Agent-Based Space Teleoperation: Mitigating Time Delays with Deep Reinforcement Learning

Bo Xia, Xianru Tian, Bo Yuan, *Senior Member, IEEE,* Chunju Yang, Zhiheng Li, *Member, IEEE,* Bin Liang *Senior Member, IEEE* and Xueqian Wang, *Member, IEEE*

***Abstract*—Space teleoperation significantly extends human reach in space missions. However, traditional approaches are constrained by factors such as the reliance on accurate dynamic models and the risk of operator fatigue during prolonged tasks. Additionally, while data-driven intelligent approaches reduce the need for prior knowledge, they have yet to adequately address the time delay issues inherent in these systems. To overcome these challenges, we introduce the Belief State Actor-Critic (BSAC) method, the first deep reinforcement learning approach tailored for space teleoperation capture tasks within a bilateral control framework. We first establish a generalized agent-based architecture for space teleoperation, shifting decision-making from human operators to autonomous agents. Following a comprehensive analysis of the time delay challenges, we propose the BSAC algorithm, which integrates state augmentation and belief state techniques to mitigate the effects of delays in teleoperated Markov decision processes. Extensive experiments are conducted on the MuJoCo simulation platform, modeling a real hardware system across various scenarios. The learned policies are then successfully transferred and validated in a real-world setup, demonstrating the effectiveness and robustness of BSAC. In summary, our results support the feasibility of agent-based frameworks capable of overcoming time delay challenges in space teleoperation.**

***Index Terms*—Space teleoperation, bilateral control, agent-based, time delay, deep reinforcement learning (DRL).**

I. Introduction

WITH the advances in teleoperation technology, space robots have significantly extended astronauts' capabilities in space missions, playing a crucial role in On-Orbit Servicing (OOS) tasks, such as satellite capture, repair, debris removal, and the assembly and maintenance of large space structures [1]–[4]. The most common control modes in teleoperation include teleprogramming control, bilateral control, and virtual predictive control [5]. In teleprogramming control, the operator only issues high-level commands, and the space robot autonomously executes the task [6], [7]. However, this approach demands a level of intelligence from the robot that surpasses current technological capabilities. As a result, the more widely used control modes are bilateral control and virtual predictive control. Both are forms of direct teleoperation, which address time delays caused by data transmission, processing, and inference in the teleoperation process by either complex controller designs [8]–[10] or simulation of the remote environment at the master end [11], [12] to mitigate the impact of delays on decision-making. Despite their effectiveness, these techniques often neglect the human factor. For instance, adults can typically maintain high levels of concentration for only 20-30 minutes [13], and stress further diminishes this focus [14]. Moreover, the remote operation of space robots requires extensive training, and even with this preparation, the operator's real-time performance remains uncertain. With the rapid advancements in artificial intelligence, an intriguing question emerges: can an AI agent, such as the one illustrated in Figure 1, take over teleoperation tasks and perform them more efficiently, accurately, and consistently than a human operator?

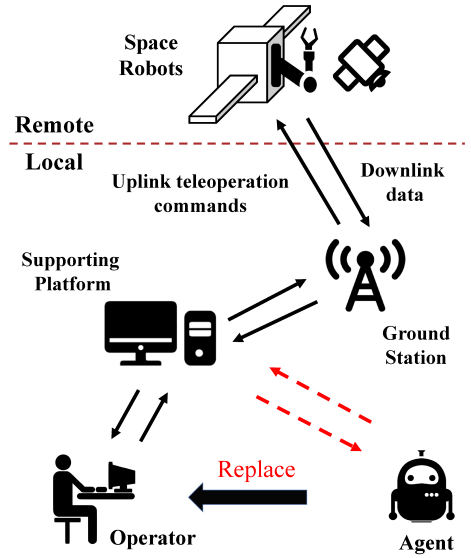


Fig. 1. AI agent replacing human operators in remote space robot operations.

Recently, Deep Reinforcement Learning (DRL) has achieved significant breakthroughs across various domains, including gaming [15], industrial control [16], and large language models [17]. DRL has also been widely employed in space robotics, particularly for tasks such as target capture and trajectory planning for robotic arms. For stationary targets, Soft Q-Learning was applied to achieve target capture with both single-arm and dual-arm configurations [18]. In [19], visual inputs were processed using the Soft Actor-Critic (SAC)

Bo Xia, Xianru Tian, Chunju Yang, Zhiheng Li, and Xueqian Wang are with the Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: xiab21@mails.tsinghua.edu.cn; txr2023214329@gmail.com; ycr24@mails.tsinghua.edu.cn; zhhli@mail.tsinghua.edu.cn; wang.xq@sz.tsinghua.edu.cn).

Bo Yuan is with School of Electrical Engineering and Computer Science, The University of Queensland, QLD 4072, Australia (e-mail: boyuan@ieee.org).

Bin Liang is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: liangbin@tsinghua.edu.cn).

algorithm to generate joint angular velocities to control the UR5 robotic arm. The Deep Deterministic Policy Gradient (DDPG) algorithm was also employed for successful trajectory tracking of a small moving target using a free-floating dual-arm space robot [20]. To enhance the efficiency and precision of exploration in DRL, an ensemble-based version of Proximal Policy Optimization (PPO) was proposed [21]. Similarly, a hybrid strategy was developed by incorporating the inverse kinematics of a fixed-base robot as a prior policy, guiding the agent towards more optimal decisions [22]. By incorporating the geometric relationships among the robotic arm's links and constraining the end-effector's velocity, the reward function was adjusted to prevent potential collisions during motion [23]. A hierarchical control system has also been designed, where the high-level policy manages collision-free trajectory planning for the end-effector's pose, and the low-level policy decomposes each pose into two sub-tasks: position and orientation [24]. Further studies have extended this approach to the end-to-end control of flexible arms, addressing more complex dynamics in space robotics [25], [26].

While these studies have demonstrated the potential of DRL for space robotics, they typically assume a high level of autonomy on the remote robot, which is unrealistic given the current state of technology. In practice, the majority of intelligence resides on the master end, which introduces significant challenges for RL-based control due to communication delays and bandwidth limitations.

Recently, progress has been made in addressing the delay problem in RL, which can be categorized based on the availability of additional information as follows. (1) **Methods with Additional Information**. This category leverages extra data from environments without delays to aid learning in delayed environments. Techniques include directly imitating expert policies from delay-free environments [27], utilizing a limited number of expert trajectories [28], [29], or using delay-free environment data without direct interaction to learn policies for delayed settings [30]. (2) **Methods without Additional Information**. This category can be further classified into three approaches. (a) **State Augmentation**. This approach constructs an information state by combining the most recent observed delayed state and action sequence, thereby transforming the original delayed Markov decision process (MDP) into a new, delay-free MDP [31], [32]. (b) **Model Prediction**. This method uses available information to predict the delayed state, which is incorporated into RL algorithms to make decisions [33]–[35]. (c) **Belief State**. In contrast to model prediction, belief state methods estimate the most informative state for decision-making based on all available information, rather than predicting actual states [36]–[38].

Driven by advancements in these methods, this paper is the first to successfully apply DRL to achieve ground-based teleoperation of a space robotic arm for capture tasks. First, a Markov model is developed for the teleoperation process, providing a theoretical framework for decision-making. Then, a novel algorithm, Belief State Actor-Critic (BSAC), is proposed, which integrates state augmentation and belief state techniques in a model-free paradigm, capable of handling both constant and random delay scenarios. Finally, the algorithm is tested in the MuJoCo environment, and transferred to a physical system for real-world validation. The results from both environments confirm the feasibility and robustness of the proposed approach.

The main contributions of this paper are as follows:

- **A novel agent-based teleoperation framework**. We propose a general framework to replace human operators with intelligent agents for space robot teleoperation, modeled as an MDP to support theoretical analysis.
- **The Belief State Actor-Critic (BSAC) algorithm**. Our proposed BSAC algorithm combines state augmentation and belief state techniques to handle both constant and random time delays, ensuring robustness and adaptability in teleoperation tasks.
- **General Markov modeling for space robotic decision-making**. We design key elements of the MDP, including the state space and reward function, to better guide agents in completing tasks, forming a foundation for effective experimental validation.
- **Validation in simulation and real-world system**. The BSAC algorithm is validated through strategy training in MuJoCo simulations and demonstrates its efficiency and robustness on a real-world physical system.

This paper is structured as follows. Section II outlines the background, including the kinematics of space robots and their modeling as a Markov process. Section III details the methodology, analyzing the time-delay issues in teleoperation and developing the BSAC algorithm to mitigate the impact of delays. Section IV presents the validation and analysis of the proposed framework and algorithm, using both simulation and real-world systems. This paper is concluded in Section V with directions for future work.

II. Background

A. Kinematics of space robots

A space robotic system typically consists of a spacecraft (base) and a manipulator with n degrees of freedom (DOF). The vector of joint angles of the manipulator is denoted as $q = [\theta_1, \theta_2, \cdots, \theta_n]^T$, with the corresponding joint velocities represented as $\dot{q}$. Additionally, $(v_i, w_i)^T$ $(i = b, e)$ represent the velocities of the base and the robotic arm, respectively. Following [39], we can derive the following equation:

$$\begin{pmatrix} v_e \\ \omega_e \end{pmatrix} = J_b \begin{pmatrix} v_b \\ \omega_b \end{pmatrix} + J_m \dot{q}, \tag{1}$$

where J_b and J_m are the Jacobian matrices of the base and the manipulator, respectively. In the absence of external forces or torques, the space robot operates in free-floating mode and its linear momentum P and angular momentum L are conserved. Therefore, it holds that:

$$\begin{pmatrix} P \\ L \end{pmatrix} = H_b \begin{pmatrix} v_b \\ \omega_b \end{pmatrix} + H_{bm} \dot{q}, \tag{2}$$

where H_b and H_{bm} are inertia matrix and coupling inertia matrix, respectively. Assuming the initial values of P and L are both zero, the velocity of the base can be expressed as:

$$\begin{pmatrix} v_b \\ \omega_b \end{pmatrix} = -\mathbf{H_b^{-1}} \mathbf{H_{bm}} \dot{q} = \mathbf{J_{bm}} \dot{q}, \tag{3}$$

where $\mathbf{J_{bm}}$ is the Jacobian matrix of the base. Substituting Eq. (3) into Eq. (1) yields the following expression:

$$\begin{pmatrix} v_e \\ \omega_e \end{pmatrix} = (\mathbf{J_b J_{bm}} + \mathbf{J_m})\dot{q} = \mathbf{J_g}\dot{q}, \tag{4}$$

where $\mathbf{J_g}$ is referred to as the generalized Jacobian matrix, which is associated not only with kinematic parameters but also with dynamic parameters.

B. Reinforcement learning for space robots

RL is a sequential decision-making process grounded in the theoretical framework of MDP. Typically, an MDP is defined by a six-element tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \rho, \gamma)$, where $\mathscr{S}$ and $\mathscr{A}$ represent the state space and action space, respectively; $\mathscr{P}$ is the state transition function and $\mathscr{R}$ denotes the reward function, while ρ represents the initial distribution of states; γ signifies the discount factor for future rewards. At time step t, the agent observes the environmental state s_t. Subsequently, based on its current policy π, it selects an action $a_t \sim \pi(\cdot|s_t)$ to apply to the environment. The environment then returns a new state $s_{t+1} \sim \mathscr{P}(\cdot|s_t, a_t)$ along with a reward $r_t = R(s_t, a_t)$. The objective of reinforcement learning is to maximize the cumulative reward $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ to obtain an optimal policy $\pi^*(\cdot|s_t)$, where T is the maximum number of steps the agent interacts with the environment in an episode.

The teleoperation process of the space robotic arm is modeled as an MDP. Specifically, the state space, action space, and reward function are defined as follows.

• State space $\mathscr{S}$

Given the critical importance of the state information in guiding the agent's decisions, it is essential to incorporate as many relevant features as possible when designing the state representation. During task execution by the space robot, key factors include the joint angles $q \in \mathbb{R}^n$, joint angular velocities $\dot{q} \in \mathbb{R}^n$, end-effector pose $p_e \in \mathbb{R}^6$ and velocity $v_e \in \mathbb{R}^6$, target pose $p_{target} \in \mathbb{R}^6$, and the distance $d \in \mathbb{R}$ between the end-effector and the target. Particularly in free-floating mode, the non-holonomic constraints between the robot's base and its manipulator can significantly influence motion planning. Hence, the base pose $p_b \in \mathbb{R}^6$ and base velocity $v_b \in \mathbb{R}^6$ are also essential components of the state space. Accordingly, the state design for the fixed-base mode is as follows:

$$s_t = (q, \dot{q}, p_e, v_e, p_{target}, d) \in \mathbb{R}^{2n+19}; \tag{5}$$

For the free-floating mode, the state is represented as:

$$s_t = (q, \dot{q}, p_e, v_e, p_b, v_b, p_{target}, d) \in \mathbb{R}^{2n+31}. \tag{6}$$

• Action space $\mathscr{A}$

The actions are defined as the torques applied to the joints of the robotic arm, with its dimensionality corresponding to the degrees of freedom of the joints. Specifically, the action $a_t \in \mathbb{R}^n$ satisfies $a_{ti} \in [-\max(\tau_i), \max(\tau_i)]$, where $i = 1, \cdots, n$, and τ_i represents the torque applied to the i-th joint.

• Reward function $\mathscr{R}$

The reward function r_t at each time step consists of three parts: task reward $r_{task}(t)$, constraint reward $r_{constraint}(t)$, and terminal reward $r_{terminal}(t)$, as defined below:

$$r_t = r_{task}(t) + r_{constraint}(t) + r_d(t), \tag{7}$$

$$r_{task}(t) = \omega_1 d_t + \omega_2 \log(d_t + 1 \times 10^{-6}), \tag{8}$$

$$\begin{aligned} r_{constraint}(t) =& \omega_3 \|a_t - a_{t-1}\| + \omega_4 \left(\|(v_e)_t\| + \|(v_b)_t\|\right) \\ &+ \delta(check\_collision) C_{obs}, \end{aligned} \tag{9}$$

$$r_d(t) = \delta(t < T)\delta(d_t \leq d_{threshold}) \times C, \tag{10}$$

where the $\delta(\cdot)$ is the indicator function, $\delta(check\_collision)$ indicates whether a collision occurs, C_{obs} is the penalty for such collisions and $d_{threshold}$ is the maximum allowable distance between the end-effector and the target for successful capture. The three rewards serve the following purposes: (1) $r_{task}(t)$ is related to the capture task. $\omega_1 d_t$ encourages the end-effector of the robotic arm to approach the target as closely as possible, while $\omega_2 \log(d_t + 1 \times 10^{-6})$ prevents the end-effector from stalling near the target by ensuring continuous progress even when the distance is close to the threshold; (2) $r_{constraint}(t)$ is associated with constraints. $\omega_3 \|a_t - a_{t-1}\|$ ensures that the applied torques remain smooth during the execution of actions, $\omega_4 \left(\|(v_e)_t\| + \|(v_b)_t\|\right)$ limits the velocity of the space robot's end-effector and base to avoid excessive motion, and $\delta(check\_collision)C_{obs}$ penalizes collisions of the robotic arm; (3) $r_d(t)$ defines the terminal reward, which provides a constant reward C if the task is completed within T and the distance to the target is below the threshold.

III. Method

This section provides an in-depth analysis of the challenges posed by time delays in space teleoperation. It also presents a novel model-free DRL algorithm, the Belief State Actor-Critic (BSAC), which employs state augmentation and belief state techniques to mitigate the negative effects of delays on decision-making.

A. Time delays in teleoperation

As shown in Figure 1, delays caused by factors such as data transmission, processing, and inference disrupt the Markov property of the original sequential decision-making process, significantly degrading the performance of RL algorithms. In this work, we propose replacing human operators with intelligent agents to carry out space teleoperation tasks. At each time step, the agent makes decisions based on the information received from the environment and its current policy. However, a delay occurs between when the agent issues a command and when the remote environment receives and acts on it. This delay, referred to as action delay [31], is denoted as d_a. Additionally, once the remote space robot processes the action, there is another delay in sending the feedback to the agent, known as observation delay [31], denoted as d_o. The entire process is illustrated in Figure 2.

Ideally, the total delay in this loop is $d = d_o + d_a$, known as the round-trip delay (RTD), which is assumed to be constant. However, network congestion and other environmental factors can introduce variability, modeled as a random delay following a uniform distribution with parameter ξ, denoted as $d_{random} \sim Uniform(\xi)$. The actual delay at any time step is then expressed as $d_{total}(t) = d + d_{random}(t)$. At time t, the

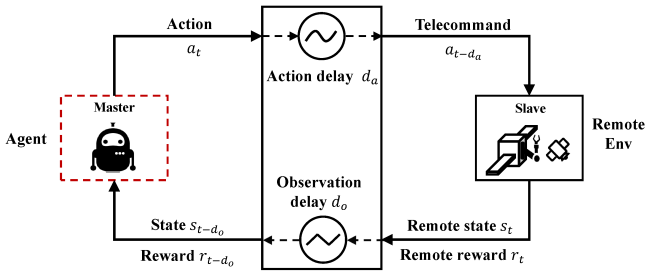


Fig. 2. Agent-based space teleoperation framework. The diagram illustrates two primary delays in the system: observation delay d_o and action delay d_a. These delays cause a lag in the space robot's actions relative to the agent's decisions by d_a, and a lag in the agent's observations relative to the space robot's feedback by d_o.

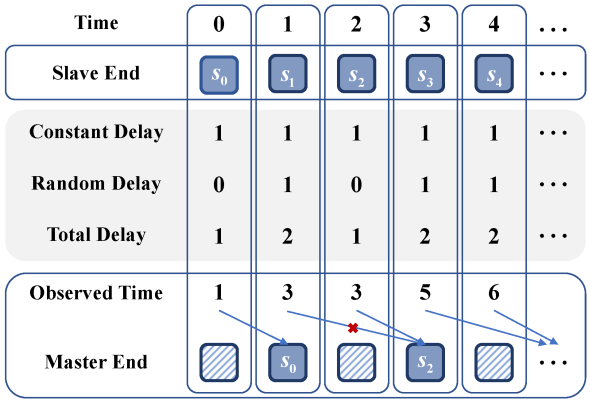


Fig. 3. Explanation of Random Delay Scenarios. The delay consists of both fixed and random components, causing variations in observation times at both ends. When multiple states occur at different times at the Master End, only the most recent state is observed, while the others are discarded.

state generated by the remote space robot is represented by s_t, while the state observed by the agent is represented by o_t.

For instance, assuming $d = 1$ and $\xi = 1$, an example of random delay is shown in Figure 3. At $t = 0$, with $d_{total}(0) = 1$, the agent does not observe the state at $t = 0$, and instead observes s_0 at $t = 1$, leading to $o_0 = \Phi$ and $o_1 = s_0$. Similarly, at $t = 1$, $d_{total}(1) = 2$, and at $t = 2$, $d_{total}(2) = 1$, implying that the agent does not observe the state at $t = 2$. Consequently, both s_1 and s_2 may be observed at $t = 3$. However, outdated states are typically discarded, so the state observed at $t = 3$ would be s_2, resulting in $o_3 = s_2$.

Remark 1. To facilitate the subsequent description of the algorithm's state space, the delay at any given time t, where t starts from 0, is defined as follows:

$$z_t = \begin{cases} t+1, & \text{if } t < d, \\ z_{t-1}+1, & \text{if } o_t = \Phi \text{ and } t \geq d, \\ d, & \text{otherwise.} \end{cases} \tag{11}$$

Remark 2. When $\xi = 0$, this corresponds to the constant delay scenario. In this case, for the first d time steps, the agent does not observe any information, i.e., $o_t = \Phi$, where Φ represents an empty or unobserved state. After the initial delay, the agent begins observing the state with a lag of d time steps, represented as:

$$o_t = \begin{cases} \Phi, & \text{if } t < d, \\ s_{t-d}, & \text{otherwise.} \end{cases} \tag{12}$$

Remark 3. In the following experiments, unless otherwise specified, the default assumption is that $d_a = d_o = d/2$.

B. Solutions to time delays

To effectively mitigate the impact of delays in space teleoperation, we propose the BSAC algorithm, which combines state augmentation and belief state techniques.

Part 1. State Augmentation

The core idea is to utilize the delayed state and historical action sequences to construct an augmented **information state**, enabling the transformation of the original delayed MDP into a delay-free MDP [31]. In the context of space teleoperation, the key components are defined as follows:

• State space $\mathcal{X}$

The augmented state is represented as:

$$x_t = \begin{cases} \Phi, & \text{if } o_t = \Phi \text{ and } t < d, \\ (o_{t-z_t}, a_{t-d}, \cdots, a_{t-1}), & \text{if } o_t = \Phi \text{ and } t \geq d, \\ (o_t, a_{t-d}, \cdots, a_{t-1}), & \text{otherwise,} \end{cases} \tag{13}$$

which will be used to determine the agent's action at time t.

• Action space $\mathcal{A}$

This follows the same definition as in Section II-B.

• State transition function $\mathcal{P}$

At time t, the state x_t follows the new state transition function $x_t \sim \mathcal{P}(\cdot|x_{t-1}, a_t)$.

• Reward function $\mathcal{R}$

To account for the delay process, and based on Eq. (7), the reward function is defined as:

$$\mathcal{R}(t) = \begin{cases} 0, & \text{if } o_t = \Phi, \\ r(x_t, a_t), & \text{otherwise.} \end{cases} \tag{14}$$

• Initial state distribution ζ

The new initial state distribution is given by:

$$\zeta(x_0) = \rho(s_0) \prod_{i=0}^{d-1} \delta(a_i - c_i), \tag{15}$$

where $\rho(s_0)$ is the original initial state distribution, and $\{c_i\}_{i=0}^{d-1}$ represents random actions taken by the operator without state feedback from the remote environment.

Part 2. Belief State

The agent's objective function in RL is typically defined as:

$$\mathcal{J}(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(x_t,a_t)\sim D}\left[r(x_t, a_t)\right], \tag{16}$$

where D is the distribution of states and actions obtained from either a replay buffer or previously sampled data. The goal is to optimize this objective function to derive an optimal policy. However, due to the complexity of end-to-end space teleoperation tasks, optimizing Eq. (16) often leads to policies that can get stuck in local optima, making it difficult to

find policy parameters that yield high cumulative rewards. To address this issue, the objective function is modified as follows:

$$\mathcal{J}(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(x_t,a_t)\sim D} \left[r(x_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|x_t)) \right], \quad (17)$$

where $\mathcal{H}(\pi(\cdot|x_t))$ denotes the entropy of the policy, and α is a coefficient that controls the balance between the reward and the entropy term. This modification not only encourages more effective exploration but also helps manage the stochasticity of the optimal policy.

As discussed in **Part 1**, the dimensionality of the state space increases from $dim(\mathscr{S})$ to $dim(\mathscr{S}) \times dim(\mathscr{A})^d$, where d is the delay. This exponential increase in dimensionality not only expands the policy search space but also imposes greater demands on storage, computation, and the volume of interaction samples, leading to what is commonly known as the "curse of dimensionality." In [38], for constant delay scenarios, the problem was addressed by employing belief projection of the augmented state, rather than directly calculating the Q/V values for the augmented state. Following this approach, we propose the BSAC algorithm to manage scenarios with random delays in space teleoperation, effectively overcoming the significant challenges posed by time delays.

During the policy evaluation phase, the soft Bellman operator is formulated as follows based on Eq. (17):

$$\begin{aligned} \mathscr{T}Q(x_t, a_t) \rightarrow & \mathbb{E}_{p(s_t|x_t)}[\mathscr{R}(s_t, a_t)] + \gamma \mathbb{E}_{x_{t+1}\sim\mathcal{P}, a_{t+1}\sim\pi}[\\ & Q(x_{t+1}, a_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|x_{t+1}))], \end{aligned} \quad (18)$$

where $p(s_t|x_t)$ represents the probability of transitioning from the augmented state to the current belief state. The "curse of dimensionality" arises from evaluating state-action pairs (x_t, a_t) in high-dimensional spaces. However, for each augmented state x_t, there exists a corresponding low-dimensional belief state s_t in the training phase. Thus, we can approximate the Q-value as:

$$Q(x_{t+1}, a_{t+1}) \approx \mathbb{E}_{p(s_{t+1}|x_{t+1})}[\mathcal{Q}(s_{t+1}, a_{t+1})], \quad (19)$$

where $\mathcal{Q}(\cdot, \cdot)$ denotes the Q-value corresponding to the belief state. To estimate the belief state action-value function, we minimize the following objective:

$$\mathcal{J}_\mathcal{Q} = \mathbb{E}_{x_t\sim\zeta, a_t\sim\pi}[\mathbb{E}_{p(s_t|x_t)}[\mathcal{Q}(s_t, a_t)] - Q(x_t, a_t)]^2. \quad (20)$$

Substituting Eq. (18) and Eq. (19) into the above equation, we obtain:

$$\begin{aligned} \mathcal{J}_\mathcal{Q} = & \mathbb{E}_{x_t\sim\zeta, a_t\sim\pi}[\mathbb{E}_{p(s_t|x_t)}[\mathcal{Q}(s_t, a_t) - \mathscr{R}(s_t, a_t)] - \\ & \gamma \mathbb{E}_{x_{t+1}\sim\mathcal{P}, a_{t+1}\sim\pi}[\mathbb{E}_{p(s_{t+1}|x_{t+1})}[\mathcal{Q}(s_{t+1}, a_{t+1})] + \\ & \alpha \mathcal{H}(\pi(\cdot|x_{t+1}))]]^2. \end{aligned} \quad (21)$$

In the policy improvement phase, the optimal policy is typically obtained by minimizing the KL-divergence associated with the augmented state x_t, as shown below:

$$argmin_\pi D_{KL} \left(\pi(\cdot|x_t) \,||\, \frac{exp(\mathbb{E}_{p(s_t|x_t)} \left[\mathcal{Q}^{\pi^{old}}(s_t, \cdot) \right])}{Z^{\pi^{old}}(x_t)} \right). \quad (22)$$

Algorithm 1 Belief State Actor-Critic for Teleoperation

```
Initialize network parameters θ and φ, target networks, temporary buffer B, replay buffer D.
for each iteration do
  for t ∈ [0, d − 1] do
    a_t ∼ random(−τ_max, τ_max).
    o_t ← delayed_env(a_t).
    Store a_t and non-null o_t in B.
  end for
  for t ∈ [d, T] do
    get x_t from B.
    a_t ∼ π_φ(x_t).
    (o_t, R(t)) ← delayed_env(a_t).
    if o_t = Φ then
      o_t = x_t [: dim(S)].
    end if
    Store a_t and o_t in B.
    if t > 2d then
      get x_{t−d}, o_{t−d}, a_{t−d}, x_{t−d+1}, o_{t−d+1} from B.
      D ← D∪{(x_{t−d}, o_{t−d}, a_{t−d}, r_{t−d}, x_{t−d+1}, o_{t−d+1})}.
    end if
  end for
  Sample data from D and update θ and φ by gradient descent to optimize Eq. (24) and Eq. (25).
end for
```

As proven in [38], Eq. (22) can be transformed into a KL-divergence related to the belief state:

$$argmin_\pi \mathbb{E}_{p(s_t|x_t)} \left[D_{KL}(\pi(\cdot|x_t)||\frac{exp(\mathcal{Q}^{\pi^{old}}(s_t, \cdot))}{Z^{\pi^{old}}(s_t)}) \right], \quad (23)$$

the corresponding proof can be found in the appendix A.

Therefore, in both the policy evaluation and improvement stages, the Q-value estimation shifts from being based on the augmented state to being based on the belief state.

We then train the actor and critic using neural networks, with parameters θ and ϕ, respectively. The corresponding Eq. (21) and Eq. (23) are reformulated as:

$$\begin{aligned} \mathcal{J}_\mathcal{Q}(\theta) = & \mathbb{E}_{(x_t, s_t, a_t, r_t, x_{t+1}, s_{t+1})\sim D}[\frac{1}{2}(\mathcal{Q}_\theta(s_t, a_t) - r_t \\ & - \gamma \mathbb{E}_{a_{t+1}\sim\pi}[\mathcal{Q}_\theta(s_{t+1}, a_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|x_{t+1}))])^2], \end{aligned} \quad (24)$$

and

$$\mathcal{J}_\pi(\phi) = \mathbb{E}_{(x_t, s_t)\sim D}[\mathbb{E}_{a_t\sim\phi}[\alpha log \pi(a_t|x_t) - \mathcal{Q}_\theta(s_t, a_t)]]. \quad (25)$$

By iteratively minimizing these two objective functions, we can obtain the actor and critic networks optimized for space teleoperation.

The pseudocode of the algorithm is shown as Algorithm 1.

IV. Experiments

This section details the experiments designed to validate the effectiveness and robustness of the BSAC algorithm. It includes the experimental setup, comparative studies, and further discussions on normal and special scenarios.

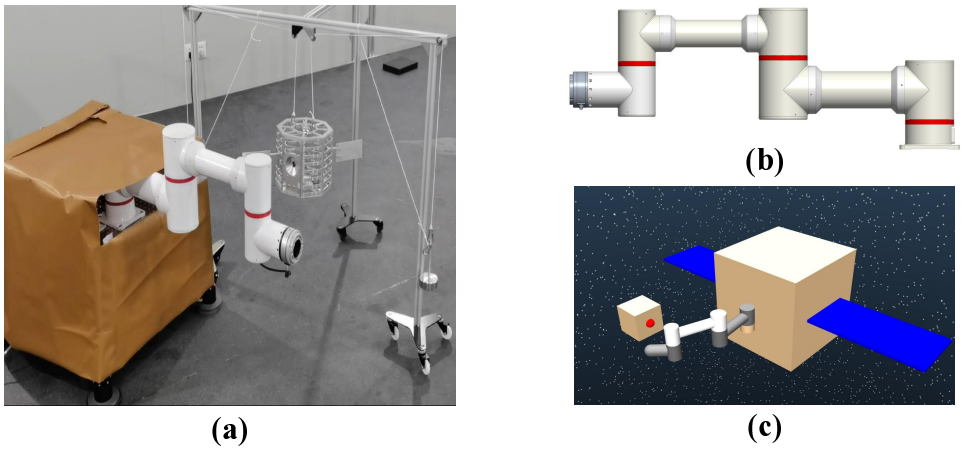


Fig. 4. System Introduction. (a) Hardware System; (b) Manipulator Configuration; (c) Simulation System.

TABLE I
KINEMATIC AND DYNAMIC PROPERTIES.

Link No.	Shape	Inertial properties
0 (base)	Box, length 0.7	$m = 60$ $I = diag(10, 9, 11)$
1	Cylinder, $r = 0.05$, $l = 0.123$ Cylinder, $r = 0.0365$, $l = 0.3$ Cylinder, $r = 0.05$, $l = 0.12$	$m = 6.6$ $I = diag(1, 1, 0.28)$
2	Cylinder, $r = 0.05$, $l = 0.108$ Cylinder, $r = 0.0325$, $l = 0.307$ Cylinder, $r = 0.043$, $l = 0.1116$	$m = 2.456$ $I = diag(1, 1, 0.14)$
3	Cylinder, $r = 0.043$, $l = 0.113$ Cylinder, $r = 0.0375$, $l = 0.144$	$m = 0.96$ $I = diag(1, 1, 0.0068)$

* All data are presented in standard units.

TABLE II
JOINT PROPERTIES.

Joint No.	Position Range	Torque range
1	$[-\pi/2, \pi/2]$	$[-0.5, 0.5]$
2	$[-\pi, \pi]$	$[-0.5, 0.5]$
3	$[-\pi, \pi]$	$[-0.5, 0.5]$

* All data are presented in standard units.

A. Experiment setup

Experiment System. The hardware system is shown in Figure 4(a) and Figure 4(b), with the parameters of the links and joints listed in Tables I and II, all using standard units. Additionally, we installed four pneumatic feet at the base to achieve free-floating capabilities for the space robot via pneumatic buoyancy. To validate our algorithm, we adopt a strategy where the policy is first trained in a simulation environment before being transferred to the real system. Accordingly, we developed a space robot simulator using the MuJoCo physics engine, referred to as "FreeFloating-v0", as shown in Figure 4(c). Furthermore, considering the hardware control frequency of 1 kHz and the time required for the agent to make and execute decisions, the simulation time step is set as $T_m = 0.04\ s$ per step. The maximum number of steps per episode is $T = 250$, corresponding to a maximum simulation duration of $T \cdot T_m = 10$ seconds.

Task description. The radius of the robotic arm's end effector is 0.0375 m, and the radius of the target point is assumed to be 0.01 m. We define a threshold distance $d_{\text{threshold}} = 0.05\ m$, where the task is considered successfully accomplished if the distance between the end effector and the target is less than this value, satisfying the condition $d_{\text{threshold}} > 0.0375 + 0.01 = 0.0475\ m$. To enhance model generalization, the target point is randomly placed within a rectangular region. Additionally, noise is introduced into the initial joint angles and velocities to further test the model's robustness. Specifically, the initial position of the base in the world coordinate system is $p_b = [0, 0, 1]$, and the target position is $p_s = [-0.25 \pm 0.15, -0.75 \pm 0.15, 1]$. The initial joint angles and velocities of the robotic arm are $q_0 = [0, 0, 0]$ and $v_0 = [0, 0, 0]$, with added noise defined as $q_{\text{noise}} = [q_{j1}, q_{j2}, q_{j3}]$ and $v_{\text{noise}} = [v_{j1}, v_{j2}, v_{j3}]$, where $q_{ji} \in Uniform(-\pi/6, \pi/6)$ and $v_{ji} \in Uniform(-0.005, 0.005)$, for $i = 1, 2, 3$. Thus, the initial joint angles and velocities are set to $q = q_0 + q_{\text{noise}}$ and $v = v_0 + v_{\text{noise}}$.

Network architecture. Based on the current hardware system and the analysis in Section II-B, the network architecture consists of two main components: the actor network and the critic network. Both networks feature two hidden layers, each with 256 units, and use ReLU as the activation function. The primary difference between them lies in their input and output dimensions: the actor network has an input size of 37 and an output size of 3, while the critic network has an input size of 37 and an output size of 1. The discount factor γ is set to 0.99, while the replay buffer size is 1,000,000, and the batch size is 256. The Adam optimizer is employed for training. In Eq. (8), Eq. (9) and Eq. (10), the weights ω_1 to ω_4 are set to -1, -0.1, -0.1, and -0.1, respectively, with a constant C of 100 and a constant C_{obs} of -1.

Operating platform. All experiments are conducted on an NVIDIA GeForce RTX 3090 graphics card. The versions

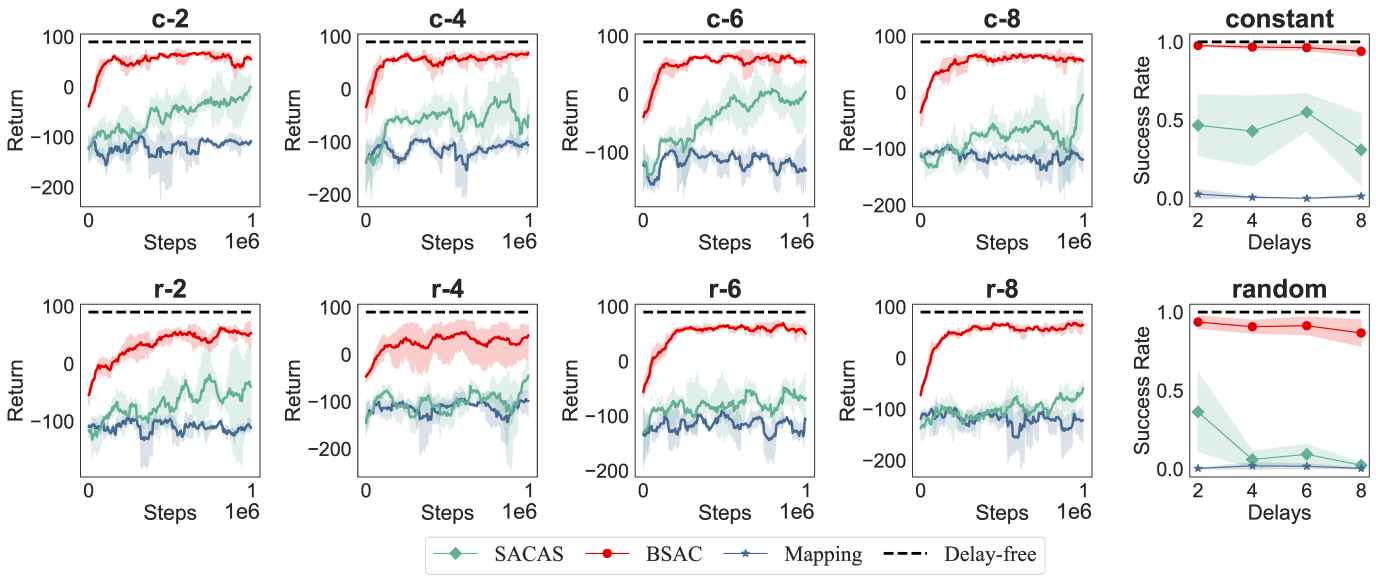


Fig. 5. Performance comparison of different algorithms in "FreeFloating-v0" on MuJoCo.

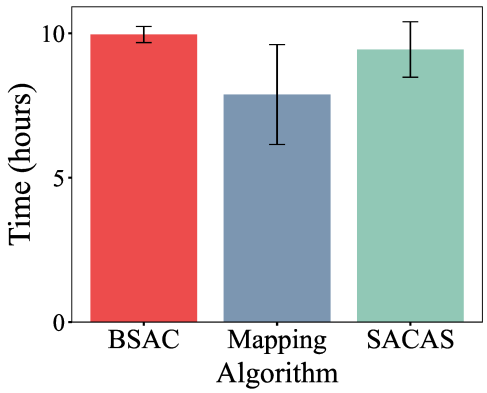


Fig. 6. Running times of different algorithms in "FreeFloating-v0" on MuJoCo.

of Gym, MuJoCo, and PyTorch used in the experiments are 0.21.0, 2.0.2.8, and 1.11, respectively.

B. Results

We compare BSAC against two widely used state-of-the-art (SOTA) algorithms: (1) a memoryless approach, which ignores delays and makes decisions based solely on the observed state (referred to as Mapping); (2) a state augmentation approach using MDP transformation (denoted as SACAS). All algorithms are SAC-based and evaluated under constant and random delay conditions with delays of 2, 4, 6, and 8 timesteps. The results, shown in Figure 5, represent the average performance over three different random seeds. The first four columns depict the training process, labeled as "scenario-delay" (with "c" for constant and "r" for random), while the last column shows the success rate over 100 trials with the trained models. Additionally, the term "Delay-free" in Figure 5 refers to the final performance achieved using the SAC algorithm in a delay-free environment, representing the best possible performance that the agent can attain.

As illustrated in Figure 5: (1) All algorithms experience a decline in performance as the delay increases in both constant and random delay scenarios, indicating that delays, particularly substantial and random delays, significantly impact the control of the agent by these algorithms. (2) BSAC outperforms the other algorithms in both convergence speed and final returns, achieving near delay-free performance and higher success rates across all delay scenarios. SACAS shows stable performance under constant delays, maintaining a success rate around 0.5, but its performance degrades significantly with increasing random delays. Mapping performs poorly in all scenarios, essentially failing to learn a useful strategy; (3) BSAC performs better in constant delay scenarios compared to random ones, although the performance drop in random delay scenarios is minimal. These results highlight the robustness and effectiveness of BSAC.

C. Discussion on normal scenarios

This section compares the time consumption of different methods, analyzes performance with fixed round time delay with different delay pairs, and evaluates the trained strategy under varying bass masses, all under normal conditions.

Time complexity. Using the same experimental setup as in Section IV-A, we measure the training times of all three algorithms across various scenarios and averaged the results, as shown in Figure 6. It is clear that Mapping has the shortest training time, followed by SACAS, with BSAC taking slightly longer. The increased time for BSAC can be attributed to the additional processing required for handling both augmented and corresponding true states. However, given BSAC's superior performance over the other algorithms, this extra time is well justified.

Fixed round time delay with different delay pairs. In all previous experiments, we evenly divide the constant delay d

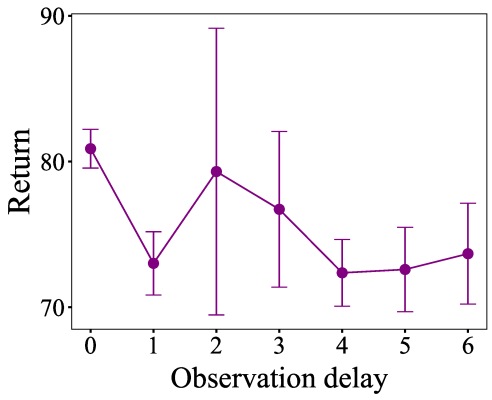


Fig. 7. Returns of BSAC with fixed round time delay and varying observation delays.

TABLE III
COMPARISON OF VARYING MASS OF THE BASE.

Base mass	**60 (original)**	30	45	75	90
Success rate	**0.95**	0.75	0.89	0.92	0.94

* All data are presented in standard units.

TABLE IV
COMPARISON OF PERIODIC JOINT FAILURE.

Faulty joint No.	**Duration T_1**	**Success Rate**	**Average Step**
1	5	0.94	120.48±39.66
	10	0.75	147.49±64.63
	15	0.65	166.25±67.55
2	5	0.95	129.9±35.76
	10	0.89	168.36±47.44
	15	0.45	212.08±55.14
3	5	1.0	121.48±22.6
	10	0.97	131.92±33.23
	15	0.98	139.46±30.19
1,2,3	5	0.98	141.34±30.22
	10	0.91	181.53±41.19
	15	0.18	247.31±15.97

TABLE V
COMPARISON OF RANDOM JOINT FAILURE.

q	**Success Rate**	**Average Step**
0.2	0.94	135.53±39.16
0.5	0.88	188.05±42.38
0.8	0.08	249.36±14.83

into observation delay d_o and action delay d_a. To investigate whether varying these delays significantly impacts algorithm performance, we conduct experiments with $d = d_o + d_a = 6$, varying the observation delay d_o across values of 0, 1, 2, 3, 4, 5, and 6. The results, presented in Figure 7, reveal that the final returns across all scenarios hover around 75 and the highest average returns occur when d_o is 0 or 2, the latter exhibiting greater variance. This suggests that, with d held constant, the specific values of d_o and d_a have a minimal effect on final returns, consistent with the conclusions drawn in [31]. This phenomenon can be further elucidated by noting that when d remains constant, the state and corresponding actions remain unchanged, leading to relatively stable performance.

Varying Mass. In real-world applications, the mass of a space robot's base can fluctuate due to factors such as fuel consumption or payload changes, potentially impacting the effectiveness of pre-trained policies. To evaluate BSAC's robustness under varying mass conditions, we use the mass parameters from Section IV-A, training a policy with a constant delay of 6. We then test this policy in scenarios where the base mass is adjusted to 30, 45, 75, and 90 kg (with corresponding changes to inertia), conducting 100 trials for each configuration. The success rates are summarized in Table III. The results indicate that when the base mass is reduced (30 kg and 45 kg), the success rates are noticeably lower than those in the original environment. However, when the base mass exceeds the original (75 kg and 90 kg), the success rates remain similar or even match the original performance. This suggests that when the mass is lower, the nonlinear holonomic constraints between the base and the robotic arm lead to stronger dynamic coupling in the free-floating scenario, degrading the policy's performance. Conversely, with a heavier base, the arm's movements exert less influence on the base, which remains more stable, allowing the policy to maintain its efficacy despite the dynamic coupling.

D. Discussion on special scenarios

This section analyzes the application of BSAC in specific scenarios, including joint faults, communication failures, task generalization, and real-system validation. Unless specified otherwise, all subsequent experiments are conducted with a delay of 6, where the previously learned strategies are directly transferred to the new scenarios.

Joint Failures. Two joint failure modes are considered: periodic and random.

In the periodic mode, a joint receives torque commands from the agent but remains inactive (applying zero torque) for a period T_1 every T_0 steps. We set $T_0 = 20$ and $T_1 = 5, 10, 15$ for testing. Experiments are conducted under two conditions: (a) a single joint failure and (b) failure of all three joints. The results are summarized in Table IV. The findings indicate that: (1) The impact of joint failures and failure durations varies across joints, with the BSAC algorithm demonstrating significant robustness to these failures. (2) The first two joints play a more critical role in task completion compared to the third joint, consistent with the arm's configuration. (3) Interestingly, for $T_1 = 10$, the success rate is lower when one of the first two joints fails compared to when all three joints fail simultaneously. However, the average number of steps per trial shows the opposite trend. This can be explained by joint coupling: when all three joints fail, the arm tends to maintain a posture closer to its pre-failure state, requiring more time for decision-making. In contrast, when only one joint fails, the arm's configuration deviates from the optimal posture, reducing the success rate.

In the random mode, each joint fails with a fixed probability q at every step. When functional, the joint executes the

TABLE VI
COMPARISON OF NOISY OBSERVED DATA.

δ	**0(original)**	**0.005**	**0.01**
Success Rate	0.95	0.91	0.88
Average Step	126.89±37.4	129.37±47.09	136.14±53.59
δ	**0.015**	**0.02**	**0.025**
Success Rate	0.84	0.68	0.62
Average Step	153.16±54.64	171.15±63.61	178.18±68.43

TABLE VII
COMPARISON OF PACKET LOSS WITH DIFFERENT RATES.

p	**Success Rate**	**Average Step**
0.2	0.94	129.07±33.79
0.5	0.92	136.71±34.49
0.8	0.87	143.98±51.21
1.0	0.0	250.0±0.0

TABLE VIII
COMPARISON OF DIFFERENT COMMUNICATION INTERRUPTIONS.

T_1	**Success Rate**	**Average Step**
5	0.94	128.17±39.96
10	0.89	146.46±47.73
15	0.81	162.48±54.06
20	0.0	250.0±0.0

TABLE IX
COMPARISON OF NOISY ACTION DATA.

δ	**0(original)**	**0.1**	**0.2**
Success Rate	0.95	0.94	0.91
Average Step	126.89±37.4	127.42±34.9	130.01±38.71
δ	**0.3**	**0.4**	**0.5**
Success Rate	0.91	0.88	0.85
Average Step	137.61±34.37	148.61±38.76	160.11±39.49

commanded torque; otherwise, it remains inactive with zero torque. Experiments are conducted with $q = 0.2, 0.5, 0.8$, and the results are summarized in Table V. The results show that as q increases, task performance deteriorates. However, even with a failure probability of $q = 0.5$, the BSAC algorithm maintains a success rate close to 90%, demonstrating strong robustness to this type of fault.

Communication Failures. This part covers three scenarios: signal distortion, packet loss, and communication interruptions.

(1) Signal Distortion: Observation noise caused by factors such as sensor and transmission is modeled as Gaussian noise, $s_\Delta(t) \sim \mathcal{N}(0, \delta^2)$, with $\delta = 0.005, 0.01, 0.015, 0.02, 0.025$. The observed state becomes $s_{noise}(t) = s_t + s_\Delta(t)$. For each δ, 100 trials are conducted, recording task success rates and the average number of steps per trial. Results are shown in Table VI. The data indicates that Gaussian noise impacts policy performance, with larger noise leading to greater degradation. However, even with $\delta = 0.015$, the agent achieved an 80% success rate, and with $\delta = 0.025$, the success rate remained above 60%, with only 60 additional steps compared to the noise-free environment.

(2) Packet Loss: To simulate packet loss, communication packets are randomly dropped with a probability p. When a packet is lost, the agent relies on the most recent observation for decision-making. The trained policy is directly tested under packet loss scenarios ($p = 0.2, 0.5, 0.8, 1.0$), with results summarized in Table VII. The results show that even with a packet loss rate of $p = 0.8$, the success rate remained at 87%, dropping to 0 only when $p = 1.0$. This reflects the robustness of the BSAC-learned policy, indicating smooth and predictable system operation despite significant packet loss.

(3) Communication Interruptions: Communication interruptions are modeled by introducing periodic blackouts, where communication is interrupted for T_1 steps every T_0 steps. During these interruptions, the agent relied on the most recent observation. Policies trained are tested with $T_0 = 20$ and $T_1 = 5, 10, 15, 20$. Results are summarized in Table VIII. As shown in table, BSAC effectively handles communication interruptions. The results demonstrate smooth and stable control of the space robot, with minimal variance in the robot's motion during interruptions. By relying on recent observations, the agent continues making reasonable decisions, albeit with slightly longer task completion times. Once accurate state information is received, BSAC promptly generates precise actions, enabling the robot to quickly resume optimal operation.

Task Generalization. This part encompasses three scenarios: action execution noise, gradual base mass reduction, and obstacle avoidance tasks.

(1) Action Execution Noise: During action execution, sensor errors can introduce noise into the actions. This is typically modeled as Gaussian noise $a_\Delta(t) \sim \mathcal{N}(0, \delta^2)$, where δ takes values of 0.1, 0.2, 0.3, 0.4, and 0.5. The executed action is then $a_{noise}(t) = a_t + a_\Delta(t)$. For each δ, we conducted 100 trials and all results are presented in Table IX. Although action noise affected performance, its impact is less significant compared to observation noise, as shown in Table VI. This can be attributed to the fact that the BSAC actor network relies on states for decision-making, and while robust, accurate state representations are essential for optimal decisions. Additionally, real robotic systems often impose constraints on control inputs, clipping actions that exceed certain limits, whereas Gaussian noise in observations remains unrestricted. The results also indicate that larger action noise increases task completion times. For instance, with $\delta = 0.2$ and $\delta = 0.3$, success rates are identical, but the latter required more steps to complete the task.

(2) Gradual Base Mass Reduction: During task execution, fuel consumption can lead to a gradual decrease in the base's mass. We simulated this by reducing the base's mass by $m\%$ of its current value every 10 steps within a single trial, with $m = 2$, 4, and 6. The results are summarized in Table X. The data show that changes in base mass have minimal impact on the task success rate. This robustness is due to the state space in BSAC being independent of mass, making the algorithm resilient to mass variations. However, the average number of steps per trial increases with greater mass reduction, reflecting the added coupling between the base and the robotic arm,

TABLE X
COMPARISON OF GRADUAL BASE MASS REDUCTION.

m	Success Rate	Average Step
2	0.95	125.39±38.43
4	0.92	135.47±37.21
6	0.91	147.03±41.76

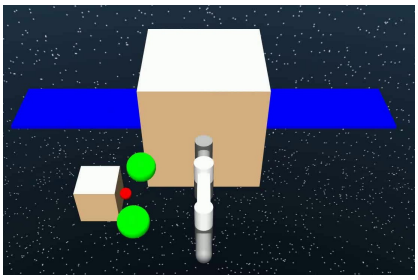


Fig. 8. Obstacle scenario. Two spherical obstacles near the target point interfering with robotic arm grasping.

which affects the agent's decision-making.

(3) Obstacle Avoidance Tasks: The new obstacle scenario is illustrated in Figure 8. We initially transferred the previously trained policy to the obstacle avoidance task without considering obstacle checks. The results, shown in Figure 9, indicate poor performance as the initial policy did not account for obstacles and thus failed to complete the task. To adapt the framework to this new task, we redesigned the reward function by incorporating a collision penalty, as shown in Eq. (9), in addition to the original components. The results of the retraining, presented in Figure 10, demonstrate that with the updated reward function, BSAC successfully guides the agent to avoid obstacles while completing the task with a smooth trajectory and minimal time, even in delayed environments. This highlights the adaptability and effectiveness of BSAC, further reinforcing its potential for a wide range of space teleoperation applications.

Real-System Validation. The real physical experiments are conducted on a smooth stone surface, with the dynamic parameters of the base and robotic arm specified in Section IV-A. To enable the free-floating capability of the space robot, we equip the base with four pneumatic feet. During the experiments, these feet control gas ejection to generate sufficient lift, effectively countering gravity. To simulate delay effects, we implement a waiting function on the agent side to introduce a deliberate delay in the agent's state after the specified time.

In the MuJoCo environment, the policy trained in a scenario with a delay of 6 is successfully transferred to the real system, as shown in Figures 11 and 12. Specifically, during the first six time steps, the agent, lacking observable information, opts to take no action, which results in the robotic arm remaining stationary. This is evident in Figure 11(a), where the arm is at its zero position, and in Figure 12(a), where the distance between the arm's end effector and the target remains unchanged. Once the agent begins to receive data, the end effector gradually approaches the target, successfully completing the task at $t = 48$. This progression is clearly illustrated in Figures 11(b) to 11(e) and Figure 12(a).

Additionally, Figures 12(b) and 12(c) represent the variations in the base's position and attitude throughout the process. Figure 12(b) indicates slight movement of the base in the x-y plane, while the z-axis exhibits minimal change. Figure 12(c) reveals that the base experiences negligible rotation about the x and y axes, with most rotation occurring around the z-axis. This demonstrates that during the approach to the target, the agent adeptly manages variations in the base's position and attitude, achieving the task with minimal adjustments while maintaining movement as close to a plane as possible.

V. CONCLUSION

In this paper, we address the inherent limitations of human operators in space teleoperation by introducing the Belief State Actor-Critic (BSAC) algorithm, which leverages deep reinforcement learning. This research represents a pioneering effort to employ an agent for remote operation in space capture tasks. We begin by modeling the space teleoperation process as a Markov decision process, thereby establishing a theoretical foundation for deeper analysis. We then present BSAC, which synergistically combines state augmentation and belief state techniques to mitigate the impact of delays in remote operations. Finally, we validate the effectiveness and robustness of our approach through extensive modeling and training in the MuJoCo physics engine, followed by empirical testing in real-world environments.

Future research directions could focus on three primary areas: (1) enhancing the robustness and real-world applicability of the control framework in actual space or equivalent environments by combining domain randomization with adaptive control strategies, (2) integrating multimodal information to enhance the agent's decision-making capabilities, such as text and visual information [40], and (3) incorporating human intervention within the current framework to facilitate human-agent collaboration in scenarios where the agent encounters challenges.

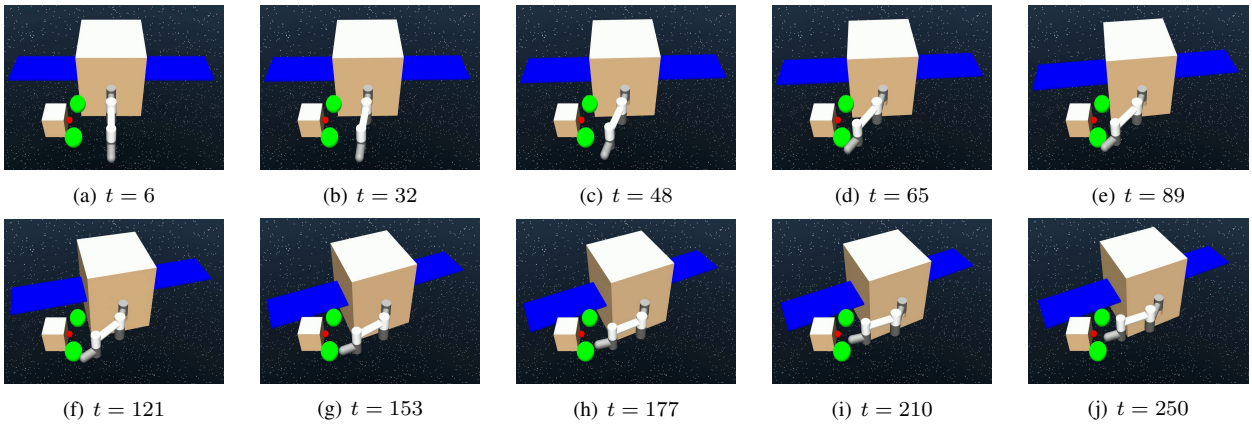
APPENDIX

The derivation process from Eq. (22) to Eq. (23) is as follows:

$$D_{KL}\left(\pi\left(\cdot|x_t\right)||\frac{exp\left(\mathbb{E}_{p(s_t|x_t)}\left[\mathcal{Q}^{\pi^{old}}(s_t,\cdot)\right]\right)}{Z^{\pi^{old}}\left(x_t\right)}\right) \tag{A.1}$$

$$\rightarrow \sum_{a_t\in A}\pi(a_t|x_t)\Bigg(log\pi(a_t|x_t)-\mathbb{E}_{p(s_t|x_t)}\left[\mathcal{Q}^{\pi^{old}}(s_t,\cdot)\right] + log Z^{\pi^{old}}\left(x_t\right)\Bigg) \tag{A.2}$$

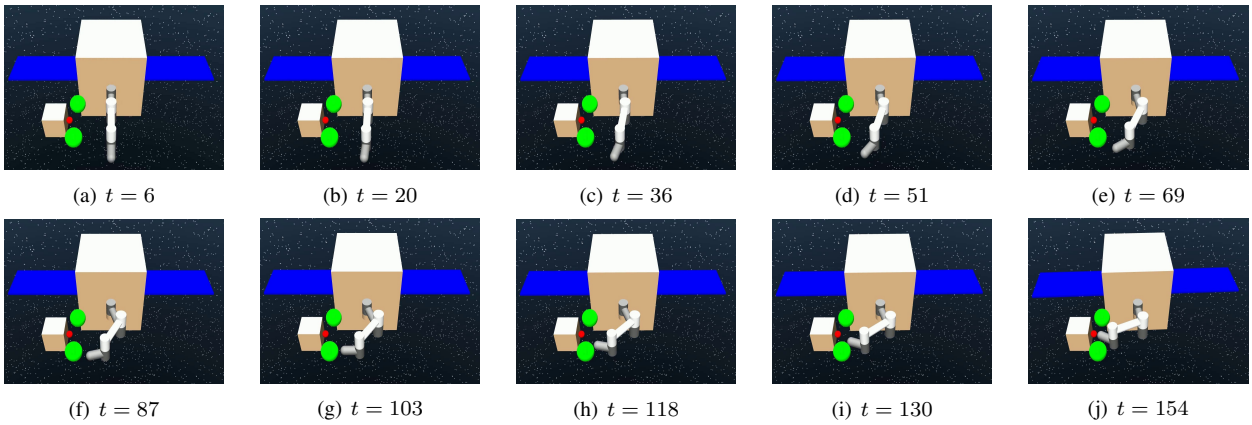
$$\rightarrow \sum_{a_t\in A}\pi(a_t|x_t)\Bigg(log\pi(a_t|x_t)-\mathbb{E}_{p(s_t|x_t)}\left[\mathcal{Q}^{\pi^{old}}\left(s_t,\cdot\right)\right] + \mathbb{E}_{p(s_t|x_t)}\left[log Z^{\pi^{old}}(s_t)\right]\Bigg) \tag{A.3}$$



(a) $t = 6$ (b) $t = 32$ (c) $t = 48$ (d) $t = 65$ (e) $t = 89$

(f) $t = 121$ (g) $t = 153$ (h) $t = 177$ (i) $t = 210$ (j) $t = 250$

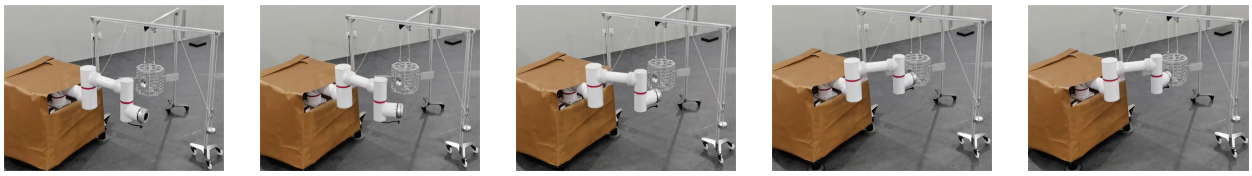
Fig. 9. Results of the direct policy transfer.



(a) $t = 6$ (b) $t = 20$ (c) $t = 36$ (d) $t = 51$ (e) $t = 69$

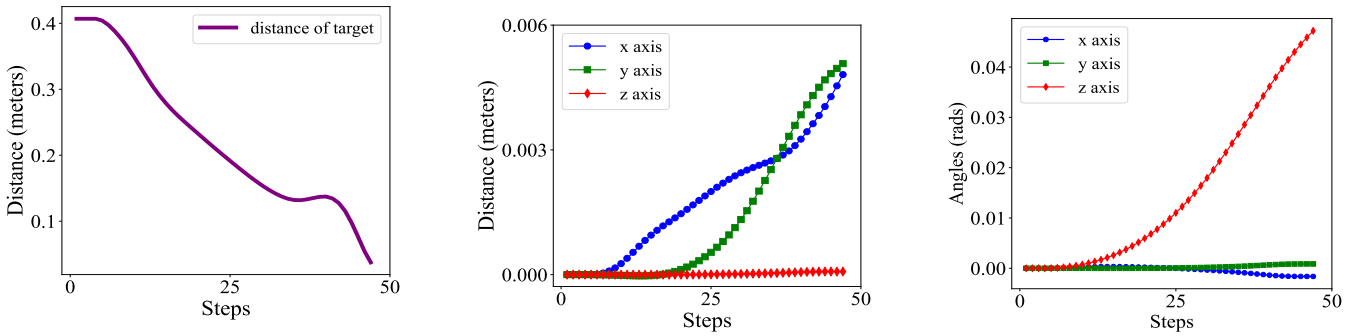
(f) $t = 87$ (g) $t = 103$ (h) $t = 118$ (i) $t = 130$ (j) $t = 154$

Fig. 10. Results of the retraining with collision penalty.



(a) $t = 6$ (b) $t = 18$ (c) $t = 32$ (d) $t = 40$ (e) $t = 48$

Fig. 11. Different stages of the space robot grasping task in the real system.



(a) Distance between the end effector and the target.

(b) Position changes of the base.

(c) Attitude changes of the base.

Fig. 12. Position and attitude changes of the space robot components during target capture.

$$\rightarrow \mathbb{E}_{p(s_t|x_t)}\left[\sum_{a_t\in A}\pi(a_t|x_t)\Big(log\pi(a_t|x_t) - \mathcal{Q}^{\pi^{old}}(s_t,\cdot) + logZ^{\pi^{old}}(s_t)\Big)\right] \tag{A.4}$$

$$\rightarrow \mathbb{E}_{p(s_t|x_t)}\left[D_{KL}(\pi(\cdot|x_t)||\ \frac{exp\left(\mathcal{Q}^{\pi^{old}}(s_t,\cdot)\right)}{Z^{\pi^{old}}(s_t)}\right] \tag{A.5}$$

Eq. (A.1) is expanded using the KL-divergence, which directly leads to Eq. (A.2). Since $logZ^{\pi^{old}}(s_t)$ and $\mathbb{E}_{p(s_t|x_t)}\left[logZ^{\pi^{old}}(s_t)\right]$ are independent of the policy π, further derivation yields Eq. (A.3). Next, by interchanging the summation and the expectation, Eq. (A.4) is obtained. Finally, applying the definition of KL-divergence, we arrive at Eq. (A.5).

REFERENCES

[1] Q. Gao, J. Li, Y. Zhu, S. Wang, J. Liufu, and J. Liu, "Hand gesture teleoperation for dexterous manipulators in space station by using monocular hand motion capture," *Acta Astronautica*, vol. 204, pp. 630–639, 2023.

[2] W. Pryor, B. P. Vagvolgyi, A. Deguet, S. Leonard, L. L. Whitcomb, and P. Kazanzides, "Interactive planning and supervised execution for high-risk, high-latency teleoperation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1857–1864.

[3] W. Zhang, F. Li, J. Li, and Q. Cheng, "Review of on-orbit robotic arm active debris capture removal methods," *Aerospace*, vol. 10, no. 1, p. 13, 2022.

[4] W. Doggett, "Robotic assembly of truss structures for space systems and future research plans," in *Proceedings, IEEE Aerospace Conference*, vol. 7. IEEE, 2002, pp. 7–7.

[5] X. Wang, W. Xu, B. Liang, and C. Li, "General scheme of teleoperation for space robot," in *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2008, pp. 341–346.

[6] X. Zhang and J. Liu, "Autonomous trajectory planner for space telerobots capturing space debris under the teleprogramming framework," *Advances in Mechanical Engineering*, vol. 9, no. 9, p. 1687814017723298, 2017.

[7] J. Funda, T. S. Lindsay, and R. P. Paul, "Teleprogramming: Toward delay-invariant remote manipulation," *Presence: Teleoperators & Virtual Environments*, vol. 1, no. 1, pp. 29–44, 1992.

[8] E. Nuño, L. Basañez, and R. Ortega, "Passivity-based control for bilateral teleoperation: A tutorial," *Automatica*, vol. 47, no. 3, pp. 485–495, 2011.

[9] Z. Wang, Z. Chen, B. Liang, and B. Zhang, "A novel adaptive finite time controller for bilateral teleoperation system," *Acta Astronautica*, vol. 144, pp. 263–270, 2018.

[10] Z. Chen, B. Liang, and T. Zhang, "A self-adjusting compliant bilateral control scheme for time-delay teleoperation in constrained environment," *Acta Astronautica*, vol. 122, pp. 185–195, 2016.

[11] A. Kheddar, E.-S. Neo, R. Tadakuma, and K. Yokoi, "Enhanced teleoperation through virtual reality techniques," *Advances in telerobotics*, pp. 139–159, 2007.

[12] Z. Deng and M. Jagersand, "Predictive display system for telemanipulation using image-based modeling and rendering," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2797–2802.

[13] A. Johnson, *Attention: Theory and Practice*. Sage Publications, 2004.

[14] R. A. Grier, J. S. Warm, W. N. Dember, G. Matthews, T. L. Galinsky, J. L. Szalma, and R. Parasuraman, "The vigilance decrement reflects limitations in effortful attention, not mindlessness," *Human factors*, vol. 45, no. 3, pp. 349–359, 2003.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] J. Park, T. Kim, S. Seong, and S. Koo, "Control automation in the heat-up mode of a nuclear power plant using reinforcement learning," *Progress in Nuclear Energy*, vol. 145, p. 104107, 2022.

[17] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[18] C. Yan, Q. Zhang, Z. Liu, X. Wang, and B. Liang, "Control of free-floating space robots to capture targets using soft q-learning," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 654–660.

[19] S. Wang, Y. Cao, X. Zheng, and T. Zhang, "An end-to-end trajectory planning strategy for free-floating space robots," in *2021 40th Chinese Control Conference (CCC)*. IEEE, 2021, pp. 4236–4241.

[20] Y.-H. Wu, Z.-C. Yu, C.-Y. Li, M.-J. He, B. Hua, and Z.-M. Chen, "Reinforcement learning in dual-arm trajectory planning for a free-floating space robot," *Aerospace Science and Technology*, vol. 98, p. 105657, 2020.

[21] S. Wang, X. Zheng, Y. Cao, and T. Zhang, "A multi-target trajectory planning of a 6-dof free-floating space robot via reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3724–3730.

[22] Y. Cao, S. Wang, X. Zheng, W. Ma, X. Xie, and L. Liu, "Reinforcement learning with prior policy guidance for motion planning of dual-arm free-floating space robot," *Aerospace Science and Technology*, vol. 136, p. 108098, 2023.

[23] Y. Li, X. Hao, Y. She, S. Li, and M. Yu, "Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning," *Aerospace Science and Technology*, vol. 109, p. 106446, 2021.

[24] S. Wang, Y. Cao, X. Zheng, and T. Zhang, "Collision-free trajectory planning for a 6-dof free-floating space robot via hierarchical decoupling optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4953–4960, 2022.

[25] C. Yang, J. Yang, X. Wang, and B. Liang, "Control of space flexible manipulator using soft actor-critic and random network distillation," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 3019–3024.

[26] D. Jiang, Z. Cai, H. Peng, and Z. Wu, "Coordinated control based on reinforcement learning for dual-arm continuum manipulators in space capture missions," *Journal of Aerospace Engineering*, vol. 34, no. 6, p. 04021087, 2021.

[27] P. Liotet, D. Maran, L. Bisi, and M. Restelli, "Delayed reinforcement learning by imitation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 13 528–13 556.

[28] M. Xie, B. Xia, Y. Yu, X. Wang, and Y. Chang, "Addressing delays in reinforcement learning via delayed adversarial imitation learning," in *International Conference on Artificial Neural Networks*. Springer, 2023, pp. 271–282.

[29] B. Xia, Y. Kong, Y. Chang, B. Yuan, Z. Li, X. Wang, and B. Liang, "Deer: A delay-resilient framework for reinforcement learning with variable delays," *arXiv preprint arXiv:2406.03102*, 2024.

[30] B. Xia, Z. Yang, M. Xie, Y. Chang, B. Yuan, Z. Li, X. Wang, and B. Liang, "Solving time-delay issues in reinforcement learning via transformers," *Applied Intelligence*, pp. 1–21, 2024.

[31] K. V. Katsikopoulos and S. E. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE transactions on automatic control*, vol. 48, no. 4, pp. 568–574, 2003.

[32] Y. Bouteiller, S. Ramstedt, G. Beltrame, C. Pal, and J. Binas, "Reinforcement learning with random delays," in *International conference on learning representations*, 2021.

[33] T. Hester and P. Stone, "Texplore: real-time sample-efficient reinforcement learning for robots," *Machine learning*, vol. 90, pp. 385–429, 2013.

[34] V. Firoiu, T. Ju, and J. Tenenbaum, "At human speed: Deep reinforcement learning with action delay," *arXiv preprint arXiv:1810.07286*, 2018.

[35] E. Derman, G. Dalal, and S. Mannor, "Acting in delayed environments with non-stationary markov policies," in *International Conference on Learning Representations*, 2020.

[36] M. Agarwal and V. Aggarwal, "Blind decision making: Reinforcement learning with delayed observations," *Pattern Recognition Letters*, vol. 150, pp. 176–182, 2021.

[37] P. Liotet, E. Venneri, and M. Restelli, "Learning a belief representation for delayed reinforcement learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[38] J. Kim, H. Kim, J. Kang, J. Baek, and S. Han, "Belief projection-based reinforcement learning for environments with delayed feedback," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[39] Y. Umetani and K. Yoshida, "Resolved motion rate control of space manipulators with generalized jacobian matrix," Ph.D. dissertation, Tohoku University, 1989.
[40] Z. Yuan, T. Wei, S. Cheng, G. Zhang, Y. Chen, and H. Xu, "Learning to manipulate anywhere: A visual generalizable framework for reinforcement learning," in *8th Annual Conference on Robot Learning*, 2024.

Bo Xia received the B.Sc. degree from the Department of Mathematics, Beihang University, Beijing, China, in 2014, and the M.Sc. degree in control engineering from the Department of Automation, Tsinghua University, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree in control science and engineering in the Shenzhen International Graduate School, Tsinghua University, Shenzhen, China.

His research interests include decision-making, reinforcement learning and space robots.

Xianru Tian received the B.Sc degree in Mechatronics engineering from Harbin Institute of Technology, Harbin, China, in 2023. He is currently pursuing the M.Sc degree in Electronic Information (Artificial Intelligence) in Shenzhen International Graduate School, Tsinghua University, Shenzhen, China.

His research interests include reinforcement learning and robot control.

Bo Yuan (Senior Member, IEEE) received the B.E. degree in computer science from the Nanjing University of Science and Technology, Nanjing, China, in 1998, and the M.Sc. and Ph.D. degrees in computer science from The University of Queensland (UQ), St Lucia, QLD, Australia, in 2002 and 2006, respectively.

From 2006 to 2007, he was a Research Officer on a project funded by the Australian Research Council, UQ. From 2007 to 2021, he was a faculty member (Lecturer: 2007-2009 and Associate Professor: 2009-2021) in the Division of Informatics, Tsinghua Shenzhen International Graduate School, P.R. China, and served as the Deputy Director of Office of Academic Affairs (2013-2020). In July 2021, he co-founded Shenzhen Wisdom & Strategy Technology Co., Ltd., an innovative K-12 AI education solution provider. He has authored or coauthored more than 110 papers in refereed international conferences and journals. His research interests include data science, evolutionary computation, and reinforcement learning.

Chunju Yang received the B.E. degree in Artificial Intelligence and Automation at Huazhong University of Science and Technology, Wuhan, China, in 2024. She is currently pursuing the M.E. degree in Electronic Information (Artificial Intelligence) in Shenzhen International Graduate School, Tsinghua University, Shenzhen China.

Her research interests include imitation learning, motion control of cable robots.

Zhiheng Li (Member, IEEE) received the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 2009. He is currently an Associate Professor with the Department of Automation, Tsinghua University, and with the Graduate School at Shenzhen, Tsinghua University, Shenzhen, China. His research interests include traffic operation, advanced traffic management system, urban traffic planning and intelligent transportation systems. Dr. Li serves as an Associate Editor for IEEE Transactions on Intelligent Transportation Systems.

Bin Liang (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in control engineering from the Honors College, Northwestern Polytechnical University, Xi'an, China, in 1989 and 1991, respectively, and the Ph.D. degree in control engineering from the Department of Precision Instrument, Tsinghua University, Beijing, China, in 1994.

From 1994 to 2003, he held his positions as a Post-Doctoral Researcher, an Associate Researcher, and a Researcher with the China Academy of Space Technology (CAST), Beijing. From 2003 to 2007, he held his positions as a Researcher and an Assistant Chief Engineer with the China Aerospace Science and Technology Corporation, Beijing. He is currently a Professor with the Research Center for Navigation and Control, Department of Automation, Tsinghua University. His research interests include modeling and control of intelligent robotic systems, teleoperation, and intelligent sensing technology.

Xueqian Wang (Member, IEEE) received the M.Sc. and Ph.D. degrees in control science and engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 2005 and 2010, respectively.

From June 2010 to February 2014, he was a Post-Doctoral Researcher with HIT. From March 2014 to November 2019, he was an Associate Professor with the Division of Informatics, Shenzhen International Graduate School, Tsinghua University, Shenzhen, China. He is currently a Professor and the Leader of the Center for Artificial Intelligence and Robotics, Shenzhen International Graduate School, Tsinghua University. His research interests include robot dynamics and control, teleoperation, intelligent decision-making and game playing, and fault diagnosis.