration efficiency can be improved greatly [6], [7]. Hence, space manipulators have attracted much attention all over the world. Since 1990s, China has started the project of building and operating a permanent space station. On September 15, 2016, the TianGong-2 manipulator system was launched with the TianGong-2 space laboratory, to gain experience for on-orbit servicing tasks using manipulators. Many on-orbit servicing tasks were scheduled, including screwing off an electrical connector, removing a multilayer covering and loosening bolts using a hand drill [8]. As a result, it is necessary to propose a proper control method to carry out the above tasks. Currently, there are some common methods of operating space manipulators, such as on-ground pre-programming offline trajectory [9], on-orbit teleoperation [10], and on-ground teleoperation [11].

However, their applications have been limited by some shortcomings. For example, on-ground pre-programming offline trajectory method has poor adaptability to the environment. Even if the method is combined with vision information [12] or laser information [9], it still works poorly when the working condition is different from the expected. For the on-orbit teleoperation method, operators must stay in the spacecraft [13].

Compared with the other two methods, on-ground teleoperation method could combine the human experience and wisdom with the implementation capacity of space manipulators [11], [14]. As an extension of human perception and behavior, the method could be applied to the tasks under expected or unexpected working conditions, especially by increasing force [15] and visual [16] feedback. Using this method, operators can operate space manipulators on the ground. However, this method also suffers some limitations. Firstly, large communication time-delay (about six seconds)

The associate editor coordinating the review of this manuscript and approving it for publication was Luigi Biagiotti.

and limited communication [17], [18] have a great effect on the teleoperation process. Two solutions are proposed currently: a) to establish a dedicated communication link [19], which can reduce the time-delay to tens of milliseconds, and improve the transparency of teleoperation, but increase the cost; b) to use the virtual reality simulation instead of real space scenes [20], [21], which is cheaper but may be dangerous, if there is a large difference between simulation models and real space scenes. Secondly, the operator can only complete one task in one teleoperation process, and thus has to repeat the operation once again even if the task is similar to the previous one. As a result, the on-ground teleoperation method may increase the workload of the operator. Thirdly, the required joint control torques and their fluctuations are large. The phenomenon results from the fact that the manipulator directed trajectories are a mapping of human arm trajectories, and it is difficult for the operator to produce smooth enough trajectories, especially when being fatigue.

The former two disadvantages of on-ground teleoperation can be overcome by semi-autonomous teleoperation method. This method is usually divided into two steps: the first step is to collect the prior knowledge from human operators; and the second is to reproduce Cartesian or joint trajectories based on the actual state and the prior knowledge [22]. Loannis Havoutis and Sylvain Calinon used this method in an underwater ROV teleoperation context to assist the operator, and completed the previously learned tasks autonomously when the communication was limited [23].

To solve all the above three shortcomings, in this paper, we present a novel semi-autonomous teleoperation method. In this method, kinesthetic demonstrations could be collected by the operator with the space manipulator teaching platform. More importantly, based on those demonstrations, space manipulators could obtain the smaller and smoother joint control torques for the tasks by applying Dynamics Constraint Learning from Demonstration (DC-LfD) algorithm. The smaller and smoother joint control torques can reduce the energy consumption and make the manipulator joint drivers work in a linear range, thus improving joint drives output accuracy. More specifically, in our work, the space manipulator teaching platform consists of virtual reality platform and teleoperation device with the force feedback, and the virtual reality platform includes the simulation models of the TianGong-2 manipulator system and the expected working environment. Compared with other Learning from Demonstration methods, DC-LfD algorithm, presented in this paper, not only reproduces task trajectories in the task space [24]–[28], but also obtains the improved joint control torques of the manipulator based on the current conditions.

The rest of the paper is structured as follows: in Section II, we introduce the TianGong-2 manipulator system, the maintenance task unit and the space manipulator teaching platform. In Section III, we present the novel semi-autonomous teleoperation method and its key methodology, DC-LfD algorithm. In Section IV, based on the on-orbit servicing task of the TianGong-2 manipulator system, a simulation and an experiment are carried out to verify the semi-autonomous teleoperation method. The results are also analyzed and discussed. Finally, the paper is summarized in Section V.

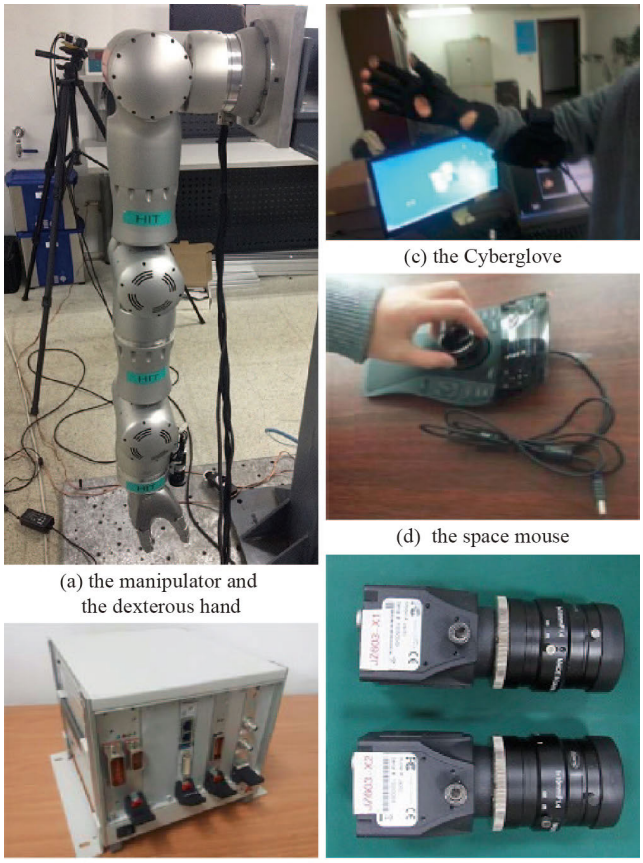

FIGURE 1. The TianGong-2 manipulator system.

II. PRELIMINARIES

In this section, we introduce the TianGong-2 manipulator system, the maintenance task unit and the space manipulator teaching platform. Among them, the TianGong-2 manipulator system and the maintenance task unit are in the space capsule for on-orbit servicing tasks, where the former is used to test the novel semi-autonomous teleoperation method, and the latter provides tools and operation objects. The space manipulator teaching platform is located on the ground, and is used to collect kinesthetic demonstrations of tasks.

A. THE TIANGONG-2 MANIPULATOR SYSTEM

The TianGong-2 manipulator system is developed for on-orbit servicing tasks, and mainly consists of four components:

(1) A six degree of freedom (6-DoFs) lightweight manipulator [29] with a five-fingered dexterous hand [30] and a hand-eye camera which is mounted at the end of the manipulator, as shown in Fig. 1 (a);

(2) An electrical cabinet including a center controller and a power management system, as shown in Fig. 1 (b);

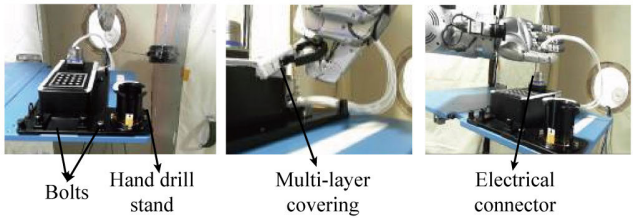

FIGURE 2. The maintenance task unit.

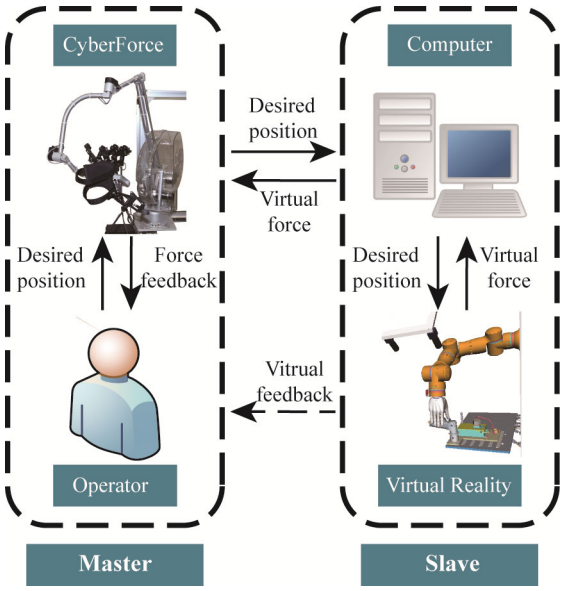

FIGURE 3. The space manipulator teaching platform.

(3) An on-orbit teleoperation platform composed of a teleoperation laptop, a space mouse and a data glove (Cyberglove), as shown in Fig. 1 (c);

(4) Two global cameras mounted at the ceiling of the space capsule.

B. THE MAINTENANCE TASK UNIT

The maintenance task unit is composed of the tools and operation objects, employed in the on-orbit servicing tasks. It includes a spring, a hand drill, a multi-layer covering, two bolts, and an electrical connector, etc., as shown in Fig. 2.

C. THE SPACE MANIPULATOR TEACHING PLATFORM BASED ON VIRTUAL REALITY

In order to get kinesthetic demonstrations of tasks, we design a space manipulator teaching platform. As shown in Fig. 3 this platform consists of two parts, the master part and the slave part.

The teleoperation device CyberForce (CyberGlove System Inc., the USA) is employed as the master part, which can track the 6-dimensional pose of the operator's wrist with the accuracy of 60μm and 0.09°. The 3-dimensional force feedback, of which the upper limit is 10N, can report the operation conditions of the slave part and thus can help the operator to adjust the operation strategy.

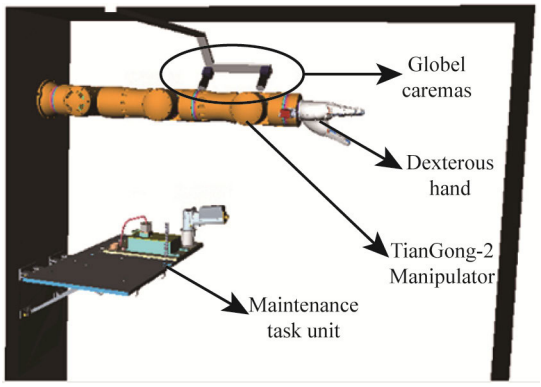

FIGURE 4. The virtual reality platform.

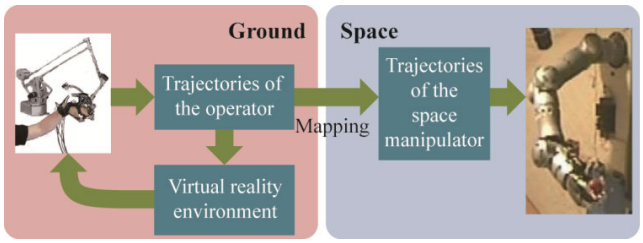

FIGURE 5. On-ground teleoperation system.

The slave part is the virtual reality platform designed by Open Inventor Coin3D. As shown in Fig. 4, the virtual reality platform includes the simulation models of the TianGong-2 manipulator system, the maintenance task unit, the global cameras and parts of the space capsule. The virtual reality platform can also detect the collision during simulation, with the method presented in [31]. Once a collision occurs, the virtual reality platform will produce a virtual force, and transfer it to the master part. Then, the operator can receive the force feedback from CyberForce, and thus the safety of the obtaining kinesthetic demonstrations is ensured.

III. METHODOLOGY

A. OVERVIEW

For the on-orbit servicing tasks, if using on-ground teleoperation directly, we have to face the problem of large time-delay [18]. In the traditional solution, the operator needs to operate the virtual manipulator in virtual reality environment instead of operating the real space manipulator. And then, the operation trajectories will be sent to the space manipulator controller, to control the real manipulator [17]. The architecture of the method is shown in Fig.5:

This method has three shortcomings: Firstly, the inaccurate virtual reality environment would bring danger; secondly, the operator has to repeat the operation once again even if the tasks are similar to each other; thirdly, the required joint control torques and their fluctuations are large.

To solve the above shortcomings, we present a novel semi-autonomous teleoperation method for the TianGong-2 manipulator system in this section. In this method, the trajectories,

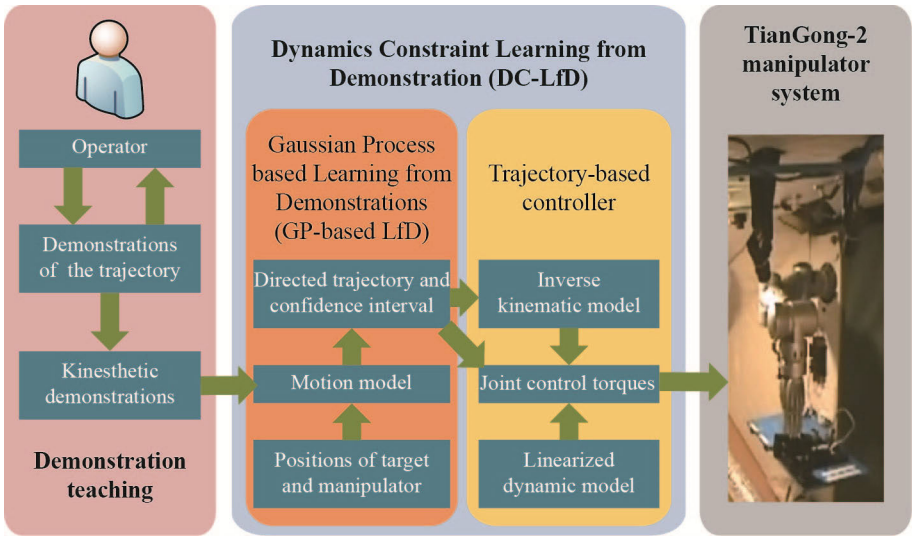

FIGURE 6. The novel semi-autonomous teleoperation method.

from the operators operating the virtual reality, are treated as the prior knowledge, and the manipulator reproduces the task trajectory based on prior knowledge and the target position. The frame of this method is shown in Fig.6. The semi-autonomous teleoperation method can be divided into three steps:

Step 1: Getting kinesthetic demonstrations of directed tasks as prior knowledge. In this step, the space manipulator teaching platform is used, and the tasks are carried out in the virtual reality platform guided by CyberForce [32]. In addition, the recorded Cartesian trajectories of the manipulator model are demonstrations of the task trajectory, and kinesthetic demonstrations of the tasks can be calculated by them.

Step 2: Designing the joint control commands, with DC-LfD algorithm. This step will be described specifically later.

Step 3: Executing the joint control torques by the TianGong-2 manipulator system.

The DC-LfD algorithm is the core algorithm of this semi-autonomous teleoperation method, and it can be done in two phases:

Phase 1: Gaussian Process-based Learning from Demonstration (GP-based LfD). In this phase, we firstly formulate the trajectories based on the dynamical system theory, and motion models are obtained. Then, for one task, with the kinesthetic demonstrations obtained in Step 1, we train the motion model by Gaussian process algorithm, and the model parameters are obtained. Finally, the motion model is applied, together with the positions and orientations of the target and space manipulator, to reproduce the directed Cartesian trajectory and its confidence intervals by Gaussian process regression.

Phase 2: Trajectory-based controller. In this phase, at first, we map the directed Cartesian trajectory from Phase 1 into joint space, using the inverse kinematics model. Next, according to the dynamics model of the manipulator, we design the joint controller according to optimal control theory, and obtain the appropriate joint control torques of this task by the differential dynamic programming (DDP) algorithm. Moreover, the confidence intervals will be the boundary conditions.

These two phases are serially connected, that is, the outputs of the Phase 1, the directed Cartesian trajectory and its confidence intervals, are the inputs of the Phase 2. More details of these two will be illustrated in Section III.B and Section III.C, respectively.

B. GAUSSIAN PROCESS-BASED LEARNING FROM DEMONSTRATION

In this section, we present a Learning from Demonstration algorithm, GP-based LfD. Compared with other similar algorithms, this algorithm trains the motion model and reproduces task trajectories by Gaussian Process. The pseudocode of the GP-based LfD algorithm is shown in Algorithm 1.

In Algorithm 1, in order to ensure the coherence and clarity of the section, we described the framework of the DC-LfD algorithm. And the GP-based LfD algorithm is located between the dash-dotted lines.

As shown in the pseudocode and Fig. 6, GP-based LfD algorithm can not only express more complex motions and the influence of sensors noise, but also calculate the confidence intervals of the reproduced trajectories, to provide inputs for subsequent algorithms. The detailed process of this algorithm is as follows:

1) FORMULATING THE TASK TRAJECTORIES AND KINESTHETIC DEMONSTRATIONS

As shown in [33], the task trajectories can be formulated into a first-order autonomous ordinary differential equation according to the dynamical system theory. In addition, this

Algorithm 1 Dynamics Constraint Learning from Demonstration

Input: kinesthetic demonstrations $\{(\boldsymbol{\xi}_i \in \mathbb{R}^D, \dot{\boldsymbol{\xi}}_i \in \mathbb{R}^D) | i = 1, \ldots, N\}$, mean function $\boldsymbol{m}(\cdot)$, covariance function $\mathrm{Ker}(\cdot, \cdot)$, noise $\boldsymbol{\varepsilon} \sim \boldsymbol{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$, initial position $\xi_0^* \in \mathbb{R}^D$, target position $\xi_{target} \in \mathbb{R}^D$, sampling period $\Delta t \in \mathbb{R}$, tolerable error $\boldsymbol{E}_{tol} \in \mathbb{R}$

Initialize $t \leftarrow 0, \xi \in \mathbb{R}^{D \times N}$ and $\dot{\xi} \in \mathbb{R}^{D \times N}$ composed with kinesthetic demonstrations

$\dot{\xi} \leftarrow f(\xi) \sim N(\boldsymbol{m}(\xi), \mathbf{K}(\xi, \xi))$ // training the motion model

while $\mathrm{abs}(\xi_t^* - \xi_{target}) > \boldsymbol{E}_{tol}$ **do**

$\boldsymbol{m}(\xi_t^*) \leftarrow \mathbf{K}(\xi_t^*, \xi) \cdot (\mathbf{K}(\xi, \xi) + \sigma^2 \boldsymbol{I})^{-1} \cdot f(\xi)$ // $\mathbf{K}_{i,j}(\xi, \xi) = \mathrm{Ker}(\xi_i, \xi_j)$

$\dot{\boldsymbol{\xi}}_t^* \leftarrow \boldsymbol{m}(\xi_t^*)$

$\Sigma(\xi_t^*) \leftarrow \mathbf{K}(\xi_t^*, \xi_t^*) + \sigma^2 \boldsymbol{I}\text{-}\mathbf{K}(\xi_t^*, \xi) \cdot (\mathbf{K}(\xi, \xi) + \sigma^2 \boldsymbol{I})^{-1} \cdot \mathbf{K}(\xi, \xi_t^*)$

$\xi_{t+1}^* \leftarrow \xi_t^* + \dot{\boldsymbol{\xi}}_t^* \cdot \Delta t$

$t \leftarrow t + 1$

Store ξ_t^* and $\Sigma(\xi_t^*)$ at each step t in ξ^*, $\Sigma(\xi^*)$

end while

$\mathbf{u}^* \leftarrow$ Trajectory-based controller $(\xi^*, \Sigma(\xi^*))$ // calculating joint control commands

Return joint control commands sequence $\mathbf{u}^*$

method is robust to spatial and temporal perturbations. Then, the dynamical system equation is:

$$\dot{\boldsymbol{\xi}} = f(\boldsymbol{\xi}) + \boldsymbol{\varepsilon} \tag{1}$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a nonlinear continuous and continuously differentiable function, and this function has only one equilibrium point $\dot{\boldsymbol{\xi}}^* = f(\boldsymbol{\xi}^*) = \mathbf{0}$. $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is the noise of the dynamical system, from inaccuracy of sensor measurement, and we assume that this noise is Gaussian noise $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{I})$. $\xi \in \mathbb{R}^n$ is the input state variable of the function, and could be joint angles, Cartesian positions of the manipulator's end-effector, Cartesian positions and velocities of the manipulator's end-effector, etc. In addition, the different ξ correspond to the different order of the equations $f(\xi)$. In this paper, we encode both positions and velocities of the manipulator's end-effector in ξ [34].

In terms of the selected form of the input variables, the kinesthetic demonstrations in this paper are the position and orientation information of N given points on L trajectories. There are many methods to get kinesthetic demonstrations, such as dragging the manipulator using the zero-force control, visual detection, and so on. We used the space manipulator teaching platform to obtain kinesthetic demonstrations. On-ground operators manipulate the TianGong-2 manipulator model in the virtual reality platform to complete the task using the master device CyberForce. The Cartesian trajectories of the end-effector would be recorded, and the position and orientation information of those trajectories would be used to calculate the kinesthetic demonstrations of the task. There are many advantages of this method, such as ease of operation, adaptation human habits, safety, and so on.

2) REPRODUCING THE DIRECTED TRAJECTORIES AND CONFIDENCE INTERVALS

In order to acquire the specific relationship of (1) by kinesthetic demonstrations, we introduce Gaussian process in this paper. Compared with other machine learning algorithms, Gaussian process provides the outputs with probability distributions. It is a non-parametric model, and can be used to describe complex and non-linear functions [35]. In addition, Gaussian process can deal with the inaccuracy of sensor measurements and errors resulted from imperfect demonstrations. As the result, we use Gaussian process in Learning from Demonstration framework to reproduce the directed trajectories and their confidence intervals, which provide the foundation for building Trajectory-based controller.

Kinesthetic demonstrations $\{(\boldsymbol{\xi}_i \in \mathbb{R}^D, \dot{\boldsymbol{\xi}}_i \in \mathbb{R}^D) | i = 1, \ldots, \mathrm{N}\}$ are obtained from L sample trajectories, and D represents the dimension of each variable. Without considering observation noise, the kinesthetic demonstrations can be treated as the training set of the motion model in Gaussian Process. ξ_i indicates the input variable, and $\dot{\boldsymbol{\xi}}_i$ represents the corresponding directed output. Then we could get a joint multivariate Gaussian distribution $f(\xi)$ as the motion model:

$$\dot{\xi} = f(\xi) \sim N(\boldsymbol{m}(\xi), \mathbf{K}(\xi, \xi)) \tag{2}$$

where $\xi \in \mathbb{R}^{D \times N}$ and $\dot{\xi} \in \mathbb{R}^{D \times N}$ are matrixes, composed with kinesthetic demonstrations. $\boldsymbol{m}(\xi) \in \mathbb{R}^N$ is the mean vector. $\mathbf{K}(\xi, \xi) \in \mathbb{R}^{N \times N}$ is the covariance matrix, and each element $\mathrm{Ker}(\xi_i, \xi_j)$ of this matrix represents the covariance of ξ_i and ξ_j. $\mathrm{Ker}(\cdot, \cdot)$ is also called kernel function of Gaussian process, and we choose the isotropic squared exponential kernel function in this paper, defined as:

$$\mathbf{K}_{i,j} = \mathrm{Ker}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j) = \frac{1}{\sigma_f^2} \cdot \exp\left(-\frac{(\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)^T (\boldsymbol{\xi}_i - \boldsymbol{\xi}_j)}{2 \cdot \ell^2}\right) \tag{3}$$

where $\sigma_f, \ell \in \mathbb{R}$ are the parameters of the isotropic squared exponential kernel, and can be calculated by minimizing the negative log marginal likelihood. Since the sum of multiple independent Gaussian random variables is also Gaussian, we can predict the output distribution of the input variable ξ^* with the prior distribution $f(\xi)$ as follows (we assume that noise $\boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon}^*$ are the same):

$$\begin{bmatrix} f(\xi) \\ f(\boldsymbol{\xi}^*) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon}^* \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{m}(\xi) \\ \boldsymbol{m}(\boldsymbol{\xi}^*) \end{bmatrix}, \begin{bmatrix} \mathbf{K}(\xi, \xi) + \sigma^2 \mathbf{I} & \mathbf{K}(\xi, \boldsymbol{\xi}^*) \\ \mathbf{K}(\boldsymbol{\xi}^*, \xi) & \mathbf{K}(\boldsymbol{\xi}^*, \boldsymbol{\xi}^*) + \sigma^2 \mathbf{I} \end{bmatrix}\right) \tag{4}$$

where $\mathbf{I} \in \mathbb{R}^{N \times N}$. Then, we can get $\dot{\boldsymbol{\xi}}^*$, based on (1) and (4):

$$\dot{\boldsymbol{\xi}}^* = (f(\boldsymbol{\xi}^*) + \boldsymbol{\varepsilon}^* | \xi, \boldsymbol{\xi}^*) \sim N(\boldsymbol{m}(\boldsymbol{\xi}^*), \Sigma(\boldsymbol{\xi}^*)) \tag{5}$$

where $\boldsymbol{m}(\xi^*)$ and $\Sigma(\xi^*)$ are defined as the following equations:

$$\begin{cases} \boldsymbol{m}\left(\boldsymbol{\xi}^*\right) = \mathbf{K}\left(\boldsymbol{\xi}^*, \xi\right) \cdot \left(\mathbf{K}\left(\xi, \xi\right) + \sigma^2\mathbf{I}\right)^{-1} \cdot f\left(\xi\right) \\ \Sigma\left(\boldsymbol{\xi}^*\right) = \mathbf{K}\left(\boldsymbol{\xi}^*, \boldsymbol{\xi}^*\right) + \sigma^2\mathbf{I} \\ \qquad\qquad - \mathbf{K}\left(\boldsymbol{\xi}^*, \xi\right) \cdot \left(\mathbf{K}\left(\xi, \xi\right) + \sigma^2\mathbf{I}\right)^{-1} \cdot \mathbf{K}\left(\xi, \boldsymbol{\xi}^*\right) \end{cases} \tag{6}$$

In this paper, the input variable is the position and orientation of the trajectories, so we can set $\xi_i = [x_i, y_i, z_i, rx_i, ry_i, rz_i]$. $x_i, y_i, z_i \in \mathbb{R}$ are the coordinates of the simulated manipulator's end-effector in the Cartesian space, and $rx_i, ry_i, rz_i \in \mathbb{R}$ are the orientation components (RPY). And the output variable is $\dot{\boldsymbol{\xi}}_i = [\dot{x}_i, \dot{y}_i, \dot{z}_i, \omega x_i, \omega y_i, \omega z_i]$, representing the linear velocities and angular velocities in Cartesian space. Then, based on kinesthetic demonstrations, we used Gaussian process to train the dynamical system (1), and obtained the motion model of this task (a mapping from ξ_i to $\dot{\boldsymbol{\xi}}_i$). Because the input and output variables of the system are both six dimensions (multiple-input and multiple-output), to ensure the independence of the output variables, each output variable should be trained individually with all input variables (multiple-input and single-output). Thus, the motion model of this task is composed of six equations. We took the equations of the velocity in the X direction as an example. In the direction, the velocity part of kinesthetic demonstrations is calculated by $\dot{x}_i = (x_i - x_{i-1}) \cdot F_s$, where $F_s \in \mathbb{R}$ is the sampling frequency of the virtual reality platform. Thus, according to (2), we obtained the motion model of $\dot{x}$ as shown in (7):

$$f_{\dot{x}}(\xi) \sim N\left(m_{\dot{x}}(\xi), \mathbf{K}_{\dot{x}}(\xi, \xi)\right) \tag{7}$$

Then, for each input position and orientation ξ_t, we calculated the distribution of $\dot{\boldsymbol{x}}_t$ as shown in (8):

$$\dot{x}_t = f_{\dot{x}}\left(\boldsymbol{\xi}_t\right) + \boldsymbol{\varepsilon}_t \sim N\left(m_{\dot{x}}\left(\boldsymbol{\xi}_t\right), \Sigma_{\dot{x}}\left(\boldsymbol{\xi}_t\right)\right) \tag{8}$$

where $m_{\dot{\boldsymbol{x}}}(\xi_t)$ and $\Sigma_{\dot{\boldsymbol{x}}}(\xi_t)$ are defined as the following equations:

$$\begin{cases} m_{\dot{x}}(\boldsymbol{\xi}_t) = \mathbf{K}_{\dot{x}}\left(\boldsymbol{\xi}_t, \xi\right) \cdot \left(\mathbf{K}_{\dot{x}}\left(\xi, \xi\right) + \sigma^2\mathbf{I}\right)^{-1} \cdot f_{\dot{x}}(\xi) \\ \Sigma_{\dot{x}}(\boldsymbol{\xi}_t) = \mathbf{K}_{\dot{x}}\left(\boldsymbol{\xi}_t, \boldsymbol{\xi}_t\right) + \sigma^2\mathbf{I} \\ \qquad\qquad - \mathbf{K}_{\dot{x}}\left(\boldsymbol{\xi}_t, \xi\right) \cdot \left(\mathbf{K}_{\dot{x}}\left(\xi, \xi\right) + \sigma^2\mathbf{I}\right)^{-1} \cdot \mathbf{K}_{\dot{x}}\left(\xi, \boldsymbol{\xi}_t\right) \end{cases} \tag{9}$$

where $m_{\dot{\boldsymbol{x}}}(\xi_t)$ represents the mean of this distribution, and $\Sigma_{\dot{\boldsymbol{x}}}(\xi_t)$ represents the covariance. By applying $m_{\dot{\boldsymbol{x}}}(\xi_t)$, the position of the next step along the X direction can be calculated as follows:

$$x_{t+1} = x_t + m_{\dot{x}}(\boldsymbol{\xi}_t) \cdot \Delta t \tag{10}$$

where $\Delta t \in \mathbb{R}$ indicates the sampling period, which can be determined in accordance with tasks.

Each position is calculated according to the last position. By iterating (8) - (10) from a given initial state until reaching the target position, we can reproduce the completely directed trajectory along the X direction. With the same method and process, we also could obtain the directed trajectory alone the Y and Z direction. For orientation trajectories, the process is the same except for the angular velocities ωx_i, ωy_i and ωz_i of the kinesthetic demonstrations as shown in (11):

$$\begin{bmatrix} \omega x_i \\ \omega y_i \\ \omega z_i \end{bmatrix} = \begin{bmatrix} 0 & -\sin(rx_i) & \cos(rx_i) \cdot \cos(ry_i) \\ 0 & \cos(rx_i) & \sin(rx_i) \cdot \cos(ry_i) \\ 1 & 0 & -\sin(ry_i) \end{bmatrix} \cdot \begin{bmatrix} rz_i - rz_{i-1} \\ ry_i - ry_{i-1} \\ rx_i - rx_{i-1} \end{bmatrix} \cdot F_s \tag{11}$$

As shown in (8), when we get the mean $m_{\dot{\boldsymbol{x}}}(\xi_t)$, we also can get the variance $\Sigma_{\dot{\boldsymbol{x}}}(\xi_t)$. And then, we can calculate the confidence interval based on confidence level. For instance, in this paper, we set the confidence level as 68.27% (3-sigma rule). Then, by iterating, we can get the confidence interval of the completely directed trajectory, and it works as the constraint of Trajectory-based controller in next phase.

C. TRAJECTORY-BASED CONTROLLER

In Section III.B, we reproduce the directed trajectory for competing tasks by GP-based LfD algorithm. However, the reproduced trajectories are all based on kinesthetic demonstrations, which are from human operators. Thus, the trajectories may not be smooth enough for manipulator control. In fact, during collecting kinesthetic demonstrations by the space manipulator teaching platform, operators have to pay more attention to the current operation tasks, instead of making demonstrations smooth. Additionally, it is difficult for operators to sense and control whether the velocity and acceleration of the human hand are smooth, especially after working a long time. That would make the required joint control torques and their fluctuations larger, and increase energy loss and cause instability in control. In order to solve this problem, we designed a trajectory-based controller. The pseudocode is shown in Algorithm 2.

As shown in the pseudocode, due to the algorithm works in joint space, the reproduced directed trajectories must be mapped into the joint space at first. Then, we derive the cost function and the state equation (the linearized dynamics model of the manipulator), and formalize the problem of designing the joint controller into an optimal control problem. Finally, we calculate the joint control torques, by the DDP algorithm which can be used to solve the optimal control problem. The confidence intervals in Section III.B would become the boundary constraint of the optimal controller problem. In order to obtain the batter joint torques, we improve the joint controller by modifying the parameters $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{Q}_f$.

1) DYNAMICS MODEL OF THE MANIPULATOR AND LINEARIZATION

According to [36], we can obtain that the manipulator dynamics model, which can be formulated by the following

Algorithm 2 Trajectory-Based Controller

Input: directed Cartesian trajectory $\xi^* \in \mathbb{R}^{D\times N}$ and confidence intervals $\Sigma(\xi^*) \in \mathbb{R}^{M\times N}$, control period ΔT, manipulator parameter ***Model***.
Boundary ← BuildBoundary(ξ^*, $\Sigma(\xi^*)$)
$\mathbf{q}$ ← InverseKinematic(***Model***, ξ^*)
$\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ ← CalculateVelocityAndAcceleration($\mathbf{q}$)
Initialize $\mathbf{Q}$, $\mathbf{Q}_f$, $\mathbf{R}$, $\mathbf{M}$ ← $\mathbf{Q}_f$, $\boldsymbol{P}$ ← − $\mathbf{Q}_f$· [$\mathbf{q}(N, :)$, $\dot{\mathbf{q}}(N, :)$], $i \leftarrow N$
$\mathbf{M}_{\text{cell}}\{N\} \leftarrow \mathbf{M}$, $\boldsymbol{P}_{\text{cell}}\{N\} \leftarrow \boldsymbol{P}$, $flag \leftarrow 0$
while $flag == 0$ **do**
 while $i > 0$ **do**
  [**A**, **B**, ***Bia***] ← LinearDynamics(***Model***, $\mathbf{q}(i, :)$, $\dot{\mathbf{q}}(i, :)$, $\ddot{\mathbf{q}}(i, :)$)
  $\dot{\mathbf{M}} \leftarrow \mathbf{M} \cdot \mathbf{B}$' · inv($\mathbf{R}$) · $\mathbf{B}$' · $\mathbf{M}$- $\mathbf{M}$ · $\mathbf{A}$- $\mathbf{A}$' · $\mathbf{M}$-$\mathbf{Q}$
  $\dot{\boldsymbol{P}} \leftarrow$ - $\mathbf{A}$' · $\boldsymbol{P}$ + $\mathbf{M}$ · $\mathbf{B}$ · inv($\mathbf{R}$) · $\mathbf{B}$' · $\boldsymbol{P}$- $\mathbf{M}$ · ***Bia*** + $\mathbf{Q}$ · [$\mathbf{q}(i, :)$, $\dot{\mathbf{q}}(i, :)$]'
  $\mathbf{M} \leftarrow \mathbf{M}$- $\dot{\mathbf{M}} \cdot \Delta T$, $\boldsymbol{P} \leftarrow \boldsymbol{P}$- $\dot{\mathbf{P}} \cdot \Delta T$
  $\mathbf{M}_{\text{cell}}\{i\} \leftarrow \mathbf{M}$, $\boldsymbol{P}_{\text{cell}}\{i\} \leftarrow \boldsymbol{P}$
  $i \leftarrow i - 1$
 end while
 Initial i ←0, $\mathbf{q}^*(0, :) \leftarrow \mathbf{q}(0, :)$, $\dot{\mathbf{q}}^*(0, :) \leftarrow (0, :)$, $\ddot{\mathbf{q}}^*(0, :) \leftarrow \ddot{\mathbf{q}}(0, :)$
 while $i < N + 1$ **do**
  [**A**, **B**, ***Bia***] ← LinearDynamics(***Model***, $\mathbf{q}^*(i, :)$, $\dot{\mathbf{q}}^*(i, :)$, $\ddot{\mathbf{q}}^*(i, :)$)
  $\mathbf{u}^*(i) \leftarrow$ -inv($\mathbf{R}$) · $\mathbf{B}$' · $\mathbf{M}_{\text{cell}}\{i\}$ · [$\mathbf{q}^*(i, :)$, $\dot{\mathbf{q}}^*(i, :)$]' -inv($\mathbf{R}$) · $\mathbf{B}$' · $\boldsymbol{P}_{\text{cell}}\{i\}$
  [$\mathbf{q}^*(i + 1, :)$, $\dot{\mathbf{q}}^*(i + 1, :)$]' ← [$\mathbf{q}^*(i, :)$, $\dot{\mathbf{q}}^*(i, :)$]' + ($\mathbf{A}$ · [$\mathbf{q}^*(i, :)$, $^*(i, :)$]' + $\mathbf{B}$ · $\mathbf{u}^*(i)$ + ***Bia***) ·ΔT
  $i \leftarrow i + 1$
 end while
 ξ ← ForwardKinematic(***Model***, $\mathbf{q}^*$)
 if ξ in ***Boundary*** && $\mathbf{u}^*$ is small enough **then**
  flag ← 1
 else
  flag ← 0, update $\mathbf{Q}$, $\mathbf{Q}_f$, $\mathbf{R}$
 end if
end while
Return joint control commands sequence $\mathbf{u}^*$

equation:

$$\mathbf{H} \cdot \ddot{\boldsymbol{q}} + C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{f}^x\right) = \boldsymbol{u} \tag{12}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ represent the vectors of joint position, velocity and acceleration of the space manipulator, respectively, and n is the number of joints. $\boldsymbol{u}, \boldsymbol{f}^x \in \mathbb{R}^n$ are vectors of the joint control torques and external forces. $\mathbf{H} \in \mathbb{R}^{n\times n}$ is the joint space inertia matrix, which is a symmetric, positive definite matrix. C is the joint space bias force, and it is the function of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\boldsymbol{f}^x$. We linearize the dynamics model based on (12), and the process is presented as follows:

When the space manipulator works in the free workspace, and has no contact with the environment, we can set $\boldsymbol{f}^x = \boldsymbol{0}$. Then (12) can be linearized by the first-order Taylor expansion each sampling time, as the following equation shows:

$$\begin{aligned}\boldsymbol{u} &\approx \hat{\boldsymbol{u}} \\ &= \mathbf{H} \cdot \ddot{\boldsymbol{q}} + C\left(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i\right) + \frac{\partial C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)}{\partial \boldsymbol{q}} \cdot \left(\boldsymbol{q} - \boldsymbol{q}_i\right) \\ &\quad + \frac{\partial C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)}{\partial \dot{\boldsymbol{q}}} \cdot \left(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_i\right)\end{aligned} \tag{13}$$

We simplify (13), shown as follows:

$$\mathbf{a} \cdot \ddot{\boldsymbol{q}} + \mathbf{b} \cdot \dot{\boldsymbol{q}} + \mathbf{c} \cdot \boldsymbol{q} + \boldsymbol{d} = \hat{\boldsymbol{u}} \tag{14}$$

where $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\mathbf{d}$ are defined as the following equations:

$$\begin{cases} \mathbf{a} = \mathbf{H} \\ \mathbf{b} = \dfrac{\partial C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)}{\partial \dot{\boldsymbol{q}}} \\ \mathbf{c} = \dfrac{\partial C\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right)}{\partial \boldsymbol{q}} \\ \boldsymbol{d} = C\left(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i\right) - \mathbf{c} \cdot \boldsymbol{q}_i - \mathbf{b} \cdot \dot{\boldsymbol{q}}_i \end{cases} \tag{15}$$

If $\boldsymbol{s} = [\dot{\boldsymbol{q}}, \boldsymbol{q}]^T$ is defined as the state variable, the state equation can be derived from (14) and (15) as follows:

$$\dot{\boldsymbol{s}} = \mathbf{A} \cdot \boldsymbol{s} + \mathbf{B} \cdot \boldsymbol{u} + \boldsymbol{Bia} \tag{16}$$

where $\mathbf{A}$, $\mathbf{B}$, and ***Bia*** are defined as the following equations:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{a}^{-1}\mathbf{c} & -\mathbf{a}^{-1}\mathbf{b} \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} \mathbf{0} \\ \mathbf{a}^{-1} \end{bmatrix} \\ \boldsymbol{Bia} &= \begin{bmatrix} \mathbf{0} \\ -\mathbf{a}^{-1}\boldsymbol{d} \end{bmatrix} \end{aligned} \tag{17}$$

2) DESIGNING THE JOINT CONTROL COMMANDS BY OPTIMAL CONTROL THEORY

In this section, according to the optimal control theory, we design the trajectory-based controller and calculate the joint control torques. The controller should be armed with the following two functions:

(1) Tracking the directed trajectory. The joint control torques, which are calculated by trajectory-based controller, can drive the space manipulator to track the directed trajectory reproduced by the GP-based LfD algorithm in Section III.B, and the tracking error should not exceed the confidence interval.

(2) Reducing joint control torques. Due to the fact that the reproduced directed trajectory relies on teaching process totally, the trajectory may not be smooth and thus the joint control torques and their fluctuation may be large. We need to reduce it to decrease the energy consumption and ensure the control stability.

Based on functions (1) and (2), we design the single step cost function of optimal controller, as shown in (18):

$$\begin{aligned} R\left(s(t), \boldsymbol{u}(t)\right) = \left(\boldsymbol{s}(t) - \boldsymbol{s}^*(t)\right)^T \cdot \mathbf{Q} \cdot \left(\boldsymbol{s}(t) - \boldsymbol{s}^*(t)\right) \\ + \boldsymbol{u}^T(t) \cdot \mathbf{R} \cdot \boldsymbol{u}(t) \end{aligned} \tag{18}$$

where $\boldsymbol{s}^*(t) = [\dot{\boldsymbol{q}}^*(t), \boldsymbol{q}^*(t)]^T \in \mathbb{R}^{2n}$ is the state variable of the directed trajectory at time t. n is the number of joints of the space manipulator. $\boldsymbol{s}(t) \in \mathbb{R}^{2n}$ is the state variable of the real trajectory. $\boldsymbol{u}(t) \in \mathbb{R}^n$ is the vector of the joint control torques. It is the output variable of the trajectory-based controller. The relationship of $\boldsymbol{s}(t)$ and $\boldsymbol{u}(t)$ is shown in (16). $\mathbf{Q} \in \mathbb{R}^{2n\times 2n}$ and $\mathbf{R} \in \mathbb{R}^{n\times n}$ are the parameters of the cost function, $\mathbf{Q}$ is a positive semi-definite matrix, and $\mathbf{R}$ is a positive definite matrix. In (18), the former item $(\boldsymbol{s}(t)\text{-}\ \boldsymbol{s}^*(t))\cdot\ \mathbf{Q}\cdot\ (\boldsymbol{s}(t)\text{-}\ \boldsymbol{s}^*(t))$ corresponds to function (1), and can make the state locate close to the directed state. The latter item $\boldsymbol{u}^T(t)\cdot\ \mathbf{R}\cdot\ \boldsymbol{u}(t)$ corresponds to function (2), and it is used to smooth the joint control torques.

The single step cost function can compute the cost at the time t. Based on that, we define the cost of the whole trajectory as shown in (19):

$$\begin{aligned}\sum_{t=0}^{t_f}\mathbf{R}\left(\boldsymbol{s}(t),\boldsymbol{u}(t)\right) &= \left(\boldsymbol{s}(t_f)-\boldsymbol{s}^*(t_f)\right)^T\cdot\mathbf{Q}_f\cdot\left(\boldsymbol{s}(t_f)-\boldsymbol{s}^*(t_f)\right)\\ &+\sum_{t=0}^{t_f-1}\Big(\left(\boldsymbol{s}(t)-\boldsymbol{s}^*(t)\right)^T\cdot\mathbf{Q}\cdot\left(\boldsymbol{s}(t)-\boldsymbol{s}^*(t)\right)\\ &+\boldsymbol{u}^T(t)\cdot\mathbf{R}\cdot\boldsymbol{u}(t)\Big)\end{aligned} \tag{19}$$

Because the cost function (19) is quadratic and the model in (16) is linear, this optimal control problem can be treated as a Linear Quadratic Tracking (LQT) problem [37]:

$$\begin{aligned}&\min\sum_{t=0}^{t_f-1}\Big(\left(\boldsymbol{s}(t)-\boldsymbol{s}^*(t)\right)^T\cdot\mathbf{Q}\cdot\left(\boldsymbol{s}(t)-\boldsymbol{s}^*(t)\right)+\boldsymbol{u}^T(t)\cdot\mathbf{R}\cdot\boldsymbol{u}(t)\Big)\\ &\quad+\left(\boldsymbol{s}(t_f)-\boldsymbol{s}^*(t_f)\right)^T\cdot\mathbf{Q}_f\cdot\left(\boldsymbol{s}(t_f)-\boldsymbol{s}^*(t_f)\right)\\ &s.t.\ \begin{cases}\dot{\boldsymbol{s}}(t)=\mathbf{A}(t)\cdot\boldsymbol{s}(t)+\mathbf{B}(t)\cdot\boldsymbol{u}(t)+\boldsymbol{Bia}(t)\\ \boldsymbol{s}(t+1)=\boldsymbol{s}(t)+\dot{\boldsymbol{s}}(t)\cdot\Delta T\end{cases}\end{aligned} \tag{20}$$

where ΔT is the control cycle of the trajectory-based controller, and is set as 1ms in this paper. All variables in (20) are the function of time t, except $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{Q}_f$ and ΔT. To solve the LQT problem, DDP algorithm is applied, and the quadratic cost function can be obtained as shown in (21):

$$J\left(\boldsymbol{s}\left(t\right),t\right)=h\left(\boldsymbol{s}\left(t_f\right),t_f\right)+\int_0^{t_f}g\left(\boldsymbol{s}\left(\tau\right),\boldsymbol{u}\left(\tau\right),\tau\right)\mathrm{d}t \tag{21}$$

where $h(\boldsymbol{s}(t_f), t_f)$ and $g(\boldsymbol{s}(t), \boldsymbol{u}(t)t)$ are defined as the following equations:

$$\begin{cases}h\left(\boldsymbol{s}\left(t_f\right),t_f\right)=\dfrac{1}{2}\cdot\left[\boldsymbol{s}\left(t_f\right)-\boldsymbol{s}^*\left(t_f\right)\right]^T\cdot\mathbf{Q}_f\cdot\left[\boldsymbol{s}\left(t_f\right)-\boldsymbol{s}^*\left(t_f\right)\right]\\ g\left(\boldsymbol{s}\left(t\right),\boldsymbol{u}\left(t\right),t\right)=\dfrac{1}{2}\cdot\left[\boldsymbol{s}\left(t\right)-\boldsymbol{s}^*\left(t\right)\right]^T\cdot\mathbf{Q}\cdot\left[\boldsymbol{s}\left(t\right)-\boldsymbol{s}^*\left(t\right)\right]\\ \qquad\qquad+\dfrac{1}{2}\boldsymbol{u}^T\left(t\right)\cdot\mathbf{R}\cdot\boldsymbol{u}\left(t\right)\end{cases} \tag{22}$$

where $\mathbf{Q}_f \in \mathbb{R}^{2n\times 2\ n}$ is the value of $\mathbf{Q}$ at time t_f.

If J^* is the minimum value of J, it should satisfy the Hamilton-Jacobi-Bellman equation (H-J-B), as shown in (23):

$$J_t^*+\min_{\boldsymbol{u}(t)}\left\{\begin{array}{l}g\left(\boldsymbol{s}\left(t\right),\boldsymbol{u}\left(t\right),t\right)\\ +J_{\boldsymbol{s}}^*\left(\mathbf{A}(t)\cdot\boldsymbol{s}(t)+\mathbf{B}(t)\cdot\boldsymbol{u}(t)+\boldsymbol{Bia}(t)\right)\end{array}\right\}=0 \tag{23}$$

where $J_t^*=\partial J^*\left(\boldsymbol{s}\left(t\right),t\right)/\partial t$, $J_{\boldsymbol{s}}^*=\partial J^*\left(\boldsymbol{s}\left(t\right),t\right)/\partial\boldsymbol{s}$.

Then, we define the Hamiltonian function as follows:

$$\begin{aligned}&Ha\left(\boldsymbol{s}\left(t\right),\boldsymbol{u}\left(t\right),J_{\boldsymbol{s}}^*,t\right)\\ &=g\left(\boldsymbol{s}\left(t\right),\boldsymbol{u}\left(t\right),t\right)+J_{\boldsymbol{s}}^{*T}\left(\boldsymbol{s}\left(t\right),t\right)\\ &\quad\cdot\left(\mathbf{A}\left(t\right)\cdot\boldsymbol{s}\left(t\right)+\mathbf{B}\left(t\right)\cdot\boldsymbol{u}\left(t\right)+\boldsymbol{Bia}\left(t\right)\right)\end{aligned} \tag{24}$$

In this case, minimizing J is the same as minimizing Ha by controlling $\boldsymbol{u}(t)$. Therefore, we obtain:

$$\frac{\partial Ha}{\partial\boldsymbol{u}}\left(\boldsymbol{s}\left(t\right),\boldsymbol{u}\left(t\right),J_{\boldsymbol{s}}^*,t\right)=\mathbf{R}\cdot\boldsymbol{u}\left(t\right)+\mathbf{B}^T\left(t\right)\cdot J_{\boldsymbol{s}}^*=0 \tag{25}$$

Since $\partial^2 Ha/\partial\boldsymbol{u}^2 = R$ is a positive definite matrix, we could obtain $\boldsymbol{u}^*$ as follows:

$$\boldsymbol{u}^*\left(t\right)=-\mathbf{R}^{-1}\cdot\mathbf{B}^T\left(t\right)\cdot J_{\boldsymbol{s}}^* \tag{26}$$

Considering (23), (24) and (26), we have:

$$\begin{aligned}0=&J_t^*+\frac{1}{2}\cdot\left[s\left(t\right)-s^*\left(t\right)\right]^{\mathrm{T}}\cdot\mathbf{Q}\cdot\left[s\left(t\right)-s^*\left(t\right)\right]\\ &-\frac{1}{2}\cdot J_{\boldsymbol{s}}^{*T}\cdot\mathbf{B}\left(t\right)\cdot R^{-1}\cdot\mathbf{B}\left(t\right)^T\cdot J_{\boldsymbol{s}}^*\\ &+J_{\boldsymbol{s}}^{*T}\cdot\mathbf{A}\left(t\right)\cdot\boldsymbol{s}\left(t\right)+J_{\boldsymbol{s}}^{*T}\cdot\boldsymbol{Bia}\left(t\right)\end{aligned} \tag{27}$$

Since the boundary condition of (21) is:

$$J^*\left(\boldsymbol{s}\left(t_f\right),t_f\right)=\frac{1}{2}\cdot\left[\boldsymbol{s}\left(t_f\right)-\boldsymbol{s}^*\left(t_f\right)\right]^{\mathrm{T}}\cdot\mathbf{Q}_f\cdot\left[\boldsymbol{s}\left(t_f\right)-\boldsymbol{s}^*\left(t_f\right)\right] \tag{28}$$

It is reasonable to assume that:

$$J^*\left(\boldsymbol{s}\left(t\right),t\right)=\frac{1}{2}\cdot\boldsymbol{s}^T\left(t\right)\cdot\mathbf{M}\left(t\right)\cdot\boldsymbol{s}\left(t\right)+\boldsymbol{p}^{\mathrm{T}}\left(t\right)\cdot\boldsymbol{s}\left(t\right)+\eta\left(t\right) \tag{29}$$

where $\mathbf{M}(t)$ is a positive semi-definite matrix, $\boldsymbol{p}(t)$ is a vector, and $\eta(t)$ is a scalar.

Substitute (29) into (27), and we can obtain:

$$\begin{aligned}0=&\frac{1}{2}\cdot\boldsymbol{s}^T(t)\\ &\cdot\begin{pmatrix}\dot{\mathbf{M}}(t)+\mathbf{Q}-\mathbf{M}(t)\cdot\mathbf{B}(t)\cdot\mathbf{R}^{-1}\cdot\mathbf{B}^T(t)\cdot\mathbf{M}(t)\\ +2\cdot\mathbf{M}(t)\cdot\mathbf{A}(t)\end{pmatrix}\cdot\boldsymbol{s}(t)\\ &+\begin{pmatrix}\dot{\boldsymbol{p}}^T(t)-\boldsymbol{s}^{*T}(t)\cdot\mathbf{Q}-\boldsymbol{p}^T(t)\cdot\mathbf{B}(t)\cdot\mathbf{R}^{-1}\cdot\mathbf{B}^T(t)\cdot\mathbf{M}(t)\\ +\boldsymbol{p}^T(t)\cdot\mathbf{A}(t)+\boldsymbol{Bia}^T(t)\cdot\mathbf{M}^T(t)\end{pmatrix}\\ &\cdot\boldsymbol{s}(t)\\ &+\begin{pmatrix}\dot{\eta}(t)+\dfrac{1}{2}\cdot\boldsymbol{s}^{*T}(t)\cdot\mathbf{Q}\cdot\boldsymbol{s}^*(t)\\ -\dfrac{1}{2}\cdot\boldsymbol{p}^T(t)\cdot\mathbf{B}(t)\cdot\mathbf{R}^{-1}\cdot\mathbf{B}^T(t)\cdot\boldsymbol{p}(t)+\boldsymbol{p}^T(t)\cdot\boldsymbol{Bia}(t)\end{pmatrix}\end{aligned} \tag{30}$$

Since that (30) is an identical equation for any $\boldsymbol{s}(t)$ and $\boldsymbol{s}^*(t)$, the following conditions have to be satisfied, with the

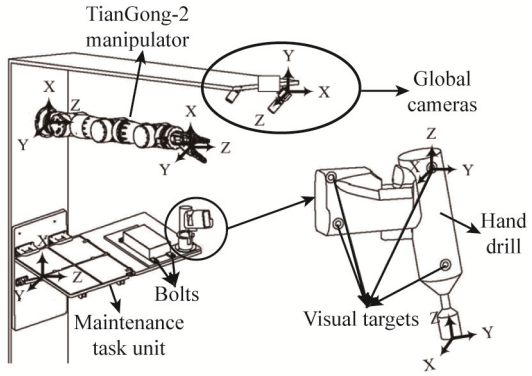

FIGURE 7. Loosening bolts using a hand drill.

boundary conditions $\mathbf{M}(t_f) = \mathbf{0}$, $\boldsymbol{p}(t_f) = \boldsymbol{0}$, and $\eta(t_f) = 0$. We obtain:

$$\begin{cases} \dot{\mathbf{M}}(t) = \mathbf{M}(t) \cdot \mathbf{B}(t) \cdot \mathbf{R}^{-1} \cdot \mathbf{B}^T(t) \cdot \mathbf{M}(t) - Q - 2 \cdot \mathbf{M}(t) \cdot \mathbf{A}(t) \\ \dot{\boldsymbol{p}}(t) = \mathbf{M}^T(t) \cdot \mathbf{B}(t) \cdot \mathbf{R}^{-1} \cdot \mathbf{B}^T(t) \cdot \boldsymbol{p}(t) + Q^T \cdot \boldsymbol{s}^*(t) \\ \qquad - \mathbf{A}^T(t) \cdot \boldsymbol{p}(t) - \mathbf{M}(t) \cdot \boldsymbol{Bia}(t) \\ \dot{\eta}(t) = -\frac{1}{2} \cdot \boldsymbol{s}^{*T}(t) \cdot Q \cdot \boldsymbol{s}^*(t) \\ \qquad + \frac{1}{2} \cdot \boldsymbol{p}^T(t) \cdot \mathbf{B}(t) \cdot \mathbf{R}^{-1} \cdot \mathbf{B}^T(t) \cdot \boldsymbol{p}(t) - \boldsymbol{p}^T(t) \cdot \boldsymbol{Bia}(t) \end{cases} \tag{31}$$

In addition, by substituting (29) into (26), the improved joint control torques would be obtain as follows:

$$\boldsymbol{u}^*(t) = -R^{-1} \cdot \mathbf{B}^T(t) \cdot (\mathbf{M}(t) \cdot \boldsymbol{s}(t) + \boldsymbol{p}(t)) \tag{32}$$

In the equation, $\mathbf{M}(t)$ and $\boldsymbol{p}(t)$ can be derived by a numerical integration of (31), and then the improved joint control torques $\boldsymbol{u}^*(t)$ are obtained. And we need to adjust the parameters $\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{Q}_f$, to make the improved joint control torques smoother, and the manipulator trajectory driven by them can satisfy the confidence intervals in Section III.B.

IV. SIMULATIONS AND EXPERIMENTS

A. OVERVIEW

In order to verify the novel semi-autonomous teleoperation method for the TianGong-2 manipulator system, we design a simulation and an experiment, locating the dexterous hand to the pre-screwing bolt. This experiment is the prerequisite of loosening bolts using the hand drill, and loosening bolts is a prototypical on-orbit operation. As shown in Fig. 7, the platform mainly consists of the TianGong-2 manipulator, the hand drill, the maintenance task unit and two global cameras. In the maintenance task unit, a device is fixed unit with two bolts, both of which should be loosed by the hand drill. The global cameras are used to sense the positions and orientations of the hand drill and bolts.

B. COLLECTING AND PROCESSING THE KINESTHETIC DEMONSTRATIONS

In this paper, the kinesthetic demonstrations work as the prior knowledge, and they are collected by the space manipulator

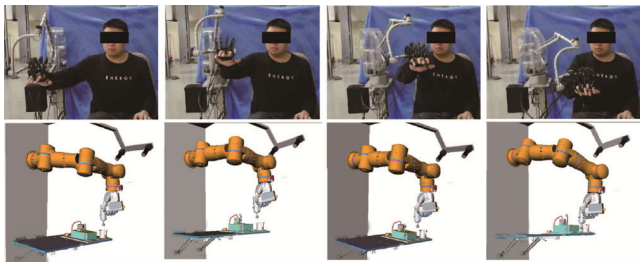

FIGURE 8. Collecting the kinesthetic demonstrations.

teaching platform. The teaching operation is carried out in the virtual reality platform by the master device, CyberForce, as shown in Fig. 8. From the figure, it can be concluded that the process starts after the hand drill is pulled out of the hand drill stand, and the manipulator model is controlled until the hand drill moves to 20mm above each one of the bolts.

To ensure the generalization of the kinesthetic demonstrations and obtain the reasonable confidence intervals by the GP-based LfD algorithm, the operator needs to repeat this process 3 times in the study, with different start positions. Moreover, to facilitate the calculation of the dynamical system framework of GP-based LfD algorithm, we build a new coordinate system, named the target coordinate system, at the target location, and all subsequent Cartesian trajectories are displayed in the coordinate system. In the experiment, the origin of the target coordinate system is 20mm above each one of the bolts, and the direction of the coordinate axis is aligned with the base coordinate system of the TianGong-2 space manipulator. The demonstrations of the task trajectory in this experiment are shown in Fig. 9. With the trajectories, the kinesthetic demonstrations can be calculated as shown in Section III.B.

As shown in Fig. 9, in order to test the adaptability of this algorithm, we select three 'bad' trajectories as the demonstrations. There are many disadvantages of these trajectories, such as severe shaking, unsmooth velocity and acceleration, a cross between of the trajectories, and so on. All of these situations might occur in the actual operation.

C. SIMULATION OF DYNAMICS CONSTRAINT LEARNING FROM DEMONSTRATION

Before the experiment, we set up a simulation to verify the semi-teleoperation method in MATLAB. The simulation includes the whole DC-LfD algorithm and the TianGong-2 manipulator system. The DC-LfD algorithm mainly consists of two steps:

Step 1 is the GP-based LfD algorithm. We calculate the reproduced directed trajectories and their confidence intervals based on the kinesthetic demonstrations. We set the step time Δt of the reproduced trajectories to 1s. And we also set the noise of the dynamical system model in (1) to $\boldsymbol{\varepsilon} \sim N(0, 0.1)$. In order to ensure the generality, we select some different initial positions to reproduce the directed trajectories. The result is shown in Fig. 10.

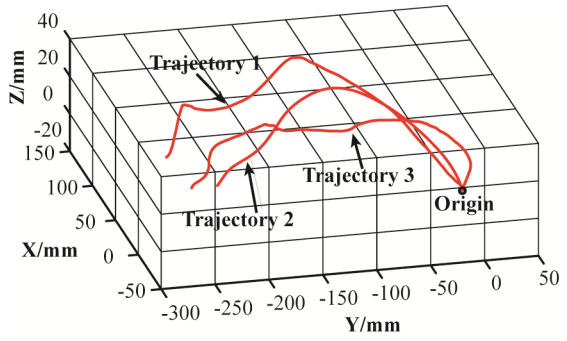

FIGURE 9. Demonstrations of the task trajectory.

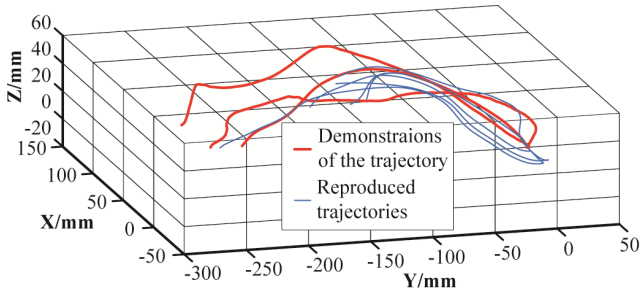

FIGURE 10. The reproduced directed trajectories.

In Fig. 10, the blue lines are the reproduced trajectories from different start positions, and they can all converge to the target position. Considering the safety of the task, the locating error of the hand drill must be less than 3mm. In the simulation, this value is set to 1mm for a better performance. To give more specific explanation, we take the initial position (98.80 mm, −93.55 mm, −2.52 mm, 0°, 0°, 0°) as an example, and the reproduced directed trajectory and its confidence intervals are shown in Fig. 11. This initial position is from an experimental data on the ground.

In Fig. 11, the dashed blue lines are the projections of the reproduced directed trajectory on three coordinate planes, and the gray region are the confidence intervals, when we set the confidence level as 68.27% (3-sigma rule). According to the process of GP-based LfD algorithm, the confidence intervals are calculated based on the prior knowledge (kinesthetic demonstrations) and the confidence level. The wide reign of the confidence intervals mean the decentralized demonstrations of the trajectory, and it can provide more choices for the joint control torques improvement, and vice versa.

Step 2 is to design the joint control commands by trajectory-based controller. Before that, we would do some pro-processing. Firstly, we need to convert the reproduced directed trajectory and its confidence intervals from the target coordinate system to the base coordinate system of the TianGong-2 manipulator. Secondly, we match the Δt in (10) and ΔT in (19) by interpolating the reproduced directed trajectory. Finally, we convert the trajectory from Cartesian space to the joint space.

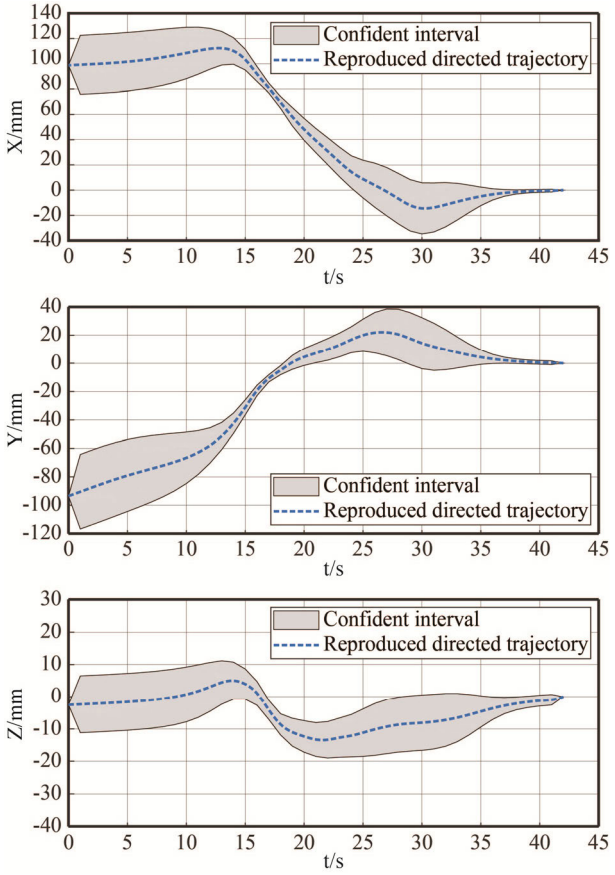

FIGURE 11. Reproduced directed trajectory and confidence interval.

In the simulation, the initial joint angle of the TianGong-2 manipulator is [79.85°, −9.44°, −76.48°, −83.47°, −81.54°, −10.46°]°, and this data is from one on-ground experiment. By adjusting the parameters in LQT, and crossing validation with the confidence intervals obtained in the last step, we select $\mathbf{Q} = 100 \cdot [\mathbf{I}\ \mathbf{O}; \mathbf{O}\ \mathbf{O}]$, $\mathbf{Q}_f = [\mathbf{I}\ \mathbf{O};\ \mathbf{O}\ \mathbf{O}]$, and $\mathbf{R} = 1000 \cdot \mathbf{I}$. The corresponding joint control torques are shown in Fig. 12. Moreover, as a comparison, we also compute the joint control torques by the traditional optimal controller and the computed torque controller. The traditional optimal controller is divided into two steps: the first step is to optimize the trajectory using the LQT algorithm; the second step is to compute the joint control torques with the computed torque method. The solid red lines represent the joint control torques generated by the trajectory-based controller. In the comparisons, the dashed green and blue lines are those generated by the traditional optimal controller and the computed torque controller.

Compared with the computed torque controller, the trajectory-based controller has obvious advantages. In Fig.12, we can find that the fluctuations of the joint control torques of trajectory-based controller are reduced significantly. For all six joints, the fluctuations are reduced from 4-5 peaks to 1-2. In addition, using the trajectory-based controller, the amplitudes of the joint control torques are also greatly reduced. Taking joint 1 as an example, the peak-to-peak value

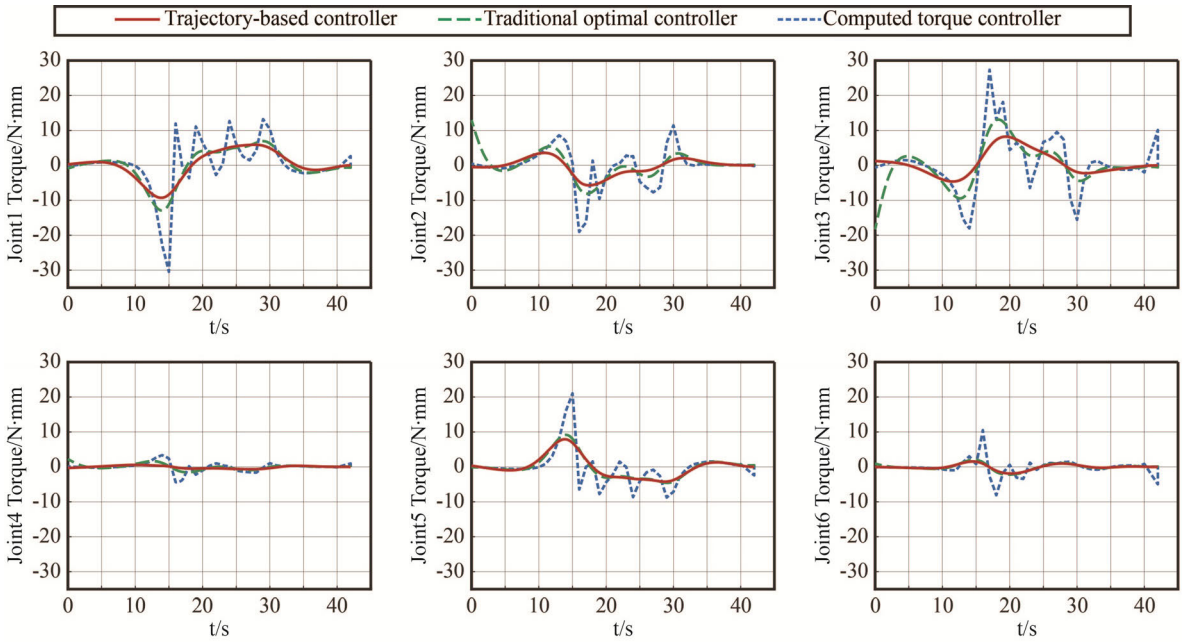

FIGURE 12. Joint control torques of the simulation.

of the joint control torques generated by the computed torque controller is 43.69N · mm, whereas the one generated by the trajectory-based controller is 15.93N · mm, and the value is reduced by 64%. And other joint control torques are also reduced by more than 60%. The reductions of the fluctuations and the peak-to-peak values not only decrease the energy consumption of the manipulator, but also improve the stability of control. Compared with the traditional optimal controller, the torques from the trajectory-based controller are smoother at the beginning of the trajectory. In particular, at that moment, the control torques of the joint 2 and 3 have a large jump to more than 10 N · mm by using traditional optimal controller, but the value is less than 1N · mm with trajectory-based controller. Moreover, in terms of the fluctuations and peak-to-peak values of the joint control torques, trajectory-based controller is also better than the other one.

For the tracking accuracy, with the joint control torques obtained by the trajectory-based controller, the Cartesian trajectory of the manipulator is in the confidence intervals, as shown in Fig. 13.

In Fig. 13, the manipulator Cartesian trajectories and the confidence intervals are all expressed in the target coordinate system. The dashed blue lines are the projections of the manipulator Cartesian trajectory on three coordinated planes, with the joint control torques obtained by the computed torque controller. The grey regions mean the confidence level is 3, which the value is 68.27%. The solid red and dashed greens lines are the projections of the manipulator trajectory, using the trajectory-based controller and the traditional optimal controller. It can be concluded from the figures that the lines are all in the grey regions, and thus can be applied in the further control.

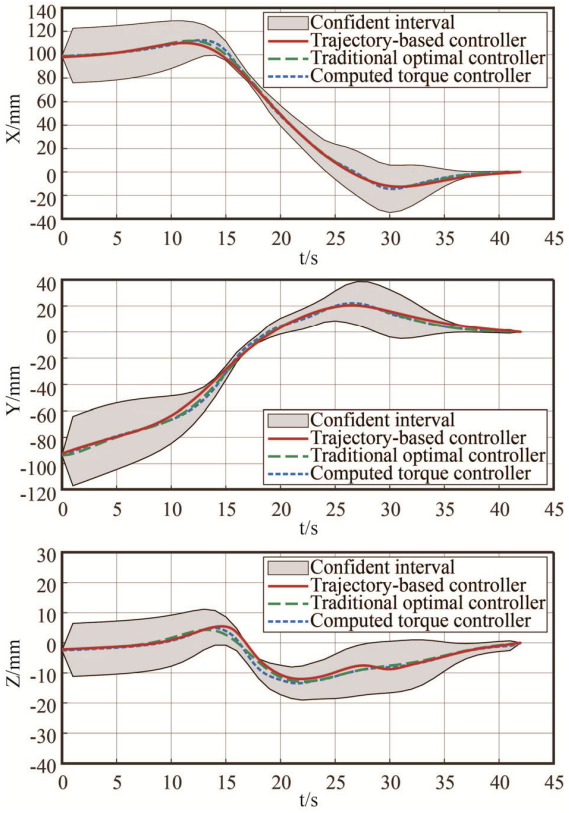

FIGURE 13. Cartesian trajectory in the confidence interval.

Smooth joint control torques contribute to smooth joint velocities, as shown in Fig. 14. The solid red lines represent the joint velocities, using trajectory-based controller, and the dashed green and blue lines are produced by the traditional

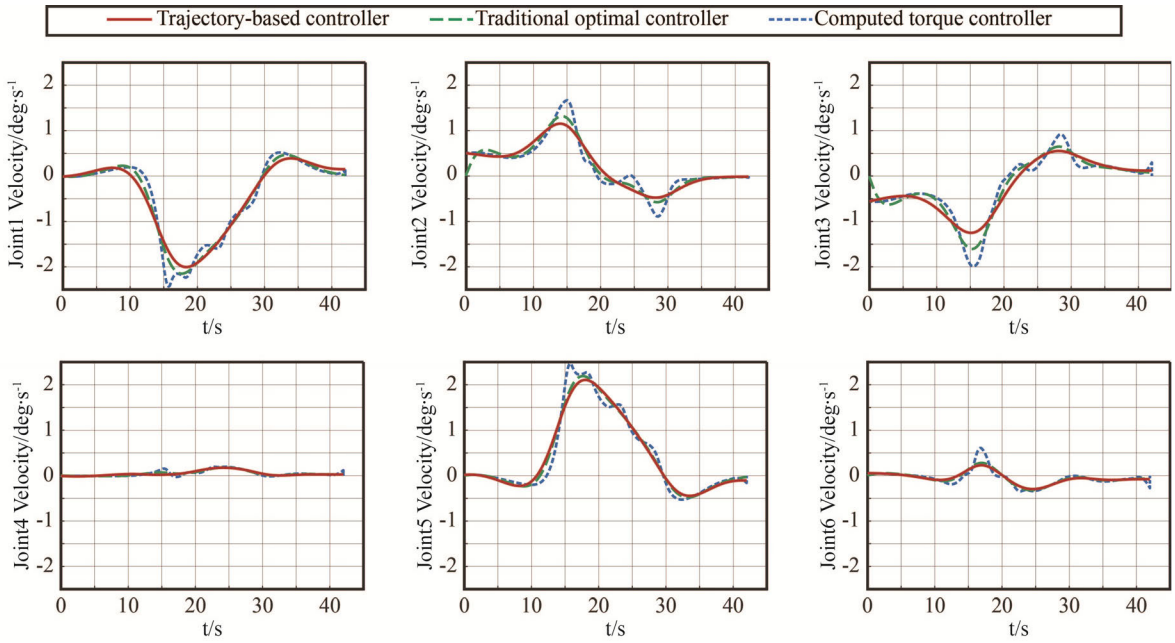

FIGURE 14. Joint velocities of the simulation.

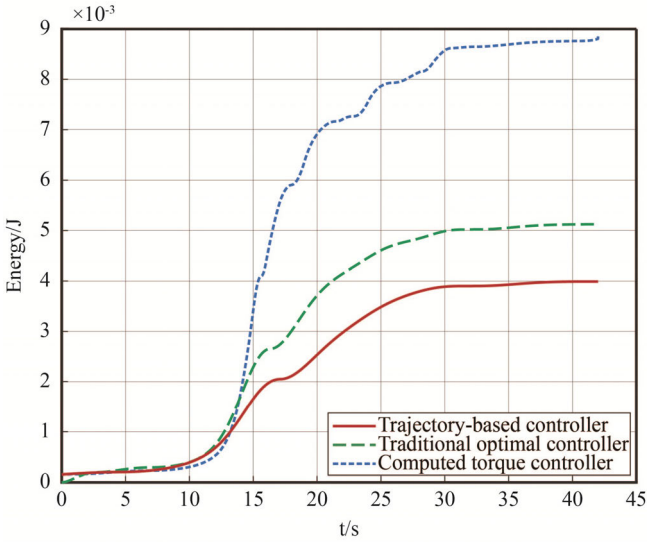

FIGURE 15. The total energy variable with time.

optimal controller and the computed torque controller. Obviously, the red solid lines are smoother, and their fluctuations are reduced from 2-4 to 1 compared with the dashed blue lines. The green lines are intermediate. And different velocity trajectories will lead the Tiangong-2 manipulator system to consume different energy. The total energy variable with time is shown in Fig. 15:

As shown in Fig. 15, clearly, using the trajectory-based controller consumes the least amount of energy to accomplish the same task. In comparison, using the traditional optimal controller and the computed torque controller need to consume 33.54% and 126.72% more energy respectively.

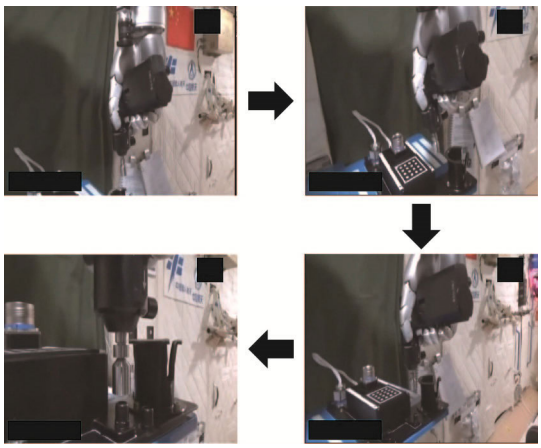

FIGURE 16. Experiment process.

D. ON-ORBIT EXPERIMENT OF THE TIANGONG-2 MANIPULATOR SYSTEM

The actual experiment is carried out with the TianGong-2 manipulator system in the space capsule. In the experiment, the process of obtaining the kinesthetic demonstrations is the same as the simulation in Section IV.C. Then, the kinesthetic demonstrations are transmitted to the center controller of the TianGong-2 manipulator system. The center controller generates the joint control torques according to the position of the bolt, with DC-LfD algorithm. Fig. 16 shows the process of locating the dexterous hand to the first bolt.

In Fig. 16, the TianGong-2 manipulator system can locate the bolt automatically and accurately. The start position and orientation of the hand drill are (131.29 mm, −93.28 mm, −2.19mm, 0°, 0°, 0°) in the target coordinate

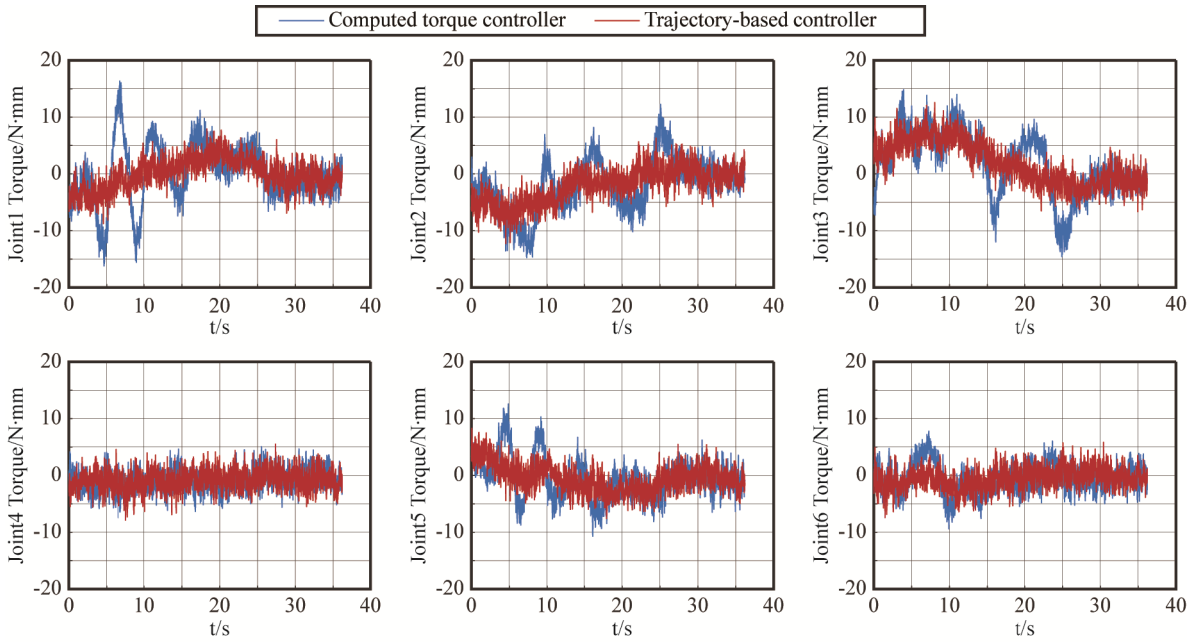

FIGURE 17. Joint control torques of the on-orbit experiment.

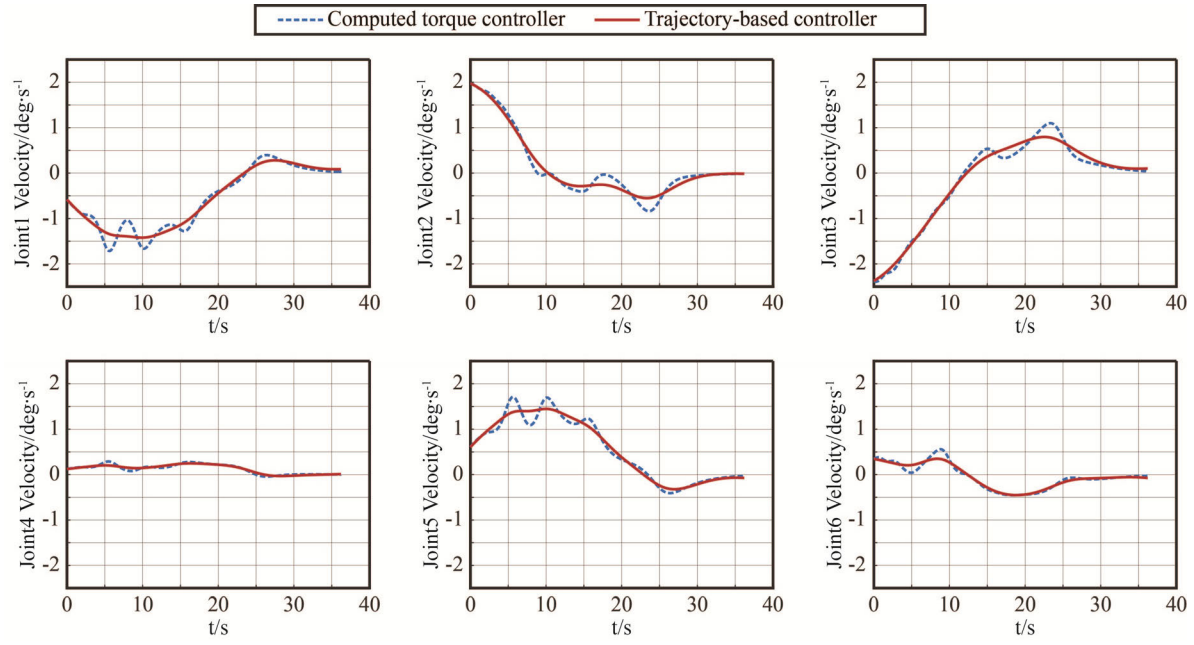

FIGURE 18. Joint velocities of the on-orbit experiment.

system. And the initial joint of the manipulator are [74.52°, −17.6°, −60.55°, −84.53°, −90.46°, −22.04°]. Because the TianGong-2 manipulator system and the maintenance task unit are reassembled on-orbit, the initial states of the two are different from the states of the simulation and the ground experiments. The joint control torques of this experiment are shown in Fig. 17, the joint velocities are shown in Fig. 18, and the total energy variable with time is shown in Fig. 19.

In Fig. 17, the joint control torques are measured by the joint torque sensors. Despite the sensor noise, we can still find that the torques produced by the trajectory-based controller perform better than the others, i.e., the fluctuations are fewer, and the peak-to-peak values are smaller.

The joint velocities in the experiment are shown in Fig. 18. In Fig. 18, compared with the computed torque controller, the trajectory-based controller can reduce the fluctuations from

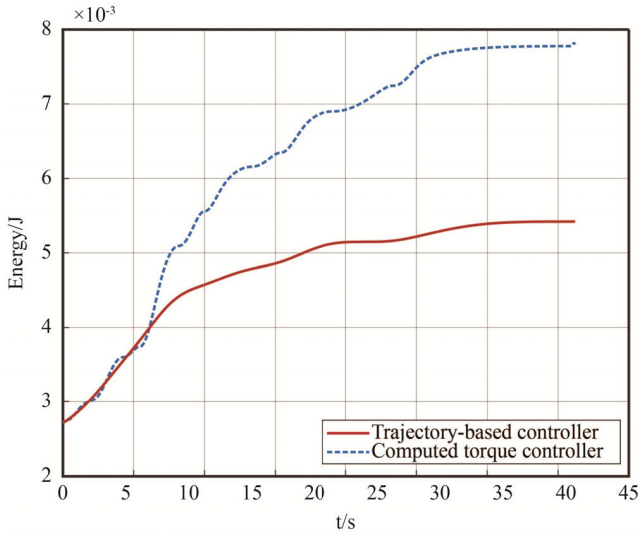

FIGURE 19. The total energy variable with time of the on-orbit experiment.

2-3 to 1, and besides, it also can reduce the peak-to-peak values.

Fig. 19, the energy variable figure, shows that using the trajectory-based controller can lead to the less energy consumption. For the experiment, using the computed torque controller have to consume 88.45% more energy.

V. CONCLUSION

In this paper, we present a novel semi-autonomous teleoperation method for the TianGong-2 manipulator system, which could be used to operate the space manipulator remotely. Compared with other semi-autonomous teleoperation methods, this method can not only reproduce trajectories for completing tasks according to the current environment, but also reduce the joint control torque fluctuations, peak-to-peak values significantly and energy consumption. There are two important steps in this method. The first step is to collect kinesthetic demonstrations as prior knowledge, by the space manipulator teaching platform. And the second step is to obtain the joint control torques with DC-LfD algorithm. The second step include two phases: GP-based LfD and trajectory-based controller. GP-based LfD algorithm is used to reproduce the directed trajectory and obtain its confidence intervals based on the current environment. Trajectory-based controller is used to generate the smoother joint control torques. To verify this method, we carry out the simulation and experiment to locate the dexterous hand to the pre-screwing bolt. The results show that this method can reduce the fluctuations and peak-to-peak value of the joint control torques, and also can reduce energy consumption. In the future, we aim to apply this method in contacting tasks and free-flying manipulators.

REFERENCES

[1] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, "A review of space robotics technologies for on-orbit servicing," *Prog. Aerosp. Sci.*, vol. 68, pp. 1–26, Jul. 2014.
[2] W. Xu, B. Liang, and Y. Xu, "Survey of modeling, planning, and ground verification of space robotic systems," *Acta Astron.*, vol. 68, nos. 11–12, pp. 1629–1649, Jun./Jul. 2011.
[3] M. Chu and X. Wu, "Modeling and self-learning soft-grasp control for free-floating space manipulator during target capturing using variable stiffness method," *IEEE Access*, vol. 6, pp. 7044–7054, 2017.
[4] P. Huang, M. Wang, Z. Meng, F. Zhang, and Z. Liu, "Attitude takeover control for post-capture of target spacecraft using space robot," *Aerosp. Sci. Technol.*, vol. 51, pp. 171–180, Apr. 2016.
[5] X. Chen and S. Qin, "Kinematic modeling for a class of free-floating space robots," *IEEE Access*, vol. 5, pp. 12389–12403, 2017.
[6] P. Laryssa, E. Lindsay, O. Marius, K. Nara, L. Aris, T. Ed, and O. Layi, "International space station robotics: A comparative study of ERA, JEM-RMS and MSS," in *Proc. ESA Work. Adv. Space Technol. Robot. Autom.*, Noordwijk, The Netherlands, 2002, pp. 1–8.
[7] C. Zhou, M. Jin, Y. Liu, Z. Xie, and H. Liu, "Hybrid task priority-based motion control of a redundant free-floating space robot," *Chin. J. Aeronaut.*, vol. 30, no. 6, pp. 2024–2033, 2017.
[8] H. Liu, Z. Q. Li, Y. W. Liu, M. Jin, F. Ni, Y. Liu, J. Xia, and Y. Zhang, "Key technologies of TianGong-2 robotic hand and its on-orbit experiments," *Sci. Sinica Technol.*, vol. 48, no. 12, pp. 1313–1320, 2018.
[9] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-based space robotics-ROTEX and its telerobotic features," *IEEE Trans. Robot. Autom.*, vol. 9, no. 5, pp. 649–663, Oct. 1993.
[10] H. C. Schubert and J. P. How, "Space construction: An experimental testbed to develop enabling technologies," *Proc. SPIE*, vol. 3206, pp. 179–188, Dec. 1997.
[11] D. Reintsema, K. Landzettel, and G. Hirzinger, "DLR's advanced telerobotic concepts and experiments for on-orbit servicing," in *Advances in Telerobotics*. Berlin, Germany: Springer, 2007, pp. 323–345.
[12] J. P. Alepuz, M. R. Emami, and J. Pomares, "Direct image-based visual servoing of free-floating space manipulators," *Aerosp. Sci. Technol.*, vol. 55, pp. 1–9, Aug. 2016.
[13] B. Azria and C. Belzile, "Dextre operations 2011 milestones," in *Proc. SpaceOps Conf.*, Stockholm, Sweden, 2012, pp. 1–16.
[14] C. Passenberg, A. Peer, and M. Buss, "A survey of environment-, operator-, and task-adapted controllers for teleoperation systems," *Mechatronics*, vol. 20, no. 7, pp. 787–801, Oct. 2010.
[15] Z. Zhao, J. Yang, S. Li, and W.-H. Chen, "Composite nonlinear bilateral control for teleoperation systems with external disturbances," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 5, pp. 1220–1229, Sep. 2018.
[16] P. Xiong, X. Zhu, A. Song, L. Hu, X. P. Liu, and L. Feng, "A target grabbing strategy for telerobot based on improved stiffness display device," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 661–667, Apr. 2017.
[17] J. Zainan, L. Hong, W. Jie, and H. Jianbin, "Virtual reality-based teleoperation with robustness against modeling errors," *Chin. J. Aeronaut.*, vol. 22, no. 3, pp. 325–333, 2009.
[18] T. B. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Trans. Robot. Autom.*, vol. 9, no. 5, pp. 592–606, Oct. 1993.
[19] C. Preusche, D. Reintsema, K. Landzettel, and G. Hirzinger, "Robotics component verification on ISS ROKVISS—Preliminary results for telepresence," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Syst.*, Beijing, China, Oct. 2006, pp. 4595–4601.
[20] C.-P. Kuan and K.-Y. Young, "VR-based teleoperation for robot compliance control," *J. Intell. Robot. Syst. Theory Appl.*, vol. 30, no. 4, pp. 377–398, 2001.
[21] N. Rodriguez, J.-P. Jessel, and P. Torguet, "A virtual reality tool for teleoperation research," *Virtual Reality*, vol. 6, no. 2, pp. 57–62, 2002.
[22] Y. Xu, C. Yang, J. Zhong, N. Wang, and L. Zhao, "Robot teaching by teleoperation based on visual interaction and extreme learning machine," *Neurocomputing*, vol. 275, pp. 2093–2103, Jan. 2018.
[23] I. Havoutis and S. Calinon, "Learning from demonstration for semi-autonomous teleoperation," *Auton. Robots*, vol. 43, no. 3, pp. 713–726, 2018.
[24] E. Gribovskaya and A. Billard, "Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators," in *Proc. IEEE-RAS Int. Conf. Humanoid Robot.*, Paris, France, Dec. 2009, pp. 472–477.
[25] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013.
[26] T. Liu, "Parallel reinforcement learning: A framework and case study," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 4, pp. 827–835, Jul. 2018.

[27] R. F. Santos, G. A. S. Pereira, and L. A. Aguirre, "Learning robot reaching motions by demonstration using nonlinear autoregressive models," *Robot. Auton. Syst.*, vol. 107, pp. 182–195, Sep. 2018.
[28] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.
[29] C. Fan, Y. Zhang, Z. Xie, Y. Liu, Z. Li, C. Li, and H. Liu, "Design of space robotic arm-hand system and operation research," in *Proc. IEEE Int. Conf. Mechatronics Autom.*, Changchun, China, Aug. 2018, pp. 481–486.
[30] H. Liu, D. Yang, S. Fan, and H. Cai, "On the development of intrinsically-actuated, multisensory dexterous robotic hands," *ROBOMECH J.*, vol. 3, Feb. 2016, Art. no. 4.
[31] J. Xia, Z. Jiang, H. Liu, H. Cai, and G. Wu, "A novel hybrid safety-control strategy for a manipulator," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 1, pp. 1–10, 2014.
[32] C. Li, Z. Jiang, Y. Wang, J. Chen, H. Liu, and H. Cai, "Teleoperation of upper-body humanoid robot platform with hybrid motion mapping strategy," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot.*, Kandima, Maldives, vol. 99, Aug. 2018, pp. 651–656.
[33] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.
[34] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1463–1467, Dec. 2008.
[35] C. E. Rasmussen and C. K. I. Williams, "Regression," in *Gaussian Processes for Machine Learning*. Berlin, Germany: Springer, 2004, ch. 2, pp. 7–31.
[36] R. Featherstone, "Forward dynamics—Inertia matrix methods," in *Rigid Body Dynamics Algorithms*. New York, NY, USA: Springer-Verlag, 2008, pp. 101–118.
[37] D. E. Kirk, "The performance measure," in *Optimal Control Theory: An Introduction*, Chelmsford, MA, USA: Dover, 2004, ch. 2, pp. 29–49.

CHONGYANG LI received the B.S. degree in mechanical and electrical engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2013, and the M.S. degree in mechatronics engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 2015, where he is currently pursuing the Ph.D. degree with the State Key Laboratory of Robotics and System. His research interests include space manipulator, learning from demonstration, optimal control, and human–robot interaction.

ZAINAN JIANG received the B.S., M.S., and Ph.D. degrees in mechatronics engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 2003, 2005, and 2010, respectively. He is currently an Associate Professor and the M.S. Supervisor with the School of Mechatronics Engineering, HIT. His research interests include space robot, space assembly, human–robot interaction, and virtual reality.

ZHIQI LI received the B.S. and M.S. degrees in control science and technology, and the Ph.D. degree in mechatronics from the Harbin Institute of Technology (HIT), in 2006, 2008, and 2016, respectively. He is currently a Research Assistant with the School of Mechatronics Engineering, HIT. His research interests include robot modeling and control, machine vision, and machine intelligence.

CHUNGUANG FAN received the B.S. and M.S. degrees in mechanical and power engineering from the Harbin University of Science and Technology (HUST), Harbin, China, in 2009 and 2012, respectively. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Robotics and System, Harbin Institute of Technology (HIT), Harbin. His research interests include space manipulator, space microgravity, and human–robot interaction.

HONG LIU received the Ph.D. degree from the Harbin Institute of Technology (HIT), China, in 1993. From 1991 to 1993, he was a Joint Ph.D. Student with the Institute of Robotics and System Dynamics, German Aerospace Research Establishment (DLR), Germany, where he has been a Research Fellow, since 1993. He became one of the First batches of Changjiang Scholars, in 1998. He is currently a Professor with HIT. His research projects include the development of dexterous robot hand and space manipulator.